

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

Μεταπτυχιακή Εξιδείκευση στα Πληροφοριακά Συστήματα (MSc)

Διπλωματική Εργασία

«Ανάπτυξη εφαρμογής έξυπνης διαχείρισης αποκομιδής απορριμάτων σε μία πόλη»

«ΜΑΡΙΑ ΔΑΚΑΝΑΛΗ»

Επιβλέπων καθηγητής: Κωνσταντίνος Παξιμάδης

Πάτρα, Σεπτέμβριος 2026

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία της φοιτήτριας Μαρίας Δακανάλη που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο ΕΑΠ, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.

ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

«Ανάπτυξη εφαρμογής έξυπνης διαχείρισης αποκομιδής απορριμμάτων σε μία πόλη»

ΜΑΡΙΑ ΔΑΚΑΝΑΛΗ

Επιτροπή Επίβλεψης Πτυχιακής / Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:
Κωνσταντίνος Παξιμάδης

Διδάκτωρ
Μηχανικός Ηλεκτρονικών
Υπολογιστών και Πληροφορικής του
Πανεπιστημίου Πατρών

Συν-Επιβλέπων Καθηγητής:
Ευθύμιος Ταμπούρης

Καθηγητής
Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας

Πάτρα, Σεπτέμβριος 2026

«Αφιερωμένο σε όσους με στήριξαν και πίστεψαν σε εμένα. Στην οικογένεια μου και τις φίλες μου, για την εμψύχωση τους σε αυτό το ταξίδι της γνώσης.»

Περίληψη

Η συλλογή και διαχείριση των αστικών στερεών απορριμμάτων είναι μια από τις δημοτικές υπηρεσίες στους Οργανισμούς Τοπικής Αυτοδιοίκησης (Ο.Τ.Α.) που, παρά τον κρίσιμο ρόλο τους στην καθημερινότητα, δεν έχουν ακόμη ενσωματώσει συστηματικά τα εργαλεία που η τεχνολογική εξέλιξη έχει στη διάθεση της όπως αυτής της ψηφιακής τεχνολογίας. Στην Ελλάδα, ορισμένοι δήμοι ανέπτυξαν μεμονωμένες εφαρμογές παρακολούθησης στόλου οχημάτων, χωρίς όμως να ενσωματωθούν πληροφορίες για την πραγματική κατάσταση των κάδων απορριμμάτων. Το αποτέλεσμα είναι οχήματα που εκτελούν προκαθορισμένα δρομολόγια ανεξάρτητα από το αν υπάρχει πράγματι ανάγκη αποκομιδής. Η ανατροπή αυτής της πρακτικής, που βασίζεται σε πραγματικά δεδομένα χρήσης, μπορεί να μειώσει τις λειτουργικές δαπάνες, να αποσυμφορήσει την κυκλοφορία και να τροφοδοτήσεις με αξιόπιστα στοιχεία τον καθορισμό δημοτικών τελών.

Η παρούσα Διπλωματική Εργασία (ΔΕ) επικεντρώνεται στην ανάπτυξη πληροφοριακού συστήματος που να αντιμετωπίζει αυτό ακριβώς το κενό. Ουσιαστικά επικεντρώνεται σε ένα δήμο ως μελέτη περίπτωσης και στην ανάπτυξη εφαρμογής έξυπνης διαχείρισης αποκομιδής απορριμάτων μιας πόλης, και πιο συγκεκριμένα στο δήμο Μαλεβιζίου, του νομού Ηρακλείου Κρήτης. Αξιοποιεί τεχνολογίες IoT για την παρακολούθηση κάδων απορριμμάτων σε πραγματικό χρόνο και υπηρεσίες για την αποθήκευση και επεξεργασία δεδομένων, η οποία καταγράφει τη στάθμη πλήρωσης και τις γεωγραφικές συντενταγμένες κα τα αποστέλλει σε κεντρική βάση δεδομένων σε τακτά χρονικά διαστήματα, καλύπτοντας τόσο σύμμεικτα όσο και ανακυκλώσιμα απόβλητα. Με βάση αυτές τις πληροφορίες, η διαδρομή κάθε οχήματος μπορεί να σχεδιαστεί δυναμικά, αντί να ακολουθεί σταθερό πρόγραμμα.

Η εφαρμογή στηρίζεται γύρω από από τέσσερις άξονες. Ο πρώτος αφορά την προσομοίωση της συσκευής αισθητήρων των κάδων που αποτελεί και το επίπεδο συλλογής και αξιοποίησης μεγάλου όγκου δεδομένων για τη βελτιστοποίηση δρομολογίων, τη διαχείριση πόρων και την παροχή ανοιχτών δεδομένων. Ο δεύτερος επικεντρώνεται στη συνεχή ενημέρωση για την κατάσταση των κάδων από την πλευρά των οδηγών των απορριματοφόρων μέσω της εφαρμογής Smart Garbage Vehicle Truck που παρέχει πρόσβαση στον οδηγό για την προτεραιότητα συγκομιδής απορριμάτων και της βέλτιστης διαδρομής και δίνοντας το στίγμα της θέσης τόσο του κάδου όσο και του οχήματος, στο διαχειριστής της πλατφόρμας. Ο τρίτος αναφέρεται στη εφαρμογή του κέντρου ελέγχου

(smart waste management dashboard) του δήμου όπου ενημερώνεται για την πληρότητα των κάδων, οργανώνει ελέγχει και στέλνει μηνύματα προς τους οδηγούς όσον αφορά τα δρομολόγια και ελέγχει τη λειτουργία, τη συντήρηση, και την διαθεσιμότητα των οχημάτων του καθως και μπορεί να εξάγει χρήσιμές αναφορές στατιστικών από το σύστημα και να διαχειριστεί επίσης και τα μηνύματα από την πύλη πολιτών. Τέλος, ο τέταρτος άξονας αφορά την επικοινωνία μέσω της εφαμοργής της πύλης των πολιτών, με τον δήμο. Μέσα από την εφαρμογή αυτή γεφυρώνεται η άμεση επικοινωνία σε ζητήματα πολιτών που αφορούν τους εκάστοτε δήμους και την αποκομιδή των απορριμμάτων τους.

Λέξεις – Κλειδιά

Έξυπνη διαχείριση αποκομιδή απορριμμάτων, έξυπνη διαχείριση απορριμματοφόρων, διαχείριση κάδων, έξυπνοι κάδοι απορριμμάτων, πύλη πολιτών, διαδίκτυο των πραγμάτων, internet of things.

«Development of a smart waste collection management application in a city»

«Maria Dakanali»

(στην αγγλική γλώσσα)

Abstract

The collection and management of municipal solid waste is one of the municipal services in Local Government Organizations that, despite their crucial role in everyday life, have not yet systematically integrated the tools that technological development has made available to them, such as digital technology. In Greece, some municipalities have developed individual vehicle fleet monitoring applications, without integrating information on the actual condition of the waste bins. The result is vehicles that run predetermined routes regardless of whether there is an actual need for collection. Overturning this practice, which is based on real usage data, can reduce operating costs, decongest traffic and provide reliable data for the determination of municipal fees. This Diploma Thesis (DT) focuses on the development of an information system that addresses this very gap. Essentially, it focuses on a municipality as a case study and on the development of a smart waste management application for a city, and more specifically in the municipality of Malevizi, in the prefecture of Heraklion, Crete. It utilizes IoT technologies for real-time monitoring of waste bins and services for data storage and processing, which records the filling level and geographic coordinates and sends them to a central database at regular intervals, covering both mixed and recyclable waste. Based on this information, the route of each vehicle can be planned dynamically, instead of following a fixed schedule. The application is based on four axes. The first concerns the simulation of the bin sensor device which is also the level of collection and utilization of large amounts of data for route optimization, resource management and the provision of open data. The second focuses on the continuous information on the status of the bins from the point of view of the garbage truck drivers through the Smart Garbage Vehicle Truck application that provides access to the driver for the priority of waste

collection and the optimal route and giving the position of both the bin and the vehicle to the platform administrator. The third refers to the application of the control center (smart waste management dashboard) of the municipality where it is informed about the fullness of the bins, organizes, controls and sends messages to the drivers regarding the routes and controls the operation, maintenance, and availability of its vehicles as well as being able to extract useful statistical reports from the system and also manage the messages from the citizen portal. Finally, the fourth axis concerns communication through the implementation of the citizen portal with the municipality. Through this application, direct communication is bridged on citizen issues concerning each municipality and their waste collection..

Keywords

Smart waste collection management, smart waste truck management, bin management, smart waste bins, citizen portal, internet of things.

Περιεχόμενα

Περίληψη v
Abstract vii
Περιεχόμενα ix
Κατάλογος Εικόνων / Σχημάτων xiv
Κατάλογος Πινάκων xv
Συντομογραφίες & Ακρωνύμια xvi
1. Εισαγωγή 1
1.1 Το πρόβλημα της διαχείρισης απορριμμάτων στις σύγχρονες πόλεις 1
1.1.1 Το ευρωπαϊκό πλαίσιο – αρχές και στόχοι 1
1.1.2 Το ελληνικό πλαίσιο – ΕΣΔΑ 2020-2030 και νομοθετικές πρωτοβουλίες 1
1.1.3 Κατάσταση σήμερα — βασικά στοιχεία και τάσεις (ελληνικός χώρος) 2
1.1.4 Περιβαλλοντικές επιπτώσεις 2
1.1.5 Λειτουργικές και οικονομικές επιπτώσεις 3
1.1.6 Υφιστάμενες προκλήσεις στον ελληνικό χώρο 4
1.1.7 Σημεία αιχμής για την ελληνική πολιτική και πρακτική 4
1.1.8 Συμπεράσματα για τη διαχείριση απορριμμάτων στην Ελλάδα 4
1.2 Κίνητρα και Αναγκαιότητα της Έρευνας: Σημασία της Έξυπνης Διαχείρισης Απορριμμάτων στο Πλαίσιο των Smart Cities στην Ελλάδα 5
1.2.1 Κίνητρα για την έρευνα 5
1.2.2 Σημασία της Έξυπνης Διαχείρισης Απορριμμάτων στα Smart Cities στην Ελλάδα 7
1.3 Σκοπός και στόχοι της εργασίας 10
1.4 Ερευνητικά Ερωτήματα 11
1.4.1 Κύριο Ερευνητικό Ερώτημα (Primary Research Question): 11
1.4.2 Δευτερεύοντα Ερευνητικά Υποερωτήματα: 11
1.4.3 Διατύπωση Υποθέσεων Έρευνας 12
Κεντρική Ερευνητική Υπόθεση 12
Εναλλακτική Ερευνητική Υπόθεση 12
Ειδικές Ερευνητικές Υποθέσεις 12
Αρνητικές Υποθέσεις (προαιρετικά για στατιστική ανάλυση) 13
1.5 Μεθοδολογία προσέγγισης 13
2. Έξυπνες πόλεις, χαρακτηριστικά και τεχνολογίες (Smart Cities) 15
2.1. Έννοια και χαρακτηριστικά των Έξυπνων Πόλεων (Smart Cities) 15
2.2. Σύγχρονες Τεχνολογίες στις Έξυπνες Πόλεις 15
2.2.1. Internet of Things (IoT) 16
2.2.2. Big Data & Predictive Analytics 16
2.2.3. Cloud Computing 16
2.2.4. GIS – Γεωγραφικά Συστήματα Πληροφοριών 16
2.2.5. Mobile Applications 16
2.3. Έξυπνη Διαχείριση Απορριμμάτων (Smart Waste Management) 17
2.3.1. Ορισμός 17
2.3.2. Στόχοι 17
2.3.3. Βασικά Συστατικά ενός Έξυπνου Συστήματος Διαχείρισης Απορριμμάτων 18

2.4. Διεθνείς Πρακτικές και Παραδείγματα Έξυπνης Διαχείρισης Απορριμμάτων.....18
2.4.1. Ευρωπαϊκές Πρωτοβουλίες 19
2.4.2. Ιδιωτικές Τεχνολογικές Λύσεις σε Διεθνές Επίπεδο.....19
2.5. Έξυπνη Διαχείριση Απορριμμάτων στον Ελληνικό Χώρο 20
2.5.1. Πιλοτικές Εφαρμογές Ελληνικών Δήμων.....20
2.5.2. Προκλήσεις που Αντιμετωπίζει η Ελλάδα 20
2.6. Θεωρητική Σύνδεση Smart Cities και Smart Waste Management.....21
2.7. Επεξεργασία και Ανάλυση Δεδομένων σε Πραγματικό Χρόνο 21
2.7.1. Ροές δεδομένων (Data Pipelines) 21
2.7.2. Εργαλεία Επεξεργασίας Δεδομένων.....22
2.8. Αλγόριθμοι Βελτιστοποίησης και Έξυπνη Δρομολόγηση Αποκομιδής 23
2.8.1. Vehicle Routing Problem (VRP).....24
2.8.2. Κλασικοί Αλγόριθμοι.....25
2.8.3. Μεταευρετικοί Αλγόριθμοι (Metaheuristics) 25
2.8.4. Εφαρμογές σε Έξυπνες Πόλεις.....26
2.9. Προγνωστική Ανάλυση (Predictive Analytics) στην Έξυπνη Διαχείριση Πόλεων 27
2.9.1. Προγνωστικά Μοντέλα 27
2.9.2. Εφαρμογές Προγνωστικής Ανάλυσης στη Διαχείριση Απορριμμάτων 28
2.10. Συστήματα Υποστήριξης Αποφάσεων και Dashboards 29
2.10.1. Κύριες Λειτουργίες των Συστημάτων Υποστήριξης Αποφάσεων (DSS) 29
2.10.2. Κατηγορίες Πινάκων Ελέγχου (Dashboards) 30
2.10.3. Εργαλεία.....31
2.11. Ανασκόπηση Συναφών Ερευνητικών και Εμπορικών Έργων 31
2.11.1. Ερευνητικά Έργα (Ευρωπαϊκά & Διεθνή) 31
2.11.2. Εμπορικές Λύσεις Διαχείρισης Αστικών Απορριμμάτων με Έξυπνες Τεχνολογίες 32
2.11.3. Συγκριτική Αποτίμηση Υφιστάμενων Λύσεων.....33
2.12. Συμπεράσματα Κεφαλαίου.....34
3. Case Study – Δήμος Μαλεβιζίου με έδρα το Γάζι της Περιφέρειας Κρήτης 35
3.1. Περιγραφή Δήμου – Διοικητικά και Χωροταξικά 35
3.2. Βασικά Χαρακτηριστικά του Δήμου 35
3.3. Δημοτικές Ενότητες Δήμου Μαλεβιζίου.....37
3.4. Οικονομική Κατάσταση Δήμου 37
3.5. Υφιστάμενη Υπηρεσία Καθαριότητας.....38
3.6. Ανάλυση Υφιστάμενου Κόστους.....39
3.6.1. Εκτίμηση Ετήσιου Κόστους 39
3.7. Σχεδιασμός Smart Waste System για το δήμο Μαλεβιζίου 39
3.7.1. Εκτίμηση Αριθμού Κάδων.....39
3.8. Τεχνικές Προδιαγραφές Συστήματος.....40
3.9. Λεπτομερής Ανάλυση Κόστους Επένδυσης.....40
3.9.1. Κόστος Hardware 40
3.9.2. Κόστος Software.....41
3.9.3. Κόστος Υλοποίησης.....41
3.9.4. Συνολική Αρχική Επένδυση.....41
3.10. Ετήσια Κόστη Λειτουργίας.....42
3.10.1. Cloud και Τηλεπικοινωνίες 42

3.10.2. Συντήρηση και Support 42
3.10.3. Ανάλυση Εξοικονομήσεων 42
3.10.3.1. Μείωση Κόστους Καυσίμων 42
3.10.3.2. Μείωση Κόστους Προσωπικού 42
3.10.3.3. Μείωση Κόστους Συντήρησης 42
3.10.3.4. Λοιπές Εξοικονομήσεις 43
3.11. Συνολική Οικονομική Ανάλυση 43
3.11.1. Ετήσιες Εξοικονομήσεις 43
3.11.2. Καθαρές Εξοικονομήσεις 43
3.11.3. ROI Ανάλυση 43
3.12. Ανάλυση Ευαισθησίας 43
3.12.1. Συντηρητικό Σενάριο (-20% εξοικονομήσεις) 43
3.12.2. Αισιόδοξο Σενάριο (+20% εξοικονομήσεις) 43
3.13. Ανάλυση Κινδύνων 44
3.14. Χρηματοδότηση και Υλοποίηση 44
3.14.1. Πηγές Χρηματοδότησης 44
3.15. Προτεινόμενο Χρονοδιάγραμμα 44
3.16. Οφέλη για τον Δήμο Μαλεβιζίου 44
3.16.1. Οικονομικά Οφέλη 44
3.16.2. Περιβαλλοντικά Οφέλη 44
3.16.3. Κοινωνικά Οφέλη 45
3.17. Συγκριτική Ανάλυση 45
3.17.1. Σύγκριση με Άλλους Δήμους 45
3.17.2. Ανταγωνιστικά Πλεονεκτήματα 45
3.18. Άμεσες Ενέργειες 45
3.19. Μακροπρόθεσμο Όραμα του Δήμου Μαλεβιζίου 45
3.20. Συμπεράσματα 46
4. Ανάλυση Απαιτήσεων και Σχεδίαση Συστήματος 48
4.1. Περιγραφή του Συστήματος και των Εμπλεκομένων 48
4.2. Ανάλυση Απαιτήσεων 48
4.2.1. Λειτουργικές Απαιτήσεις 48
4.2.2. Μη Λειτουργικές Απαιτήσεις 49
4.3. Αρχιτεκτονική του Συστήματος 49
4.4. Μοντέλο Δεδομένων 50
4.5. Διαγράμματα UML 50
4.5.1. Use Case Diagram 50
4.5.2. Sequence Diagram 51
4.5.3. Activity Diagram 54
4.6. Σχεδίαση Διεπαφών Χρήστη 54
4.7. Επιλογή Τεχνολογιών 54
4.8. Συμπεράσματα Κεφαλαίου 55
5. Ανάπτυξη Λογισμικού Smart Waste Management System 56
5.1. Καταγραφή Δραστηριοτήτων 56
5.1.1. Καταγραφή απαιτήσεων εφαρμογής Smart Waste Management System 56

5.1.2. Καταγραφή απαιτήσεων web server 58
5.1.2.1. Τι είναι το Docker Engine – Docker container 58
5.1.2.2. Καθορισμός Τεχνικών Προδιαγραφών 59
5.1.2.3. Docker images – Αναλυτικές Προδιαγραφές 60
5.2. Ανάλυση ΠΣ 61
5.2.1. Διάγραμμα Περιπτώσεων Χρήσης του Smart Waste Management System 61
5.2.2. Διάγραμμα Δραστηριοτήτων εφαρμογής Smart Waste Management System 62
5.2.3. Λεκτική περιγραφή Περιπτώσεων Χρήσης (ΠΧ) 65
5.3. Ανάλυση Πληροφοριακού Συστήματος Smart Waste Management System 67
5.4. Σχεδιασμός 67
5.4.1. Υλοποίηση Βάσης Δεδομένων 68
5.4.1.1. Εγκατάσταση Βάσης Δεδομένων PostgreSQL – TimescaleDB 68
Βήμα 1: Εγκατάσταση και Εκκίνηση του TimescaleDB με Docker 69
Βήμα 2: Σύνδεση στη Βάση Δεδομένων μας 70
Εγκατάσταση FASTAPI 73
5.5. Υλοποίηση Λειτουργιών 74
5.5.1. FastAPI Backend — Ανάπτυξη 74
5.5.2. React Frontend — Ανάπτυξη 74
5.6. Εξωτερικά Πληροφοριακά Συστήματα 75
5.6.1. Συσκευή SmartBin στο simulator Wokwi 75
5.6.1.1. Τι είναι το ngrok 75
5.6.1.2. Ανάπτυξη λογισμικού συσκευής στο wokwi 75
5.6.2. Συσκευή Προσομοίωσης Αισθητήρων στους κάδους με χρήση Γλώσσας Python 83
5.7. Ανάπτυξη κώδικα εφαρμογής Smart Waste Management System 87
5.7.1. Στόχοι Συστήματος 87
5.7.2. Γεωγραφικές Ζώνες Δήμου Μαλεβιζίου 87
5.7.3. Στρώματα Αρχιτεκτονικής 88
5.8. Τεχνολογίες και Εργαλεία ΠΣ Smart Waste Management System 89
5.8.1. Backend 89
5.8.2. Frontend 89
5.8.3. IoT & Επικοινωνία 89
5.9. Δομή Αρχείων του ΠΣ Smart Waste management System 90
5.10. Σχεδιασμός Βάσης Δεδομένων 91
5.10.1. Πίνακες Βάσης Δεδομένων 91
5.10.2. Database Views (Προβολές Βάσης Δεδομένων) 92
5.10.3. Παραμετροποίηση Σύνδεσης 92
5.11. Τεκμηρίωση REST API 92
5.11.1. Πιστοποίηση (Auth) 93
5.11.2. Κάδοι & Αισθητήρες 93
5.11.3. Βελτιστοποίηση Διαδρομής (ΠΧ2) 94
5.12. Αποστολή & Οχήματα 95
5.13. Πύλη Πολιτών (ΠΧ3) 96

6. Υλοποίηση Εφαρμογής και Περιπτώσεων Χρήσης 98
6.1. ΠΧ1 - Πιστοποίηση Χρήστη 98
6.2. ΠΧ2 - Βελτιστοποίηση Διαδρομής 99
6.3. ΠΧ3 - Πύλη Πολιτών 100
6.4. Οδηγίες Εγκατάστασης και Εκτέλεσης 103
6.4.1. Προαπαιτούμενα 103
6.4.2. Εγκατάσταση 104
6.5. Εκτέλεση Συστήματος 104
6.6. Προεπιλεγμένοι Χρήστες 105
6.7. Λειτουργίες Κεντρικού Dashboard 105
6.8. CollectionGarbageTruck - Εφαρμογή Οδηγού Οχήματος Απορριμματοφόρου 107
6.8.1. Σύνδεση 107
6.8.2. Καρτέλες 107
6.8.3. Διαχείριση Χωρητικότητας 107
6.9. Ασφάλεια και Προσβασιμότητα 108
6.9.1. Ασφάλεια 108
6.9.2. Προσβασιμότητα (WCAG) 108
6.10. Συμπεράσματα και Μελλοντικές Επεκτάσεις 108
6.10.1. Συμπεράσματα 108
6.10.2. Μελλοντικές Επεκτάσεις 109
Βιβλιογραφία 110
Παράρτημα Α: «Κώδικας του backend του ΠΣ Smart Waste Management System» 114
Παράρτημα Β: «Κώδικας του frontend του ΠΣ Smart Waste Management System» 211
Παράρτημα Γ: «Κώδικας του CollectGarbageTruck του ΠΣ Smart Waste Management System» 444
Παράρτημα Δ: «Κώδικας της Πύλης Πολιτών του ΠΣ Smart Waste Management System» 474

Κατάλογος Εικόνων / Σχημάτων

Εικόνα 1 3.1 Τοποθεσία του Δήμου Μαλεβιζίου στη Κρήτη 35
Εικόνα 2 4.5.1 Use Case Diagram 51
Εικόνα 3 4.5.2 Sequence Diagram 54
Εικόνα 4 Λειτουργία Docker Container 58
Εικόνα 5 Εγκατάστση Docker Desktop 59
Εικόνα 6 Use Case ΠΧ Smart Waste Management System 62
Εικόνα 7 Διάγραμμα Δραστηριοτήτων με προσομοίωση αισθητήρων μέσω Wokwi 63
Εικόνα 8 Διάγραμμα Δραστηριοτήτων Smart Waste Managenemt System με προσομοίωση iot σε Python 65
Εικόνα 9 Επικοινωνία Wokwi με Dashboard 75
Εικόνα 10 Deployment Wokwi-ngrok-FastAPI 76
Εικόνα 11 Το νέο σχήμα της ΒΔ 92
Εικόνα 12 ΠΣ Πύλης Πολιτών 97
Εικόνα 13 ΠΣ Πιστοποίηση Χρήστη Συστήματος 99
Εικόνα 14 ΠΣ citizen-portal 1 101
Εικόνα 15 ΠΣ citizen-portal 2 101
Εικόνα 16 ΠΣ citizen portal 3 102
Εικόνα 17 ΠΣ citizen portal 4 102
Εικόνα 18 ΠΣ citizen portal 5 103
Εικόνα 19 ΠΣ citizen portal 6 103
Εικόνα 20 ΠΣ Διαχείρισης Smart Waste Management 106
Εικόνα 21 ΠΣ CollectionGarbageTruck για οδηγούς 107

Κατάλογος Πινάκων

Πίνακας 1-1 Στοιχεία φοιτητώ
Πίνακας 1 2.11.2 Πίνακας εταιρειών παροχής εμπορικών λύσεων σε IoT πλατφόρμες...33
Πίνακας 2 4.7 Επιλογή Τεχνολογιών.....55
Table 3 Τεχνικές Προδιαγραφές μηχανήματος Server.....60
Table 4 Αναλυτικές Προδιαγραφές του Docker container τοπικά.....61
Table 5 Σχεδιασμός ΠΣ Smart Waste Management.....68
Table 6 Πίνακες της Βάσης Δεδομένων (ΒΔ).....71
Table 7 ER Σχεσιακό Μοντέλο ΒΔ.....72
Table 8 FastAPI Backend - Ανάπτυξη.....74
Table 9 Ανάπτυξη React Frontend.....74
Table 10 Γεωγραφικές Ζώνες στο ΠΣ.....88
Table 11 Στρώματα Αρχιτεκτονικής του ΠΣ.....88
Table 12 Πίνακες ΒΔ.....91
Table 13 Πιστοποίηση χρηστών ΠΣ.....93
Table 14 Πίνακας εμφάνισης δεδομένων ΠΣ.....94
Table 15 Πίνακας με μεθόδους βέλτιστης διαδρομής.....95
Table 16 Πίνακας κατάστασης οχημάτων του CollectionGarbageTruck.....96
Table 17 Πίνακας αναφορών του ΠΣ Πύλης Πολιτών.....97
Table 18 Πίνακας Εκτέλεσης του ΠΣ.....105
Table 19 Χρήστες του ΠΣ Smart Waste Management System & CollectionGarbageTruck
.....105
Table 20 Λειτουργίες κεντρικού Dashboard του ΠΣ.....106
ν με δικαίωμα εκπόνησης Διπλωματικής Εργασίας.....**Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.**

Συντομογραφίες & Ακρωνύμια

IoT	Internet Of Things – Διαδίκτυο των Πραγμάτων
ΛΣ	Λειτουργικό Σύστημα
ΠΣ	Πληροφοριακό Σύστημα
SQL	Structure Query Language
IT	Τεχνολογία της πληροφορίας
ΧΥΤΑ	Χώρος υγειονομικής ταφής απορριμάτων
Python	Γλώσσα προγραμματισμού
Ts	Typescript
ΕΣΔΑ	Εθνικό Σχέδιο Διαχείρισης Αποβλήτων
ΕΛΣΤΑΤ	Ελληνική Στατιστική Αρχή
ΕΕ/ΕΕΑ	Ευρωπαϊκή Ένωση και του Ευρωπαϊκός Οργανισμός Περιβάλλοντος
JSON	JavaScript Object Notation
ΠΧ	Περίπτωση χρήσης
ΒΔ	Βάση δεδομένων
ΔΑΕ	Δομή ανάλυσης εργασιών
Web app	Εφαρμογή που εκτελείται σε Φυλλομετρητή
GPS	Global Positioning System
GPRS	General Packet Radio Service
ΜΕΑ	Μονάδες Επεξεργασίας Αποβλήτων
RFID	Radio Frequency Identification - Ταυτοποίηση μέσω Ραδιοσυχνοτήτων
PWA	Progressive Web App
ΤΠΕ	Τεχνολογία Πληροφορικής και Επικοινωνιών
GIS	Γεωγραφικά Συστήματα Πληροφοριών
LoRaWAN	πρωτόκολλο MAC για εφαρμογές IoT που βασίζεται στη μέθοδο διαμόρφωσης LoRa (Long Range)
NB-IoT	Narrowband Internet of Things- ασύρματη τεχνολογία δικτύου LPWAN χαμηλής κατανάλωσης

LTE-M	Long-Term Evolution Machine Type Communication - πρότυπο ασύρματης επικοινωνίας τεχνολογίας χαμηλής κατανάλωσης (LPWAN)
MQTT	Message Queuing Telemetry Transport - ελαφρύ πρωτόκολλο επικοινωνίας ανοιχτού κώδικα, σχεδιασμένο ειδικά για το Internet of Things (IoT)
VRP	Vehicle Routing Problem – Πρόβλημα δρομολόγησης οχημάτων
CVRP	Capacitated Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
DVRP	Dynamic Vehicle Routing Problem
GA	Genetic Algorithms
DSS	Decision Support Systems – Συστήματα Υποστήριξης Αποφάσεων
R.O.I.	Return on Investment - Δείκτης Απόδοσης Επένδυσης
KPIs	Key Performance Indicators - Βασικοί Δείκτες Απόδοσης

1. Εισαγωγή

1.1 Το πρόβλημα της διαχείρισης απορριμμάτων στις σύγχρονες πόλεις

Η ραγδαία αύξηση του αστικού ιστού, η αύξηση της κατανάλωσης και η πολυπλοκότητα των συσκευασιών κι άλλων αποβλήτων έχουν οδηγήσει σε σημαντική αύξηση των παραγόμενων απορριμμάτων στις σύγχρονες πόλεις. Η αποτελεσματική διαχείριση των απορριμμάτων αποτελεί κρίσιμο ζήτημα για την προστασία του περιβάλλοντος, τη δημόσια υγεία και την οικονομική βιωσιμότητα των αστικών πόλεων. Για την αντιμετώπιση αυτών των προκλήσεων έχει διαμορφωθεί ένα ευρωπαϊκό ρυθμιστικό πλαίσιο που έχει ως στόχο την πρόληψη, επαναχρησιμοποίηση και ανακύκλωση, καθώς και εθνικά σχέδια δράσης που επιχειρούν την προσαρμογή αυτού του πλαισίου στην πράξη.

1.1.1 Το ευρωπαϊκό πλαίσιο – αρχές και στόχοι

Το ευρωπαϊκό νομικό πλαίσιο για τη διαχείριση αποβλήτων βασίζεται στην αρχή της «ιεράρχησης των αποβλήτων», η οποία περιλαμβάνει τα παρακάτω διαδοχικά στάδια: πρόληψη, επαναχρησιμοποίηση, ανακύκλωση, ανάκτηση και, ως έσχατη επιλογή, διάθεση. Παράλληλα, θεσπίζονται σαφείς και μετρήσιμοι στόχοι ανακύκλωσης, τόσο για τα δημοτικά όσο και για τα συσκευασιακά απόβλητα. Η αναθεωρημένη ευρωπαϊκή νομοθεσία, όπως αποτυπώνεται στην Οδηγία-Πλαίσιο για τα Απόβλητα και στις τροποποιήσεις της (2018/851), προσδιορίζει σταδιακά αυξανόμενους στόχους ανακύκλωσης για τα δημοτικά απορρίμματα: 55% έως το 2025, 60% έως το 2030 και 65% έως το 2035. Ιδιαίτερη έμφαση δίνεται, επίσης, στην πρόληψη παραγωγής αποβλήτων και στην προαγωγή της κυκλικής οικονομίας, ως βασικούς άξονες της ευρωπαϊκής στρατηγικής για τη βιώσιμη διαχείριση των πόρων.

1.1.2 Το ελληνικό πλαίσιο – ΕΣΔΑ 2020-2030 και νομοθετικές πρωτοβουλίες

Η Ελλάδα προχώρησε στην έγκριση του Εθνικού Σχεδίου Διαχείρισης Αποβλήτων (ΕΣΔΑ) για την περίοδο 2020–2030, το οποίο θέτει φιλόδοξους και συγκεκριμένους στόχους, όπως η μείωση του ποσοστού ταφής αποβλήτων σε 10% έως το 2030, η πλήρης εξάλειψη των

παράνομων χώρων ανεξέλεγκτης διάθεσης αποβλήτων, καθώς και η αύξηση των ποσοστών ανακύκλωσης σε 55% έως το 2025 και 60% έως το 2030. Οι εν λόγω στόχοι ευθυγραμμίζονται ή και υπερβαίνουν, κατά περίπτωση, τις δεσμεύσεις που έχουν τεθεί σε ευρωπαϊκό επίπεδο. Παράλληλα, το σχέδιο περιλαμβάνει εξειδικευμένες παρεμβάσεις στην οργάνωση των συστημάτων συλλογής, την ανάπτυξη και αναβάθμιση των υποδομών επεξεργασίας αποβλήτων, καθώς και την υιοθέτηση κινήτρων και οικονομικών εργαλείων για την ενίσχυση της αποτελεσματικότητας των δράσεων διαχείρισης αποβλήτων.

1.1.3 Κατάσταση σήμερα — βασικά στοιχεία και τάσεις (ελληνικός χώρος)

Αναφορικά με τη συνολική παραγωγή αποβλήτων, βάσει των προσωρινών στοιχείων της Ελληνικής Στατιστικής Αρχής (ΕΛΣΤΑΤ) για το έτος 2022, εκτιμάται ότι η συνολική ποσότητα αποβλήτων στην Ελλάδα ανήλθε σε περίπου 29,8 εκατομμύρια τόνους, παρουσιάζοντας αύξηση της τάξεως του 5,2% σε σύγκριση με τα αντίστοιχα δεδομένα του 2020.

Ειδικότερα, όσον αφορά τα δημοτικά απόβλητα και την ανακύκλωση, σύμφωνα με τα στοιχεία της Ευρωπαϊκής Ένωσης και του Ευρωπαϊκού Οργανισμού Περιβάλλοντος (ΕΕ/ΕΕΑ), το πραγματικό ποσοστό ανακύκλωσης των δημοτικών αποβλήτων στην Ελλάδα για το 2022 ανέρχεται σε μόλις 16–17%, ποσοστό σημαντικά χαμηλότερο σε σχέση με τον μέσο όρο της Ευρωπαϊκής Ένωσης, ο οποίος διαμορφώνεται περίπου στο 49–50% για το ίδιο έτος. Επιπλέον, η παραγωγή δημοτικών αποβλήτων ανά κάτοικο στην Ελλάδα ανήλθε σε περίπου 519 κιλά ετησίως για το 2022.

Συνοψίζοντας, τα ανωτέρω δεδομένα καταδεικνύουν την ύπαρξη ενός ουσιώδους χάσματος μεταξύ των θεσμοθετημένων στόχων (σε επίπεδο Ευρωπαϊκής Ένωσης και ΕΣΔΑ) και της πραγματικής επίδοσης που παρατηρείται στην πράξη.

1.1.4 Περιβαλλοντικές επιπτώσεις

Τα βασικότερα περιβαλλοντικά ζητήματα που σχετίζονται με τις υφιστάμενες πρακτικές διαχείρισης απορριμμάτων στην Ελλάδα, καθώς και γενικότερα στον αστικό χώρο, συνοψίζονται ως εξής:

- Εκπομπές αερίων του θερμοκηπίου, κυρίως μεθανίου (CH4), ως αποτέλεσμα της αναερόβιας αποσύνθεσης οργανικών υλικών σε χώρους υγειονομικής ταφής.

- Ρύπανση του εδάφους και των υπόγειων υδάτων εξαιτίας διαρροών και ανεπαρκούς στεγανότητας των χώρων ταφής ή/και παράνομων αποθέσεων αποβλήτων.
- Απώλεια πολύτιμων πρώτων υλών και ενέργειας, η οποία οφείλεται στη μη αποτελεσματική ανακύκλωση και επαναχρησιμοποίηση υλικών.
- Εμφάνιση τοπικών οχλήσεων, όπως δυσοσμίες, εστίες ρύπανσης και προβλήματα δημόσιας υγείας, ιδίως σε περιπτώσεις παράνομων χωματερών.

Οι ανωτέρω περιβαλλοντικές επιπτώσεις εντείνονται όταν σημαντικό ποσοστό των παραγόμενων αποβλήτων οδηγείται προς ταφή χωρίς να προηγηθεί διαδικασία προεπεξεργασίας.

1.1.5 Λειτουργικές και οικονομικές επιπτώσεις

Σε λειτουργικό επίπεδο, οι προκλήσεις που αντιμετωπίζουν οι αστικές περιοχές συνοψίζονται στα εξής:

- Υψηλό λειτουργικό κόστος που σχετίζεται με τη συλλογή και μεταφορά αποβλήτων, το οποίο επιβαρύνεται περαιτέρω από την ανάγκη διαχείρισης πολλαπλών ρευμάτων απορριμμάτων, ιδίως όταν η διαλογή στην πηγή παραμένει σε ανεπαρκή επίπεδα.
- Ανεπάρκεια κατάλληλων υποδομών επεξεργασίας, όπως Μονάδες Επεξεργασίας Αποβλήτων (ΜΕΑ), εγκαταστάσεις κομποστοποίησης και κέντρα διαλογής, γεγονός που περιορίζει τις δυνατότητες ανακύκλωσης και οδηγεί σε αυξημένη διάθεση αποβλήτων σε χώρους υγειονομικής ταφής.
- Ελλιπής εφαρμογή της αρχής «ο ρυπαίνων πληρώνει» και χαμηλού ύψους τέλη ταφής, τα οποία δεν λειτουργούν επαρκώς ως οικονομικά κίνητρα για τη μείωση της ταφής αποβλήτων.
- Λειτουργικές δυσχέρειες που αντιμετωπίζουν οι μικροί ή περιφερειακοί δήμοι, κυρίως λόγω περιορισμένων διαθέσιμων πόρων, έλλειψης τεχνικής επάρκειας και δυσκολιών ως προς τη διασφάλιση της οικονομικής βιωσιμότητας συλλογικών έργων διαχείρισης αποβλήτων.

1.1.6 Υφιστάμενες προκλήσεις στον ελληνικό χώρο

Εν συντομία, οι σημαντικότερες προκλήσεις που επισημαίνονται στη σχετική βιβλιογραφία και στις εκθέσεις συνοψίζονται ως εξής:

- Ισχυρή εξάρτηση από την ταφή, με σημαντικό ποσοστό των δημοτικών αποβλήτων να συνεχίζει να οδηγείται σε χώρους υγειονομικής ταφής.
- Χαμηλά ποσοστά ανακύκλωσης και ανεπαρκής διαλογή στην πηγή, με τα επίπεδα ανακύκλωσης των δημοτικών αποβλήτων να υπολείπονται αισθητά του ευρωπαϊκού μέσου όρου.
- Παρουσία παράνομων αποθέσεων και ανεπαρκώς αποκατεστημένων παλαιών χωματερών, που συνεχίζουν να δημιουργούν προβλήματα διαχείρισης.
- Έλλειψη επαρκών υποδομών και επενδύσεων, με ανάγκη για ανάπτυξη κέντρων διαλογής, μονάδων επεξεργασίας αποβλήτων και υποδομών για βιοαπόβλητα, ως προϋποθέσεις για τη μετάβαση σε κυκλική οικονομία.
- Ανεπαρκή οικονομικά και θεσμικά κίνητρα ή εφαρμογή αυτών, όπως η χαμηλή τιμολόγηση της ταφής, ο περιορισμένος έλεγχος και επιβολή της νομοθεσίας καθώς και η ανάγκη για ολοκληρωμένα χρηματοδοτικά εργαλεία.
- Προβλήματα στο πολιτικό και διοικητικό πλαίσιο, αλλά και στην ποιότητα των δεδομένων, με ασυνέπειες στην εφαρμογή πολιτικών, διαφορετικό ρυθμό προόδου ανά περιφέρεια και ζητήματα αξιοπιστίας και συνέπειας στα στοιχεία καταγραφής.

1.1.7 Σημεία αιχμής για την ελληνική πολιτική και πρακτική

Το ΕΣΔΑ δίνει σαφείς ποσοτικούς στόχους, αλλά η υλοποίηση απαιτεί περαιτέρω επενδύσεις και λειτουργική αναδιοργάνωση των ΟΤΑ. Η μείωση της ταφής στο 10% έως το 2030 θα απαιτήσει συντονισμένες δράσεις: αύξηση χωρητικότητας επεξεργασίας, χωροθέτηση κατάλληλων εγκαταστάσεων, εφαρμογή συστημάτων πληρωμής ανά όγκο/βάρος (PAYT) και ενίσχυση ανακύκλωσης οργανικών.

1.1.8 Συμπεράσματα για τη διαχείριση απορριμμάτων στην Ελλάδα

Η διαχείριση απορριμμάτων στην Ελλάδα παρουσιάζει αξιοσημείωτη απόκλιση μεταξύ των θεσμοθετημένων στόχων και της εφαρμοζόμενης πρακτικής. Τα ποσοστά ανακύκλωσης παραμένουν σε ιδιαίτερα χαμηλά επίπεδα, ενώ η εξάρτηση από την ταφή εξακολουθεί να

είναι έντονη. Ταυτόχρονα, υφίστανται δομικές και λειτουργικές ανεπάρκειες τόσο στους δήμους όσο και στις σχετικές υποδομές. Τα παραπάνω αποτελούν σημείο αφετηρίας για πληθώρα ερευνητικών κατευθύνσεων, όπως η αξιολόγηση των εφαρμοζόμενων πολιτικών, η ανάλυση κόστους-οφέλους επενδυτικών παρεμβάσεων σε υποδομές (ενδεικτικά ΜΕΑ ή διαχείριση βιοαποβλήτων), η διερεύνηση της συμπεριφοράς των πολιτών ως προς τη διαλογή στην πηγή, καθώς και η πιλοτική εφαρμογή συστημάτων πληρωμής βάσει παραγόμενου όγκου ή βάρους και κινήτρων για ενίσχυση της ανακύκλωσης.

1.2 Κίνητρα και Αναγκαιότητα της Έρευνας: Σημασία της Έξυπνης Διαχείρισης Απορριμμάτων στο Πλαίσιο των Smart Cities στην Ελλάδα

1.2.1 Κίνητρα για την έρευνα

Αύξηση αστικού πληθυσμού και επιπτώσεις στις υποδομές διαχείρισης απορριμμάτων

Η διαρκής αύξηση του αστικού πληθυσμού επιφέρει σημαντικές πιέσεις στις υφιστάμενες υποδομές διαχείρισης απορριμμάτων. Τα παραδοσιακά συστήματα συλλογής, τα οποία βασίζονται στη χειροκίνητη εποπτεία των κάδων και σε προκαθορισμένα δρομολόγια αποκομιδής, παρουσιάζουν σημαντικές αδυναμίες ως προς την αποτελεσματική και αποδοτική διαχείριση. Η ενσωμάτωση “έξυπνων” τεχνολογιών, όπως το Διαδίκτυο των Πραγμάτων (IoT) και προηγμένοι αισθητήρες, δύναται να προσδώσει δυναμικότητα, ευελιξία και υψηλότερα επίπεδα αποδοτικότητας στη διαχείριση των αστικών αποβλήτων.

Οικονομική αποδοτικότητα και ορθολογική διαχείριση πόρων

Η αξιοποίηση συστημάτων έξυπνων κάδων, εξοπλισμένων με αισθητήρες μέτρησης πληρότητας, προσφέρει τη δυνατότητα βελτιστοποίησης των δρομολογίων αποκομιδής. Ως αποτέλεσμα, επιτυγχάνεται μείωση της σπατάλης καυσίμων, του λειτουργικού κόστους του ανθρώπινου δυναμικού, καθώς και της φθοράς των οχημάτων συλλογής. Η εξοικονόμηση πόρων καθίσταται ιδιαίτερα σημαντική, δεδομένων των περιορισμένων δημοτικών προϋπολογισμών, και συμβάλλει στη βιωσιμότητα των υπηρεσιών διαχείρισης απορριμμάτων.

Περιβαλλοντικά οφέλη και συμβολή στην κυκλική οικονομία

Η εφαρμογή έξυπνων συστημάτων διαχείρισης απορριμμάτων συμβάλλει ουσιαστικά στη μείωση των αρνητικών περιβαλλοντικών επιπτώσεων. Ειδικότερα, η αποφυγή υπερχειλισμένων κάδων περιορίζει τη ρύπανση και τις οσμές, ενώ η μείωση των εκπομπών ρύπων από τα απορριμματοφόρα οχήματα υποστηρίζει τη βελτίωση της ποιότητας του αστικού περιβάλλοντος. Παράλληλα, διευκολύνεται ο καλύτερος διαχωρισμός των ανακυκλώσιμων υλικών και ενισχύεται η ανακύκλωση, στοιχείο κρίσιμο για την επίτευξη του οράματος της "κυκλικής πόλης".

Ενίσχυση συμμετοχής των πολιτών και διαμόρφωση κουλτούρας ανακύκλωσης

Η χρήση έξυπνων πλατφορμών, όπως οι κάδοι ανακύκλωσης με τεχνολογία RFID, προσφέρει τη δυνατότητα άμεσης εμπλοκής των πολιτών στη διαδικασία διαχείρισης απορριμμάτων. Μέσω της επιβράβευσης ανάλογα με την ποσότητα και ποιότητα των ανακυκλώσιμων που απορρίπτονται, της παροχής πληροφόρησης σε πραγματικό χρόνο και της χρήσης διαδραστικών εργαλείων ενημέρωσης, ενισχύεται η περιβαλλοντική συνείδηση και η ενεργή συμμετοχή των πολιτών στην ανακύκλωση.

Ευθυγράμμιση με εθνικές και ευρωπαϊκές στρατηγικές

Η Ελλάδα έχει θέσει φιλόδοξους στόχους για την προώθηση της κυκλικής οικονομίας και την υιοθέτηση "έξυπνων" λύσεων στο πλαίσιο των Smart Cities. Η παρούσα έρευνα δύναται να υποστηρίξει τις προσπάθειες των φορέων τοπικής αυτοδιοίκησης και της κεντρικής κυβέρνησης, προσφέροντας πρακτικά μοντέλα εφαρμογής, ανάλυση κόστους-οφέλους και τεκμηριωμένες προτάσεις πολιτικής που θα συμβάλουν στην επίτευξη των στρατηγικών αυτών.

Ανάγκη για περαιτέρω επιστημονική διερεύνηση και καινοτομία

Παρότι σε διεθνές επίπεδο έχουν καταγραφεί πολυάριθμες μελέτες σχετικά με τα έξυπνα συστήματα διαχείρισης απορριμμάτων, στην ελληνική πραγματικότητα παρατηρείται έλλειψη τεκμηριωμένων παραδειγμάτων, αξιολόγησης επιδόσεων και ανάλυσης των ειδικών περιορισμών που ενυπάρχουν (πολιτικών, οικονομικών, τεχνικών). Συνεπώς, κρίνεται επιτακτική η ανάγκη να διερευνηθεί, μέσω της παρούσας εργασίας, ο τρόπος

προσαρμογής και ενσωμάτωσης των τεχνολογιών αυτών στις ιδιαιτερότητες του ελληνικού αστικού τοπίου, της πολιτικής και της λειτουργικής δομής.

1.2.2 Σημασία της Έξυπνης Διαχείρισης Απορριμμάτων στα Smart Cities στην Ελλάδα

1.2.2.1 Πρακτικά παραδείγματα στην Ελλάδα

Στο πλαίσιο της υιοθέτησης καινοτόμων λύσεων διαχείρισης απορριμμάτων στις ελληνικές "έξυπνες πόλεις", αξίζει να αναφερθούν ορισμένες χαρακτηριστικές εφαρμογές που υλοποιούνται από εταιρείες και φορείς του χώρου. Ενδεικτικά, η εταιρεία Intracom Telecom έχει αναπτύξει την πλατφόρμα Smart Waste, η οποία αξιοποιεί αισθητήρες Διαδικτύου των Πραγμάτων (IoT) εντός των κάδων για την καταγραφή της πληρότητας, επιτρέποντας την αυτοματοποίηση ειδοποιήσεων και τη βελτιστοποίηση των δρομολογίων αποκομιδής. Επιπλέον, η συγκεκριμένη πλατφόρμα υποστηρίζει τη διαχείριση της πρόσβασης στους κάδους, καθώς και την επιβράβευση των πολιτών μέσω χρήσης καρτών RFID.

Παράλληλα, ο Όμιλος Τεχνολογιών Συστήματος (OTS) προσφέρει την εφαρμογή Open1|WasteManagement, η οποία παρέχει στους αρμόδιους φορείς τη δυνατότητα παρακολούθησης της πληρότητας των κάδων σε πραγματικό χρόνο, την έγκαιρη λήψη προειδοποιήσεων και τη δυναμική αναπροσαρμογή των δρομολογίων αποκομιδής.

Αξιοσημείωτη είναι και η δραστηριοποίηση της εταιρείας EPTA Environmental Engineers στον τομέα των "έξυπνων πόλεων", με σχεδιασμό και υλοποίηση συστημάτων pay-as-you-throw (πληρωμή βάσει όγκου ή βάρους απορριμμάτων), έξυπνων κάδων με τεχνολογίες RFID ή QR code και ψηφιακών πλατφορμών για την επιβράβευση της ανακύκλωσης.

Επιπροσθέτως, η εταιρεία Brainbox Technology έχει προχωρήσει στην υλοποίηση λύσεων που αφορούν την παρακολούθηση της καθαριότητας των κάδων και την οργάνωση των δρομολογίων συλλογής, αξιοποιώντας αισθητήρες πληρότητας σε πραγματικό χρόνο.

Τέλος, αξίζει να σημειωθεί ότι η Περιφέρεια Πελοποννήσου τιμήθηκε με διάκριση στα Smart Cities Awards για το έργο διαχείρισης απορριμμάτων που υλοποίησε. Το γεγονός αυτό αναδεικνύει την "έξυπνη διαχείριση απορριμμάτων" ως θεμελιώδες στοιχείο της βιώσιμης ανάπτυξης που αναγνωρίζεται και από τους περιφερειακούς φορείς της χώρας.

1.2.2.2 Οικονομική και Περιβαλλοντική Βιωσιμότητα

Η ενσωμάτωση έξυπνων συστημάτων διαχείρισης απορριμμάτων δύναται να προσφέρει ουσιώδη οφέλη τόσο σε οικονομικό όσο και σε περιβαλλοντικό επίπεδο για τους δήμους. Συγκεκριμένα, η εφαρμογή τέτοιων τεχνολογιών συμβάλλει στη μείωση των λειτουργικών δαπανών, καθώς επιτρέπει τη βελτιστοποίηση των δρομολογίων συλλογής απορριμμάτων, περιορίζοντας τις περιττές μετακινήσεις των οχημάτων και, κατ' επέκταση, τις ανάγκες σε ανθρώπινο δυναμικό και καύσιμα. Παράλληλα, η μείωση της κατανάλωσης πόρων οδηγεί σε σημαντική ελάττωση του περιβαλλοντικού αποτυπώματος των αστικών περιοχών. Τα παραπάνω οφέλη ενισχύουν τη μακροπρόθεσμη βιωσιμότητα των ελληνικών πόλεων και συνεισφέρουν ουσιαστικά στην επίτευξη των εθνικών και ευρωπαϊκών στόχων περιβαλλοντικής πολιτικής.

1.2.2.3 Κοινωνική Αποδοχή και Εμπλοκή των Πολιτών

Η επιτυχία των συστημάτων έξυπνης διαχείρισης απορριμμάτων εξαρτάται σε σημαντικό βαθμό από την κοινωνική αποδοχή και τη δραστήρια συμμετοχή των πολιτών. Η ενσωμάτωση μηχανισμών που προάγουν τη συμμετοχή και επιβραβεύουν υπεύθυνες περιβαλλοντικές πρακτικές δύναται να ενισχύσει ουσιαστικά την περιβαλλοντική συνείδηση και συμπεριφορά του πληθυσμού. Ο κοινωνικός αυτός παράγοντας αναγνωρίζεται ως καταλυτικός για την επιτυχία κάθε πρωτοβουλίας που εντάσσεται στο πλαίσιο των "έξυπνων πόλεων". Η ενεργή εμπλοκή των κατοίκων συμβάλλει καθοριστικά στην επίτευξη ρεαλιστικών και μετρήσιμων στόχων, όπως η αύξηση των ποσοστών ανακύκλωσης και η αποτελεσματική μείωση της παραγωγής αποβλήτων.

1.2.2.4 Στρατηγική ευθυγράμμιση με τις «Έξυπνες Πόλεις» και τις περιβαλλοντικές πολιτικές

Η Ελλάδα, στο πλαίσιο του εκσυγχρονισμού της δημόσιας διοίκησης και της ενίσχυσης των "έξυπνων πόλεων", προωθεί συστηματικά την υιοθέτηση καινοτόμων πρακτικών που εναρμονίζονται με τις αρχές της κυκλικής οικονομίας. Ειδικότερα, η ορθολογική διαχείριση απορριμμάτων αναδεικνύεται ως βασικό συστατικό στοιχείο της εθνικής στρατηγικής, συμβάλλοντας ουσιαστικά στην επίτευξη των στόχων αειφορίας και περιβαλλοντικής προστασίας. Στο πλαίσιο αυτό, η παρούσα ερευνητική εργασία ενισχύει και τεκμηριώνει τις υφιστάμενες πολιτικές κατευθύνσεις, ενώ δύναται να προσφέρει χρήσιμες προτάσεις και

τεκμηριωμένα συμπεράσματα που μπορούν να αξιοποιηθούν στη διαδικασία λήψης αποφάσεων τόσο σε τοπικό όσο και σε περιφερειακό επίπεδο.

1.2.2.5 Έρευνα και Καινοτομία

Από ακαδημαϊκή και ερευνητική σκοπιά, η μελέτη της έξυπνης διαχείρισης απορριμμάτων συνιστά ένα δυναμικό πεδίο ανάπτυξης καινοτόμων τεχνολογιών και μεθοδολογιών. Ειδικότερα, η αξιοποίηση προηγμένων συστημάτων Διαδικτύου των Πραγμάτων (IoT), η συστηματική ανάλυση δεδομένων (“smart data”), καθώς και η εφαρμογή τεχνικών μηχανικής μάθησης για την αυτόματη ταξινόμηση απορριμμάτων, διαμορφώνουν το επιστημονικό υπόβαθρο για τη δημιουργία αποτελεσματικότερων πρακτικών στον τομέα αυτό. Παράλληλα, η ανάπτυξη και η χρήση περιβαλλοντικών-οικονομικών μοντέλων, όπως η ανάλυση κόστους-οφέλους, συμβάλλει ουσιαστικά στην τεκμηριωμένη αξιολόγηση της αποδοτικότητας και της βιωσιμότητας των προτεινόμενων λύσεων.

Η παρούσα έρευνα δύναται να συνεισφέρει σημαντικά στον σχεδιασμό και την επιστημονική αξιολόγηση τέτοιων καινοτόμων συστημάτων, με ιδιαίτερη έμφαση στην προσαρμογή τους στις ιδιαιτερότητες του ελληνικού πλαισίου. Η εστίαση σε τοπικές ανάγκες και προκλήσεις αποτελεί κρίσιμη παράμετρο για την επιτυχία της εφαρμογής των έξυπνων τεχνολογιών διαχείρισης απορριμμάτων, ενισχύοντας παράλληλα την προστιθέμενη αξία των ερευνητικών αποτελεσμάτων σε εθνικό και περιφερειακό επίπεδο.

1.2.2.6 Συμπέρασμα

Η διεξοδική ανάλυση της έξυπνης διαχείρισης απορριμμάτων στο πλαίσιο των ελληνικών “Έξυπνων Πόλεων” καθίσταται όχι μόνο επίκαιρη, αλλά και ζωτικής σημασίας για τη βιώσιμη αστική ανάπτυξη. Ο συγκεκριμένος τομέας εστιάζει στη σύνθεση τεχνολογικής καινοτομίας, περιβαλλοντικής βιωσιμότητας και ενεργού κοινωνικής συμμετοχής, προσφέροντας στοχευμένες και εφαρμόσιμες λύσεις στα σύνθετα προβλήματα που αντιμετωπίζουν οι δήμοι, οι περιφέρειες και οι πολίτες. Η παρούσα ερευνητική εργασία δύναται να συμβάλει ουσιαστικά στην τεκμηρίωση της αποτελεσματικότητας των έξυπνων συστημάτων διαχείρισης απορριμμάτων στην Ελλάδα, ενώ παράλληλα μπορεί να διατυπώσει προτάσεις για περαιτέρω βελτιστοποίηση και να προτείνει μοντέλα ευρύτερης εφαρμογής τους σε εθνικό ή/και διεθνές επίπεδο.

1.3 Σκοπός και στόχοι της εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η διερεύνηση και ανάλυση της έξυπνης διαχείρισης συλλογής απορριμμάτων, ενός πεδίου που αποκτά ολοένα και μεγαλύτερη σημασία για την αντιμετώπιση των σύγχρονων προκλήσεων σε αστικά περιβάλλοντα, ιδιαίτερα στις αναπτυσσόμενες πόλεις. Η έξυπνη διαχείριση αποσκοπεί στη βελτιστοποίηση της αποδοτικότητας και της βιωσιμότητας των συστημάτων συλλογής απορριμμάτων, μέσω της ολοκληρωμένης αξιοποίησης τεχνολογιών συλλογής δεδομένων (όπως το Διαδίκτυο των Πραγμάτων – IoT), προηγμένων αναλυτικών μεθόδων και εργαλείων υποστήριξης αποφάσεων, όπως οπτικοί πίνακες ελέγχου.

Ειδικότερα, η εργασία επιδιώκει:

- Να αναλύσει την υφιστάμενη κατάσταση της διαχείρισης απορριμμάτων στην Ελλάδα και να αξιολογήσει τον βαθμό ετοιμότητας των ελληνικών δήμων για την υιοθέτηση έξυπνων τεχνολογιών.
- Να εξετάσει τις τεχνολογικές λύσεις έξυπνης διαχείρισης απορριμμάτων (IoT αισθητήρες, πλατφόρμες παρακολούθησης, συστήματα pay-as-you-throw) και να μελετήσει την προσαρμογή τους στις ανάγκες των ελληνικών αστικών περιοχών.
- Να εντοπίσει τα εμπόδια, τις προκλήσεις και τους επιταχυντές υιοθέτησης (θεσμικούς, χρηματοδοτικούς, τεχνολογικούς, κοινωνικούς) των έξυπνων λύσεων σε ελληνικούς δήμους.
- Να εκτιμήσει τα περιβαλλοντικά, λειτουργικά και οικονομικά οφέλη που δύνανται να προκύψουν από την εφαρμογή έξυπνων τεχνολογιών διαχείρισης απορριμμάτων.
- Να προτείνει ένα πρακτικό πλαίσιο στρατηγικής εφαρμογής ή μοντέλο προσέγγισης για τους ελληνικούς δήμους, βασισμένο στα ερευνητικά ευρήματα.

Στο πλαίσιο της εργασίας προβλέπεται η ανάπτυξη σχετικών εφαρμογών εφ’ εξής θα το ονομάσουμε Πληροφοριακό Σύστημα (ΠΣ), όπως πίνακας ελέγχου ή/και ΠΣ για τις δημοτικές αρχές και τους υπαλλήλους καθαριότητας, με στόχο τη διευκόλυνση της λήψης αποφάσεων και την αποτελεσματική διαχείριση των απορριμμάτων. Ιδιαίτερη έμφαση δίνεται στην πρακτική εφαρμογή των προτεινόμενων λύσεων, με σκοπό να καταστεί δυνατή η πιλοτική υλοποίησή τους σε τοπικούς δήμους.

Οι στόχοι της εργασίας διαρθρώνονται σε θεωρητικό, τεχνολογικό και εφαρμοσμένο επίπεδο, και αποσκοπούν στην παραγωγή τεκμηριωμένων, εφαρμόσιμων καινοτόμων

λύσεων που θα συμβάλουν ουσιαστικά στη βιώσιμη αστική ανάπτυξη και στην ενίσχυση της αποτελεσματικότητας της διαχείρισης απορριμμάτων στην Ελλάδα.

1.4 Ερευνητικά Ερωτήματα

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, τα ερευνητικά ερωτήματα διαμορφώνονται με στόχο την καθοδήγηση της υλοποίησης της εφαρμογής (ΠΣ) που θα σχεδιαστεί. Τα ερωτήματα αυτά αποτυπώνουν τη θεωρητική και πρακτική προσέγγιση της έρευνας, εστιάζοντας στη διερεύνηση της συμβολής των έξυπνων τεχνολογιών στη βελτίωση της διαχείρισης απορριμμάτων στους ελληνικούς δήμους, σύμφωνα με τις αρχές των έξυπνων πόλεων (Smart Cities).

1.4.1 Κύριο Ερευνητικό Ερώτημα (Primary Research Question):

Πώς μπορούν οι έξυπνες τεχνολογίες διαχείρισης απορριμμάτων να συμβάλουν στη βελτίωση της αποτελεσματικότητας, της βιωσιμότητας και της οικονομικής λειτουργίας των ελληνικών πόλεων στο πλαίσιο των Smart Cities;

1.4.2 Δευτερεύοντα Ερευνητικά Υποερωτήματα:

1. Ποιο είναι το υφιστάμενο επίπεδο οργάνωσης, ψηφιακής ωριμότητας και αποτελεσματικότητας στη διαχείριση απορριμμάτων σε επιλεγμένους ελληνικούς δήμους;
2. Ποιες έξυπνες τεχνολογίες (όπως IoT αισθητήρες, πλατφόρμες παρακολούθησης σε πραγματικό χρόνο, συστήματα τιμολόγησης ανά παραγόμενα απόβλητα – PAYT) θεωρούνται καταλληλότερες για εφαρμογή στο ελληνικό αστικό περιβάλλον και υπό ποιες προϋποθέσεις;
3. Ποιοι παράγοντες λειτουργούν ως εμπόδια ή ως καταλύτες για την υιοθέτηση συστημάτων έξυπνης διαχείρισης απορριμμάτων στην Ελλάδα (ενδεικτικά: οικονομικοί, θεσμικοί, κοινωνικοί, τεχνικοί, πολιτισμικοί);
4. Ποια περιβαλλοντικά, κοινωνικά και οικονομικά οφέλη τεκμηριώνονται από τη διεθνή βιβλιογραφία και ποια από αυτά δύνανται να εφαρμοστούν ρεαλιστικά στον ελληνικό χώρο;
5. Με ποια μεθοδολογική προσέγγιση μπορεί να σχεδιαστεί ένα προσαρμοσμένο μοντέλο εφαρμογής (implementation roadmap) για την υιοθέτηση έξυπνης διαχείρισης απορριμμάτων σε ελληνικούς δήμους;

Η διατύπωση των παραπάνω ερευνητικών ερωτημάτων αποτελεί τον θεμέλιο λίθο για τη μεθοδολογική προσέγγιση της εργασίας, διασφαλίζοντας την παραγωγή επιστημονικά τεκμηριωμένων και πρακτικά εφαρμόσιμων λύσεων, που θα συμβάλουν ουσιαστικά στη βελτίωση των συστημάτων διαχείρισης απορριμμάτων στην Ελλάδα.

1.4.3 Διατύπωση Υποθέσεων Έρευνας

Λαμβάνοντας υπόψη τη βιβλιογραφική θεμελίωση, το ερευνητικό πρόβλημα, τη σημασία της παρούσας έρευνας, καθώς και τους στόχους και τα ερευνητικά ερωτήματα που έχουν τεθεί, διατυπώνονται οι ακόλουθες ερευνητικές υποθέσεις, οι οποίες θα αποτελέσουν το πλαίσιο ελέγχου και ανάλυσης των εμπειρικών δεδομένων:

Κεντρική Ερευνητική Υπόθεση

Η εφαρμογή συστημάτων έξυπνης διαχείρισης απορριμμάτων δεν επιφέρει στατιστικώς σημαντικές διαφοροποιήσεις στο επίπεδο αποτελεσματικότητας της συλλογής, στο λειτουργικό κόστος και στην περιβαλλοντική επίδοση των δήμων, συγκριτικά με τα παραδοσιακά συστήματα διαχείρισης απορριμμάτων.

Εναλλακτική Ερευνητική Υπόθεση

Η υιοθέτηση και εφαρμογή συστημάτων έξυπνης διαχείρισης απορριμμάτων συμβάλλει ουσιωδώς στη βελτίωση της αποτελεσματικότητας της συλλογής, στη μείωση του λειτουργικού κόστους και στην ενίσχυση της περιβαλλοντικής επίδοσης των ελληνικών δήμων.

Ειδικές Ερευνητικές Υποθέσεις

- Η ενσωμάτωση τεχνολογιών αισθητήρων και συστημάτων Διαδικτύου των Πραγμάτων (IoT) στους κάδους απορριμμάτων οδηγεί σε μείωση των περιττών δρομολογίων συλλογής, καθώς και του συνολικού κόστους καυσίμων των δημοτικών υπηρεσιών καθαριότητας.
- Η παροχή πληροφόρησης και κινήτρων προς τους πολίτες μέσω ψηφιακών εφαρμογών συμβάλλει στην αύξηση των ποσοστών ανακύκλωσης και ενισχύει τη συμμετοχή στη διαλογή στην πηγή.
- Η αξιοποίηση συστημάτων προγνωστικής ανάλυσης δεδομένων στη διαχείριση απορριμμάτων από τους δήμους βελτιώνει τον σχεδιασμό των διαθέσιμων πόρων και μειώνει τον χρόνο απόκρισης σε περιστατικά υπερπλήρωσης κάδων.

- Η ενσωμάτωση έξυπνων λύσεων στη διαχείριση απορριμμάτων ενδυναμώνει τη θετική στάση των πολιτών απέναντι σε καινοτόμες δημοτικές υπηρεσίες και προωθεί την περιβαλλοντική ευαισθητοποίηση και κουλτούρα.
- Οι ελληνικοί δήμοι αντιμετωπίζουν σημαντικά εμπόδια στην υλοποίηση έξυπνων συστημάτων διαχείρισης απορριμμάτων, τα οποία σχετίζονται με οικονομικούς, οργανωτικούς και τεχνικούς παράγοντες, σε μεγαλύτερο βαθμό σε σύγκριση με άλλες ευρωπαϊκές πόλεις.

Αρνητικές Υποθέσεις (προαιρετικά για στατιστική ανάλυση)

Δεν παρατηρούνται στατιστικά σημαντικές διαφορές μεταξύ των παραδοσιακών και των έξυπνων συστημάτων διαχείρισης απορριμμάτων στους παραπάνω άξονες αξιολόγησης.
Οι ανωτέρω υποθέσεις δύνανται να διερευνηθούν μέσω ποσοτικών μεθοδολογιών (όπως ανάλυση κόστους, αξιολόγηση δρομολογίων συλλογής και καταγραφή ποσοστών ανακύκλωσης), ποιοτικών μεθόδων (ενδεικτικά: συνεντεύξεις με στελέχη της τοπικής αυτοδιοίκησης, ερωτηματολόγια πολιτών) ή με χρήση μικτής μεθοδολογικής προσέγγισης, διασφαλίζοντας την επιστημονική εγκυρότητα και την πρακτική εφαρμοσιμότητα των ερευνητικών αποτελεσμάτων.

1.5 Μεθοδολογία προσέγγισης

Στόχος της μεθοδολογίας είναι να αξιολογήσει το ισχύον νομοθετικό και λειτουργικό πλαίσιο της συλλογής/αποκομιδής απορριμμάτων στον Δήμο Γαζίου ως Case Study, να εντοπίσει κενά, εμπόδια και ευκαιρίες για έξυπνες λύσεις, και να παράσχει συστηματική τεκμηρίωση για προτάσεις πολιτικής και εφαρμογής.
Ένα πληροφοριακό σύστημα (ΠΣ) το οποίο θα παρέχει πληροφορίες σχετικές με την πληρότητα των κάδων απορριμμάτων της πόλης ανά τακτά χρονικά διαστήματα για κάθε ημέρα, την βέλτιστη διαδρομή κάθε απορριμματοφόρου, το πλήθος των οχημάτων και των εργαζομένων που απαιτούνται σε ημερήσια βάση, την κατάσταση των κάδων μετά την διαδικασία της αποκομιδής, το στίγμα του κάδου και του απορριμματοφόρου, στατιστικά στοιχεία και αναφορές που είναι χρήσιμες για την αποτελεσματικότερη και βέλτιστη λειτουργία της υπηρεσίας αποκομιδής απορριμμάτων, θα αποτελέσει απαραίτητο εργαλείο στον τομέα καθαριότητας κάθε δήμου και συγκεκριμένα του δήμου Γαζίου.
Η παρούσα διπλωματική εργασία έχει σκοπό την ανάπτυξη ενός πληροφοριακού συστήματος (ΠΣ) το οποίο θα αποτελείται από τέσσερα επιμέρους συστήματα (συσκευή

προσομοίωσης «Έξυπνου» Κάδου, εφαρμογή Web App «Σύστημα Διαχείρισης Απορριμμάτων» και εφαρμογή PWA (Progressive Web App) ΈξυπνοΑπορριματοφόρο – για τους οδηγούς των οχημάτων και τέλος την πύλη πολιτών για ενημέρωση του δήμου απευθείας από τους πολίτες για τυχόν προβλήματα που αφορούν την αποκομιδή των απορριμμάτων της περιοχής. Το ΠΣ θα αξιοποιεί τις τεχνολογίες Internet of Things (IoT) και θα παρέχει πληροφορίες για την πληρότητα των κάδων, την διαχείριση διαδρομών του στόλου απορριμματοφόρων, την απεικόνιση των απορριμματοφόρων και των κάδων απορριμμάτων σε χάρτη, την μείωση της κυκλοφορίας, την αύξηση της καθαριότητας, την μείωση του κόστους αποκομιδής, τον έλεγχο της ποσότητας των απορριμμάτων που συλλέγονται με την ποσότητα που παραδίδεται στους ΧΥΤΑ και έμμεσα την καλύτερη αξιοποίηση του ανθρώπινου δυναμικού.

2. Έξυπνες πόλεις, χαρακτηριστικά και τεχνολογίες (Smart Cities)

2.1. Έννοια και χαρακτηριστικά των Έξυπνων Πόλεων (Smart Cities)

Ο όρος «Έξυπνη Πόλη» (Smart City) περιγράφει ένα σύγχρονο αστικό περιβάλλον, το οποίο αξιοποιεί προηγμένες τεχνολογίες πληροφορικής και επικοινωνιών (ΤΠΕ) με σκοπό την αναβάθμιση της ποιότητας ζωής των πολιτών, την αποδοτικότερη παροχή δημοτικών υπηρεσιών και τη βιώσιμη διαχείριση των διαθέσιμων πόρων. Στο πλαίσιο των έξυπνων πόλεων, ιδιαίτερη έμφαση δίνεται στη συστηματική συλλογή, επεξεργασία και αξιοποίηση δεδομένων που προέρχονται από διάφορους τομείς της αστικής ζωής. Η διαχείριση αυτών των δεδομένων επιτρέπει τη λήψη τεκμηριωμένων αποφάσεων από τους αρμόδιους φορείς και συμβάλλει στην αυτοματοποίηση κρίσιμων λειτουργιών, ενισχύοντας τη λειτουργικότητα και τη βιωσιμότητα της πόλης.

Σύμφωνα με το ευρωπαϊκό μοντέλο έξυπνης πόλης (European Smart City Model), οι έξυπνες πόλεις χαρακτηρίζονται από ένα σύνολο διακριτών αξόνων που διαμορφώνουν τη λειτουργία και την ανάπτυξή τους. Συγκεκριμένα, τα βασικά χαρακτηριστικά των έξυπνων πόλεων περιλαμβάνουν την έξυπνη διακυβέρνηση, η οποία προάγει τη διαφάνεια, τη συμμετοχή των πολιτών και την παροχή ψηφιακών δημόσιων υπηρεσιών· την έξυπνη οικονομία, με έμφαση στην καινοτομία, την επιχειρηματικότητα και τον ψηφιακό μετασχηματισμό· την έξυπνη κινητικότητα, που εστιάζει στη βιώσιμη μεταφορά, τη βελτιστοποιημένη διαχείριση της κυκλοφορίας και τις έξυπνες υποδομές· το έξυπνο περιβάλλον, με στόχο την ενεργειακή αποδοτικότητα, την προστασία των φυσικών πόρων και την εφαρμογή έξυπνων μεθόδων διαχείρισης αποβλήτων· τους έξυπνους πολίτες, οι οποίοι διαθέτουν ψηφιακές δεξιότητες, συμμετέχουν ενεργά και προάγουν την κοινωνική καινοτομία· και τέλος, την έξυπνη διαβίωση, που αφορά τη βελτίωση της ποιότητας ζωής μέσω της παροχής προηγμένων υπηρεσιών υγείας, ασφάλειας και αναψυχής.

Ιδιαίτερη σημασία αποδίδεται στον άξονα του έξυπνου περιβάλλοντος, όπου η διαχείριση των απορριμμάτων εντάσσεται ως ένα κρίσιμο πεδίο για την προώθηση της βιωσιμότητας και της κυκλικής οικονομίας στις σύγχρονες αστικές περιοχές.

2.2. Σύγχρονες Τεχνολογίες στις Έξυπνες Πόλεις

Οι έξυπνες πόλεις αξιοποιούν ένα ευρύ φάσμα τεχνολογιών, οι οποίες λειτουργούν συντονισμένα με σκοπό την παρακολούθηση, ανάλυση και βελτιστοποίηση των αστικών λειτουργιών. Οι κυριότερες από αυτές τις τεχνολογίες παρουσιάζονται παρακάτω:

2.2.1. Internet of Things (IoT)

Το Internet of Things (IoT) παρέχει τη δυνατότητα διασύνδεσης συσκευών και αισθητήρων με δίκτυα δεδομένων, προσφέροντας πληροφόρηση σε πραγματικό χρόνο. Στο πλαίσιο της διαχείρισης απορριμμάτων, αξιοποιούνται αισθητήρες για τη μέτρηση της στάθμης πληρότητας των κάδων, της θερμοκρασίας (με στόχο την πρόληψη πυρκαγιών), της κίνησης και του ανοίγματος του καπακιού, καθώς και της θέσης μέσω GPS.

2.2.2. Big Data & Predictive Analytics

Η συλλογή μεγάλου όγκου δεδομένων από αισθητήρες επιτρέπει στους δήμους να αναλύουν τα μοτίβα παραγωγής απορριμμάτων, να προβλέπουν περιόδους αιχμής και να σχεδιάζουν βέλτιστα δρομολόγια συλλογής, ενισχύοντας έτσι την αποδοτικότητα των διαδικασιών.

2.2.3. Cloud Computing

Η χρήση τεχνολογιών cloud για την αποθήκευση και επεξεργασία δεδομένων διευκολύνει την πρόσβαση στις πληροφορίες, επιτρέπει την κεντρική διαχείριση, μειώνει το κόστος των υποδομών και προάγει τη διαλειτουργικότητα μεταξύ διαφορετικών υπηρεσιών του δήμου.

2.2.4. GIS – Γεωγραφικά Συστήματα Πληροφοριών

Τα Γεωγραφικά Συστήματα Πληροφοριών (GIS) χρησιμοποιούνται για τη χαρτογράφηση των κάδων απορριμμάτων, τον σχεδιασμό δρομολογίων συλλογής, την ανάλυση της χωρικής κατανομής των απορριμμάτων και την επιλογή κατάλληλων σημείων για τη χωροθέτηση νέων κάδων.

2.2.5. Mobile Applications

Οι εφαρμογές για κινητές συσκευές παρέχουν στους πολίτες τη δυνατότητα να αναφέρουν προβλήματα σχετικά με τη διαχείριση απορριμμάτων, να ενημερώνονται για τις συχνότητες συλλογής, να λαμβάνουν κίνητρα όπως πόντους ανακύκλωσης και να συμμετέχουν ενεργά σε δράσεις που προάγουν την κυκλική οικονομία.

2.3. Έξυπνη Διαχείριση Απορριμμάτων (Smart Waste Management)

Η έξυπνη διαχείριση απορριμμάτων συνιστά έναν εξειδικευμένο υποτομέα των «Έξυπνων Πόλεων», ο οποίος αποσκοπεί στη βελτιστοποίηση του συνόλου των διαδικασιών συλλογής, μεταφοράς, ανακύκλωσης και τελικής διάθεσης των στερεών αποβλήτων. Η ενσωμάτωση καινοτόμων τεχνολογικών λύσεων συμβάλλει στην αναβάθμιση των υπηρεσιών καθαριότητας, ενισχύοντας την αποδοτικότητα και τη βιωσιμότητα των αστικών περιβαλλόντων.

2.3.1. Ορισμός

Η «Έξυπνη Διαχείριση Απορριμμάτων» ορίζεται ως η συστηματική αξιοποίηση τεχνολογιών του Διαδικτύου των Πραγμάτων (IoT), αυτοματισμών, ανάλυσης μεγάλου όγκου δεδομένων και προηγμένων ψηφιακών συστημάτων, με σκοπό την αποτελεσματική, οικονομικά συμφέρουσα και περιβαλλοντικά φιλική διαχείριση των αστικών στερεών αποβλήτων.

2.3.2. Στόχοι

Οι βασικοί στόχοι της έξυπνης διαχείρισης απορριμμάτων συνοψίζονται ως ακολούθως:

1. Η μείωση του συνολικού κόστους που σχετίζεται με τη διαδικασία συλλογής των αστικών στερεών αποβλήτων, μέσω της αξιοποίησης τεχνολογικών καινοτομιών και της βελτιστοποίησης των επιχειρησιακών διαδικασιών.
2. Η αναβάθμιση της αποδοτικότητας στον σχεδιασμό και την εκτέλεση των δρομολογίων συλλογής, με τη χρήση συστημάτων δυναμικής προσαρμογής και ανάλυσης δεδομένων, ώστε να επιτυγχάνεται η βέλτιστη κατανομή πόρων και η ελαχιστοποίηση των λειτουργικών δαπανών.
3. Ο περιορισμός των φαινομένων υπερπλήρωσης των κάδων απορριμμάτων, μέσω της παρακολούθησης της κατάστασής τους σε πραγματικό χρόνο, συμβάλλοντας στην άμεση ανταπόκριση και την αποτελεσματική διαχείριση των υποδομών συλλογής.
4. Η ελαχιστοποίηση των εκπομπών διοξειδίου του άνθρακα (CO_2) που προκύπτουν από τις δραστηριότητες διαχείρισης απορριμμάτων, με στόχο τη μείωση του περιβαλλοντικού αποτυπώματος και την ενίσχυση της βιωσιμότητας των αστικών περιοχών.

5. Η αύξηση των ποσοστών ανακύκλωσης, μέσω της εφαρμογής στοχευμένων δράσεων, κινήτρων και ενημερωτικών εκστρατειών προς τους πολίτες, προωθώντας τη συμμετοχή και τη συλλογική ευθύνη.
6. Η ενίσχυση της κυκλικής οικονομίας και η προαγωγή της περιβαλλοντικής συνείδησης των πολιτών, μέσα από την υιοθέτηση βιώσιμων πρακτικών και την ενσωμάτωση καινοτόμων λύσεων στη διαχείριση των απορριμμάτων.

2.3.3. Βασικά Συστατικά ενός Έξυπνου Συστήματος Διαχείρισης Απορριμμάτων

Οι κάδοι απορριμμάτων δύναται να εξοπλιστούν με αισθητήρες πληρότητας, οι οποίοι επιτρέπουν την παρακολούθηση της κατάστασής τους σε πραγματικό χρόνο, ενισχύοντας τη διαφάνεια και την αποδοτικότητα του συστήματος συλλογής. Παράλληλα, η χρήση ολοκληρωμένων πληροφοριακών συστημάτων λογισμικού καθίσταται απαραίτητη για τη συλλογή, αποθήκευση και ανάλυση δεδομένων που αφορούν τη διαχείριση των απορριμμάτων, διασφαλίζοντας την αποτελεσματική αξιοποίηση των πληροφοριών προς όφελος της λειτουργίας του συστήματος.

Ο δυναμικός σχεδιασμός των δρομολογίων συλλογής, ο οποίος βασίζεται στα δεδομένα που συλλέγονται από τους αισθητήρες και τις πλατφόρμες, προσφέρει τη δυνατότητα προσαρμογής στις πραγματικές ανάγκες της εκάστοτε περιοχής, συμβάλλοντας στη βέλτιστη κατανομή των διαθέσιμων πόρων και στη μείωση του λειτουργικού κόστους. Επιπλέον, η ανάπτυξη εφαρμογών για φορητές συσκευές παρέχει στους πολίτες τη δυνατότητα ενημέρωσης, λήψης κινήτρων και ενεργούς συμμετοχής στη διαδικασία διαχείρισης των απορριμμάτων, ενισχύοντας την περιβαλλοντική συνείδηση και τη συλλογική ευθύνη.

Η υιοθέτηση συστημάτων αναφοράς περιστατικών, όπως η καταγραφή υπερχειλίσεων κάδων ή η εκδήλωση πυρκαγιάς, συμβάλλει στην άμεση αντιμετώπιση προβλημάτων και στην ενίσχυση της ασφάλειας των υποδομών. Τέλος, η ενσωμάτωση του συστήματος διαχείρισης απορριμμάτων στο ευρύτερο οικοσύστημα της Έξυπνης Πόλης διασφαλίζει τη διαλειτουργικότητα και τη συνέργεια μεταξύ των υπηρεσιών, δημιουργώντας τις προϋποθέσεις για την ολιστική και βιώσιμη διαχείριση των αστικών στερεών αποβλήτων.

2.4. Διεθνείς Πρακτικές και Παραδείγματα Έξυπνης Διαχείρισης Απορριμμάτων

2.4.1. Ευρωπαϊκές Πρωτοβουλίες

Στο πλαίσιο της υιοθέτησης καινοτόμων πρακτικών έξυπνης διαχείρισης απορριμμάτων σε ευρωπαϊκό επίπεδο, η πόλη της Βαρκελώνης (Ισπανία) έχει προχωρήσει στην εγκατάσταση αισθητήρων σε χιλιάδες κάδους απορριμμάτων, αξιοποιώντας παράλληλα δυναμικά δρομολόγια συλλογής με στόχο τη βέλτιστη διαχείριση των πόρων και τη μείωση του λειτουργικού κόστους. Αντίστοιχα, το Άμστερνταμ (Ολλανδία) εφαρμόζει τεχνολογίες ανάλυσης μεγάλων δεδομένων (Big Data) για τη χωρική αποτύπωση και αξιολόγηση των ροών απορριμμάτων, γεγονός που ενισχύει τη στοχευμένη κατανομή των διαθέσιμων πόρων και τη βελτίωση των διαδικασιών συλλογής. Στην περίπτωση της Κοπεγχάγης (Δανία), οι κάδοι απορριμμάτων έχουν διασυνδεθεί με το δίκτυο LoRaWAN, επιτυγχάνοντας επικοινωνία χαμηλής ενεργειακής κατανάλωσης και συνεχή παρακολούθηση της πληρότητάς τους, συμβάλλοντας στην αποδοτικότερη λειτουργία του συστήματος. Τέλος, η Εσθονία αναδεικνύεται ως πρότυπο πλήρους ψηφιακής διακυβέρνησης στον τομέα των περιβαλλοντικών υπηρεσιών, εφαρμόζοντας προηγμένες ψηφιακές λύσεις για την ολοκληρωμένη διαχείριση των απορριμμάτων και την ενίσχυση της διαφάνειας και της αποδοτικότητας του συστήματος.

2.4.2. Ιδιωτικές Τεχνολογικές Λύσεις σε Διεθνές Επίπεδο

Σε παγκόσμια κλίμακα, ο ιδιωτικός τομέας διαδραματίζει καθοριστικό ρόλο στην προώθηση καινοτόμων τεχνολογικών λύσεων για την έξυπνη διαχείριση απορριμμάτων. Ενδεικτικά, η φινλανδική εταιρεία Enevo έχει αναπτύξει προηγμένους αισθητήρες υπερήχων και συστήματα προγνωστικής ανάλυσης, τα οποία συμβάλλουν στη βελτιστοποίηση των διαδικασιών συλλογής απορριμμάτων, μέσω της ακριβούς πρόβλεψης των αναγκών αδειάσματος των κάδων και της δυναμικής προσαρμογής των δρομολογίων συλλογής. Επιπλέον, η αμερικανική εταιρεία Bigbelly εισήγαγε στον τομέα της διαχείρισης απορριμμάτων κάδους εξοπλισμένους με συμπιεστές που λειτουργούν με ηλιακή ενέργεια, αυξάνοντας σημαντικά τη χωρητικότητα των κάδων και μειώνοντας τη συχνότητα αποκομιδής. Παράλληλα, η εταιρεία SmartBin έχει υλοποιήσει λύσεις διαχείρισης κάδων που βασίζονται σε τεχνολογίες cloud και παρέχουν προηγμένες διαδραστικές πλατφόρμες παρακολούθησης (dashboards), επιτρέποντας την απομακρυσμένη εποπτεία και βελτιστοποίηση της λειτουργίας των υποδομών συλλογής απορριμμάτων.

2.5. Έξυπνη Διαχείριση Απορριμμάτων στον Ελληνικό Χώρο

Την τελευταία περίοδο, η Ελλάδα καταγράφει σημαντικά βήματα εκσυγχρονισμού στον τομέα της διαχείρισης απορριμμάτων, με εμφανείς διαφοροποιήσεις μεταξύ των δήμων. Οι βασικές εξελίξεις περιλαμβάνουν:

2.5.1. Πιλοτικές Εφαρμογές Ελληνικών Δήμων

Ο Δήμος Τρικκαίων αναδεικνύεται ως πρότυπο καινοτομίας στον τομέα των «Έξυπνων Πόλεων», προχωρώντας στη συστηματική αξιοποίηση αισθητήρων και ψηφιακών υπηρεσιών για τη βελτίωση της διαχείρισης των απορριμμάτων. Παράλληλα, ο Δήμος Χανίων εφαρμόζει προηγμένες τεχνολογικές λύσεις, όπως έξυπνους κάδους και διαδραστικούς χάρτες, οι οποίοι επιτρέπουν την αποτελεσματική παρακολούθηση και διαχείριση των ροών απορριμμάτων. Ο Δήμος Ηρακλείου συμμετέχει ενεργά σε ευρωπαϊκά προγράμματα, μεταξύ των οποίων και το CIVITAS, με στόχο την ενίσχυση της βιώσιμης αστικής κινητικότητας και την προώθηση καινοτόμων πρακτικών στον τομέα της συλλογής και διαχείρισης των απορριμμάτων. Επιπρόσθετα, δήμοι της Περιφέρειας Αττικής υλοποιούν πιλοτικές εφαρμογές που βασίζονται στην τεχνολογία του Διαδικτύου των Πραγμάτων (IoT), εστιάζοντας σε δράσεις ανακύκλωσης και τη δημιουργία πράσινων σημείων, ενισχύοντας με τον τρόπο αυτό τη συλλογική προσπάθεια για τον ψηφιακό μετασχηματισμό της διαχείρισης αστικών απορριμμάτων.

2.5.2. Προκλήσεις που Αντιμετωπίζει η Ελλάδα

Οι προκλήσεις που αντιμετωπίζει η Ελλάδα στον τομέα της έξυπνης διαχείρισης απορριμμάτων συνοψίζονται στα εξής: η ύπαρξη περιορισμένων διαθέσιμων πόρων και η ανεπαρκής χρηματοδότηση αποτελούν σημαντικά εμπόδια για την υλοποίηση σύγχρονων λύσεων. Επιπλέον, καταγράφεται έλλειψη εξειδικευμένου τεχνικού προσωπικού, γεγονός που δυσχεραίνει την εγκατάσταση και λειτουργία προηγμένων συστημάτων. Οι γραφειοκρατικές διαδικασίες και οι καθυστερήσεις στη διενέργεια δημόσιων διαγωνισμών επιβραδύνουν περαιτέρω την υιοθέτηση καινοτόμων πρακτικών. Παρατηρείται επίσης ανισομερής κατανομή των υποδομών μεταξύ των περιφερειών, με αποτέλεσμα να δημιουργούνται ανισότητες στην πρόσβαση σε έξυπνες υπηρεσίες. Το χαμηλό επίπεδο περιβαλλοντικής συνείδησης και η περιορισμένη συμμετοχή των πολιτών δυσχεραίνουν την αποτελεσματική εφαρμογή των σχετικών πολιτικών. Επιπρόσθετα, τα ελλείμματα στη

διαλειτουργικότητα των πληροφοριακών συστημάτων περιορίζουν την ανταλλαγή και αξιοποίηση δεδομένων. Τέλος, σε πολλούς Οργανισμούς Τοπικής Αυτοδιοίκησης καταγράφεται περιορισμένη ωριμότητα ως προς τον ψηφιακό μετασχηματισμό, γεγονός που επιβραδύνει τη μετάβαση προς το μοντέλο της «έξυπνης πόλης» και την ολοκληρωμένη υιοθέτηση των σχετικών τεχνολογικών λύσεων.

2.6. Θεωρητική Σύνδεση Smart Cities και Smart Waste Management

Η διαχείριση απορριμμάτων με έξυπνες τεχνολογίες δεν συνιστά μια αυτόνομη ή αποσπασματική δραστηριότητα, αλλά αποτελεί θεμελιώδη και στρατηγικό πυλώνα στο πλαίσιο της δομής των «Έξυπνων Πόλεων». Ειδικότερα, η εφαρμογή σύγχρονων πρακτικών και εργαλείων συμβάλλει καθοριστικά στην ενίσχυση της έξυπνης διακυβέρνησης, καθώς προάγει τη διαφάνεια στη λήψη αποφάσεων και ενδυναμώνει τη συμμετοχή των πολιτών στις διαδικασίες διαχείρισης. Επιπλέον, συνεισφέρει στη βελτίωση του αστικού περιβάλλοντος, μέσω της μείωσης της ρύπανσης και της αποτελεσματικής αξιοποίησης των διαθέσιμων πόρων. Παράλληλα, υποστηρίζει την ανάπτυξη έξυπνης κινητικότητας, εξασφαλίζοντας ορθολογική διαχείριση και βελτιστοποίηση των δρομολογίων συλλογής απορριμμάτων. Τέλος, η έξυπνη διαχείριση απορριμμάτων ενισχύει την οικονομική αποδοτικότητα των Οργανισμών Τοπικής Αυτοδιοίκησης, περιορίζοντας τα λειτουργικά κόστη. Συνοψίζοντας, η εν λόγω πρακτική αποτελεί ολοκληρωμένη προσέγγιση ψηφιακού μετασχηματισμού, με άμεσες και έμμεσες θετικές επιδράσεις στο περιβάλλον, την οικονομία και το κοινωνικό σύνολο.

2.7. Επεξεργασία και Ανάλυση Δεδομένων σε Πραγματικό Χρόνο

Η έξυπνη διαχείριση απορριμμάτων θεμελιώνεται σε προηγμένα συστήματα συλλογής, επεξεργασίας και ανάλυσης δεδομένων σε πραγματικό χρόνο. Η ενσωμάτωση των τεχνολογιών αυτών παρέχει στους δήμους τη δυνατότητα άμεσης απόκρισης σε κρίσιμα συμβάντα, όπως η υπερχείλιση κάδων, οι βλάβες σε οχήματα συλλογής ή η αιφνίδια αύξηση της παραγωγής απορριμμάτων. Παράλληλα, συμβάλλει ουσιαστικά στη βελτιστοποίηση των επιχειρησιακών διαδικασιών, ενισχύοντας την αποτελεσματικότητα και την αποδοτικότητα της διαχείρισης των αστικών αποβλήτων.

2.7.1. Ροές δεδομένων (Data Pipelines)

Μία τυπική ροή δεδομένων στο πλαίσιο συστημάτων Έξυπνης Διαχείρισης Απορριμμάτων (Smart Waste Management) διαρθρώνεται σε διαδοχικά στάδια, τα οποία περιγράφονται αναλυτικά ως εξής:

Συλλογή Δεδομένων (Data Acquisition):

Η διαδικασία εκκινεί με την τοποθέτηση εξειδικευμένων αισθητήρων στους κάδους απορριμμάτων, όπως αισθητήρες υπερήχων, υπερύθρων, θερμοκρασίας και συσκευές γεωεντοπισμού (GPS), οι οποίοι παράγουν συνεχές και υψηλής πυκνότητας ρεύμα δεδομένων σχετικά με την πληρότητα, τη θερμοκρασία και τη θέση των κάδων.

Μετάδοση Δεδομένων (Data Transmission):

Για τη μεταφορά των συλλεγόμενων δεδομένων χρησιμοποιούνται τεχνολογίες ασύρματων δικτύων χαμηλής κατανάλωσης ενέργειας, όπως LoRaWAN, NB-IoT και LTE-M, ενώ για απαιτήσεις υψηλότερου εύρους ζώνης αξιοποιούνται δίκτυα κινητής τηλεφωνίας 4G/5G.

Επεξεργασία Δεδομένων (Edge ή Cloud Processing):

Η επεξεργασία των δεδομένων πραγματοποιείται είτε τοπικά μέσω edge computing, επιτρέποντας την άμεση και ταχεία ανάλυση πληροφορίας κοντά στην πηγή, είτε σε κεντρικό επίπεδο μέσω cloud computing, όπου τα δεδομένα υπόκεινται σε εκτενή ανάλυση και αποθήκευση σε μεγάλης κλίμακας υποδομές.

Αποθήκευση Δεδομένων (Data Storage):

Τα δεδομένα αποθηκεύονται σε κατάλληλες βάσεις, όπως NoSQL (π.χ. MongoDB, Cassandra) για μη δομημένα δεδομένα, καθώς και σε βάσεις δεδομένων χρονικών σειρών (time-series databases), όπως η InfluxDB, που εξειδικεύονται στη διαχείριση δεδομένων αισθητήρων.

Ανάλυση και Οπτικοποίηση (Analytics & Visualization):

Στο τελικό στάδιο, αξιοποιούνται εργαλεία επιχειρησιακής ευφυΐας (Business Intelligence) ή εξειδικευμένοι πίνακες ελέγχου γεωγραφικών πληροφοριών (GIS dashboards) για την ανάλυση και απεικόνιση των δεδομένων, διευκολύνοντας τη λήψη τεκμηριωμένων αποφάσεων από τους αρμόδιους φορείς.

2.7.2. Εργαλεία Επεξεργασίας Δεδομένων

Το Apache Kafka αποτελεί ένα προηγμένο σύστημα διαχείρισης και διαμεσολάβησης μηνυμάτων, κατάλληλο για τη διαχείριση μεγάλου όγκου δεδομένων ροής (streaming data)

σε πραγματικό χρόνο, παρέχοντας αξιόπιστη και κλιμακούμενη υποδομή για την ενσωμάτωση και επεξεργασία δεδομένων από πολλαπλές πηγές.

Το Apache Spark Streaming προσφέρει δυνατότητες επεξεργασίας δεδομένων σε πραγματικό χρόνο, ενώ υποστηρίζει τεχνικές μηχανικής μάθησης (machine learning), επιτρέποντας την ανάλυση και αξιοποίηση των εισερχόμενων δεδομένων με υψηλή απόδοση και ευελιξία.

Οι MQTT Brokers, όπως το Mosquitto, χρησιμοποιούνται για την αποδοτική και ασφαλή επικοινωνία μεταξύ συσκευών του Διαδικτύου των Πραγμάτων (IoT), διευκολύνοντας τη μεταφορά μηνυμάτων χαμηλού όγκου με ελάχιστη καθυστέρηση και χαμηλή κατανάλωση πόρων.

Οι πλατφόρμες ThingSpeak, AWS IoT και Azure IoT Hub παρέχουν ολοκληρωμένες λύσεις για την διαχείριση συσκευών IoT και την ανάλυση δεδομένων σε πραγματικό χρόνο, ενσωματώνοντας εργαλεία για συλλογή, επεξεργασία και οπτικοποίηση των δεδομένων, με στόχο τη βελτιστοποίηση των επιχειρησιακών διαδικασιών και την υποστήριξη τεκμηριωμένης λήψης αποφάσεων.

2.8. Αλγόριθμοι Βελτιστοποίησης και Έξυπνη Δρομολόγηση Αποκομιδής

Η διαμόρφωση βέλτιστων δρομολογίων για την αποκομιδή απορριμμάτων συνιστά μία από τις πλέον κομβικές και τεχνολογικά προηγμένες εφαρμογές στο πεδίο της ευφυούς διαχείρισης αποβλήτων (Smart Waste Management). Η υλοποίηση τέτοιων συστημάτων εστιάζει στην αξιοποίηση αλγοριθμικών μεθόδων και τεχνολογιών αιχμής με κύριο σκοπό τη βελτιστοποίηση των επιχειρησιακών διαδικασιών. Ειδικότερα, ο σχεδιασμός των δρομολογίων αποσκοπεί στην ελαχιστοποίηση μιας σειράς κρίσιμων παραμέτρων, οι οποίες περιλαμβάνουν:

- τη συνολική διανυόμενη απόσταση από τα οχήματα συλλογής,
- την κατανάλωση καυσίμων, η οποία σχετίζεται άμεσα με το περιβαλλοντικό αποτύπωμα και το οικονομικό κόστος,
- τον χρόνο ολοκλήρωσης των δρομολογίων, με στόχο την ταχύτερη και αποτελεσματικότερη εξυπηρέτηση των περιοχών,
- το συνολικό λειτουργικό κόστος των υπηρεσιών αποκομιδής, ενισχύοντας τη βιωσιμότητα και αποδοτικότητα των δημοτικών και ιδιωτικών φορέων διαχείρισης.

Η επίτευξη των παραπάνω στόχων προϋποθέτει τη συστηματική ανάλυση δεδομένων, την εφαρμογή εξελιγμένων αλγορίθμων βελτιστοποίησης και την ενσωμάτωση δυναμικών πληροφοριών σε πραγματικό χρόνο, ώστε να προσαρμόζονται τα δρομολόγια στις εκάστοτε ανάγκες και συνθήκες. Συνεπώς, ο αποτελεσματικός σχεδιασμός δρομολογίων αποτελεί αναπόσπαστο μέρος μιας σύγχρονης στρατηγικής για την αειφόρο και έξυπνη διαχείριση των στερεών αποβλήτων.

2.8.1. Vehicle Routing Problem (VRP)

Το πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem - VRP) αποτελεί το θεμελιώδες θεωρητικό πλαίσιο για τον βέλτιστο σχεδιασμό δρομολογίων συλλογής και διανομής σε συστήματα διαχείρισης αποβλήτων και εφοδιαστικής αλυσίδας. Στη διεθνή βιβλιογραφία, έχουν αναπτυχθεί πληθώρα επεκτάσεων του βασικού προβλήματος, ώστε να καλύπτονται οι ιδιαίτερες απαιτήσεις που προκύπτουν από την επιχειρησιακή πραγματικότητα. Ενδεικτικά, συγκαταλέγονται οι ακόλουθες παραλλαγές:

- **CVRP (Capacitated Vehicle Routing Problem):** Στη συγκεκριμένη εκδοχή, ενσωματώνεται ο περιορισμός της χωρητικότητας των οχημάτων, γεγονός που επιβάλλει την κατανομή των αποστολών με βάση τα μέγιστα επιτρεπτά φορτία κάθε οχήματος.
- **VRPTW (Vehicle Routing Problem with Time Windows):** Η παραλλαγή αυτή προσθέτει χρονικούς περιορισμούς (time windows), εντός των οποίων πρέπει να πραγματοποιηθούν οι συλλογές ή οι παραδόσεις, καθιστώντας τον προγραμματισμό ακόμη πιο ρεαλιστικό και προσαρμοσμένο στις ανάγκες των χρηστών ή των επιχειρήσεων.
- **DVRP (Dynamic Vehicle Routing Problem):** Το δυναμικό VRP λαμβάνει υπόψη δεδομένα που μεταβάλλονται σε πραγματικό χρόνο, όπως για παράδειγμα η πληρότητα των κάδων ή η κυκλοφοριακή συμφόρηση, επιτρέποντας την άμεση αναπροσαρμογή των δρομολογίων σύμφωνα με τις τρέχουσες συνθήκες.

Η υιοθέτηση και εφαρμογή των παραπάνω μοντέλων είναι καθοριστικής σημασίας για την ανάπτυξη ευέλικτων και αποδοτικών συστημάτων αποκομιδής, συμβάλλοντας ουσιαστικά στη βελτιστοποίηση των διαθέσιμων πόρων και στη μείωση του περιβαλλοντικού αποτυπώματος.

2.8.2. Κλασικοί Αλγόριθμοι

Στο πλαίσιο της δρομολόγησης οχημάτων, ιδιαίτερη σημασία αποδίδεται στη χρήση κλασικών αλγορίθμων αναζήτησης της βέλτιστης διαδρομής σε γραφήματα, οι οποίοι αποτελούν τη βάση για την ανάπτυξη πιο σύνθετων και εξειδικευμένων μεθόδων. Ενδεικτικά, οι πλέον διαδεδομένοι αλγόριθμοι που εφαρμόζονται σε στατικά περιβάλλοντα είναι οι εξής:

- **Αλγόριθμος Dijkstra:** Χρησιμοποιείται για τον προσδιορισμό της συντομότερης διαδρομής μεταξύ κόμβων σε γραφήματα με μη αρνητικά βάρη, εξασφαλίζοντας αποτελεσματική πλοήγηση σε δίκτυα σταθερής δομής.
- **Αλγόριθμος A*:** Αποτελεί εξελιγμένη εκδοχή του Dijkstra, ενσωματώνοντας ευρετικές συναρτήσεις που επιταχύνουν τη διαδικασία εύρεσης της βέλτιστης διαδρομής, ιδιαίτερα σε μεγάλα και σύνθετα γραφήματα.
- **Αλγόριθμος Bellman-Ford:** Εφαρμόζεται σε γραφήματα που ενδέχεται να περιέχουν ακμές με αρνητικά βάρη, παρέχοντας τη δυνατότητα εντοπισμού της συντομότερης διαδρομής ακόμη και σε περιπτώσεις όπου ο αλγόριθμος Dijkstra δεν είναι κατάλληλος.

Η αξιοποίηση των παραπάνω αλγορίθμων αποτελεί βασικό εργαλείο στην ανάπτυξη συστημάτων δρομολόγησης, καθώς προσφέρει αξιόπιστες και αποδοτικές λύσεις σε προβλήματα εύρεσης διαδρομών, θέτοντας τα θεμέλια για την περαιτέρω εφαρμογή ευρετικών και μεταευρετικών τεχνικών σε δυναμικά και πολυδιάστατα περιβάλλοντα.

2.8.3. Μεταευρετικοί Αλγόριθμοι (Metaheuristics)

Λόγω της εγγενούς πολυπλοκότητας του προβλήματος δρομολόγησης οχημάτων (VRP), η επίλυση του συχνά καθίσταται ιδιαίτερα απαιτητική με τη χρήση αποκλειστικά ακριβών αλγορίθμων. Για τον λόγο αυτό, αξιοποιούνται ευρέως ευρετικές και μεταευρετικές τεχνικές, οι οποίες παρέχουν αποτελεσματικές και ποιοτικές προσεγγίσεις σε σύνθετα και μεγάλης κλίμακας προβλήματα.

Ενδεικτικά, ορισμένες από τις πλέον διαδεδομένες μεταευρετικές μεθόδους που εφαρμόζονται στη βελτιστοποίηση του VRP περιλαμβάνουν:

- **Γενετικοί Αλγόριθμοι (Genetic Algorithms - GA):** Βασίζονται στις αρχές της φυσικής επιλογής και της εξέλιξης, επιτυγχάνοντας τη βελτίωση των λύσεων μέσω

διαδικασιών αναπαραγωγής, διασταύρωσης και μετάλλαξης, με στόχο την ανεύρεση βέλτιστων ή εγγυημένα καλών λύσεων.

- **Βελτιστοποίηση μέσω Αποικίας Μυρμηγκιών (Ant Colony Optimization - ACO):** Εμπνέεται από τη συλλογική συμπεριφορά των μυρμηγκιών κατά την αναζήτηση τροφής, προσομοιώνοντας τη διαδικασία κατά την οποία τα μυρμήγκια αφήνουν φερομόνες σε διαδρομές που οδηγούν σε αποδοτικές λύσεις, ενισχύοντας έτσι την πιθανότητα επιλογής τους σε επόμενα βήματα του αλγορίθμου.
- **Προσομοιωμένη Ανόπτηση (Simulated Annealing - SA):** Προσομοιώνει τη φυσική διαδικασία της ανόπτησης μετάλλων, όπου η σταδιακή μείωση της «θερμοκρασίας» οδηγεί στη σταθεροποίηση της λύσης, επιτρέποντας την αποφυγή τοπικών ακροτάτων και την προσέγγιση της παγκόσμιας βέλτιστης λύσης.
- **Αναζήτηση Tabu (Tabu Search):** Εφαρμόζει μηχανισμούς μνήμης για την αποτροπή επανάληψης προηγούμενων λύσεων, διευρύνοντας έτσι το πεδίο αναζήτησης και ενισχύοντας τις πιθανότητες εύρεσης καλύτερων λύσεων μέσω της εξερεύνησης νέων, μη επισκεπτόμενων περιοχών του χώρου λύσεων.

Η ενσωμάτωση των παραπάνω μεταευρετικών τεχνικών στη διαδικασία δρομολόγησης οχημάτων συμβάλλει καθοριστικά στην αντιμετώπιση των προκλήσεων που προκύπτουν από την πολυπλοκότητα και τη δυναμική φύση του προβλήματος, προσφέροντας αξιόπιστες και ευέλικτες λύσεις σε πραγματικές συνθήκες λειτουργίας.

2.8.4. Εφαρμογές σε Έξυπνες Πόλεις

Οι προαναφερθέντες αλγόριθμοι ενσωματώνονται σε προηγμένες τεχνολογικές υποδομές που υποστηρίζουν τη λειτουργία των «Έξυπνων Πόλεων», συμβάλλοντας ουσιαστικά στη βελτιστοποίηση κρίσιμων διαδικασιών αστικής διαχείρισης. Ειδικότερα, αξιοποιούνται σε:

- Πλατφόρμες έξυπνης δρομολόγησης που υιοθετούνται από δημοτικές αρχές για την αποτελεσματική διαχείριση πόρων και υπηρεσιών.
- Συστήματα σχεδιασμού διαδρομών βασισμένα σε Γεωγραφικά Πληροφοριακά Συστήματα (GIS), τα οποία προσφέρουν δυναμική ανάλυση και προσαρμοστικότητα στις ανάγκες της αστικής κινητικότητας.
- Μηχανισμούς βελτιστοποίησης που λειτουργούν σε υποδομές υπολογιστικού νέφους (cloud-based optimization engines), επιτρέποντας την επεξεργασία μεγάλου όγκου δεδομένων και την ταχεία εξαγωγή βέλτιστων λύσεων σε πραγματικό χρόνο.

Η συνδυασμένη εφαρμογή των ανωτέρω τεχνολογικών λύσεων ενισχύει τη λειτουργικότητα των έξυπνων πόλεων, προάγοντας την αποδοτικότητα, την ευελιξία και την αειφορία σε τομείς όπως η δρομολόγηση οχημάτων, η διαχείριση απορριμμάτων και η παροχή δημοτικών υπηρεσιών.

2.9. Προγνωστική Ανάλυση (Predictive Analytics) στην Έξυπνη Διαχείριση Πόλεων

Η εφαρμογή της προγνωστικής ανάλυσης στη διαχείριση πόλεων παρέχει στις αρμόδιες δημοτικές αρχές τη δυνατότητα πρόβλεψης συμπεριφορών, αναγνώρισης μοτίβων και εντοπισμού περιόδων αιχμής στη δημιουργία απορριμμάτων. Μέσω της αξιοποίησης προηγμένων αναλυτικών τεχνικών, επιτυγχάνεται η ενίσχυση της αποτελεσματικότητας των διαδικασιών συλλογής και διαχείρισης, διαμορφώνοντας τις συνθήκες για βέλτιστη λήψη αποφάσεων και ορθολογική κατανομή των διαθέσιμων πόρων.

2.9.1. Προγνωστικά Μοντέλα

Για τις ανάγκες της προγνωστικής ανάλυσης στην έξυπνη διαχείριση απορριμμάτων, εφαρμόζονται εξειδικευμένα μοντέλα, όπως:

- **ARIMA και SARIMA:** Χρησιμοποιούνται για την ανάλυση και πρόβλεψη χρονοσειρών σχετικών με την πληρότητα των κάδων απορριμμάτων, προσφέροντας ακριβείς εκτιμήσεις με βάση ιστορικά δεδομένα.
- **Νευρωνικά Δίκτυα (RNN, LSTM):** Εφαρμόζονται για την πρόβλεψη διακυμάνσεων στην παραγωγή απορριμμάτων, αξιοποιώντας τις δυνατότητες της τεχνητής νοημοσύνης για την αναγνώριση σύνθετων σχέσεων και μοτίβων.
- **Random Forest & Gradient Boosting:** Αποτελούν προηγμένες μεθόδους μηχανικής μάθησης που επικεντρώνονται στην εκτίμηση της πιθανότητας υπερπλήρωσης των κάδων, συμβάλλοντας στην έγκαιρη λήψη προληπτικών μέτρων.
- **Μοντέλα Παλινδρόμησης:** Περιλαμβάνουν γραμμικές και πολυωνυμικές προσεγγίσεις για την πραγματοποίηση απλών προβλέψεων, οι οποίες διευκολύνουν την ανάλυση και τον σχεδιασμό των επιχειρησιακών διαδικασιών.

Η συστηματική αξιοποίηση των ανωτέρω μοντέλων ενισχύει τη δυνατότητα των ΟΤΑ να αντιμετωπίζουν με αποτελεσματικότητα τις προκλήσεις της σύγχρονης αστικής

διαχείρισης, διαμορφώνοντας ένα πλαίσιο λειτουργίας που χαρακτηρίζεται από ευελιξία, προσαρμοστικότητα και υψηλή αποδοτικότητα.

2.9.2. Εφαρμογές Προγνωστικής Ανάλυσης στη Διαχείριση Απορριμμάτων

Η αξιοποίηση των τεχνικών προγνωστικής ανάλυσης στη διαχείριση απορριμμάτων προσφέρει πληθώρα εφαρμογών που συμβάλλουν ουσιαστικά στη βελτιστοποίηση των λειτουργιών των Οργανισμών Τοπικής Αυτοδιοίκησης. Ενδεικτικά, οι κυριότερες εφαρμογές περιλαμβάνουν:

- **Πρόβλεψη του χρόνου υπερπλήρωσης κάθε κάδου:** Μέσω ανάλυσης ιστορικών και τρεχόντων δεδομένων, παρέχεται η δυνατότητα ακριβούς εκτίμησης του χρονικού διαστήματος που απαιτείται έως την πλήρωση ενός κάδου απορριμμάτων, διευκολύνοντας τον προγραμματισμό των συλλογών.
- **Υπολογισμός αναγκών δρομολογίων για επερχόμενες ημέρες:** Η προγνωστική ανάλυση επιτρέπει τον σχεδιασμό και τη βελτιστοποίηση των δρομολογίων των απορριμματοφόρων, λαμβάνοντας υπόψη τις εκτιμώμενες ποσότητες απορριμμάτων και τις τάσεις πληρότητας των κάδων.
- **Αναγνώριση εποχικών τάσεων:** Η ανίχνευση περιοδικών μεταβολών, όπως η αυξημένη παραγωγή απορριμμάτων κατά τη διάρκεια της τουριστικής περιόδου στην Κρήτη, επιτρέπει την προσαρμογή των επιχειρησιακών στρατηγικών και την ορθολογική κατανομή των πόρων.
- **Έγκαιρη συντήρηση οχημάτων (predictive maintenance):** Μέσω της ανάλυσης δεδομένων λειτουργίας των οχημάτων, είναι δυνατή η πρόβλεψη πιθανών βλαβών και η προγραμματισμένη συντήρηση, μειώνοντας τα απρόβλεπτα κόστη και τις διακοπές λειτουργίας.
- **Ανάλυση συμπεριφοράς πολιτών ως προς την ανακύκλωση:** Η εξέταση των δεδομένων σχετικών με τη συμμετοχή των πολιτών σε δράσεις ανακύκλωσης συμβάλλει στην κατανόηση τάσεων και στην ανάπτυξη στοχευμένων παρεμβάσεων για την ενίσχυση της ανακύκλωσης.

Συνολικά, η εφαρμογή προγνωστικών μεθόδων ενισχύει σημαντικά την αποδοτικότητα των συστημάτων διαχείρισης απορριμμάτων, οδηγεί σε μείωση λειτουργικού κόστους και

παρέχει τη βάση για τη λήψη βελτιστοποιημένων και τεκμηριωμένων αποφάσεων πολιτικής.

2.10. Συστήματα Υποστήριξης Αποφάσεων και Dashboards

Τα Συστήματα Υποστήριξης Αποφάσεων (Decision Support Systems – DSS) συνιστούν θεμελιώδες στοιχείο των σύγχρονων «έξυπνων» αστικών περιβαλλόντων, καθώς προσφέρουν στα στελέχη των Οργανισμών Τοπικής Αυτοδιοίκησης (ΟΤΑ) τη δυνατότητα λήψης τεκμηριωμένων και επιστημονικά αιτιολογημένων αποφάσεων. Μέσω της ενσωμάτωσης προηγμένων τεχνολογιών συλλογής, επεξεργασίας και ανάλυσης δεδομένων, τα DSS συμβάλλουν ουσιαστικά στη βελτιστοποίηση των επιχειρησιακών διαδικασιών, ενισχύοντας τη διαφάνεια και την αποτελεσματικότητα της διοικητικής λειτουργίας.

2.10.1. Κύριες Λειτουργίες των Συστημάτων Υποστήριξης Αποφάσεων (DSS)

Τα Σύγχρονα Συστήματα Υποστήριξης Αποφάσεων (Decision Support Systems – DSS) διαδραματίζουν καταλυτικό ρόλο στη διαχείριση των αστικών λειτουργιών και ειδικότερα στη βελτιστοποίηση των διαδικασιών που σχετίζονται με τη διαχείριση απορριμμάτων από τους Οργανισμούς Τοπικής Αυτοδιοίκησης. Οι βασικές λειτουργίες που επιτελούν τα συστήματα αυτά περιλαμβάνουν:

- **Συλλογή και ενοποίηση δεδομένων από αισθητήρες:** Τα DSS συγκεντρώνουν και ενοποιούν δεδομένα που συλλέγονται από ένα πλήθος αισθητήρων και συστημάτων παρακολούθησης, διασφαλίζοντας την πληρότητα και την αξιοπιστία των πληροφοριών που απαιτούνται για τη λήψη αποφάσεων.
- **Οπτικοποίηση δεδομένων και δεικτών απόδοσης (KPIs):** Μέσω εξειδικευμένων εργαλείων, παρέχεται η δυνατότητα οπτικοποίησης των κρίσιμων δεδομένων και των βασικών δεικτών απόδοσης, διευκολύνοντας την άμεση κατανόηση της τρέχουσας κατάστασης και την έγκαιρη ανίχνευση αποκλίσεων από τους στόχους.
- **Παρουσίαση προτεινόμενων λύσεων (recommendation engines):** Τα DSS αξιοποιούν αλγορίθμους συστάσεων για την υποβολή τεκμηριωμένων προτάσεων και εναλλακτικών λύσεων στους υπεύθυνους λήψης αποφάσεων, ενισχύοντας τη διαδικασία επιλογής των βέλτιστων στρατηγικών παρεμβάσεων.

- **Χωρική ανάλυση μέσω Γεωγραφικών Συστημάτων Πληροφοριών (GIS):** Ενσωματώνουν εργαλεία γεωγραφικής ανάλυσης, επιτρέποντας τη μελέτη και απεικόνιση των δεδομένων σε χωρικό επίπεδο, γεγονός που συμβάλλει ουσιαστικά στον εντοπισμό τοπικών τάσεων και προβλημάτων.
- **Προσομοίωση σεναρίων (scenario modeling):** Παρέχουν τη δυνατότητα δημιουργίας και αξιολόγησης εναλλακτικών σεναρίων, επιτρέποντας τη δοκιμή των επιπτώσεων διαφορετικών στρατηγικών δράσης πριν από την υλοποίησή τους.

Συνολικά, οι λειτουργίες αυτές καθιστούν τα DSS απαραίτητα εργαλεία για την υποστήριξη της τεκμηριωμένης και αποτελεσματικής λήψης αποφάσεων στους σύγχρονους ΟΤΑ, ενισχύοντας τη διαφάνεια, την αποδοτικότητα και τη βιωσιμότητα της διοικητικής τους λειτουργίας.

2.10.2. Κατηγορίες Πινάκων Ελέγχου (Dashboards)

Οι πίνακες ελέγχου που χρησιμοποιούνται στο πλαίσιο των Συστημάτων Υποστήριξης Αποφάσεων διακρίνονται σε τρεις βασικές κατηγορίες, ανάλογα με τον σκοπό και τις απαιτήσεις του εκάστοτε Οργανισμού Τοπικής Αυτοδιοίκησης (ΟΤΑ):

- Λειτουργικοί πίνακες ελέγχου (Operational dashboards):

  Αυτοί οι πίνακες εστιάζουν στην παρακολούθηση κρίσιμων λειτουργικών παραμέτρων σε πραγματικό χρόνο, όπως η πληρότητα των κάδων απορριμμάτων. Η άμεση πρόσβαση σε ενημερωμένα δεδομένα επιτρέπει την ταχεία λήψη αποφάσεων και την αποτελεσματική διαχείριση των καθημερινών λειτουργιών.

- Αναλυτικοί πίνακες ελέγχου (Analytical dashboards):

  Οι συγκεκριμένοι πίνακες προσφέρουν τη δυνατότητα εις βάθος ανάλυσης δεδομένων, μέσω της συγκριτικής αξιολόγησης, της απεικόνισης θερμικών χαρτών (heatmaps) και της παρακολούθησης βασικών δεικτών απόδοσης (KPIs). Με τον τρόπο αυτό ενισχύεται η διερεύνηση προτύπων και τάσεων που συμβάλλουν στη βελτιστοποίηση των επιχειρησιακών διαδικασιών.

- Στρατηγικοί πίνακες ελέγχου (Strategic dashboards):

  Οι στρατηγικοί πίνακες υποστηρίζουν τον μακροπρόθεσμο σχεδιασμό και τη στρατηγική λήψη αποφάσεων στους Οργανισμούς Τοπικής Αυτοδιοίκησης. Παρέχουν συγκεντρωτική εικόνα της συνολικής απόδοσης, διευκολύνοντας την

αξιολόγηση των αποτελεσμάτων και τη διαμόρφωση βιώσιμων πολιτικών και παρεμβάσεων.

2.10.3. Εργαλεία

Στο πλαίσιο της υλοποίησης Συστημάτων Υποστήριξης Αποφάσεων, αξιοποιείται πληθώρα σύγχρονων εργαλείων ανάλυσης και οπτικοποίησης δεδομένων, τα οποία συμβάλλουν στην αποτελεσματική παρακολούθηση και διαχείριση των επιχειρησιακών παραμέτρων στους Οργανισμούς Τοπικής Αυτοδιοίκησης. Ενδεικτικά, το Power BI (Microsoft) προσφέρει προηγμένες δυνατότητες ανάλυσης και δυναμικής παρουσίασης δεδομένων μέσω διαδραστικών αναφορών. Αντίστοιχα, το Grafana εξειδικεύεται στην απεικόνιση χρονοσειρών (time-series dashboards), επιτρέποντας τη σύνδεση με βάσεις δεδομένων όπως InfluxDB ή Prometheus για την παρακολούθηση δεδομένων σε πραγματικό χρόνο.
Το Tableau αποτελεί ένα ακόμη ισχυρό εργαλείο επιχειρησιακής ευφυΐας (Business Intelligence), προσφέροντας προηγμένες μεθόδους ανάλυσης και οπτικοποίησης. Παράλληλα, αναπτύσσονται εξατομικευμένοι διαδικτυακοί πίνακες ελέγχου (Custom Web Dashboards) με χρήση τεχνολογιών όπως React, Node.js, Python/Flask και Leaflet/GIS, οι οποίοι δύνανται να προσαρμοστούν στις ιδιαίτερες ανάγκες κάθε Οργανισμού. Ιδιαίτερη αναφορά αξίζει στα Esri ArcGIS Dashboards, που εξειδικεύονται στη χαρτογραφική απεικόνιση δεδομένων, ενισχύοντας τη γεωχωρική ανάλυση και λήψη αποφάσεων.
Στην πράξη, οι δήμοι υιοθετούν συχνά έναν συνδυασμό εργαλείων, όπως πίνακες ελέγχου Power BI ή Grafana σε συνδυασμό με γεωγραφικούς χάρτες GIS, προκειμένου να επιτύχουν ολοκληρωμένη και πολυεπίπεδη εποπτεία των κρίσιμων λειτουργιών τους.

2.11. Ανασκόπηση Συναφών Ερευνητικών και Εμπορικών Έργων

2.11.1. Ερευνητικά Έργα (Ευρωπαϊκά & Διεθνή)

Στο πλαίσιο της παρούσας μελέτης, ιδιαίτερο ενδιαφέρον παρουσιάζουν ορισμένα ερευνητικά έργα που υλοποιήθηκαν σε ευρωπαϊκό και διεθνές επίπεδο και τα οποία συνέβαλαν ουσιαστικά στην ανάπτυξη καινοτόμων προσεγγίσεων στη διαχείριση αστικών απορριμμάτων και την εφαρμογή τεχνολογιών έξυπνων πόλεων.

- SmartSantander (Ευρωπαϊκή Ένωση):

Πρόκειται για ένα από τα μεγαλύτερα πιλοτικά έργα ανάπτυξης και δοκιμής τεχνολογιών Διαδικτύου των Πραγμάτων (IoT), στο οποίο αναπτύχθηκε εκτεταμένο δίκτυο αισθητήρων για τη συλλογή δεδομένων σε τομείς όπως ο οδοφωτισμός, η διαχείριση στάθμευσης και τα αστικά απορρίμματα.

- Waste4Think (Horizon 2020):

  Το συγκεκριμένο έργο προώθησε την υιοθέτηση πρακτικών κυκλικής οικονομίας μέσα από την αξιοποίηση ψηφιακών τεχνολογιών και την ενεργή συμμετοχή των πολιτών σε διαδικασίες διαχείρισης απορριμμάτων.

- R4R – Regions for Recycling:

  Στόχος του έργου αποτέλεσε η ανάλυση και συγκριτική αξιολόγηση των συστημάτων ανακύκλωσης σε δέκα ευρωπαϊκές περιφέρειες, με στόχο την ανάδειξη βέλτιστων πρακτικών και τη βελτίωση των σχετικών επιδόσεων.

- LIFE-IP CEI-Greece (Ελλάδα):

  Το έργο αυτό επικεντρώνεται στην εφαρμογή αρχών κυκλικής οικονομίας σε επίπεδο δήμων, δίνοντας ιδιαίτερη έμφαση στη βέλτιστη διαχείριση αποβλήτων, μέσω ολοκληρωμένων στρατηγικών και καινοτόμων παρεμβάσεων.

2.11.2. Εμπορικές Λύσεις Διαχείρισης Αστικών Απορριμμάτων με Έξυπνες Τεχνολογίες

Στο πλαίσιο της σύγχρονης διαχείρισης αστικών απορριμμάτων, η διεθνής αγορά προσφέρει μια σειρά από εμπορικές λύσεις που αξιοποιούν τεχνολογίες αιχμής με στόχο τη βελτιστοποίηση των διαδικασιών συλλογής, παρακολούθησης και διαχείρισης των αποβλήτων. Οι εν λόγω λύσεις ενσωματώνουν τεχνολογίες όπως αισθητήρες Διαδικτύου των Πραγμάτων (IoT), συστήματα προγνωστικής ανάλυσης, συμπιεστές απορριμμάτων με ηλιακή ενέργεια, καθώς και πλατφόρμες διαχείρισης στόλου και κάδων. Ακολουθεί συνοπτική παρουσίαση βασικών εμπορικών λύσεων, όπως αυτές διαμορφώνονται στον παρακάτω πίνακα:

Εταιρεία / Λύση	**Τεχνολογία**	**Χαρακτηριστικά**
Enevo	IoT αισθητήρες & προγνωστική ανάλυση	Προγνωστική εκτίμηση πληρότητας κάδων και δυναμική δρομολόγηση οχημάτων συλλογής

Bigbelly	Συμπιεστές απορριμμάτων με ηλιακή ενέργεια	Αυτόματη συμπίεση απορριμμάτων και παρακολούθηση μέσω IoT
SmartBin	Υπερηχητικοί αισθητήρες	Συλλογή δεδομένων σε cloud και βελτιστοποίηση δρομολογίων συλλογής
Evreka	Διαχείριση στόλου & παρακολούθηση κάδων	Ενιαία πλατφόρμα παρακολούθησης για οχήματα και κάδους απορριμμάτων
BH Technologies (Γαλλία)	Ραδιοσυχνικοί αισθητήρες	Παρακολούθηση πληρότητας σε πραγματικό χρόνο και αυτόματες ειδοποιήσεις

Πίνακας 1 2.11.2 Πίνακας εταιρειών παροχής εμπορικών λύσεων σε IoT πλατφόρμες

Οι παραπάνω λύσεις συμβάλλουν σημαντικά στη βελτίωση της αποτελεσματικότητας και της βιωσιμότητας των συστημάτων διαχείρισης απορριμμάτων, προσφέροντας εργαλεία για την αυτοματοποίηση διαδικασιών, την εξοικονόμηση πόρων και την ενίσχυση της διαφάνειας στη λειτουργία των δημοτικών υπηρεσιών.

2.11.3. Συγκριτική Αποτίμηση Υφιστάμενων Λύσεων

Για την αξιολόγηση των υφιστάμενων τεχνολογικών λύσεων στη διαχείριση απορριμμάτων, υιοθετήθηκαν τα ακόλουθα κριτήρια σύγκρισης:

- Κόστος εγκατάστασης και λειτουργίας
- Αξιοπιστία αισθητήρων
- Υποστήριξη αλγορίθμων δρομολόγησης οχημάτων (Vehicle Routing Problem - VRP) και προγνωστικών μοντέλων
- Διασύνδεση με εργαλεία γεωγραφικών πληροφοριακών συστημάτων (GIS) και επιχειρηματικής ευφυΐας (BI)
- Επεκτασιμότητα και συμβατότητα με πρόσθετες υπηρεσίες έξυπνης πόλης (Smart City)
- Δυνατότητα αποστολής ειδοποιήσεων σε πραγματικό χρόνο (real-time alerts)
- Ευκολία χρήσης από το προσωπικό των Οργανισμών Τοπικής Αυτοδιοίκησης (ΟΤΑ)

Συνολικά, οι διεθνείς πλατφόρμες παρέχουν ώριμες και δοκιμασμένες λύσεις, οι οποίες καλύπτουν σε μεγάλο βαθμό τις σύγχρονες απαιτήσεις των συστημάτων διαχείρισης απορριμμάτων. Παρ' όλα αυτά, το αυξημένο κόστος υλοποίησης και συντήρησης, καθώς και η ανάγκη εξειδικευμένης τεχνικής υποστήριξης, συχνά συνιστούν σημαντικά εμπόδια για την υιοθέτηση τους από μικρούς και μεσαίους ελληνικούς δήμους, όπως ο Δήμος Γαζίου. Ως εκ τούτου, καθίσταται αναγκαία η διερεύνηση λύσεων που να ανταποκρίνονται στις ιδιαίτερες ανάγκες και δυνατότητες των συγκεκριμένων ΟΤΑ, διασφαλίζοντας ταυτόχρονα τη βιωσιμότητα και την αποτελεσματικότητα των παρεμβάσεων στον τομέα της αστικής διαχείρισης απορριμμάτων.

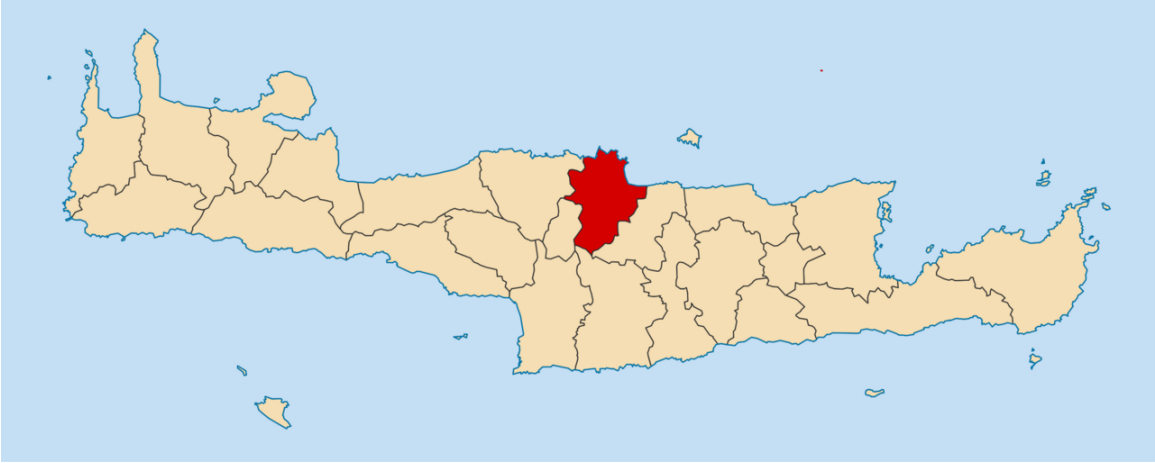
2.12. Συμπεράσματα Κεφαλαίου

Το θεωρητικό πλαίσιο που αναπτύχθηκε στο παρόν κεφάλαιο αναδεικνύει ότι οι «Έξυπνες Πόλεις» αποτελούν το πλέον σύγχρονο μοντέλο αστικής ανάπτυξης, βασιζόμενο στη χρήση τεχνολογιών όπως το IoT, τα Big Data, τα GIS και το cloud computing. Η διαχείριση απορριμμάτων αναδεικνύεται ως ένας από τους πιο σημαντικούς τομείς εφαρμογής των τεχνολογικών αυτών λύσεων, καθώς συνδέεται στενά με την περιβαλλοντική βιωσιμότητα, τη δημόσια υγεία, την ποιότητα ζωής και τη λειτουργική αποτελεσματικότητα των δήμων. Η Ελλάδα βρίσκεται σε στάδιο μετάβασης, με σημαντικούς Οργανισμούς Τοπικής Αυτοδιοίκησης να υιοθετούν έξυπνες λύσεις, παρά τις υφιστάμενες προκλήσεις. Η παρούσα ερευνητική προσέγγιση, με επίκεντρο τον Δήμο Γαζίου, συμβάλλει στην εις βάθος κατανόηση της εν εξελίξει διαδικασίας και στην αναζήτηση βέλτιστων πρακτικών για την περαιτέρω βελτιστοποίηση της συλλογής και διαχείρισης των αστικών απορριμμάτων.

3. Case Study – Δήμος Μαλεβιζίου με έδρα το Γάζι της Περιφέρειας Κρήτης

3.1. Περιγραφή Δήμου – Διοικητικά και Χωροταξικά

Ο Δήμος Μαλεβιζίου έχει συνολική έκταση 291,907 τ. χλμ., χωροθετείται δυτικά του Πολεοδομικού Συγκροτήματος Ηρακλείου και συνορεύει στα ανατολικά με το Δήμο Ηρακλείου, στα νοτιοδυτικά με το Δήμο Ανωγείων, στα νότια με τους Δήμους Ηρακλείου και Γόρτυνα και στα δυτικά με το Δήμο Μυλοποτάμου ενώ στα βόρεια, βρέχεται από το Κρητικό Πέλαγος. Διοικητικά υπάγεται στο Νομό Ηρακλείου. Σύμφωνα με την απογραφή της ΕΛ.ΣΤΑΤ. του 2021, ο πραγματικός πληθυσμός του Δήμου ανέρχεται σε 25.734 κατοίκους αυξημένος κατά 870 κατοίκους από την απογραφή της ΕΛ.ΣΤΑΤ του 2011.



Εικόνα 1 3.1 Τοποθεσία του Δήμου Μαλεβιζίου στη Κρήτη

Ο Δήμος Μαλεβιζίου είναι δήμος της Περιφέρειας Κρήτης με έδρα το Γάζι, δημιουργήθηκε την 01.01.2011, με το Ν.3852/4.6.2010 (ΦΕΚ 87Α/7-6-2010) "Νέα Αρχιτεκτονική της Αυτοδιοίκησης και της Αποκεντρωμένης Διοίκησης – Πρόγραμμα Καλλικράτης". Συστάθηκε επομένως με το Πρόγραμμα Καλλικράτης και προέκυψε από τη συνένωση των προϋπάρχοντων δήμων Γαζίου, Κρουσώνα και Τυλίσου.

3.2. Βασικά Χαρακτηριστικά του Δήμου

Το Γάζι χωροθετείται σε απόσταση 7 χλμ από το Ηράκλειο. Η ευρύτερη περιοχή του Γαζίου συνιστά σημαντικό κυκλοφοριακό κόμβο του Βόρειου Οδικού Άξονα της Κρήτης (ΒΟΑΚ), καθώς συνδέει τους Νομούς Ηρακλείου και Ρεθύμνου αλλά και τα βόρεια παράλια με την ενδοχώρα και τον Νότιο Οδικό Άξονα του νησιού.

Σημαντικό χαρακτηριστικό του Δήμου Μαλεβιζίου είναι ότι αποτελεί περιοχή ιδιαίτερης ενεργειακής σημασίας καθώς στο Δημοτικό Διαμέρισμα της Ροδιάς βρίσκεται εγκατεστημένο το εργοστάσιο παραγωγής ηλεκτρικού ρεύματος της ΔΕΗ. Μεγάλο μέρος των ενεργειακών αναγκών ολόκληρου του νησιού εξυπηρετούνται από το συγκεκριμένο εργοστάσιο, το οποίο ταυτόχρονα αποτελεί σημαντική βιομηχανική εγκατάσταση στην περιοχή αλλά και σε ολόκληρο το Νομό.

Η συγκέντρωση σημαντικών βιομηχανικών αλλά και βιοτεχνικών μονάδων, καθώς και η έντονη τουριστική ανάπτυξη που συγκεντρώνεται στο παράκτιο τμήμα της περιοχής, καθιστά σημαντικό το ρόλο του Δήμου Μαλεβιζίου στη γενικότερη οικονομική ανάπτυξη του Νομού Ηρακλείου. Επιπρόσθετα, λόγω των νέων τάσεων στην τουριστική αγορά και τη στροφή των τουριστών τα τελευταία χρόνια σε στις εναλλακτικές μορφές τουρισμού - πέραν του ήλιου και της θάλασσας- παρατηρείται αύξηση της ροής των επισκεπτών και των διανυκτερεύσεων στο Δήμο Μαλεβιζίου, όπως άλλωστε συμβαίνει σε όλη τη χώρα.

Το οικοσύστημα του Αλμυρού ποταμού έχει επίσης ιδιαίτερη σημασία για την περιοχή. Ένας από τους δύο σημαντικότερους υγροτόπους του Ηρακλείου είναι ο υγρότοπος του Αλμυρού, ο οποίος μαζί με το φαράγγι αποτελούν σημαντική τοποθεσία για ολόκληρη την Κρήτη. Η περιοχή αποτελεί σημαντικό πόλο έλξης επισκεπτών και «πνεύμονα» πρασίνου και αναψυχής για τους επισκέπτες αλλά και τους κατοίκους των γειτονικών Δήμων. Μεγάλη σημασία έχει και το Δάσος της Κέρης όπου αναδασώθηκε από τον πρώην Δήμο Τυλίσου με σκοπό να αποτελέσει πνεύμονα πρασίνου.

Οι δύο περιοχές Natura, Κρουσώνας – Βρωμόνερο Ίδης (κωδικό GR4310009) και Ίδη Όρος (κωδικό GR 4330005) οι οποίες χαρακτηρίζονται από ποικιλομορφία τοπίων υψηλής βιοποικιλότητας, σπάνιες φυτικές διαπλάσεις, την παρουσία ενδημικών ειδών πανίδας και χλωρίδας της Κρήτης και της Ελλάδας, σπάνια και απειλούμενα είδη, είναι ιδιαίτερης σπουδαιότητας για την περιοχή.

Δυνατά εντόπια στοιχεία συνιστούν πολιτιστικοί πόροι όπως π.χ. το Μουσείο του Δομίνικου Θεοτοκόπουλου, η Μονή Σαββαθιανών, το Μνημείο Κερατίδι Δαμάστας, η Μονή του Αγίου Παντελεήμονα, ο αρχαιολογικός χώρος στο ακρωτήρι Σούδα της Αγίας Πελαγίας, το μοναστήρι της Παναγίας στα Φρασκιά, ο αρχαιολογικός χώρος Τυλίσου, η Δημοτική Πινακοθήκη, η αρχαιολογική συλλογή Μαλεβιζίου, η έκθεση του Γ. Κλινάκη στο

Μάραθος, τα μεσαιωνικά πατητήρια κ.α. οι οποίοι συγκεντρώνονται στα όρια του Δήμου αλλά και στην ευρύτερη από αυτόν περιοχή. Η ύπαρξη όλων αυτών των πολιτιστικών πόρων, συμβάλει στην ενίσχυση της ανάπτυξης του Δήμου. [1 https://malevizi.gov.gr/malevizi-map/]

3.3. Δημοτικές Ενότητες Δήμου Μαλεβιζίου

Ο Δήμος Μαλεβιζίου αποτελείται από τις ακόλουθες δημοτικές ενότητες και δημοτικά διαμερίσματα:

Δημοτική Ενότητα Γαζίου

- Δ.Δ. Γαζίου
- Δ.Δ. Αχλάδας
- Δ.Δ. Καλεσιών
- Δ.Δ. Ροδιάς
- Δ.Δ. Φόδελε

Δημοτική Ενότητα Κρουσώνα

- Δ.Δ. Κρουσώνος
- Δ.Δ. Κορφών
- Δ.Δ. Λουτρακίου
- Δ.Δ. Σάρχου

Δημοτική Ενότητα Τυλισού

- Δ.Δ. Τυλισού
- Δ.Δ. Αηδονοχωρίου
- Δ.Δ. Αστυρακίου
- Δ.Δ. Γωνιών Μαλεβιζίου
- Δ.Δ. Δαμάστας
- Δ.Δ. Καμαρίου
- Δ.Δ. Καμαριώτου
- Δ.Δ. Κεραμουτσίου
- Δ.Δ. Μαράθου
- Δ.Δ. Μονής

3.4. Οικονομική Κατάσταση Δήμου

Μελετώντας τη διεθνή, τοπική του δήμου Μαλεβιζίου βιβλιογραφία καθώς και τα διάφορα Ευρωπαϊκά προγράμματα που τέθηκαν σε ισχύ με το σύμφωνο των Δημάρχων όπου πρέπει οι Δήμαρχοι να έχουν εξασφαλίσει μια σημαντική μείωση των απορριμμάτων και σύμφωνα και με τη μελέτη «Τοπικού Σχέδιου Αποκεντρωμένης Διαχείρισης Στερεών Αποβλήτων Δήμου Μαλεβιζίου», του 2016 και μετά από επικοινωνία με φορέα του τμήματος της Υπηρεσίας Καθαριότητας του Δήμου Μαλεβιζίου παρακάτω παραθέτω την οικονομική κατάσταση του δήμου σύμφωνα με τον προϋπολογισμό του 2024 όπου ανέρχεται για το Δήμο Μαλεβιζίου στο ύψος των 50 € εκατομμυρίων όπου παρατηρήθηκε μια αύξηση των 8 € εκατομμυρίων από το οικονομικό έτος 2023. Η οικονομική πορεία του δήμου βρίσκεται σε σταθερή πορεία με πλεόνασμα 1.285 € εκατομμυρίων για το 2024. Επίσης, υπήρξε και μια πρόσφατη χρηματοδότηση του δήμου ύψους 1,2 € εκατομμυρίων από το ΥΠΕΣ για ανανέωση του στόλου απορριμματοφόρων.

Η οικονομική ανάλυση για την εφαρμογή του συστήματος Smart Waste Management που θέλουμε να υλοποιήσουμε στον Δήμο Μαλεβιζίου δείχνει **εξαιρετική εφικτότητα** με Δείκτη Απόδοσης Επένδυσης R.O.I. 187% σε 5 χρόνια και payback period μόλις 16 μήνες. Το έργο προσφέρει ετήσιες εξοικονομήσεις 156.000 €-198.000 € έναντι αρχικής επένδυσης 285.000 €.

3.5. Υφιστάμενη Υπηρεσία Καθαριότητας

Οργανωτική Δομή: Η Διεύθυνση Καθαριότητας & Ανακύκλωσης αποτελείται από αποκεντρωμένα γραφεία σε κάθε δημοτική ενότητα. Το προσωπικό της Υπηρεσίας καθαριότητας απαρτίζεται από 15 οδηγούς απορριμματοφόρων και 35 εργάτες καθαριότητας , καθώς και το Διοικητικό προσωπικό της υπηρεσίας.

Στόλος Οχημάτων: Τον Οκτώβριο του 2024, ολοκληρωτική καταστροφή υπέστη ο στόλος απορριμματοφόρων του Δήμου Μαλεβιζίου μετά από πυρκαγιά που ξέσπασε στο χώρο του αμαξοστασίου, προκαλώντας έκτακτη ανάγκη στη διαχείριση απορριμμάτων. Η κατάσταση αυτή ενεργοποίησε κύμα συμπαράστασης από άλλους δήμους της περιοχής για την παροχή οχημάτων. Η ζημιά από την καταστροφή του στόλου ανήλθε σε 2 εκατομμύρια €. Πρόσφατα ενισχύθηκε με 3 νέα οχήματα (2 δορυφορικά απορριμματοφόρα). Επίσης, υπάρχει προγραμματισμός αγοράς 5 επιπλέον απορριμματοφόρων - Ιστορικό: Καταστροφή στόλου από πυρκαγιά (ζημιά 2 εκατομμυρίων). Η λειτουργία της υπηρεσίας καθαριότητας είναι με υπερωριακή εργασία.

3.6. Ανάλυση Υφιστάμενου Κόστους

3.6.1. Εκτίμηση Ετήσιου Κόστους

Βασισμένη σε συγκριτικά στοιχεία από παρόμοιους δήμους της Κρήτης:

Κόστος Προσωπικού (ετήσια):

15 οδηγοί × 28.000€ = 420.000 €

35 εργάτες × 24.000€ = 840.000 €

Διοικητικό προσωπικό (5 άτομα) × 32.000€ = €160.000

Υπερωρίες και επιδόματα: 180.000 €

Συνεπώς το **Συνολικό κόστος προσωπικού ανέρχεται στις 1.600.000 €**

Κόστος Καυσίμων και Συντήρησης:

Καύσιμα (εκτίμηση 120.000L/έτος × 1.45€): 174.000 €

Συντήρηση οχημάτων: 85.000 €

Ανταλλακτικά και επισκευές: 65.000 €

Συνολικό κόστος λειτουργίας: 324.000 €

Λοιπά Κόστη: - Διαχείριση απορριμμάτων (ΕΣΔΑΚ): 280.000€ - Εξοπλισμός και αναλώσιμα: €45.000 - Ασφάλιστρα και λοιπά: 35.000€ - **Συνολικά λοιπά κόστη: 360.000€**

ΣΥΝΟΛΙΚΟ ΕΤΗΣΙΟ ΚΟΣΤΟΣ: 2.284.000 €

Κόστος ανά Κάτοικο

2.284.000€ ÷ 25.734 κάτοικοι = **88.8 € ανά κάτοικο ετησίως**

Το κόστος είναι εντός των αναμενόμενων ορίων για δήμους παρόμοιου μεγέθους στην Κρήτη.

3.7. Σχεδιασμός Smart Waste System για το δήμο Μαλεβιζίου

3.7.1. Εκτίμηση Αριθμού Κάδων

Κατανομή κάδων ανά περιοχή:

Αστικές Περιοχές (Γάζι, Αμμουδάρα, Αγία Πελαγία): Στις Πυκνοδομημένες περιοχές τοποθετείται κάδος απορριμμάτων περίπου ένας (1) κάδος ανά 150-200m – Εκτίμηση που μου έδωσαν για το πόσοι κάδοι υπάρχουν σε αυτές τις περιοχές είναι **180 κάδοι 1100L**.

Ημιαστικές Περιοχές (Τύλισος, μεγάλα χωριά): Σε αυτές τις περιοχές έχει διατηρηθεί μια μέση πυκνότητα τοποθέτησης κάδων απορριμμάτων περίπου ένας (1) κάδος ανά 300-400m, έτσι η εκτίμηση που μου έδωσαν για το πόσοι κάδοι υπάρχουν σε αυτές τις ημιαστικές περιοχές είναι **85 κάδοι 1100L**.

Αγροτικές Περιοχές (μικροί οικισμοί): Στους μικρούς οικισμούς του δήμου Μαλεβιζίου υπάρχει μια χαμηλή πυκνότητα τοποθέτησης κάδων περίπου ένας (1) κάδος ανά 500-800m με εκτίμηση περίπου στους **45 κάδους 1100L.**

Τουριστικές Ζώνες (παραλίες, κεντρικοί δρόμοι): Στην τουριστική ζώνη του δήμου Μαλεβιζίου υπάρχει μια υψηλή συχνότητα τοποθέτησης κάδων όπου αντιστοιχεί περίπου ένας (1) κάδος ανά 100-150m με εκτιμώμενη τοποθέτηση 65 κάδοι 1100L.

ΣΥΝΟΛΙΚΟΣ ΑΡΙΘΜΟΣ ΚΑΔΩΝ: 375 κάδοι

3.8. Τεχνικές Προδιαγραφές Συστήματος

Οι τεχνικές προδιαγραφές του συστήματος που ενδεικτικά θα μπορέσουμε να υλοποιήσουμε στο δήμο Μαλεβιζίου είναι να τοποθετηθούν SmartBin Συσκευές: 375 όσοι είναι οι κάδοι απορριμμάτων με πλήρη εξοπλισμό αισθητήρων. Η **LoRaWAN Υποδομή**: Για την κάλυψη όλου του δήμου θα χρειαστούμε 8 gateways. Ένα κεντρικό gateway στο Δημαρχείο Γαζίου θα υπάρχει στο δημαρχείο Γαζίου και τα υπόλοιπα δευτερεύοντα gateways σε στρατηγικά σημεία. Μια **Cloud Platform** όπου θα υλοποιήσουμε μια ολοκληρωμένη πλατφόρμα με web dashboard και mobile εφαρμογές. Και τέλος θα υπάρχει και μια **ενσωμάτωση** και σύνδεση με υπάρχοντα συστήματα του δήμου.

3.9. Λεπτομερής Ανάλυση Κόστους Επένδυσης

3.9.1. Κόστος Hardware

SmartBin Συσκευές: Το κόστος των συσκευών σύμφωνα θα είναι 375 συσκευές × 175€ (μέσος όρος) = 65.625€ . Η Εγκατάσταση και παραμετροποίηση υπολογίζεται να κοστίσει: €18.750 . Συνεπώς το **Συνολικό κόστος των SmartBin θα ανέλθει στα: €84.375**

LoRaWAN Υποδομή: Αφού έχουμε υπολογίσει ότι θα χρειαστούμε 8 gateways το κόστος θα κυμανθεί στα **8 gateways × €1.200 = €9.600** . Η Εγκατάσταση και παραμετροποίηση: €8.000 και ο κεντρικός εξοπλισμός δικτύου: **€12.000 Συνολικό κόστος δικτύου: €29.600**

Άρα το Συνολικό Hardware θα κυμανθεί στα €113.975

3.9.2. Κόστος Software

Ανάπτυξη Εφαρμογών: - Αν η πλατφόρμα δοθεί σε εξωτερικό συνεργάτη υλοποίησης το κόστος θα είναι περίπου ManageBins Web Platform: €55.000 – Αν υπολογίσουμε ότι ίσως να χρειαστεί μια εφαρμογή για κινητές συσκευές τότε το κόστος ενδεικτικά μπορεί να υπολογστεί **SmartGarbageTruck Android App**: €30.000 και υπολογίζοντας το **Backend APIs και integrations**: €40.000 τότε το **Συνολικό κόστος ανάπτυξης της εφαρμογής ανέρχεται στα €125.000**

Customization (προσαρμογή) για το δήμο Μαλεβιζίου: Για να μπορέσει να προσαρμοστεί στις ανάγκες του δήμου υπολογίζουμε ένα ενδεικτικό κόστος στις: €15.000 και να ενσωματωθεί με υπάρχοντα συστήματα που μπορεί να έχει ο δήμος υπολογίζεται σε: €12.000 κι αν υπολογίσουμε και τους ελέγχους και την πιλοτική ανάπτυξη της εφαρμογής μπορεί να κοστίσει: €8.000 . Επομένως, το **Συνολικό κόστος προσαρμογής υπολογίζεται στα: €35.000**

Τέλος το Συνολικό κόστος ανάπτυξης του Software ανέρχεται στα €160.000

3.9.3. Κόστος Υλοποίησης

Το κόστος Υλοποίησης του Project Management υπολογίζουμε για το δήμο ότι θα είναι : €8.000 . Η εκπαίδευση του προσωπικού υπολογίζεται ότι θα στοιχίσει στο δήμο περίπου (**Training προσωπικού)**: €6.000 **Documentation**: €4.000 **Contingency (5%)**: €11.000

Συνολικό κόστος υλοποίησης: €29.000

3.9.4. Συνολική Αρχική Επένδυση

Σύμφωνα με τους προηγούμενους υπολογισμούς το κόστος επένδυσης σε **Hardware ανέρχεται στα**: €113.975 και το κόστος επένδυσης στο Λογισμικό Εφαρμογής ανέρχεται **Software στα** : €160.000 και τέλος το κόστος της **Υλοποίησης υπολογίσαμε ότι ανέρχεται στα**: €29.000 . Άρα το σύνολο της αρχικής επένδυσης **είναι €302.975 και το**

στρογγυλοποιούμε για την οικονομική ανάλυση της εφικτότητας σε συνολικό κόστος αρχικής επένδυσης στα *€305.000.*

3.10. Ετήσια Κόστη Λειτουργίας

Φυσικά θα πρέπει να υπολογίσουμε και τα ετήσια κόστη λειτουργίας της ανάπτυξης αυτής της εφαρμογής.

3.10.1. Cloud και Τηλεπικοινωνίες

Cloud Hosting: €12.000/έτος **LoRaWAN Network**: €8.000/έτος **Data Storage και Analytics**: €6.000/έτος **Συνολικό κόστος cloud: €26.000/έτος**

3.10.2. Συντήρηση και Support

Hardware Maintenance: €15.000/έτος **Software Updates**: €8.000/έτος **Technical Support**: €12.000/έτος **Συνολικό κόστος συντήρησης: €35.000/έτος**

Αντικαταστάσεις

Μπαταρίες (5% ετησίως): €3.200/έτος **Αισθητήρες (2% ετησίως)**: €2.600/έτος **Συνολικό κόστος αντικαταστάσεων: €5.800/έτος**

ΣΥΝΟΛΙΚΟ ΕΤΗΣΙΟ ΚΟΣΤΟΣ ΛΕΙΤΟΥΡΓΙΑΣ: €66,800

3.10.3. Ανάλυση Εξοικονομήσεων

3.10.3.1. Μείωση Κόστους Καυσίμων

Βελτιστοποίηση Διαδρομών: - Μείωση άδειων διαδρομών κατά 35% - Εξοικονόμηση καυσίμων: 42.000L × €1.45 = €60.900/έτος

Προγραμματισμένη Συλλογή: - Μείωση έκτακτων διαδρομών κατά 60% - Εξοικονόμηση καυσίμων: 18.000L × €1.45 = €26.100/έτος

Συνολική εξοικονόμηση καυσίμων: €87,000/έτος

3.10.3.2. Μείωση Κόστους Προσωπικού

Βελτιστοποίηση Εργατοωρών: - Μείωση υπερωριών κατά 40%: €72.000/έτος - Αποδοτικότερος προγραμματισμός: €25.000/έτος

Συνολική εξοικονόμηση προσωπικού: €97.000/έτος

3.10.3.3. Μείωση Κόστους Συντήρησης

Προληπτική Συντήρηση Οχημάτων: - Μείωση βλαβών κατά 25%: €21.250/έτος - Αύξηση διάρκειας ζωής οχημάτων: €15.000/έτος

Συνολική εξοικονόμηση συντήρησης: €36.250/έτος

3.10.3.4. Λοιπές Εξοικονομήσεις

Μείωση Παραπόνων Πολιτών: - Λιγότερο διοικητικό κόστος: €8.000/έτος

Βελτίωση Ανακύκλωσης: - Αύξηση εσόδων από ανακυκλώσιμα: €12.000/έτος

Συνολικές λοιπές εξοικονομήσεις: €20.000/έτος

3.11. Συνολική Οικονομική Ανάλυση

3.11.1. Ετήσιες Εξοικονομήσεις

Καύσιμα: €87.000 **Προσωπικό**: €97.000

Συντήρηση: €36.250 **Λοιπά**: €20.000 **ΣΥΝΟΛΟ**: **€240.250/έτος**

3.11.2. Καθαρές Εξοικονομήσεις

Ετήσιες εξοικονομήσεις: € 240.250 **Μείον ετήσιο κόστος λειτουργίας**: €66.800 **ΚΑΘΑΡΕΣ ΕΤΗΣΙΕΣ ΕΞΟΙΚΟΝΟΜΗΣΕΙΣ**: **€173.450**

3.11.3. ROI Ανάλυση

Αρχική επένδυση: €305.000 **Ετήσιες καθαρές εξοικονομήσεις**: €173.450

Payback Period: €305.000 ÷ €173.450 = **1.76 έτη (21 μήνες)**

5-Year ROI: - Συνολικές εξοικονομήσεις 5 ετών: €173.450 × 5 = €867.250 - Μείον αρχική επένδυση: 867.250 € - €305.000 = 562.250€ - **ROI = 184%**

NPV (8% discount rate, 5 έτη): 387.420€

3.12. Ανάλυση Ευαισθησίας

3.12.1. Συντηρητικό Σενάριο (-20% εξοικονομήσεις)

Ετήσιες καθαρές εξοικονομήσεις: 138.760€ **Payback Period**: 2.2 έτη (26 μήνες) **5-Year ROI**: 127%

3.12.2. Αισιόδοξο Σενάριο (+20% εξοικονομήσεις)

Ετήσιες καθαρές εξοικονομήσεις: 208.140€ **Payback Period**: 1.47 έτη (18 μήνες) **5-Year ROI**: 241%

3.13. Ανάλυση Κινδύνων

Τεχνικοί Κίνδυνοι: Χαμηλοί (τεχνολογία αποδεδειγμένη) **Οικονομικοί Κίνδυνοι**: Μέτριοι (εξάρτηση από τιμές καυσίμων) **Λειτουργικοί Κίνδυνοι**: Χαμηλοί (απλή υιοθέτηση)

3.14. Χρηματοδότηση και Υλοποίηση

3.14.1. Πηγές Χρηματοδότησης

Ίδιοι Πόροι Δήμου: 150.000€ (49%) - Διαθέσιμα από πλεόνασμα 2024

Ευρωπαϊκά Κονδύλια: 100.000€ (33%) - ΕΣΠΑ 2021-2027 Smart Cities - Horizon Europe προγράμματα

Εθνική Χρηματοδότηση: 55.000€ (18%) - ΥΠΕΣ προγράμματα ψηφιοποίησης - Πράσινο Ταμείο

3.15. Προτεινόμενο Χρονοδιάγραμμα

Φάση 1 (3 μήνες): Εξασφάλιση χρηματοδότησης και διαγωνισμοί **Φάση 2 (6 μήνες)**: Ανάπτυξη και pilot testing (50 κάδοι) **Φάση 3 (4 μήνες)**: Πλήρης ανάπτυξη (325 επιπλέον κάδοι) **Φάση 4 (2 μήνες)**: Training και go-live

Συνολική διάρκεια: 15 μήνες

3.16. Οφέλη για τον Δήμο Μαλεβιζίου

3.16.1. Οικονομικά Οφέλη

Άμεσα: - Εξοικονόμηση 173.450€ ετησίως - Μείωση λειτουργικών κοστών κατά 7.6% - Βελτίωση cash flow

Μακροπρόθεσμα: - Συνολική εξοικονόμηση 867.250€ σε 5 χρόνια - Αύξηση αξίας δημοτικής περιουσίας - Δυνατότητα επέκτασης σε άλλους τομείς

3.16.2. Περιβαλλοντικά Οφέλη

Μείωση Εκπομπών: - 35% λιγότερες διαδρομές = -42 τόνοι CO2/έτος - Βελτίωση ποιότητας αέρα

Βελτίωση Ανακύκλωσης: - Παρακολούθηση τύπων απορριμμάτων - Αύξηση ποσοστών ανακύκλωσης κατά 15-20%

3.16.3. Κοινωνικά Οφέλη

Βελτίωση Ποιότητας Ζωής: - Καθαρότερες περιοχές - Μείωση οσμών και παρασίτων - Καλύτερη εικόνα τουριστικών περιοχών

Τεχνολογική Πρωτοπορία: - Πρώτος δήμος στην περιοχή με smart waste - Προσέλκυση επενδύσεων και τουρισμού

3.17. Συγκριτική Ανάλυση

3.17.1. Σύγκριση με Άλλους Δήμους

Δήμος Ρεθύμνου (παρόμοιος μέγεθος): - Κόστος διαχείρισης: 3.137.643€ (2024) - Κόστος/κάτοικο: 95.2€ (υψηλότερο από Μαλεβίζι)

Δήμος Ηρακλείου (μεγαλύτερος): - Υλοποιεί ήδη smart waste projects - Αποδεδειγμένα οφέλη 25-30% εξοικονόμηση

3.17.2. Ανταγωνιστικά Πλεονεκτήματα

Γεωγραφικά: - Συμπαγής περιοχή = καλύτερη κάλυψη LoRaWAN - Μικτό αστικό-αγροτικό περιβάλλον = ιδανικό test case

Οικονομικά: - Υγιής οικονομική κατάσταση - Διαθέσιμα κονδύλια από πλεόνασμα

Τεχνολογικά: - Υπάρχουσα ψηφιακή υποδομή - Έμπειρο τεχνικό προσωπικό

Συστάσεις και Επόμενα Βήματα

3.18. Άμεσες Ενέργειες

1. **Έγκριση από Δημοτικό Συμβούλιο**: Παρουσίαση business case
2. **Αίτηση Χρηματοδότησης**: Υποβολή σε ΕΣΠΑ και άλλα προγράμματα
3. **Pilot Program**: Ξεκίνημα με 50 κάδους στο Γάζι
4. **Διαγωνισμός Προμηθευτών**: Επιλογή τεχνολογικών εταίρων

3.19. Μακροπρόθεσμο Όραμα του Δήμου Μαλεβιζίου

Το μακροπρόθεσμο όραμα του δήμου είναι να ανανεωθεί σε έξυπνη πόλη (Smart City) χρησιμοποιώντας και επεκτείνοντας το έργο που θα ξεκινήσει με τη έξυπνη αποκομιδή απορριμμάτων κάνοντας χρήση νέων εφαρμογών IoT όπως είναι ο έξυπνος φωτισμός σε σπίτια και πόλεις, το έξυπνο Parking, οι χάρτες αστικού θορύβου, η ρύθμιση κυκλοφορίας, οι έξυπνοι δρόμοι, και οι έξυπνες ψηφιακές υπηρεσίες πολιτών.

Μοντέλα επικοινωνίας IoT (Internet of Things)

- **Device-to-Device (Συσκευή σε συσκευή):** άμεση επικοινωνίας μεταξύ δύο συσκευών συνήθως μέσω Bluetooth, Wifi, ZigBee, όπου ο ρυθμός μετάδοσης δεδομένων δεν απαιτείται να είναι υψηλός. Π.χ. εφαρμογή σε κλειδαριές, λαμπτήρες, smartwatch κ.α.

- **Device-to-Cloud (Συσκευή σε Internet cloud):** επικοινωνία συσκευή με Interet Cloud για ενημερώσεις συσκευής, διαχείριση της ή μεταφορά δεδομένων από συσκευή σε εφαρμογή ή βάση δεδομένων στο Internet.

- **Device-to-Gateway (Συσκευή σε πύλη):** επικοινωνία συσκευής μέσω πύλης (συσκευή ή λογισμικό) που αποκαθιστά την επικοινωνία της συσκευής με το Internet Cloud. Χρησιμοποιείται συνήθως για επιπλέον ασφάλεια.

- **Device-to-Data Sharing (Συσκευή σε εφαρμογή Cloud):** επικοινωνία συσκευής σε εφαρμογή cloud με δυνατότητα κοινοποίησης δεδομένων από αισθητήρες της συσκευής.

3.20. Συμπεράσματα

Το έργο Smart Waste Management για τον Δήμο Μαλεβιζίου παρουσιάζει **εξαιρετική οικονομική εφικτότητα** με:

- **ROI 184%** σε 5 χρόνια
- **Payback period 21 μήνες**
- **Ετήσιες εξοικονομήσεις 173.450 €**
- **Χαμηλό ρίσκο** υλοποίησης

Η **σταθερή οικονομική κατάσταση** του δήμου, η **υπάρχουσα τεχνολογική υποδομή**, και η **τουριστική ανάπτυξη** δημιουργούν ιδανικές συνθήκες για επιτυχημένη υλοποίηση. **Άμεση προώθηση του έργου** με έμφαση στο pilot program για validation των οικονομικών προβλέψεων και στη σταδιακή επέκταση για μεγιστοποίηση των οφελών. Το έργο όχι μόνο θα βελτιώσει την αποδοτικότητα της υπηρεσίας καθαριότητας, αλλά θα τοποθετήσει τον Δήμο Μαλεβιζίου στην πρωτοπορία της τεχνολογικής καινοτομίας στην Κρήτη.

4. Ανάλυση Απαιτήσεων και Σχεδίαση Συστήματος

4.1. Περιγραφή του Συστήματος και των Εμπλεκομένων

Το προτεινόμενο σύστημα αφορά την ανάπτυξη μιας **έξυπνης πλατφόρμας διαχείρισης απορριμμάτων** για τον Δήμο Γαζίου, με στόχο τη βελτιστοποίηση της διαδικασίας αποκομιδής, τη μείωση του λειτουργικού κόστους και τη βελτίωση της περιβαλλοντικής απόδοσης. Το σύστημα αξιοποιεί τεχνολογίες IoT, ανάλυσης δεδομένων και υποστήριξης αποφάσεων.

Εμπλεκόμενοι Φορείς (Stakeholders)

- **Δήμος Γαζίου (Διοίκηση):**
  Υπεύθυνος για τον στρατηγικό σχεδιασμό, την πολιτική απορριμμάτων και τη λήψη αποφάσεων.
- **Υπάλληλοι Καθαριότητας:**
  Χρήστες του συστήματος για παρακολούθηση δρομολογίων, ειδοποιήσεων και επιχειρησιακής πληροφόρησης.
- **Διαχειριστές Συστήματος (Administrators):**
  Υπεύθυνοι για τη ρύθμιση, τη συντήρηση και την ασφάλεια της πλατφόρμας.
- **Πολίτες:**
  Έμμεσοι χρήστες που επωφελούνται από την καλύτερη ποιότητα υπηρεσιών και ενδεχομένως αλληλεπιδρούν μέσω εφαρμογής (αναφορές, ειδοποιήσεις).

4.2. Ανάλυση Απαιτήσεων

Η ανάλυση απαιτήσεων αποτελεί κρίσιμο στάδιο ώστε το σύστημα να ανταποκρίνεται στις πραγματικές ανάγκες του δήμου.

4.2.1. Λειτουργικές Απαιτήσεις

- Παρακολούθηση πληρότητας κάδων σε πραγματικό χρόνο
- Αυτόματη αποστολή ειδοποιήσεων υπερπλήρωσης
- Διαχείριση στόλου απορριμματοφόρων
- Δυναμικός σχεδιασμός και αναπροσαρμογή δρομολογίων

- Καταγραφή ιστορικών δεδομένων αποκομιδής
- Προβολή δεικτών απόδοσης (KPIs)
- Διαχείριση χρηστών και ρόλων
- Αναφορές και στατιστικά στοιχεία
- Υποστήριξη αναφορών από πολίτες (προαιρετικά)

4.2.2. Μη Λειτουργικές Απαιτήσεις

- **Απόδοση:** Απόκριση συστήματος < 2 δευτερόλεπτα
- **Επεκτασιμότητα:** Δυνατότητα προσθήκης νέων κάδων/αισθητήρων
- **Ασφάλεια:** Έλεγχος πρόσβασης, κρυπτογράφηση δεδομένων
- **Αξιοπιστία:** Υψηλή διαθεσιμότητα συστήματος
- **Διαλειτουργικότητα:** Συμβατότητα με GIS και άλλα δημοτικά συστήματα
- **Χρηστικότητα:** Φιλικό περιβάλλον για μη εξειδικευμένους χρήστες

4.3. Αρχιτεκτονική του Συστήματος

Το σύστημα σχεδιάζεται με **πολυεπίπεδη αρχιτεκτονική (layered architecture)**:

Επίπεδο Αισθητήρων (IoT Layer)

- Αισθητήρες πληρότητας κάδων (ultrasonic)
- Μικροελεγκτές (Arduino / ESP32)
- Επικοινωνία μέσω MQTT / LoRaWAN

Επίπεδο Διακομιστή (Backend Layer)

- Διακομιστής εφαρμογής (Node.js ή Python)
- Μηχανισμός συλλογής και επεξεργασίας δεδομένων
- Βάση δεδομένων (NoSQL ή relational)

Επίπεδο Εφαρμογών (Application Layer)

- Web dashboard για δήμο & διαχειριστές

- Mobile εφαρμογή για προσωπικό καθαριότητας
- API για διασύνδεση τρίτων συστημάτων

4.4. Μοντέλο Δεδομένων

Το μοντέλο δεδομένων του Smart Waste Management System υποστηρίζει τη συλλογή, αποθήκευση και ανάλυση πληροφοριών.

Βασικές Οντότητες

- **Bin (Κάδος):** ID, τοποθεσία, πληρότητα, τύπος
- **Sensor:** ID, τύπος, ημερομηνία συντήρησης
- **Vehicle:** ID, χωρητικότητα, κατάσταση
- **Route:** ID, ημερομηνία, απόσταση
- **User:** ID, ρόλος, στοιχεία πρόσβασης
- **Measurement:** χρονική σήμανση, επίπεδο πληρότητας

Οι σχέσεις είναι κυρίως τύπου **1:N** (π.χ. ένας κάδος → πολλές μετρήσεις).

4.5. Διαγράμματα UML

4.5.1. Use Case Diagram

Κύριες περιπτώσεις χρήσης:

- Παρακολούθηση κάδων
- Διαχείριση δρομολογίων
- Λήψη ειδοποιήσεων
- Προβολή αναφορών
- Διαχείριση χρηστών

Το Use Case Diagram όπως σχεδιάστηκε με το Visual Paradigm.



Εικόνα 2 4.5.1 Use Case Diagram

4.5.2. Sequence Diagram

Απεικονίζει:

1. Αισθητήρας → Διακομιστής (αποστολή δεδομένων)
2. Διακομιστής → Βάση Δεδομένων
3. Διακομιστής → Dashboard (ενημέρωση)
4. Σύστημα → Υπάλληλος (ειδοποίηση)

Το Sequence Diagram όπως σχεδιάστηκε με το Visual Paradigm


Sequence Diagram — Πλήρης Ροή
Smart Waste Management — Δήμος Μαλεβιζίου
Αισθητήρας
Διαχειριστής
Οδηγός
Πολίτης
FastAPI
PostgreSQL
Dashboard
PWA Οδηγού
SMTP Email
ΠΧ1: Πιστοποίηση Χρήστη
Σύνδεση (username, password, captcha)
POST /api/v1/auth/login
SELECT users WHERE username=?
user record
JWT Token
Πρόσβαση στο Dashboard
IoT: Αποστολή Μέτρησης
POST /api/v1/sensors/readings
(bin_id, fill_level, battery, lat, lon)
INSERT sensor_data (TimescaleDB)
OK
UPDATE bin_status_summary view
200 OK
Παρακολούθηση Κάδων
Προβολή Πληρότητας
GET /api/v1/fillness
SELECT bin_status_summary
δεδομένα κάδων
JSON response
Γράφημα & Χάρτης κάδων
ΠΧ2: Βελτιστοποίηση Διαδρομής
Επιλογή ζώνης & threshold (70%)
GET /api/v1/optimization/route
?zone_code=GAZ&threshold=70
alt
[Κάδοι >= 3 με threshold 70%]
SELECT bins WHERE fill_level >= 70%
Λίστα κάδων
Nearest Neighbor TSP
(Haversine αποστάσεις)
[Κάδοι < 3 — Auto-fallback]
SELECT bins WHERE fill_level >= 50%
Λίστα κάδων (fallback)
TSP με threshold 50%
INSERT collection_routes
+ collection_route_bins
route_id
INSERT vehicle_alerts
(route_assigned, zone_code)
OK
διαδρομή + χάρτης
Βήμα 9: Ειδοποίηση Οδηγών
Ειδοποίηση route_assigned
(vehicle_id, zone, κάδοι)
Νέα διαδρομή αποκομιδής!
Φόρτωση κάδων ζώνης
GET /api/v1/dispatch/alerts
?vehicle_id=X&zone_code=GAZ
SELECT vehicle_alerts
ειδοποιήσεις
Λίστα κάδων
loop
[Αποκομιδή κάδων]
Καταγραφή αποκομιδής
POST /api/v1/bins/{id}/collect
UPDATE sensor_data
(fill_level = 0)
OK
επιβεβαίωση
alt
[Πληρότητα οχήματος >= 85%]
POST /api/v1/dispatch/xyta-trip
INSERT xyta_trips
OK
Κατευθυνθείτε στο ΧΥΤΑ
Αναφορά Βλάβης
Αναφορά βλάβης
POST /api/v1/dispatch/vehicle-breakdown
(vehicle_id, reason)
UPDATE vehicles SET status='inactive'
SELECT nearest active vehicle
κοντινότερο όχημα
INSERT vehicle_alerts (zone_change)
OK
Κοντινότερο όχημα: VEH-XXX
Ειδοποίηση βλάβης
Επαναφορά Οχήματος
Επιλογή οχήματος → Επαναφορά
PUT /api/v1/vehicles/{id}
{status: active}
UPDATE vehicles SET status='active'
OK
INSERT vehicle_alerts
(vehicle_restored)
OK
Επιβεβαίωση επαναφοράς
ΠΧ3: Πύλη Πολιτών
GET /api/v1/citizen/bins
SELECT bin_status_summary
κάδοι με συντεταγμένες
χάρτης κάδων
POST /api/v1/citizen/reports
(report_type, location, photo)
INSERT citizen_reports
(tracking_code=CR1-XXXX)
tracking_code
alt
[Email δόθηκε]
Αποστολή επιβεβαίωσης
Email με tracking_code
CR1-XXXX
GET /api/v1/citizen/reports/CR1-XXXX
SELECT citizen_reports
WHERE tracking_code=CR1-XXXX
κατάσταση αναφοράς
Ανοιχτή → Σε Εξέλιξη → Επιλύθηκε
Προβολή αναφορών πολιτών
GET /api/v1/citizen/reports
SELECT citizen_reports
Λίστα αναφορών
αναφορές με φίλτρα
Αλλαγή κατάστασης → Επιλύθηκε
PUT /api/v1/citizen/reports/{id}
{status: resolved}
UPDATE citizen_reports
OK
Επιβεβαίωση
ΠΧ1: Ανάκτηση Κωδικού
Ξέχασα τον κωδικό
POST /api/v1/auth/password-reset-request
(username)
SELECT users WHERE username=?
email χρήστη
INSERT password_reset_tokens
Αποστολή reset link
Email με σύνδεσμο
POST /api/v1/auth/password-reset-confirm
(token, new_password)
UPDATE users SET password_hash=?
UPDATE tokens SET used=TRUE
Κωδικός άλλαξε επιτυχώς!

Εικόνα 3 4.5.2 Sequence Diagram

4.5.3. Activity Diagram

Ροή διαδικασίας αποκομιδής:

- Έλεγχος πληρότητας
- Δημιουργία δρομολογίου
- Εκτέλεση αποκομιδής
- Ενημέρωση συστήματος
- Αξιολόγηση αποτελεσμάτων

4.6. Σχεδίαση Διεπαφών Χρήστη

Η σχεδίαση διεπαφών βασίζεται σε αρχές **χρηστικότητας και απλότητας**.

Web Dashboard

- Χάρτης κάδων (GIS)
- Πίνακας ειδοποιήσεων
- Στατιστικά και KPIs
- Διαχείριση χρηστών

Progressive Web App (αντί για Mobile App) - (προσωπικό)

- Λίστα δρομολογίων
- Πλοήγηση
- Ενημέρωση κατάστασης κάδων
- Άμεση επικοινωνία με κέντρο

4.7. Επιλογή Τεχνολογιών

Επίπεδο	**Τεχνολογία**	**Τεκμηρίωση**
IoT	Arduino / ESP32	Χαμηλό κόστος, αξιοπιστία

Επίπεδο	**Τεχνολογία**	**Τεκμηρίωση**
Επικοινωνία	MQTT	Lightweight, ιδανικό για IoT
Backend	Node.js / Python	Επεκτασιμότητα, REST APIs
Βάση	Firebase / MongoDB	Real-time δεδομένα
Dashboard	Power BI / Web App	Οπτικοποίηση & BI
Mobile	Web App	Ευρεία χρήση, ανοιχτό περιβάλλον

Πίνακας 2 4.7 Επιλογή Τεχνολογιών

4.8. Συμπεράσματα Κεφαλαίου

Το Κεφάλαιο 4 ανέπτυξε αναλυτικά τις απαιτήσεις και τη σχεδίαση ενός ολοκληρωμένου συστήματος έξυπνης διαχείρισης απορριμμάτων για τον Δήμο Γαζίου. Μέσα από τη σαφή ανάλυση απαιτήσεων, την πολυεπίπεδη αρχιτεκτονική, το μοντέλο δεδομένων και τα διαγράμματα UML, τεκμηριώνεται ότι το προτεινόμενο σύστημα είναι τεχνικά εφικτό, λειτουργικά επαρκές και προσαρμοσμένο στις ανάγκες της τοπικής αυτοδιοίκησης. Το κεφάλαιο αυτό αποτελεί τη γέφυρα μεταξύ θεωρητικού πλαισίου και πρακτικής υλοποίησης, η οποία αναπτύσσεται στο επόμενο κεφάλαιο.

5. Ανάπτυξη Λογισμικού Smart Waste Management System

5.1. Καταγραφή Δραστηριοτήτων

Δομή Ανάλυσης Εργασιών (ΔΑΕ)

1. Καταγραφή απαιτήσεων λογισμικού
2. Πακέτο Φιλοξενίας Εφαρμογής
   2.1. Καθορισμός Τεχνικών Προδιαγραφών
3. Ανάπτυξη λογισμικού
   3.1. Καταγραφή λειτουργικότητας του πληροφοριακού συστήματος
   3.2. Ανάλυση ΠΣ
   3.3. Σχεδιασμός
   3.4. Υλοποίηση Βάσης Δεδομένων
   3.5. Υλοποίηση λειτουργιών
4. Εξωτερικά Πληροφοριακά Συστήματα
   4.1. Συσκευή iot_smartwastebin στον προσομοιωτή Wokwi
      4.1.1. Ανάπτυξη λογισμικού για τη λειτουργία της συσκευής
   4.2. Google Maps
      4.2.1. Δημιουργία API για τη χρήση Google Maps
5. Εγκατάσταση Εφαρμογής Smart Waste Management System
6. Δοκιμές καλής λειτουργίας

5.1.1. Καταγραφή απαιτήσεων εφαρμογής Smart Waste Management System

Το ΠΣ Smart Waste Management System στηρίζεται στη λήψη των δεδομένων από την εικονική συσκευή iot_smartwastebin , την διαχείριση τους, την παραγωγή σημαντικών πληροφοριών στους χρήστες του δήμου Γαζίου του τμήματος Καθαριότητας, την απεικόνιση των κάδων απορριμμάτων και τη διαδρομή των απορριμματοφόρων στους χάρτες Google, την ενημέρωση των απορριμματοφόρων για τη βέλτιστη διαδρομή τους, την παρακολούθηση των απορριματοφόρων και την παραγωγή σημαντικών πληροφοριών για την κατάσταση του ολοκληρωμένου κυκλώματος και εφαρμογής του συστήματος.

Παρακάτω περιγράφονται αναλυτικά οι λειτουργίες που προσφέρονται στο Υπεύθυνο Καθαριότητας μέσω του Πληροφοριακού Συστήματος (ΠΣ) Smart Waste Management

System. Η αλληλεπίδραση του χρήστη με την εφαρμογή διέπετε από τις αρχές της ασφάλειας και της βελτιστοποίησης των επιχειρησιακών διαδικασιών.

Διαδικασία Αυθεντικοποίησης και Πρόσβασης: Η είσοδος τους υπεύθυνου του τμήματος στο σύστημα πραγματοποιείται μέσω της διαδικασίας αυθεντικοποίησης, με χρήση μοναδικών διαπιστευτηρίων (όνομα χρήστη και κωδικό πρόσβασης). Κατόπιν της επιτυχούς ταυτοποίησης των στοιχείων του στο ΠΣ, ο χρήστης κατευθύνεται στο κεντρικό περιβάλλον διεπαφής (Dashboard), το οποίο αποτελεί το σημείο ελέγχου της εφαρμογής.

Λειτουργικές Δυνατότητες: Μέσω του κεντρικού μενού, ο χρήστης έχει τη δυνατότητα να εκτελέσει τις παρακάτω ενέργειες:

1. **Εποπτεία Πληρότητας:** Παρακολούθηση και ενημέρωση για τη στάθμη πλήρωσης του συνόλου των κάδων ή συγκεκριμένων μονάδων σε δεδομένη χρονική στιγμή. Η άντληση των δεδομένων πραγματοποιείται μέσω των διαθέσιμων φίλτρων και εργαλείων αναζήτησης της εφαρμογής.
2. **Γεωχωρική Απεικόνιση:** Εντοπισμός των γεωγραφικών συντεταγμένων κάθε κάδου και απεικόνισης τους σε διαδραστικό περιβάλλον OpenStreetMap. Οι κάδοι αποτυπώνονται με διακριτά εικονίδια στις ακριβείς θέσεις τους, ενώ η επιλογή κάθε εικονιδίου παρέχει πρόσβαση στον μοναδικό κωδικό ταυτοποίησης του κάδου.
3. **Βελτιστοποίηση Δρομολογίων:** Σχεδιασμός των βέλτιστων (συντομότερων) διαδρομών για τα απορριμματοφόρα οχήματα, λαμβάνοντας υπόψη την τρέχουσα πληρότητα των κάδων και τη χωρητικότητα του στόλου. Στόχος είναι η ελαχιστοποίηση των απαιτούμενων πόρων και οχημάτων. Ο χρήστης μπορεί να παραμετροποιήσει τον αλγόριθμο ορίζοντας την ημερομηνία, το σημείο εκκίνησης και το κατώφλι πληρότητας (threshold) των κάδων προς αποκομιδή.
4. **Ανάλυση Επιχειρησιακού Πλάνου:** Λήψη λεπτομερών πληροφοριών για τη διαδρομή κάθε οχήματος, συμπεριλαμβανομένων των σημείων επίσκεψης, των αποστάσεων που πρόκειται να διανυθούν και της αλληλουχίας των προς άδειασμα κάδων.
5. **Εποπτεία σε Πραγματικό Χρόνο (Real-time Tracking):** Παρακολούθηση της ακριβούς θέσης των απορριμματοφόρων κατά τη διάρκεια της αποκομιδής. Το στίγμα των οχημάτων προβάλλεται σε πραγματικό χρόνο πάνω σε χάρτες της OpenStreetMap, διασφαλίζοντας τον πλήρη έλεγχο της διαδικασίας.
6. **Παραμετροποίηση Συστήματος:** Διαμόρφωση των συστημικών παραμέτρων που καθορίζουν τη λειτουργία του ΠΣ, εξασφαλίζοντας την απαραίτητη ευελιξία και προσαρμοστικότητα στις μεταβαλλόμενες απαιτήσεις του εκάστοτε φορέα ή χρήστη.

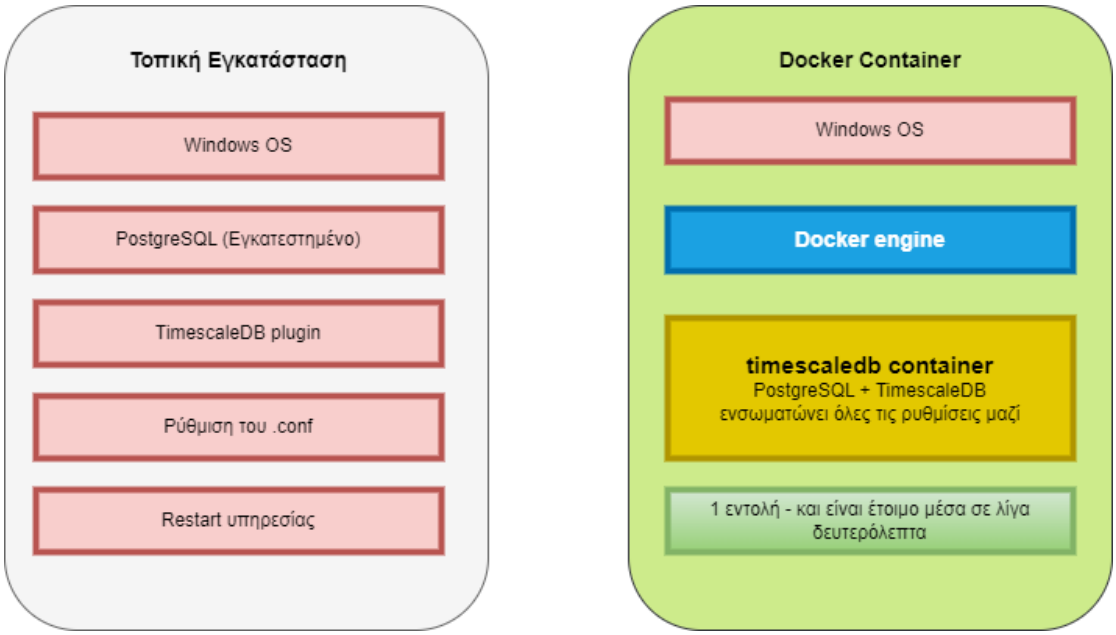
7. **Διαχείριση Πρόσβασης:** Έλεγχος και απόδοση δικαιωμάτων στους χρήστες του συστήματος, διασφαλίζοντας την ιεραρχική διαστρωμάτωση της πρόσβασης στις πληροφορίες.

5.1.2. Καταγραφή απαιτήσεων web server

Το πακέτο φιλοξενίας αποτελεί την υποδομή πάνω στην οποία θα τρέξει το Smart Waste Management System. Επιλέχθηκε η αρχιτεκτονική Docker containers.

5.1.2.1. Τι είναι το Docker Engine – Docker container

Το Docker είναι μια πλατφόρμα containerization. Πιο συγκεκριμένα, είναι μια εφαρμογή «συσκευασίας-πακέτου» που εμπεριέχει μέσα όλες τις εξαρτήσεις (βιβλιοθήκες, ρυθμίσεις,runtime), σε ένα αυτόνομο περιβάλλον που ονομάζεται **container**. Αυτό το container τρέχει το ίδιο σε οποιοδήποτε υπολογιστή, ανεξάρτητα από το είδος λειτουργικού συστήματος ή τις τοπικές ρυθμίσεις του. Επίσης, μειώνεται και ο χρόνος εγκατάστασης με τον παραδοσιακό τρόπο εγκατάστασης όπως φαίνεται και στην παρακάτω εικόνα διαγράμματος.

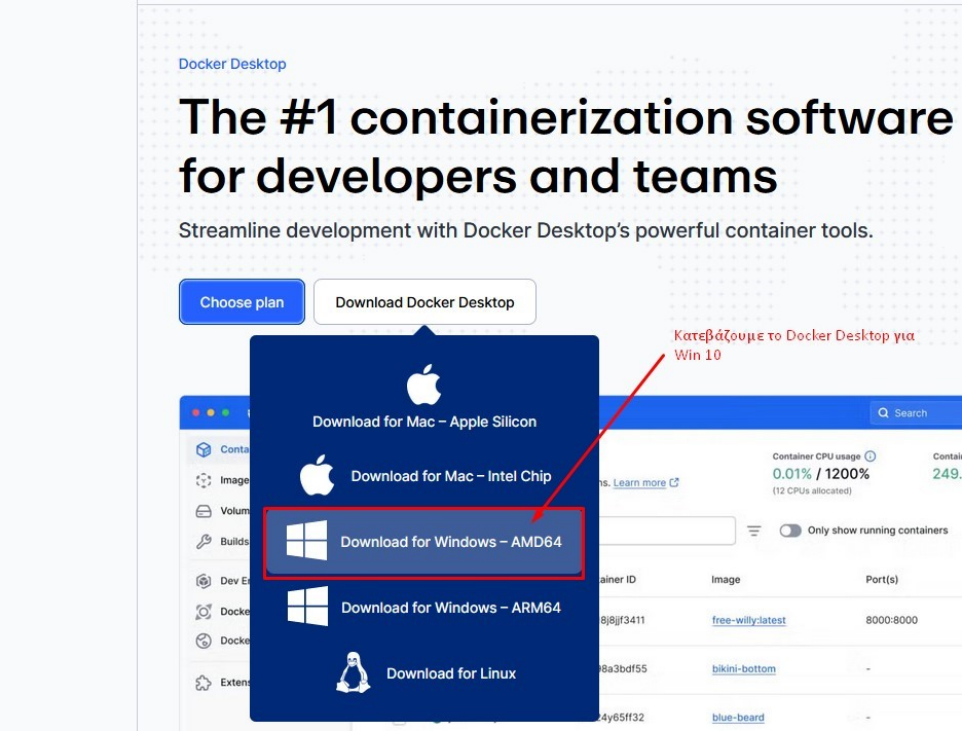


Εικόνα 4 Λειτουργία Docker Container

Βήμα 1: Εγκατάσταση Εφαρμογής Docker Desktop

Πηγαίνουμε στο https://www.docker.com/products/docker-desktop/ και κατεβάζουμε την έκδοση για το Λειτουργικό Σύστημα του υπολογιστή μας.

Εγκαθιστούμε το **Docker Desktop Installer.exe** στο υπολογιστή μας και μόλις ολοκληρωθεί η εγκατάσταση ζητείται να γίνει επανεκκίνηση του υπολογιστή μας. Μετά την επανεκκίνηση, ανοίγουμε το Docker Desktop και περιμένουμε να δούμε την πράσινη ένδειξη κάτω αριστερά Engine running μέσα στην εφαρμογή. Μας ζήτησε να κάνουμε εγκατάσταση και του WSL 2 (Windows Subsystem for Linux) όπου γίνεται αυτόματα.



Εικόνα 5 Εγκατάσταση Docker Desktop

5.1.2.2. Καθορισμός Τεχνικών Προδιαγραφών

Οι ελάχιστες απαιτήσεις που χρειάζεται σε μηχάνημα Server για να μπορέσει να υποστηρίξει την υλοποίηση και συντήρηση της εφαρμογής είναι οι παρακάτω:

Χαρακτηριστικά	Ελάχιστη Απαίτηση	Προτεινόμενη Απαίτηση (αν τοποθετηθεί σε Δήμο)
CPU	8 cores (x86, x64)	8+ cores
RAM	8GB	16-32 GB
Storage Αποθηκευτικός Χώρος	50GB SSD	200+ GB SSD (TimescaleDB, data)
OS	Ubuntu 22.04 LTS / Debian 12 , Win10, Win11	Ubuntu 22.04 LTS, Win11

Docker Engine	≥ 24.0	Latest stable
Docker Compose	v.2.20+	Latest stable
Network (Δίκτυο)	100 Mbps	1 Gbps
Ports (Θύρες)	80, 443, 5432,8000, 1883, 9001, 3000, 5050	80, 443, 5432,8000, 1883, 9001, 3000, 5050

Table 3 Τεχνικές Προδιαγραφές μηχανήματος Server

5.1.2.3. Docker images – Αναλυτικές Προδιαγραφές

Οι αναλυτικές προδιαγραφές που απαιτούνται στην εφαρμογή μας για το Docker container είναι τα images που θα εγκατασταθούν μέσω command prompts για να μπορέσουμε να τρέξουμε τοπικά το server μας με τις απαιτούμενες προδιαγραφές όπως περιγράφονται στο παρακάτω πίνακα.

Container	Image	Port(s)	Τεχνικές Προδιαγραφές & Ρυθμίσεις
timescaledb	timescale/timescaledb:latest-pg16	5432	PostgreSQL 16 + TimescaleDB extension
backend	python:3.14 (custom)	8000	FastAPI + uvicorn[standard]. Volumes: ./backend:/app, ml_models:/app/models
frontend	node:20-alpine (build)	3000	React 18 dev server (development)
nginx	nginx:1.25-alpine	80, 443	Reverse proxy. Upstream: backend:8000, frontend:3000. SSL termination.
Mosquitto / httppost	eclipse-mosquitto:2.0 ή httppost	1883, 9001	MQTT broker. QoS 1 για IoT μηνύματα. WebSocket (9001) για browser clients. Auth: username/password
redis	redis:7.2-alpine	6379	Cache: API responses (TTL 30s). Session store:

			JWT blacklist. Rate limiting counter.
pgadmin	dpage/pgadmin4:latest	5050	Web-based PostgreSQL management. Development only

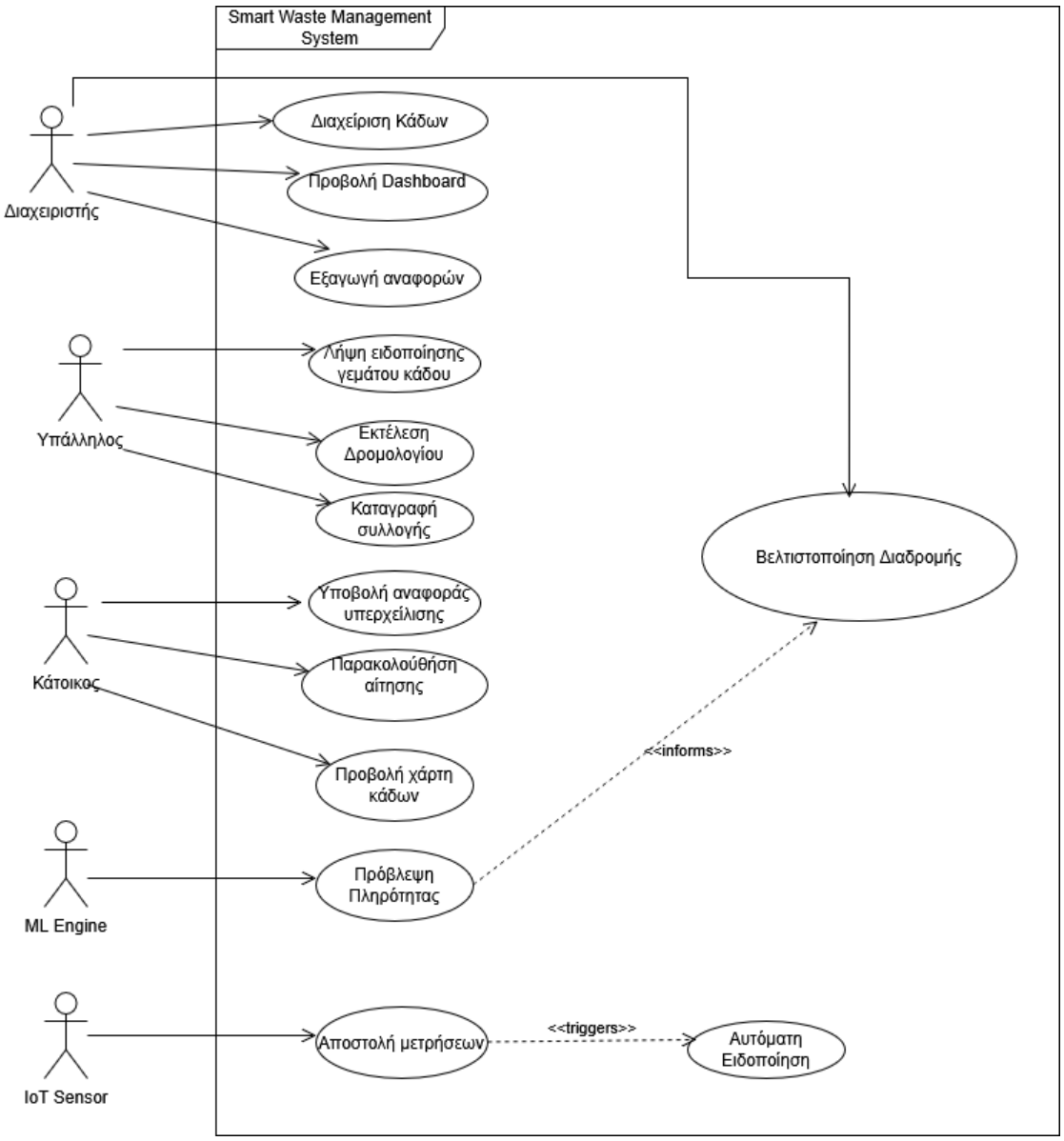
Table 4 Αναλυτικές Προδιαγραφές του Docker container τοπικά

5.2. Ανάλυση ΠΣ

Η ανάλυση του ΠΣ μας μπορεί να γίνει μέσα από τα διαγράμματα περιπτώσεων χρήσης και δραστηριοτήτων καθώς και τη λεκτική περιγραφή ΠΧ.

5.2.1. Διάγραμμα Περιπτώσεων Χρήσης του Smart Waste Management System

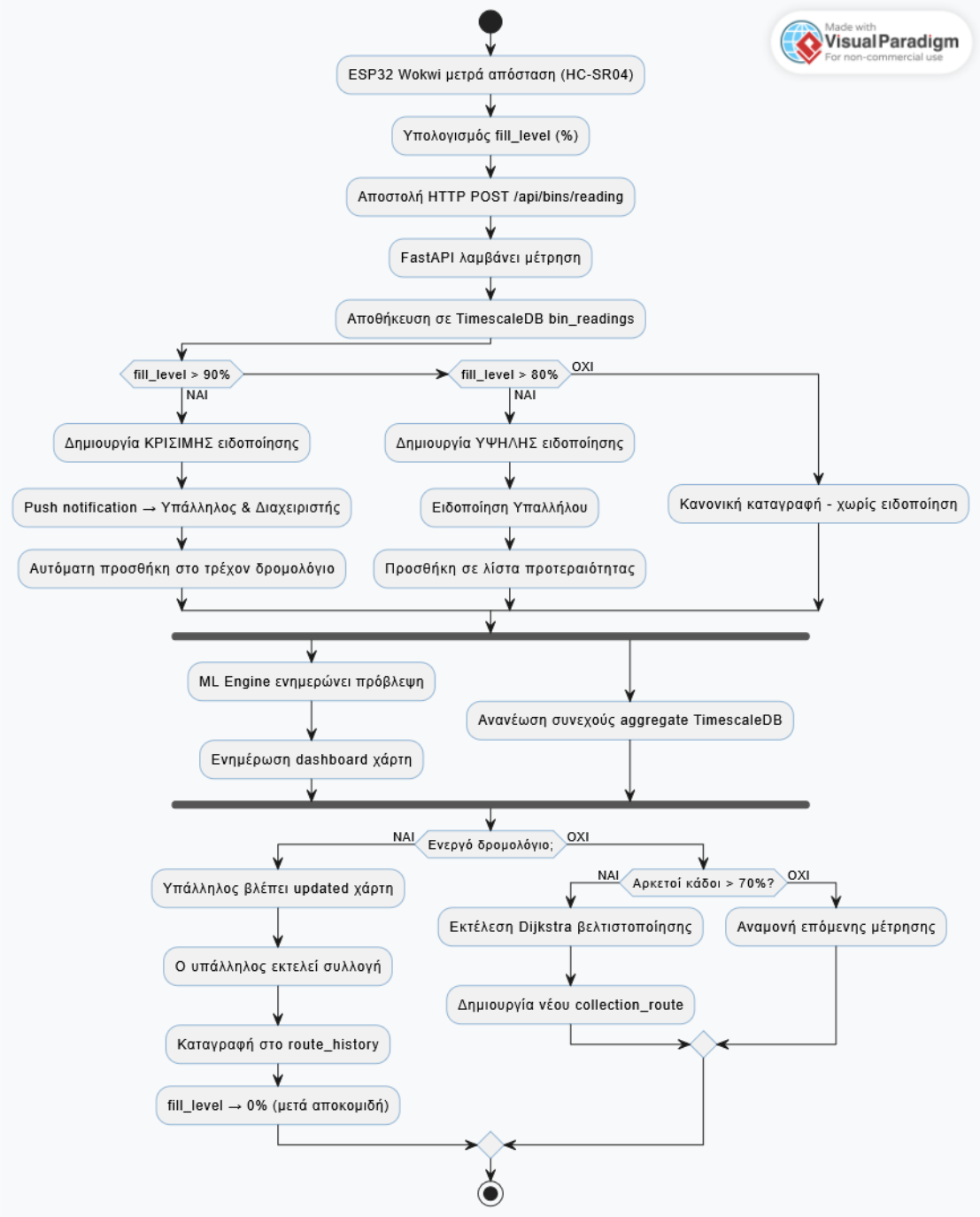
Σχεδιάζοντας το διάγραμμα Use Case (ΠΧ) Smart Waste Management System αποτυπώνεται η λειτουργικότητα του ΠΣ καθώς και των ενεργειών που εκτελεί το σύστημα όταν αλληλεπιδρά με το χρήστη και τα εξωτερικά πληροφοριακά συστήματα που στην πορεία θα προστεθούν.



Εικόνα 6 Use Case ΠΧ Smart Waste Management System

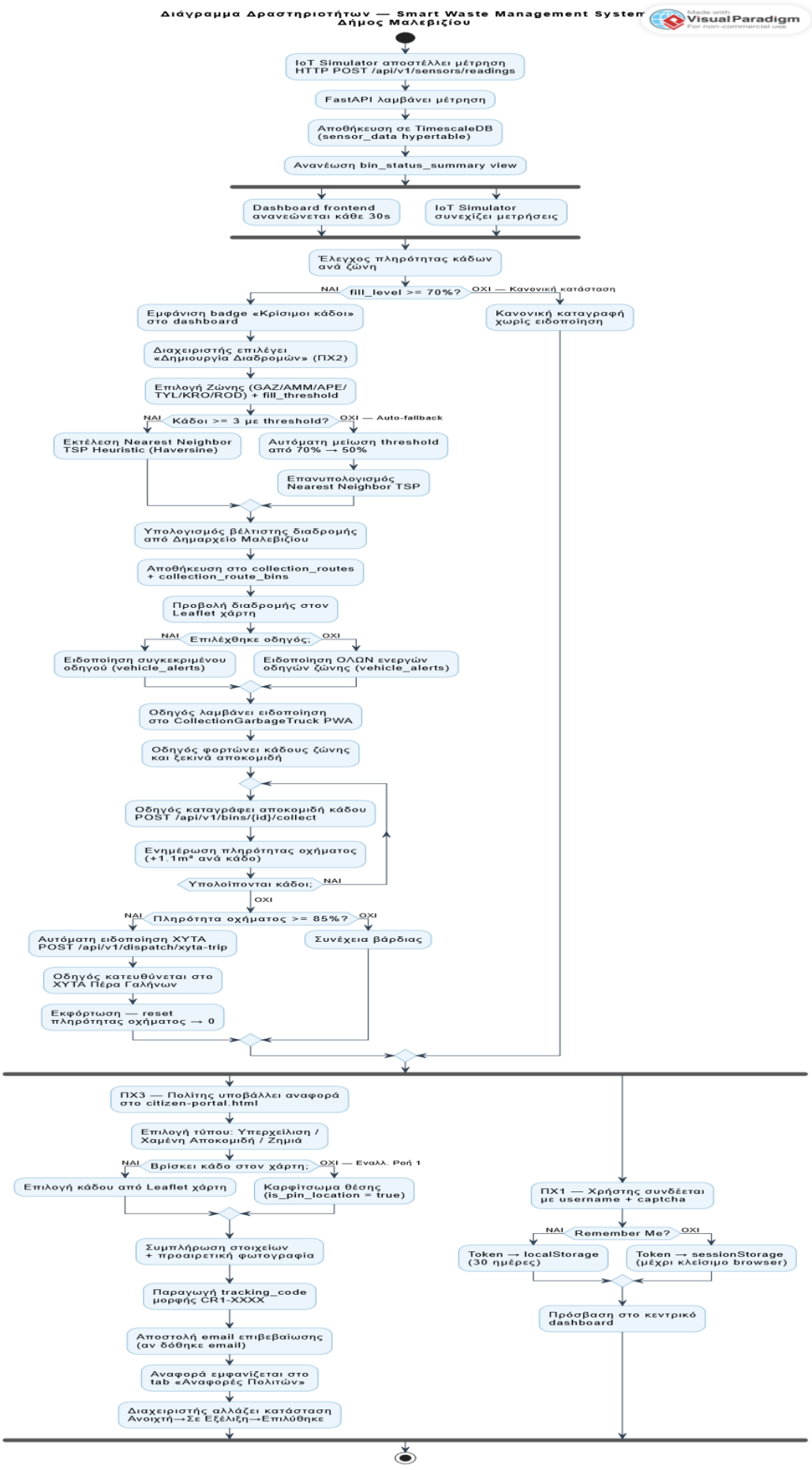
5.2.2. Διάγραμμα Δραστηριοτήτων εφαρμογής Smart Waste Management System

Στο παρακάτω διάγραμμα δραστηριοτήτων αποτυπώνονται οι δραστηριότητες , οι ενέργειες και οι ροές της εφαρμογής Smart Waste Management System.



Εικόνα 7 Διάγραμμα Δραστηριοτήτων με προσομοίωση αισθητήρων μέσω Wokwi

Λόγω αντιμετώπισης μερικών προβλημάτων με την πλατφόρμα προσομοίωσης Αισθητηρίων – Wokwi και τη συνδεσιμότητας της το διάγραμμα δραστηριοτήτων προσαρμόστηκε με χρήση γλώσσας python και δημιουργίας iot_simulator. Επομένως, το νέο διάγραμμα δραστηριοτήτων αποτυπώνει τις δραστηριότητες, τι ενέργειες και τις ροές της ήδη υλοποιήσιμής εφαρμογής Smart Waste Management System. Παρακάτω είναι το νέο διάγραμμα στο οποίο αυτή τη στιγμή στηρίζεται η εφαρμογή που υλοποίησα.


Διάγραμμα Δραστηριοτήτων — Smart Waste Management System
Δήμος Μαλεβιζίου
Made with Visual Paradigm
For non-commercial use
IoT Simulator αποστέλλει μέτρηση HTTP POST /api/v1/sensors/readings
FastAPI λαμβάνει μέτρηση
Αποθήκευση σε TimescaleDB (sensor_data hypertable)
Ανανέωση bin_status_summary view
Dashboard frontend ανανεώνεται κάθε 30s
IoT Simulator συνεχίζει μετρήσεις
Έλεγχος πληρότητας κάδων ανά ζώνη
ΝΑΙ
fill_level >= 70%?
ΟΧΙ — Κανονική κατάσταση
Εμφάνιση badge «Κρίσιμοι κάδοι» στο dashboard
Κανονική καταγραφή χωρίς ειδοποίηση
Διαχειριστής επιλέγει «Δημιουργία Διαδρομών» (ΠΧ2)
Επιλογή Ζώνης (GAZ/AMM/APE/TYL/KRO/ROD) + fill_threshold
ΝΑΙ
Κάδοι >= 3 με threshold?
ΟΧΙ — Auto-fallback
Εκτέλεση Nearest Neighbor TSP Heuristic (Haversine)
Αυτόματη μείωση threshold από 70% → 50%
Επανυπολογισμός Nearest Neighbor TSP
Υπολογισμός βέλτιστης διαδρομής από Δημαρχείο Μαλεβιζίου
Αποθήκευση στο collection_routes + collection_route_bins
Προβολή διαδρομής στον Leaflet χάρτη
ΝΑΙ
Επιλέχθηκε οδηγός;
ΟΧΙ
Ειδοποίηση συγκεκριμένου οδηγού (vehicle_alerts)
Ειδοποίηση ΟΛΩΝ ενεργών οδηγών ζώνης (vehicle_alerts)
Οδηγός λαμβάνει ειδοποίηση στο CollectionGarbageTruck PWA
Οδηγός φορτώνει κάδους ζώνης και ξεκινά αποκομιδή
Οδηγός καταγράφει αποκομιδή κάδου POST /api/v1/bins/{id}/collect
Ενημέρωση πληρότητας οχήματος (+1.1m³ ανά κάδο)
Υπολοίπονται κάδοι;
ΝΑΙ
ΟΧΙ
ΝΑΙ
Πληρότητα οχήματος >= 85%?
ΟΧΙ
Αυτόματη ειδοποίηση ΧΥΤΑ POST /api/v1/dispatch/xyta-trip
Συνέχεια βάρδιας
Οδηγός κατευθύνεται στο ΧΥΤΑ Πέρα Γαλήνων
Εκφόρτωση — reset πληρότητας οχήματος → 0
ΠΧ3 — Πολίτης υποβάλλει αναφορά στο citizen-portal.html
Επιλογή τύπου: Υπερχείλιση / Χαμένη Αποκομιδή / Ζημιά
ΝΑΙ
Βρίσκει κάδο στον χάρτη;
ΟΧΙ — Εναλλ. Ροή 1
Επιλογή κάδου από Leaflet χάρτη
Καρφίτσωμα θέσης (is_pin_location = true)
Συμπλήρωση στοιχείων + προαιρετική φωτογραφία
Παραγωγή tracking_code μορφής CR1-XXXX
Αποστολή email επιβεβαίωσης (αν δόθηκε email)
Αναφορά εμφανίζεται στο tab «Αναφορές Πολιτών»
Διαχειριστής αλλάζει κατάσταση Ανοιχτή→Σε Εξέλιξη→Επιλύθηκε
ΠΧ1 — Χρήστης συνδέεται με username + captcha
ΝΑΙ
Remember Me?
ΟΧΙ
Token → localStorage (30 ημέρες)
Token → sessionStorage (μέχρι κλείσιμο browser)
Πρόσβαση στο κεντρικό dashboard

Εικόνα 8 Διάγραμμα Δραστηριοτήτων Smart Waste Managenemt System με προσομοίωση iot σε Python

5.2.3. Λεκτική περιγραφή Περιπτώσεων Χρήσης (ΠΧ)

Αναγνωριστικό: ΠΧ1

Όνομα: Πιστοποίηση χρήστη

Περιγραφή: Η περίπτωση χρήσης περιγράφει τα βήματα που χρειάζεται να κάνει ο διαχειριστής χρήστης του συστήματος στην εφαρμογή για να μπορεί να χειριστεί προβλήματα που μπορεί να προκύψουν.

Βασική ροή γεγονότων

1. Το ΠΣ προτρέπει το χρήστη να εισάγει τα στοιχεία πρόσβασής του
2. Ο χρήστης εισάγει τα στοιχεία
3. Ο χρήστης επιλέγει «Θυμήσου με»
4. Ο χρήστης επιλέγει σύνδεση
5. Το ΠΣ επιβεβαιώνει τα στοιχεία πρόσβασης
6. Το ΠΣ εμφανίζει το κεντρικού μενού

Εναλλακτική ροή 1

2.α.1 Ο χρήστης επιλέγει να κάνει ανάκτηση κωδικού πρόσβασης

2.α.2 Το ΠΣ εμφανίζει φόρμα για καταχώρηση ονόματος χρήστη

2.α.3 Ο χρήστης συμπληρώνει το όνομα χρήστη

2.α.4 Το ΠΣ στέλνει email με σύνδεσμο για τον ορισμό νέου κωδικού πρόσβασης

2.α.5 Ο χρήστης επιλέγει τον σύνδεσμο

2.α.6 Το ΠΣ ζητάει νέο κωδικό δύο φορές από τον χρήστη

2.α.7 Το ΠΣ αλλάζει τον κωδικό πρόσβασης

2.α.8 Η ροή συνεχίζεται από το βήμα 1

Εναλλακτική ροή 2

3.α.1 Ο χρήστης δεν επιλέγει «Θυμήσου με»

3.α.2 Η ροή συνεχίζεται από το βήμα 4 της βασικής ροής

Εναλλακτική ροή 3

6.α.1 Το ΠΣ δεν επιβεβαιώνει τα στοιχεία πρόσβασης του χρήστη

6.α.2 Το ΠΣ εμφανίζει μήνυμα σφάλματος

6.α.3 Η ροή συνεχίζεται από το βήμα 1 της βασικής ροής

Αναγνωριστικό: ΠΧ2

Όνομα: Βελτιστοποίηση Διαδρομής

Περιγραφή: Η περίπτωση χρήσης περιγράφει τα βήματα που χρειάζεται να κάνει ο διαχειριστής χρήστης του συστήματος στην εφαρμογή για να μπορεί να χειριστεί τη βέλτιστη διαδρομή αν υπάρξει πληρότητα κάδων .

Βασική ροή γεγονότων

1. Το σύστημα βγάζει μήνυμα ότι υπάρχουν τουλάχιστον 3 κάδοι σε μια επιλεγμένη ζώνη π.χ. ζώνη 5 όπου έχουν το επίπεδο του κάδου είναι πάνω από το 70% όπου είναι το κατώφλι fill_threshold (default 70%).
2. Ο χρήστης – διαχειριστής επιλέγει στο dashboard της εφαρμογής «Βελτιστοποίηση Διαδρομής»
3. Ορίζει το zone_id και το ελέγχει το κατώφλι fill_thershold (default 70%).
4. Ορίζει το σημείο εκκίνησης του χώρου των απορριμματοφόρων, από όπου ο εφεδρεία οδηγός θα οδηγήσει το όχημα.
5. Η εφαρμογή μέσω το OSMnx φορτώνει την απεικόνιση του γράφου OSM
6. Ενεργοποιείται ο αλγόριθμος βέλτιστης διαδρομής Dijkstra Nearest Neighbor TSP heuristic
7. Αποθήκευση της διαδρομής στο collection_routes
8. Προβολή της διαδρομής στο χάρτη
9. Αποστολή διαδρομής στον υπάλληλο του απορριμματοφόρου που έχει οριστεί.

Εναλλακτική Ροή 1

1.α.1 Αν οι κάδοι είναι λιγότεροι από 3 κάδοι το σύστημα χαμηλώνει το fill_threshold στο 50% αυτόματα και ξανά υπολογίζει

1.α.2 Ξεκινά ξανά η διαδικασία από τ βήμα 1

Εναλλακτική Ροή 2

5.α.1 Αν OSM API δεν είναι δυνατόν να μας δείξει τη διαδρομή με τους κάδους γίνεται χρήση του cached γράφου last successful fetch και μήνυμα προειδοποίησης

5.α.2 Η ροή συνεχίζεται από το βήμα 4

Αναγνωριστικό: ΠΧ3

Όνομα: Πρόσβαση σε πύλη πολιτών

Περιγραφή: Η περίπτωση χρήσης περιγράφει τα βήματα που χρειάζεται να κάνει ο κάτοικος του δήμου Μαλεβιζίου για να μπορεί να μπορεί να στείλει ένα μήνυμα για υπερχείλιση/ χαμένη αποκομιδή/ζημιά κάδου .

Βασική ροή γεγονότων

1. Ο κάτοικος πλοηγείτε στο citizen-portal.
2. Πατά «Νέα Αναφορά»
3. Επιλέγει τύπο: Υπερχείλιση / Χαμένη Αποκομιδή / Ζημιά κάδου
4. Εντοπίζει τον κάδο στο χάρτη
5. Συμπληρώνει περιγραφή .
6. Ανεβάζει φωτογραφία (προαιρετικά)
7. Πατά «Υποβολή»
8. Το σύστημα παράγει ένα tracking_code
9. Εμφανίζει Επιβεβαίωση του αιτήματος και αποστολή email (αν δόθηκε email)

Εναλλακτική ροή 1

4.α.1 Ο κάτοικος δεν βρίσκει το κάδο στον χάρτη

4.α.2 Καρφιτσώνει τη θέση

4.α.3 Γίνεται αναφορά για τοποθέτηση κάδου στο χάρτη

4.α.4 Η ροή συνεχίζεται από το βήμα 3

Εναλλακτική ροή 2

1.α.1 Ο κάτοικος μπορεί να είναι εγγεγραμμένο χρήστης

1.α.2 Γίνεται αυτόματη προσυμπλήρωση στοιχείων

1.α.3 Ιστορικό αναφορών στο προφιλ του

1.α.4 Η ροή συνεχίζεται από το βήμα 1

5.3. Ανάλυση Πληροφοριακού Συστήματος Smart Waste Management System

5.4. Σχεδιασμός

Layer	Pattern / Τεχνολογία	Αιτιολόγηση Επιλογής
IoT Transport	HTTP REST + MQTT (dual protocol)	HTTP: εύκολο debugging. MQTT: χαμηλό latency, publish/subscribe για πολλούς αισθητήρες

Layer	Pattern / Τεχνολογία	Αιτιολόγηση Επιλογής
API Architecture	RESTful + Async (FastAPI)	Υψηλές επιδόσεις, async I/O, αυτόματο OpenAPI docs, Pydantic validation
Database Pattern	TimescaleDB Hypertables	Αυτόματο partitioning χρόνου, 10–100x γρηγορότερες χρονο-queries από plain PostgreSQL
Caching	Redis (multi-purpose)	API response cache, session store, rate limiting, pub/sub για real-time notifications
Frontend Pattern	React SPA + React Context	Component reusability, real-time updates, Google Maps integration, mobile-responsive
Auth Pattern	JWT RS256 + Refresh Tokens	Stateless, κατάλληλο για microservices, asymmetric signing για ασφάλεια
Routing Algorithm	Greedy TSP via Dijkstra (NetworkX)	Αποδεκτή ποιότητα λύσης σε polynomial χρόνο. OSMnx παρέχει πραγματικά δεδομένα οδών
ML Model	RandomForestRegressor (scikit-learn)	Καλή επίδοση με tabular δεδομένα, ανθεκτικό σε noise, explainable, χωρίς GPU

Table 5 Σχεδιασμός ΠΣ Smart Waste Management

5.4.1. Υλοποίηση Βάσης Δεδομένων

5.4.1.1. Εγκατάσταση Βάσης Δεδομένων PostgreSQL – TimescaleDB

Μετά από αρκετή έρευνα για το ποια βάση δεδομένων θα χρησιμοποιήσουμε για την εφαρμογή Διαχείριση Συστήματος Απορριμμάτων καταλήξαμε στην βάση δεδομένων PostgreSQL και την επέκταση TimescaleDB.

PostgreSQL

Η PostgreSQL είναι ένα ισχυρό σύστημα βάσης δεδομένων σχεσιακών αντικειμένων ανοιχτού κώδικα που χρησιμοποιεί και επεκτείνει τη γλώσσα SQL σε συνδυασμό με πολλά χαρακτηριστικά που αποθηκεύουν και κλιμακώνουν με ασφάλεια τα πιο περίπλοκα φόρτα εργασίας δεδομένων. Η προέλευση της PostgreSQL χρονολογείται από το 1986 ως μέρος του έργου POSTGRES στο Πανεπιστήμιο της Καλιφόρνια στο Μπέρκλεϊ και έχει σχεδόν 40 χρόνια ενεργής ανάπτυξης στην κεντρική πλατφόρμα.

Η PostgreSQL έχει κερδίσει ισχυρή φήμη για την αποδεδειγμένη αρχιτεκτονική της, την αξιοπιστία της, την ακεραιότητα των δεδομένων, το ισχυρό σύνολο χαρακτηριστικών, την επεκτασιμότητα και την αφοσίωση της κοινότητας ανοιχτού κώδικα πίσω από το λογισμικό για την παροχή με συνέπεια αποδοτικών και καινοτόμων λύσεων. Η PostgreSQL εκτελείται σε όλα τα μεγάλα λειτουργικά συστήματα, είναι συμβατή με ACID από το 2001 και διαθέτει

ισχυρά πρόσθετα όπως το δημοφιλές πρόγραμμα επέκτασης γεωχωρικών βάσεων δεδομένων PostGIS. Δεν αποτελεί έκπληξη το γεγονός ότι η PostgreSQL έχει γίνει η σχεσιακή βάση δεδομένων ανοιχτού κώδικα που προτιμούν πολλοί άνθρωποι και οργανισμοί.

TimescaleDB

Το TimescaleDB είναι μια επέκταση PostgreSQL που κατασκευάστηκε για χρονοσειρά υψηλής απόδοσης και ανάλυση δεδομένων συμβάντων. Επιτρέπει στους χρήστες να αποθηκεύουν και να αναλύουν αποτελεσματικά μεγάλους όγκους μετρήσεων και συμβάντων χωρίς να διακυβεύεται η ταχύτητα ή η αξιοπιστία.

Αξιοποιώντας το PostgreSQL, το TimescaleDB παρέχει υποστήριξη SQL, ενώ βελτιστοποιεί το φόρτο εργασίας δεδομένων με χρονοσειρές (time-series). Αυτό επιτρέπει στους προγραμματιστές και τους αναλυτές να εκτελούν σύνθετα ερωτήματα, συναθροίσεις και αναλύσεις σε πραγματικό χρόνο σε μεγάλα σύνολα δεδομένων με ελάχιστη ρύθμιση.

Το TimescaleDB είναι σχεδιασμένο με προδιαγραφές επεκτασιμότητας και ευελιξίας, ιδανικό για συστήματα παρακολούθησης, τηλεμετρία IoT, οικονομικά δεδομένα και άλλες εφαρμογές χρονοσειρών. Η απρόσκοπτη ενσωμάτωσή του με τα υπάρχοντα εργαλεία PostgreSQL εξασφαλίζει γρήγορα, αξιόπιστα και εύχρηστα αναλυτικά στοιχεία για σύγχρονο φόρτο εργασίας δεδομένων.

Η Εγκατάσταση της PostgreSQL μπορεί να γίνει με δύο τρόπους: α) Τοπική Εγκατάσταση στο Υπολογιστή μας και β) Docker Desktop (docker engine).

Βήμα 1: Εγκατάσταση και Εκκίνηση του TimescaleDB με Docker

Ενώ έχουμε ανοιχτό το Docker Desktop ανοίγουμε Γραμμή Εντολών και τρέχουμε τον παρακάτω κώδικα όπου θα εγκαταστήσει το container timescaledb και θα δημιουργήσει αυτόματα τη βάση δεδομένων μας με όνομα wastebin_db.

```
docker run -d
  --name timescaledb
  -e POSTGRES_PASSWORD=κωδικός για τη βάση
  -e POSTGRES_USER=postgres
  -e POSTGRES_DB=wastebin_db
  -p 5432:5432
  timescale/timescaledb:latest-pg16
```

Βήμα 2: Σύνδεση στη Βάση Δεδομένων μας

Στην βάση δεδομένων μας μπορούμε να συνδεθούμε με δύο τρόπους, είτε μέσα από το Docker γράφοντας στο terminal την εντολή docker exec -it timescaledb psql -U postgres -d wastebin_db είτε μέσω την εφαρμογής pgadmin4 το οποίο είναι γραφικό περιβάλλον για να μπορούμε να διαχειριζόμαστε και να βλέπουμε την βάση δεδομένων μέσω του περιβάλλοντος. Για να δείξουμε το περιβάλλον της PgAdmin 4 κατεβάζουμε και εγκαθιστούμε το γραφικό περιβάλλον της βάσης δεδομένων.

Μόλις ολοκληρωθεί η εγκατάσταση του pgAdmin 4 για να συνδεθούμε χρειάζεται στην PostgreSQL και στη Βάση Δεδομένων μας χρειάζεται να δώσουμε τα στοιχεία:

Host: localhost

Port: 5432

Username: postgres

Password: ο κωδικός που δημιουργήσαμε σε προηγούμενο βήμα για τη Βάση Δεδομένων

Database: wastebin_management_db

Η Βάση Δεδομένων αποτελείται από εννέα (9) πίνακες:

Πίνακας	Τύπος	Κύρια Πεδία	Σημειώσεις
zones	Standard	id, name, city, geom	PostGIS polygon
bins	Standard	id, bin_code, zone_id, lat, lon, sensor_id	Soft delete
bin_readings	HYPERTABLE ⚡	time, bin_id, fill_level, temp, rssi	Partition: 1 ημέρα
collection_routes	Standard	id, vehicle_id, zone_id, bin_sequence[], geom	Array + GeoJSON
route_history	HYPERTABLE ⚡	time, route_id, bin_id, fill_at_pickup, event_type	Partition: 1 ημέρα
users	Standard	id, email, role, password_hash, zone_id	BCrypt hashing

citizen_reports	Standard	id, bin_id, report_type, tracking_code, status	Tracking code unique
alerts	Standard	id, bin_id, severity, is_acknowledged, created_at	Auto-cleanup 90 ημ.
vehicles	Standard	id, plate, capacity_m3, zone_id, driver_id	Συνδέεται με routes

Table 6 Πίνακες της Βάσης Δεδομένων (ΒΔ)

Αφού καταγράφηκαν και μελετήθηκαν οι περιπτώσεις χρήσης και το διάγραμμα δραστηριοτήτων προέκυψαν οι οντότητες που αποτελούν τους πίνακες τις βάσης δεδομένων.

Στην συνέχεια δημιουργήθηκε η βάση δεδομένων και ο χρήστης με τα αντίστοιχα δικαιώματα.

Με την χρήση του λογισμικού phpmyadmin δημιουργήθηκαν οι πίνακες της βάσης δεδομένων. Το σχήμα της βάσης απεικονίζεται στην παρακάτω εικόνα.

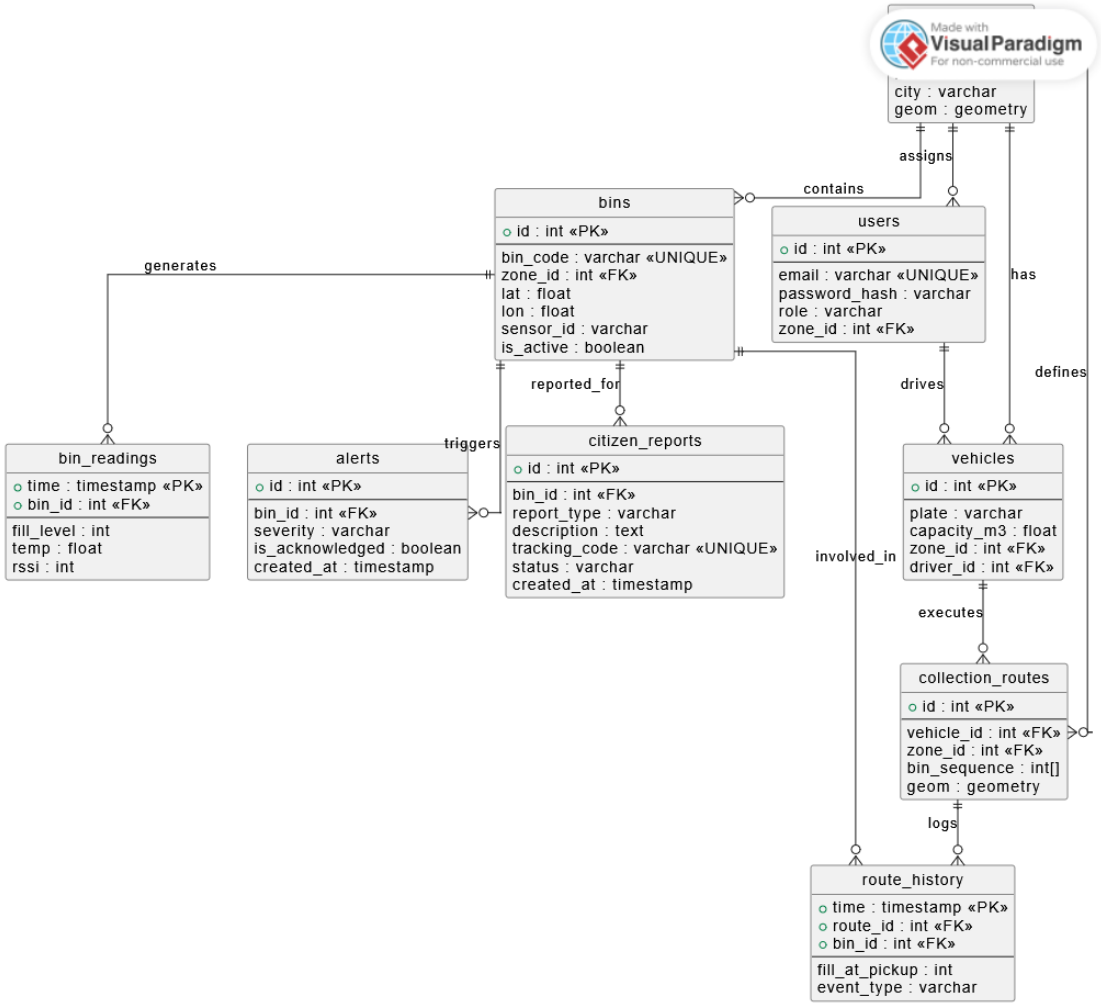


Table 7 ER Σχεσιακό Μοντέλο ΒΔ

Συσχετίσεις μεταξύ πινάκων – οντοτήτων:

zones ——— bins (1:N)

zones ——— users (1:N)

zones ——— vehicles (1:N)

zones ——— routes (1:N)

bins ——— bin_readings (1:N)

bins ——— alerts (1:N)

bins ——— citizen_reports (1:N)

bins ——— route_history (1:N)

vehicles (1) ——— (N) collection_routes

collection_routes (1) ——— (N) route_history

users (1) ——— (N) vehicles (driver σχέση)

Εγκατάσταση FASTAPI

pip install "fastapi[standard]"

pip install fastapi uvicorn psycopg2-binary

5.5. Υλοποίηση Λειτουργιών

5.5.1. FastAPI Backend — Ανάπτυξη

Module	Αρχεία Υλοποίησης
Project Setup + Config	main.py, core/config.py, core/database.py
Authentication	routers/auth.py, core/security.py
IoT Ingestion Endpoint	routers/bins.py + MQTT client
Alert Engine	services/alert_service.py
Routing Service	services/routing_service.py
ML Service	services/ml_service.py
Citizen Reports API	routers/reports.py
Dashboard Stats API	routers/dashboard.py
Scheduler Tasks	scheduler/tasks.py

Table 8 FastAPI Backend - Ανάπτυξη

5.5.2. React Frontend — Ανάπτυξη

Component	Σκοπός
App Router + Auth Context	React Router v6, protected routes, JWT management
Login / Register Pages	Forms, JWT fetch, localStorage token
BinMap + BinMarker	OpenStreetMap integration, color-coded markers
Dashboard (KPIs + Charts)	Recharts: fill trends, collections/day, zone heatmap
Alerts Panel	Real-time list, acknowledge, filter by severity
Routes Manager	Trigger optimization, view GeoJSON on map, assign
Citizen Portal	Report form, photo upload, tracking status page
Admin Panel	CRUD: bins, zones, users, vehicles
Prediction Cards	ML predictions visualization per bin
MQTT WebSocket Client	Real-time fill updates without polling

Table 9 Ανάπτυξη React Frontend

5.6. Εξωτερικά Πληροφοριακά Συστήματα

Το ΠΣ Smart Waste Management System ενσωματώνει δύο εξωτερικά πληροφοριακά συστήματα: (1) το Wokwi IoT Simulator λειτουργία ESP32 και (2) το OpenStreetMap της Leaflet API για χαρτογράφηση.

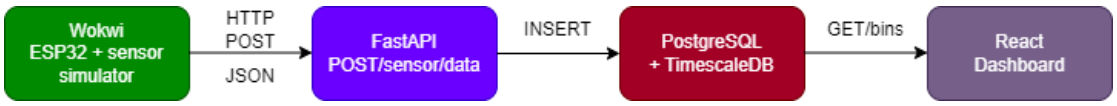
5.6.1. Συσκευή SmartBin στο simulator Wokwi

Το Wokwi προσομοιώνει τον microcontroller ESP32 που διαβάζει αισθητήρες και στέλνει HTTP requests. Το δύσκολο κομμάτι είναι ότι το Wokwi τρέχει στο cloud, οπότε χρειάζόμαστε με κάποιο τρόπο να φτάσει στο local FastAPI στον υπολογιστή μας. Αυτό λύνεται χρησιμοποιώντας το ngrok.

5.6.1.1. Τι είναι το ngrok

Το ngrok είναι ένα χρήσιμο εργαλείο που δημιουργεί ένα ασφαλές tunnel (σήραγγα) από το internet προς τον τοπικό μας υπολογιστή. Με απλά λόγια, κάνει τον τοπικό μας server (localhost) προσβάσιμο από οπουδήποτε στο internet. Αυτό το πετυχαίνει δημιουργώντας ένα δημόσιο url της μορφής https://abc123.ngrok-free.app . Για να γίνει αυτό κατεβάζουμε το αρχείο ngrok.exe και το τοποθετούμε σε ένα φάκελο με όνομα ngrok στο φάκελο με τα αρχεία του προγράμματος μας . Στη συνέχεια, δημιουργούμε στο site του ngrok λογαριασμό και μας δίνει ένα authentication token και το αντιγράφουμε στο config αρχείο του ngrok γράφοντας σε command prompt την εντολή ngrok config add-authtoken [MYAUTHTOKEN]

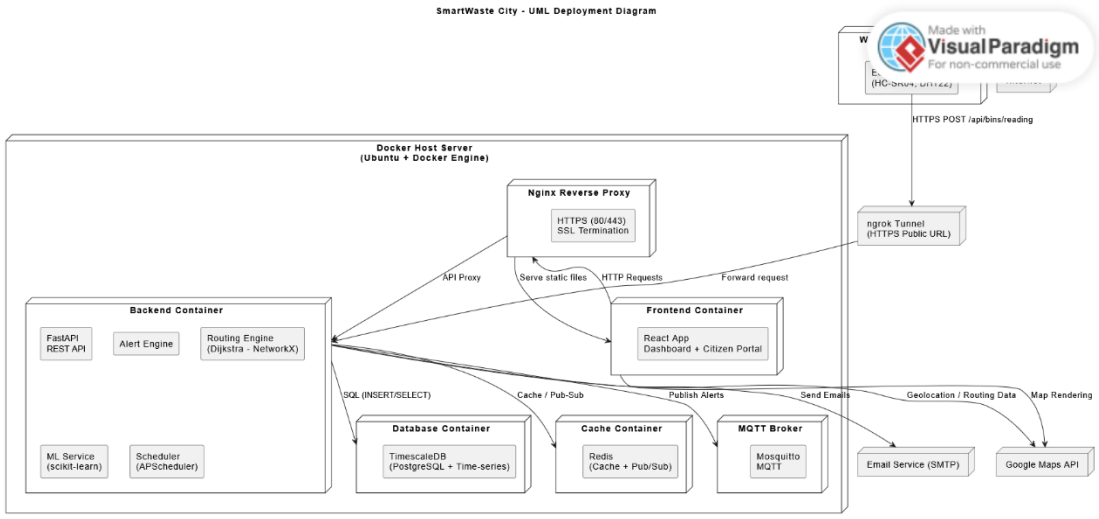
Ουσιαστικά το ngrok λειτουργεί ως ενδιάμεσος proxy, όπου δέχεται αιτήματα από το internet και τα προωθεί στο local backend της εφαρμογής , όπως μπορείτε να δείτε και στο παρακάτω διάγραμμα απεικόνισης.



Εικόνα 9 Επικοινωνία Wokwi με Dashboard

5.6.1.2. Ανάπτυξη λογισμικού συσκευής στο wokwi

Παρακάτω μπορείτε να δείτε και το σχετικό διάγραμμα Deployment Wokwi-ngrok-FastAPI



Εικόνα 10 Deployment Wokwi-ngrok-FastAPI

Αρχείο sketch.ino

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <DHT.h>

// ——— ΡΥΘΜΙΣΕΙΣ ———————————————————————————————

// ΑΝΤΙΚΑΤΑΣΤΗΣΗ με το ngrok  public URL:
const char* API_URL = "https://abc123.ngrok-free.app/api/bins/reading";
const char* WIFI_SSID = "Wokwi-GUEST";
const char* WIFI_PASS = "";
const char* BIN_ID = "BIN-GAZ-001";
const char* API_KEY = "sw_iot_key_2025";  // Αντιστοιχεί με FastAPI

// ——— PINS ————————————————————————————————————

#define TRIG_PIN    5
#define ECHO_PIN    18
#define DHT_PIN     23
#define LED_GREEN   2
```

```
#define LED_RED     4
#define DHT_TYPE    DHT22

// ——— ΣΤΑΘΕΡΕΣ ΚΑΔΟΥ ———————————————————————————

#define BIN_HEIGHT_CM  100   // Πλήρες ύψος κάδου (εσωτερικό)
#define SEND_INTERVAL  15000 // Αποστολή κάθε 15 δευτ.

DHT dht(DHT_PIN, DHT_TYPE);
unsigned long lastSendTime = 0;

// ——— SETUP ———————————————————————————————————

void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_RED, OUTPUT);
  dht.begin();

  // Σύνδεση WiFi
  Serial.println("Σύνδεση WiFi...");
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.println("\nΣυνδέθηκε! IP: " + WiFi.localIP().toString());
  digitalWrite(LED_GREEN, HIGH);
}

// ——— ΜΕΤΡΗΣΗ ΑΠΟΣΤΑΣΗΣ HC-SR04 ——————————————————
```

```
float measureDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  if (duration == 0) return -1;  // Timeout
  return (duration * 0.0343) / 2;  // cm
}

// ——— ΥΠΟΛΟΓΙΣΜΟΣ ΠΛΗΡΟΤΗΤΑΣ ———————————————————

int calculateFillLevel(float distance_cm) {
  if (distance_cm < 0) return -1;  // Σφάλμα αισθητήρα
  // Αντίστροφη σχέση: μικρή απόσταση = γεμάτος κάδος
  int fill = map((int)distance_cm, 0, BIN_HEIGHT_CM, 100, 0);
  return constrain(fill, 0, 100);
}

// ——— ΑΠΟΣΤΟΛΗ ΜΕΤΡΗΣΗΣ ————————————————————————

void sendReading(int fillLevel, float temp) {
  if (WiFi.status() != WL_CONNECTED) return;

  HTTPClient http;
  http.begin(API_URL);
  http.addHeader("Content-Type", "application/json");
  http.addHeader("X-API-Key", API_KEY);
  http.addHeader("ngrok-skip-browser-warning", "true");

  // Δημιουργία JSON payload
```

```
  DynamicJsonDocument doc(256);
  doc["bin_code"] = BIN_ID;
  doc["fill_level"] = fillLevel;
  doc["temperature_c"] = temp;
  doc["battery_voltage"] = 3.7;  // Simulated
  doc["signal_rssi"] = WiFi.RSSI();

  String payload;
  serializeJson(doc, payload);

  int httpCode = http.POST(payload);

  if (httpCode == 200) {
    Serial.println("✓ Επιτυχής αποστολή: " + payload);
    // LED status
    if (fillLevel > 80) {
      digitalWrite(LED_RED, HIGH);
      digitalWrite(LED_GREEN, LOW);
    } else {
      digitalWrite(LED_GREEN, HIGH);
      digitalWrite(LED_RED, LOW);
    }
  } else {
    Serial.println("✗ Σφάλμα HTTP: " + String(httpCode));
  }
  http.end();
}

//                                    ——                                    LOOP
————————————————————————————————————————————————————————————————————————————
—
void loop() {
```

```
 unsigned long now = millis();
 if (now - lastSendTime >= SEND_INTERVAL) {
  lastSendTime = now;

  float distance = measureDistance();
  int fillLevel = calculateFillLevel(distance);
  float temp = dht.readTemperature();

  Serial.println("━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━");
  Serial.println("Απόσταση: " + String(distance) + " cm");
  Serial.println("Πληρότητα: " + String(fillLevel) + "%");
  Serial.println("Θερμ/σία:  " + String(temp) + " °C");

  if (fillLevel >= 0) {
   sendReading(fillLevel, temp);
  } else {
   Serial.println("Σφάλμα αισθητήρα!");
  }
 }
}
```

Αρχείο Diagram.json

```
{
 "version": 1,
 "author": "SmartWaste Management System",
 "editor": "wokwi",
 "parts": [
  {
   "type": "board-esp32-devkit-v1",
   "id": "esp32",
   "top": 0, "left": 0,
   "attrs": { "env": "arduino-idf" }
  },
```

```
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic",
      "top": 80, "left": 200,
      "attrs": { "distance": "20" }
    },
    {
      "type": "wokwi-dht22",
      "id": "dht22",
      "top": 160, "left": 200,
      "attrs": { "temperature": "24", "humidity": "60" }
    },
    {
      "type": "wokwi-led",
      "id": "led_green",
      "top": 240, "left": 200,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-led",
      "id": "led_red",
      "top": 260, "left": 240,
      "attrs": { "color": "red" }
    }
  ],
  "connections": [
    ["esp32:D5", "ultrasonic:TRIG", "orange", []],
    ["esp32:D18", "ultrasonic:ECHO", "yellow", []],
    ["esp32:D23", "dht22:SDA", "blue", []],
    ["esp32:GND.1", "ultrasonic:GND", "black", []],
    ["esp32:3V3", "ultrasonic:VCC", "red", []],
    ["esp32:GND.2", "dht22:GND", "black", []],
    ["esp32:3V3", "dht22:VCC", "red", []],
```

```
    ["esp32:D2", "led_green:A", "green", []],
    ["esp32:D4", "led_red:A", "red", []]
  ]
}
```

5.6.2. Συσκευή Προσομοίωσης Αισθητήρων στους κάδους με χρήση Γλώσσας Python

Ο Simulator στέλνει τα δεδομένα από τα αισθητήρια των κάδων ανά 30 δευτερόλεπτα. Δημιούργησα λίστα με τους κάδους των ζωνών της περιοχής του Δήμου Μαλεβιζίου. Στη συνέχεια υπολόγισα το επίπεδο της κατάστασης κάθε κάδου. Επίσης σε μια λίστα κρατάω τα ενεργά οχήματα του στόλου (απορριμματοφόρα), κρατάω σε ένα Λεξικό (Dectionary) τις αρχικές θέσεις των οχημάτων. Δημιούργησα και μια συνάρτηση send_sensor_data όπου υπολογίζει και στέλνει τα δεδομένα στη βάση δεδομένων. Διαβάζει τα δεδομένα ως json και τα εμφανίζει στην κεντρική εφαρμογή στο μενού της Πληρότητας. Στη συνάρτηση update_vehicle_position προσομοιώνουμε την κίνηση του οχήματος μέσω GPS των ενεργών οχημάτων που βρίσκονται σε κίνηση. Με την επόμενη συνάρτηση send_vehicle_gps στέλνουμε την θέση GPS του οχήματος στο χάρτη μέσω json δεδομένων στην εφαρμογή.Τέλος, η συνάρτηση run_simulator αρχίζει την προσομοίωση των δεδομένων από τους κάδους και ανανεώνει κάθε 30 δευτερόλεπτα τα δεδομένα. Όλα αυτά περνάνε δυναμικά ως json αρχεία

Ο κώδικας του **iot_simulator.py** είναι ο παρακάτω:

```
import requests
import random
import time
import math
from datetime import datetime

API_URL = "http://127.0.0.1:8000"

BIN_IDS = [
    "BIN-GAZ-001", "BIN-GAZ-002", "BIN-GAZ-003", "BIN-GAZ-004",
    "BIN-GAZ-005", "BIN-GAZ-006", "BIN-GAZ-007",
    "BIN-AMM-001", "BIN-AMM-002", "BIN-AMM-003", "BIN-AMM-004",
    "BIN-AMM-005", "BIN-AMM-006", "BIN-AMM-007",
    "BIN-APE-001", "BIN-APE-002", "BIN-APE-003", "BIN-APE-004",
    "BIN-APE-005", "BIN-APE-006", "BIN-APE-007",
    "BIN-TYL-001", "BIN-TYL-002", "BIN-TYL-003",
    "BIN-TYL-004", "BIN-TYL-005", "BIN-TYL-006",
    "BIN-KRO-001", "BIN-KRO-002", "BIN-KRO-003", "BIN-KRO-004",
    "BIN-ROD-001", "BIN-ROD-002", "BIN-ROD-003", "BIN-ROD-004"
]

# Κατάσταση κάθε κάδου
```

```
fill_levels = {bin_id: random.uniform(5, 30) for bin_id in BIN_IDS}

def send_sensor_data(bin_id: str, fill_level: float):
    data = {
        "bin_id": bin_id,
        "fill_level": round(fill_level, 1),
        "temperature": round(random.uniform(20, 35), 1),
        "humidity": round(random.uniform(40, 80), 1),
        "battery_level": round(random.uniform(70, 100), 1),
        "signal_strength": random.randint(-80, -40)
    }
    try:
        response = requests.post(f"{API_URL}/api/v1/sensors/readings",
json=data)
        if response.status_code == 200:
            print(f" {bin_id}: fill={fill_level:.1f}%
temp={data['temperature']}°C")
        else:
            print(f" {bin_id}: HTTP {response.status_code}")
    except Exception as e:
        print(f" {bin_id}: {e}")

# Αρχικές θέσεις οχημάτων (lat, lon) περιοχές του Δήμου Μαλεβιζίου
VEHICLE_POSITIONS = {
    "VEH-001": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},
    "VEH-002": {"lat": 35.3362, "lon": 25.0856, "area": "Αμμουδάρα"},
    "VEH-003": {"lat": 35.4050, "lon": 25.0185, "area": "Αγία Πελαγία"},
    "VEH-004": {"lat": 35.2963, "lon": 25.0184, "area": "Τύλισος"},
    "VEH-005": {"lat": 35.2300, "lon": 24.9815, "area": "Κρουσώνας"},
    "VEH-006": {"lat": 35.3596, "lon": 25.0278, "area": "Ροδιά"},
    "VEH-007": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},  #
Συντήρηση
    "VEH-008": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},
    "VEH-009": {"lat": 35.3362, "lon": 25.0856, "area": "Αμμουδάρα"},
    "VEH-010": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},  #
Ανενεργό
    "VEH-P01": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},
    "VEH-P02": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},
    "VEH-P03": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},
    "VEH-P04": {"lat": 35.3259, "lon": 25.0659, "area": "Γάζι"},  #
Συντήρηση
}

# Οχήματα που κινούνται (active)
ACTIVE_VEHICLES = ["VEH-001", "VEH-002", "VEH-003", "VEH-004",
                   "VEH-005", "VEH-006", "VEH-008", "VEH-009",
                   "VEH-P01", "VEH-P02", "VEH-P03"]
```

```python
def update_vehicle_position(vehicle_id: str):
    """Προσομοίωση κίνησης οχήματος GPS"""
    if vehicle_id not in ACTIVE_VEHICLES:
        return None

    pos = VEHICLE_POSITIONS[vehicle_id]

    # Μικρή τυχαία κίνηση (περίπου 50-200 μέτρα)
    delta_lat = random.uniform(-0.001, 0.001)
    delta_lon = random.uniform(-0.001, 0.001)

    # Κρατάμε εντός ορίων Δήμου Μαλεβιζίου
    new_lat = pos["lat"] + delta_lat
    new_lon = pos["lon"] + delta_lon

    # Bounds check
    new_lat = max(35.20, min(35.43, new_lat))
    new_lon = max(24.95, min(25.15, new_lon))

    # Ενημέρωση θέσης
    VEHICLE_POSITIONS[vehicle_id]["lat"] = new_lat
    VEHICLE_POSITIONS[vehicle_id]["lon"] = new_lon

    return {"lat": new_lat, "lon": new_lon}

def send_vehicle_gps(vehicle_id: str, lat: float, lon: float):
    """Αποστολή GPS δεδομένων οχήματος"""
    data = {
        "latitude": round(lat, 7),
        "longitude": round(lon, 7),
        "fuel_level": round(random.uniform(30, 95), 1)
    }
    try:
        response = requests.put(
            f"{API_URL}/api/v1/vehicles/{vehicle_id}",
            json=data
        )
        if response.status_code == 200:
            print(f"  {vehicle_id}: lat={lat:.4f}, lon={lon:.4f}")
        else:
            print(f"  {vehicle_id}: HTTP {response.status_code}")
    except Exception as e:
        print(f"  {vehicle_id}: {e}")

def run_simulation():
    print("IoT Simulator Starting...")
```

```
    print(f"API: {API_URL}")
    print(f"Bins: {len(BIN_IDS)}")
    print("-" * 50)

    cycle = 1
    while True:
        print(f"\n Cycle #{cycle} — {datetime.now().strftime('%H:%M:%S')}")

        for bin_id in BIN_IDS:
            # Σταδιακή αύξηση fill level
            fill_levels[bin_id] += random.uniform(1, 8)
            if fill_levels[bin_id] > 100:
                fill_levels[bin_id] = random.uniform(5, 15)

            send_sensor_data(bin_id, fill_levels[bin_id])
            time.sleep(0.2)
        # GPS tracking οχημάτων
        print(f"\n Updating vehicle GPS positions...")
        for vehicle_id in ACTIVE_VEHICLES:
            new_pos = update_vehicle_position(vehicle_id)
            if new_pos:
                send_vehicle_gps(vehicle_id, new_pos["lat"],
new_pos["lon"])
                time.sleep(0.1)
        cycle += 1
        print(f"\n Waiting 30 seconds...")
        time.sleep(30)

if __name__ == "__main__":
    run_simulation()
```

5.7. Ανάπτυξη κώδικα εφαρμογής Smart Waste Management System

Το Smart Waste Management System είναι ένα ολοκληρωμένο πληροφοριακό σύστημα διαχείρισης απορριμμάτων που αναπτύχθηκε για τον Δήμο Μαλεβιζίου της Κρήτης. Το σύστημα αξιοποιεί τεχνολογίες IoT (Internet of Things), επεξεργασία δεδομένων σε πραγματικό χρόνο, αλγορίθμους βελτιστοποίησης διαδρομών και σύγχρονα web frameworks για την αυτοματοποίηση και βελτίωση της διαδικασίας αποκομιδής απορριμμάτων.

Η ανάπτυξη καλύπτει τρεις κύριες περιπτώσεις χρήσης: την πιστοποίηση χρηστών (ΠΧ1), τη βελτιστοποίηση διαδρομής αποκομιδής (ΠΧ2) και την πρόσβαση πολιτών σε πύλη αναφορών (ΠΧ3). Ο Δήμος Μαλεβιζίου διαθέτει 35 έξυπνους κάδους κατανεμημένους σε 6 γεωγραφικές ζώνες και 14 απορριμματοφόρα οχήματα.

5.7.1. Στόχοι Συστήματος

- Παρακολούθηση πληρότητας κάδων σε πραγματικό χρόνο μέσω αισθητήρων IoT
- Αυτόματη βελτιστοποίηση διαδρομών αποκομιδής με αλγόριθμο Nearest Neighbor TSP
- Ειδοποίηση οδηγών απορριμματοφόρων μέσω CollectionGarbageTruck (PWA) εφαρμογής
- Παροχή εφαρμογής Citizen-portal πύλης αναφορών για τους πολίτες του Δήμου
- Διαχείριση ζωνών, οχημάτων και χρηστών από κεντρικό dashboard της Smart Waste Management System - εφαρμογής.

5.7.2. Γεωγραφικές Ζώνες Δήμου Μαλεβιζίου

Δημιούργησα τις εξής γεωγραφικές ζώνες με ενδεικτικά μερικούς κάδους σε κάθε ζώνη σύνολο κάδων 35 και οχήματα (απορριμματοφόρα) 14 ενεργά.

Κωδικός	Περιοχή	Κάδοι	Οχήματα	Κύρια Οχήματα
GAZ	Γάζι	7	5	VEH-001, 007, 008, P01, P04
AMM	Αμμουδάρα	7	5	VEH-002, 009, 010, P02, P03
APE	Αγία Πελαγία	7	1	VEH-003
TYL	Τύλισος	6	1	VEH-004
KRO	Κρουσώνας	4	1	VEH-005

Κωδικός	Περιοχή	Κάδοι	Οχήματα	Κύρια Οχήματα
ROD	Ροδιά	4	1	VEH-006

Table 10 Γεωγραφικές Ζώνες στο ΠΣ

Για την ανάπτυξη της εφαρμογής Smart Waste Management System χρησιμοποιήθηκε το λογισμικό Visual Studio Code (VS Code) και έγινε χρήση όπως βλέπετε παρακάτω στο πίνακα τα στρώματα της αρχιτεκτονικής που χρησιμοποιήθηκαν.

Το σύστημα ακολουθεί αρχιτεκτονική τριών επιπέδων (3-tier architecture) με σαφή διαχωρισμό παρουσίασης, επιχειρηματικής λογικής και δεδομένων. Επιπλέον, περιλαμβάνει IoT στρώμα για τη συλλογή δεδομένων από τους αισθητήρες των κάδων όπως προαναφέραμε στην προηγούμενη παράγραφο.

5.7.3. Στρώματα Αρχιτεκτονικής

Στρώμα	Τεχνολογία	Περιγραφή
Παρουσίασης (Frontend)	React 18 + TypeScript	Κεντρικό dashboard διαχείρισης
Παρουσίασης (PWA)	HTML5 + Leaflet.js	Εφαρμογή οδηγού CollectionGarbageTruck
Παρουσίασης (Portal)	HTML5 + CSS3 + Leaflet.js	Πύλη αναφορών πολιτών (citizen-portal)
Επιχειρηματικής Λογικής	Python FastAPI + SQLAlchemy	REST API με async υποστήριξη
Δεδομένων	PostgreSQL + TimescaleDB	Σχεσιακή DB με χρονοσειρές
Cache	Redis	Session management & caching
IoT	Python Simulator	Προσομοίωση 35 αισθητήρων κάδων και 11 GPS οχήματα

Table 11 Στρώματα Αρχιτεκτονικής του ΠΣ

5.8. Τεχνολογίες και Εργαλεία ΠΣ Smart Waste Management System

5.8.1. Backend

- FastAPI 0.100+ (Χρησιμοποιήσαμε ένα σύγχρονο async Python web framework)
- SQLAlchemy 2.0 (Εγκαταστήσαμε το ORM με async υποστήριξη)
- PostgreSQL 15 + TimescaleDB (Χρησιμοποιήσαμε τη PostgreSQL βάση δεδομένων με υποστήριξη χρονοσειρών timescaleDB)
- JWT (JSON Web Tokens) (Για πιστοποίηση χρηστών κάναμε χρήση του JSON Web Token)
- bcrypt (Οι κωδικοί πρόσβασης υλοποιήθηκαν με κρυπτογράφηση κωδικών πρόσβασης)
- OSMnx + NetworkX (Χρησιμοποιήθηκαν οι βιβλιοθήκες γράφου για τη βελτιστοποίηση των διαδρομών των απορριμματοφόρων)
- Docker + Docker Compose (Εγκατέστησα στον υπολογιστή μου το Docker Engine για να κάνει containerization («πακετάρισμα») των containers (των περιεχομένων της εφαρμογής στη μνήμη. Το Docker Engine χρησιμοποιεί το WSL (Windows Subsystem for Linux) ως τη μηχανή Server του backend της εφαρμογής μας για να τρέχει τα Linux containers με ταχύτητα και σταθερότητα.)

5.8.2. Frontend

- React 18 + TypeScript (Το κεντρικό dashboard γράφτηκε με γλώσσα TypeScirpt κάνοντας χρήση της βιβλιοθήκης React για πιο γρήγορα αποτελέσματα)
- React-Leaflet (Χρησιμοποιήθηκε το Leaflet για την απόδοση των διαδραστικών χαρτών στην εφαρμογή μας μέσω του OpenStreetMap)
- Recharts (Γραφήματα bars, line charts)
- Flaticon UIcons (Για τα εικονίδια προσβασιμότητας σε όλη την εφαρμογή χρησιμοποιήθηκαν τα Flaticon)
- Html2canvas + jsPDF (Για την εξαγωγή αναφορών σε μορφή pdf)
- CSS Modules (Δημιουργήσαμε ξεχωριστά αρχεία στυλ ανά component)

5.8.3. IoT & Επικοινωνία

- Python IoT Simulator (Για την προσομοίωση της εφαρμογής μας δημιουργήθηκε κώδικας με χρήση γλώσσας Python 35 αισθητήρων, 11 GPS οχημάτων)

- REST API polling (Έγινε χρήση του rest api polling για την ανανέωση δεδομένων κάθε 30 δευτερόλεπτα)
- Geolocation API (Έγινε χρήση του Geolocation API για την άμεση ενεργοποίηση του GPS tracking των οχημάτων μέσω PWA της εφαρμογής CollectionGarbageTruck που είναι αποκλειστικά για τους οδηγούς των απορριμματοφόρων)

5.9. Δομή Αρχείων του ΠΣ Smart Waste management System

Παρακάτω βλέπετε πως είναι δομημένη η αρχιτεκτονική του συστήματος τοπικά στον υπολογιστή.

```
smart-waste-management/
├── backend/
│   └── app/
│       ├── main.py              # FastAPI εφαρμογή
│       ├── core/                # config, database, security
│       └── api/routes/          # API endpoints
│           ├── auth.py          # ΠΧ1: Πιστοποίηση
│           ├── optimization.py  # ΠΧ2: Βελτιστοποίηση
│           ├── citizen_portal.py # ΠΧ3: Πύλη Πολιτών
│           ├── bins.py, sensors.py, fillness.py
│           ├── vehicles.py, zones.py, reports.py
│           └── smart_dispatch.py
├── frontend/src/
│   ├── App.tsx + App.css
│   ├── LoginPage.tsx + LoginPage.css
│   ├── FillnessView.tsx, MapView.tsx
│   ├── RoutesView.tsx + RoutesView.css
│   ├── VehiclesView.tsx, TrackingView.tsx
│   ├── ZonesView.tsx, DispatchView.tsx
│   ├── ReportsView.tsx
│   └── CitizenReportsView.tsx + CitizenReportsView.css
├── driver-app/
│   ├── index.html               # PWA Οδηγού
│   └── driver-app.css
├── citizen-portal/
│   ├── citizen-portal.html      # Πύλη Πολιτών
│   └── citizen-portal.css
├── iot_simulator.py
├── docker-compose.yml
└── migration_v2.sql ... migration_v6.sql
```

5.10. Σχεδιασμός Βάσης Δεδομένων

Η βάση δεδομένων υλοποιείται με PostgreSQL 15 και την επέκταση TimescaleDB για αποδοτική αποθήκευση χρονοσειρών (sensor data, route history). Χρησιμοποιείται Docker container στη θύρα 5433.

5.10.1. Πίνακες Βάσης Δεδομένων

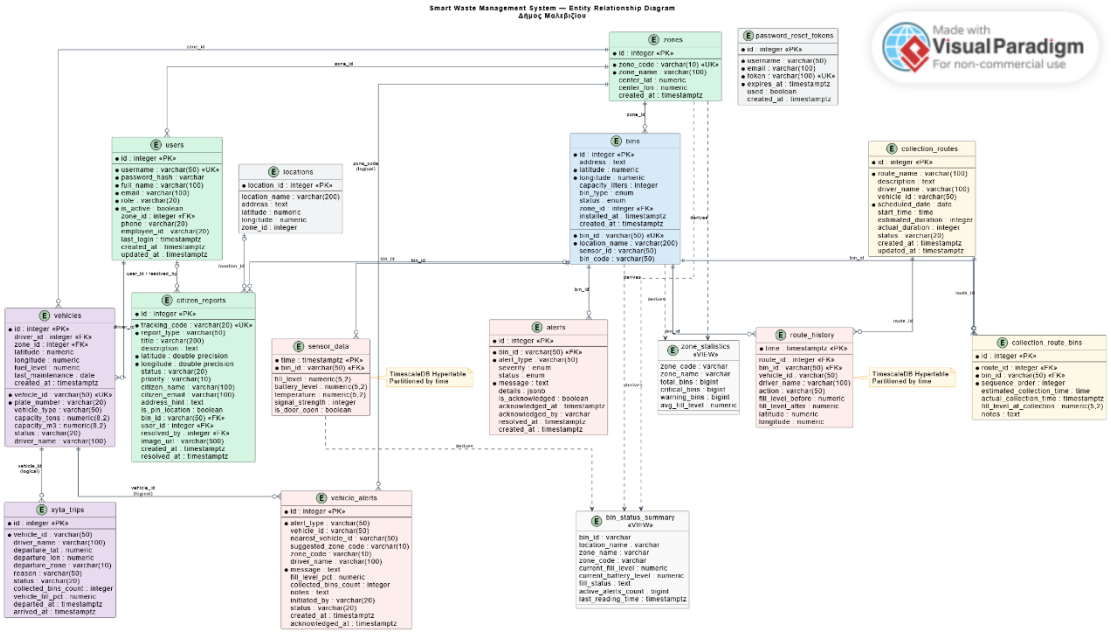
Πίνακας	Περιγραφή	Κύρια Πεδία
bins	Έξυπνοι κάδοι απορριμμάτων	bin_id, zone_id, sensor_id, location
sensor_data	Μετρήσεις αισθητήρων (TimescaleDB)	bin_id, fill_level, battery, timestamp
zones	6 γεωγραφικές ζώνες Δήμου	zone_code, zone_name, center_lat/lon
vehicles	Απορριμματοφόρα οχήματα	vehicle_id, zone_id, driver_name, status
users	Χρήστες συστήματος	username, password_hash, role, email
collection_routes	Αποθηκευμένες διαδρομές ΠΧ2	route_name, vehicle_id, scheduled_date
collection_route_bins	Κάδοι ανά διαδρομή	route_id, bin_id, sequence_order
vehicle_alerts	Ειδοποιήσεις οδηγών	alert_type, vehicle_id, message, status
xyta_trips	Μεταφορές στο ΧΥΤΑ	vehicle_id, departure_zone, collected_bins
citizen_reports	Αναφορές πολιτών ΠΧ3	tracking_code, report_type, status
password_reset_tokens	Tokens επαναφοράς κωδικού	token, username, expires_at
route_history	Ιστορικό διαδρομών (TimescaleDB)	route_id, vehicle_id, timestamp

Table 12 Πίνακες ΒΔ

5.10.2. Database Views (Προβολές Βάσης Δεδομένων)

- **bin_status_summary** (Οι προβολές Views συνδυάζουν τους πίνακες bins, sensor_data, zones για εύκολη ανάγνωση κατάστασης κάδων)
- **zone_statistics** (Οι προβολές Views zone_statistics συνδυάζουν στατιστικά ανά ζώνη (κρίσιμοι κάδοι, μέση πληρότητα)

Το νέο σχήμα της Βάσης Δεδομένων με τα στοιχεία μέχρι στιγμής της υλοποίησης της εφαρμογής το οποίο σχεδιάστηκε με το λογισμικό Visual Paradigm είναι το παρακάτω:



Εικόνα 11 Το νέο σχήμα της ΒΔ

5.10.3. Παραμετροποίηση Σύνδεσης

```
Host: localhost:5433
Database: wastebin_management
User: admin / Password: ************
Redis: localhost:6379 / Password: redis123
```

5.11. Τεκμηρίωση REST API

Το backend εκθέτει REST API μέσω FastAPI στη διεύθυνση http://localhost:8000. Η τεκμηρίωση Swagger είναι διαθέσιμη στο /docs και το ReDoc στο /redoc.

5.11.1. Πιστοποίηση (Auth)

Με τη μέθοδο get/post στέλνουμε στη FAST API της εφαρμογής τα αιτήματα για την πιστοποίηση του χρήστη του συστήματος.

Method	Endpoint	Περιγραφή
GET	/api/v1/auth/captcha	Δημιουργία captcha για login
POST	/api/v1/auth/login	Σύνδεση χρήστη (επιστρέφει JWT token)
POST	/api/v1/auth/password-reset-request	Αίτημα επαναφοράς κωδικού με username
GET	/api/v1/auth/password-reset-validate/{token}	Επαλήθευση reset token
POST	/api/v1/auth/password-reset-confirm	Ορισμός νέου κωδικού
GET	/api/v1/auth/users	Λίστα χρηστών (admin)
POST	/api/v1/auth/users	Δημιουργία νέου χρήστη

Table 13 Πιστοποίηση χρηστών ΠΣ

5.11.2. Κάδοι & Αισθητήρες

Με τη μέθοδο get / post στέλνουμε στα fast api της εφαρμογής τα αιτήματα στο αντίστοιχο endpoint για τη εμφάνιση των δεδομένων στην εφαρμογή μας από τους κάδους και τους αισθητήρες.

Method	Endpoint	Περιγραφή
GET	/api/v1/bins	Λίστα όλων των κάδων με κατάσταση
POST	/api/v1/bins/{id}/collect	Καταγραφή αποκομιδής κάδου
GET	/api/v1/fillness	Δεδομένα πληρότητας για γράφημα
GET	/api/v1/fillness/bins	Κατάσταση κάδων με φίλτρα
POST	/api/v1/sensors/readings	Αποστολή μέτρησης αισθητήρα (IoT)

Method	Endpoint	Περιγραφή
GET	/api/v1/zones	Στατιστικά ζωνών
GET	/api/v1/zones/{code}/bins	Κάδοι συγκεκριμένης ζώνης

Table 14 Πίνακας εμφάνισης δεδομένων ΠΣ

5.11.3. Βελτιστοποίηση Διαδρομής (ΠΧ2)

Με τη μέθοδο get στέλνουμε στο FAST API της εφαρμογής τον υπολογισμό της βέλτιστης διαδρομής και το ιστορικό των αποθηκευμένων διαδρομών.

Εδώ θα ήθελα να αναφέρων ότι το ΠΣ μας υποστηρίζει OSMnx αλγοριθμική αρχιτεκτονική για τον υπολογισμό πραγματικών οδικών αποστάσεων, αλλά λόγω περιορισμών του τοπικού δικτύου ίντερνετ χρησιμοποιείται ο αλγόριθμος **Haversine** (αεροπορική απόσταση) ως fallback, ο οποίος παρέχει αξιόπιστα αποτελέσματα.

Το ΠΣ για τη βέλτιστη διαδρομή υλοποιείται με τον αλγόριθμο **Nearest Neighbor TSP Heuristic,** ο οποίος είναι ένας ευρετικός αλγόριθμος για Travelling Salesman Problem (TSP).

Για τον υπολογισμό των αποστάσεων πάνω στο χάρτη χρησιμοποιούμε το τύπο Haversine (Great Circle Distance) Κυκλικής διαδρομής, ο οποίος μας παρέχει αξιόπιστα αποτελέσματα για τις αποστάσεις εντός του Δήμου Μαλεβιζίου (<15km).

Η αρχιτεκτονική του συστήματος της εφαμοργής προβλέπει ενσωμάτωση OSMnx για πραγματικές οδικές αποστάσεις σε μελλοντική επέκταση κι έκδοση της εφαρμογής, με caching mechanism για offline λειτουργία (Εναλλακτική Ροή 2) της λεκτικής περιγραφής ΠΧ2 όπως αναφέρθηκε σε προηγούμενη παράγραφο.

Method	Endpoint	Περιγραφή
GET	/api/v1/optimization/route	Υπολογισμός βέλτιστης διαδρομής (TSP)
GET	/api/v1/optimization/routes-history	Ιστορικό αποθηκευμένων διαδρομών
GET	/api/v1/optimization/osm-status	Κατάσταση OSMnx cache

Table 15 Πίνακας με μεθόδους βέλτιστης διαδρομής

Παράμετροι GET /api/v1/optimization/route:

- threshold (float, default: 70.0) (Δημιούργησα ένα κατώφλι πληρότητας κάδων 70%)
- zone_code (string, optional) : (Δημιούργησα ένα φίλτρο ζώνης (GAZ/AMM/APE/TYL/KRO/ROD) για κάθε περιοχή στο χάρτη έτσι ώστε να είναι πιο χρήσιμη και ευανάγνωστη η κάθε περιοχή ποιους κάδους εμπεριέχει στη ζώνη της)
- vehicle_id (string, optional) (Ενδεικτικά δημιούργησα με το vehicle_id να μπορεί ο διαχειριστής του συστήματος να στέλνει ειδοποίηση σε κάθε οδηγό για τυχόν αποκομιδή κάποιου κάδου ή να αναφέρει στο σύστημα της εφαρμογής του οδηγού τυχόν αναφορά σε κάποιο πρόβλημα και αντίστροφα να μπορεί και ο οδηγός να στέλνει ανάλογες ειδοποιήσεις)
- save_route (boolean, default: false) : (αποθήκευση στο collection_routes)

5.12. Αποστολή & Οχήματα

Με τη μέθοδο get/post/put ενημερώνουμε το FAST API της εφαρμογής και τα αντίστοιχα endpoints για τα οχήματα της εφαρμογής της κατάστασής τους, τη θέση τους μέσω GPS τόσο τη κεντρική εφαρμογή μας όσο και την εφαρμογή για τους οδηγούς των απορριμματοφόρων (οχημάτων) CollectionGarbageTruck καθώς και τις ειδοποιήσεις που στέλνει ο χειριστής του συστήματος προς τους οδηγούς των οχημάτων και οι ίδιοι οι οδηγοί μέσω του δικού τους συστήματος.

Method	Endpoint	Περιγραφή
GET	/api/v1/vehicles	Λίστα οχημάτων με κατάσταση
PUT	/api/v1/vehicles/{id}	Ενημέρωση θέσης GPS οχήματος
GET	/api/v1/tracking/vehicles	Real-time tracking οχημάτων
GET	/api/v1/dispatch/status	Κατάσταση ζωνών/οχημάτων
POST	/api/v1/dispatch/driver-action	Ενέργεια οδηγού (αποκομιδή, βλάβη)

Method	Endpoint	Περιγραφή
POST	/api/v1/dispatch/xyta-trip	Καταγραφή μεταφοράς στο ΧΥΤΑ
GET	/api/v1/dispatch/alerts	Ειδοποιήσεις (φίλτρα: vehicle_id, zone_code)
PUT	/api/v1/dispatch/alerts/{id}/acknowledge	Επιβεβαίωση ειδοποίησης
GET	/api/v1/dispatch/zone-vehicles/{zone}	Οχήματα ζώνης για PWA

Table 16 Πίνακας κατάστασης οχημάτων του CollectionGarbageTruck

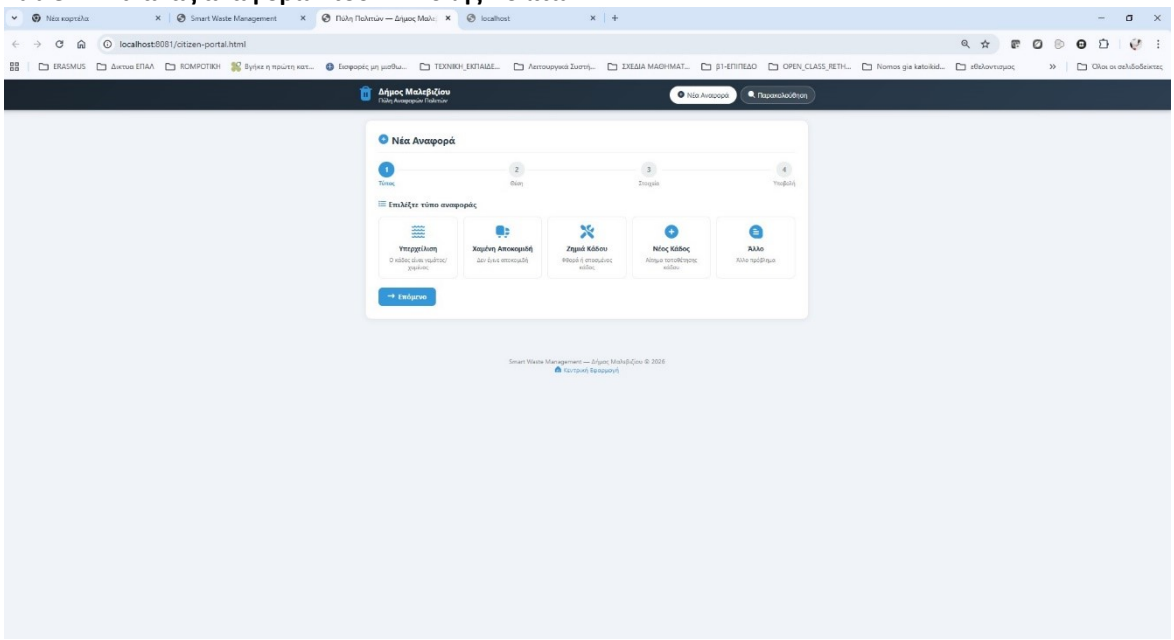
5.13. Πύλη Πολιτών (ΠΧ3)

Στη λεκτική Περιγραφή ΠΧ3 μιλήσαμε για την δημιουργία της πύλης των πολιτών του δήμου, όπου θα μπορούν να στέλνουν αναφορές για τυχόν προβλήματα που παρατηρούν και αφορούν την αποκομιδή των απορριμμάτων της περιοχής του δήμου τους.

Έτσι υλοποιήσαμε και αυτή την ξεχωριστή εφαρμογή την λεγόμενη πύλη των πολιτών citizen-portal όπου με τη μέθοδο get/post/put στο FAST API της εφαρμογής μας ο πολίτης μπορεί να υποβάλει αναφορά στο πληροφοριακό σύστημα της εφαρμογής μας και να ειδοποιείται ο κεντρικός διαχειριστής του συστήματος για τις αναφορές των πολιτών. Επίσης, η εφαρμογή στέλνει τις αναφοράς μέσω μιας φόρμας που συμπληρώνει ο πολίτης ανάλογα το πρόβλημα που έχει εντοπίσει και μπορεί επίσης με τον κωδικό που του δίνεται να παρακολουθήσει την αναφορά του σε τη στάδιο βρίσκεται και να έχει ικανοποιηθεί.

Method	Endpoint	Περιγραφή
GET	/api/v1/citizen/bins	Κάδοι για χάρτη πολιτών
POST	/api/v1/citizen/reports	Υποβολή αναφοράς (multipart/form-data)
GET	/api/v1/citizen/reports/{tracking_code}	Παρακολούθηση αναφοράς
GET	/api/v1/citizen/reports	Λίστα αναφορών για admin
PUT	/api/v1/citizen/reports/{id}	Ενημέρωση κατάστασης αναφοράς

Table 17 Πίνακας αναφορών του ΠΣ Πύλης Πολιτών



Εικόνα 12 ΠΣ Πύλης Πολιτών

6. Υλοποίηση Εφαρμογής και Περιπτώσεων Χρήσης

6.1. ΠΧ1 - Πιστοποίηση Χρήστη

Η ΠΧ1 υλοποιεί την ασφαλή πρόσβαση στο σύστημα με πολλαπλά επίπεδα ασφαλείας.

Βασική Ροή

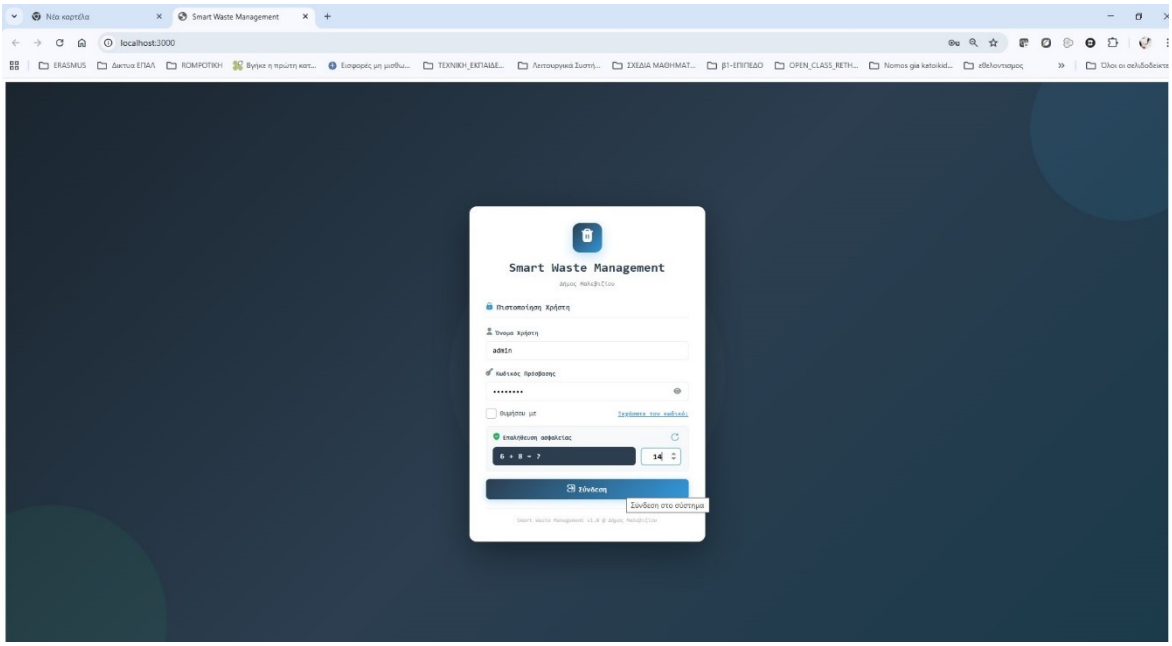
1. Το ΠΣ εμφανίζει φόρμα σύνδεσης με πεδία username/password και captcha επαλήθευσης
2. Ο χρήστης εισάγει τα στοιχεία του και συμπληρώνει το αριθμητικό captcha
3. Ο χρήστης επιλέγει «Θυμήσου με» (30 ημέρες localStorage) ή όχι (sessionStorage)
4. Το ΠΣ επαληθεύει τα στοιχεία μέσω bcrypt + JWT και εμφανίζει το κεντρικό dashboard

Εναλλακτικές Ροές

- Ροή 1 (Ανάκτηση Κωδικού): Ο χρήστης εισάγει username γίνεται αποστολή email με σύνδεσμο reset και ζητείται ορισμός νέου κωδικού (2 φορές) και τέλος γίνεται επιστροφή στη σύνδεση
- Ροή 2 (Χωρίς Remember Me): Το token αποθηκεύεται στο sessionStorage και διαγράφεται στο κλείσιμο του browser
- Ροή 3 (Λάθος Στοιχεία): Εμφάνιση μηνύματος σφάλματος, ανανέωση captcha, επιστροφή στη φόρμα

Τεχνική Υλοποίηση

- bcrypt με cost factor 12 για hashing κωδικών
- JWT tokens με configurable διάρκεια (8 ώρες ή 30 ημέρες)
- password_reset_tokens πίνακας με 2-ωρη λήξη
- SMTP email αποστολή (Gmail App Password)
- Captcha server-side αποθήκευση σε memory store



Εικόνα 13 ΠΣ Πιστοποίηση Χρήστη Συστήματος

6.2. ΠΧ2 - Βελτιστοποίηση Διαδρομής

Η ΠΧ2 υλοποιεί αυτόματη δημιουργία βέλτιστης διαδρομής αποκομιδής με βάση την πληρότητα των κάδων.

Βασική Ροή

1. Το σύστημα εντοπίζει κάδους με πληρότητα >= fill_threshold (default: 70%)
2. Ο διαχειριστής επιλέγει «Δημιουργία Διαδρομών» από το dashboard
3. Ορίζει ζώνη (zone_id) και κατώφλι πληρότητας (fill_threshold)
4. Ορίζει προαιρετικά οδηγό για αποστολή ειδοποίησης
5. Εκτελείται ο αλγόριθμος Nearest Neighbor TSP Heuretic με αποστάσεις Haversine
6. Η διαδρομή αποθηκεύεται στον πίνακα collection_routes και collection_route_bins
7. Προβολή στον Leaflet OpenStreetMap χάρτη με αριθμημένες στάσεις και dashed γραμμή
8. Αποστολή ειδοποίησης σε όλους τους ενεργούς οδηγούς της ζώνης μέσω vehicle_alerts

Εναλλακτικές Ροές

- Ροή 1 (Λίγοι Κάδοι): Αν βρεθούν <3 κάδοι με threshold 70%, το σύστημα μειώνει αυτόματα το threshold στο 50% και επανυπολογίζει. Εμφανίζεται badge μήνυμα για το «Auto-fallback».

- Ροή 2 (OSM Αποτυχία): Αν το Overpass API δεν είναι διαθέσιμο, χρησιμοποιείται ο αλγόριθμος Haversine (αεροπορική απόσταση) που παρέχει αξιόπιστα αποτελέσματα για αποστάσεις <15km.

Αλγόριθμος Nearest Neighbor TSP

Ο αλγόριθμος ξεκινά από το Δημαρχείο Μαλεβιζίου (35.3259°N, 25.0655°E) και επιλέγει επαναληπτικά τον κοντινότερο δεύτερο κάδο με βάση score = απόσταση / (πληρότητα/100), δίνοντας προτεραιότητα σε γεμάτους κοντινούς κάδους. Η πολυπλοκότητα είναι $O(N^2)$.

6.3. ΠΧ3 - Πύλη Πολιτών

Η ΠΧ3 παρέχει web εφαρμογή (http://localhost:8081/citizen-portal.html) για υποβολή αναφορών προβλημάτων κάδων από πολίτες χωρίς λογαριασμό.

Βασική Ροή

1. Ο πολίτης πλοηγείται στο citizen-portal.html
2. Επιλέγει τύπο αναφοράς: Υπερχείλιση / Χαμένη Αποκομιδή / Ζημιά Κάδου / Νέος Κάδος / Άλλο
3. Εντοπίζει τον κάδο στον Leaflet OpenStreetMap χάρτη και τον επιλέγει
4. Συμπληρώνει τίτλο, περιγραφή και προαιρετικά ανεβάζει φωτογραφία
5. Εισάγει όνομα και email (προαιρετικά για επιβεβαίωση)
6. Υποβολή - το σύστημα παράγει μοναδικό tracking_code μορφής CR1-XXXX
7. Εμφάνιση επιβεβαίωσης και αποστολή email στον πολίτη (αν δόθηκε)

Εναλλακτικές Ροές

- Ροή 1 (Καρφίτσωμα Θέσης): Αν ο πολίτης δεν βρίσκει τον κάδο, ενεργοποιεί το «Καρφίτσωμα Θέσης», κάνει κλικ στον χάρτη και υποβάλλει αίτημα τοποθέτησης νέου κάδου με is_pin_location=true

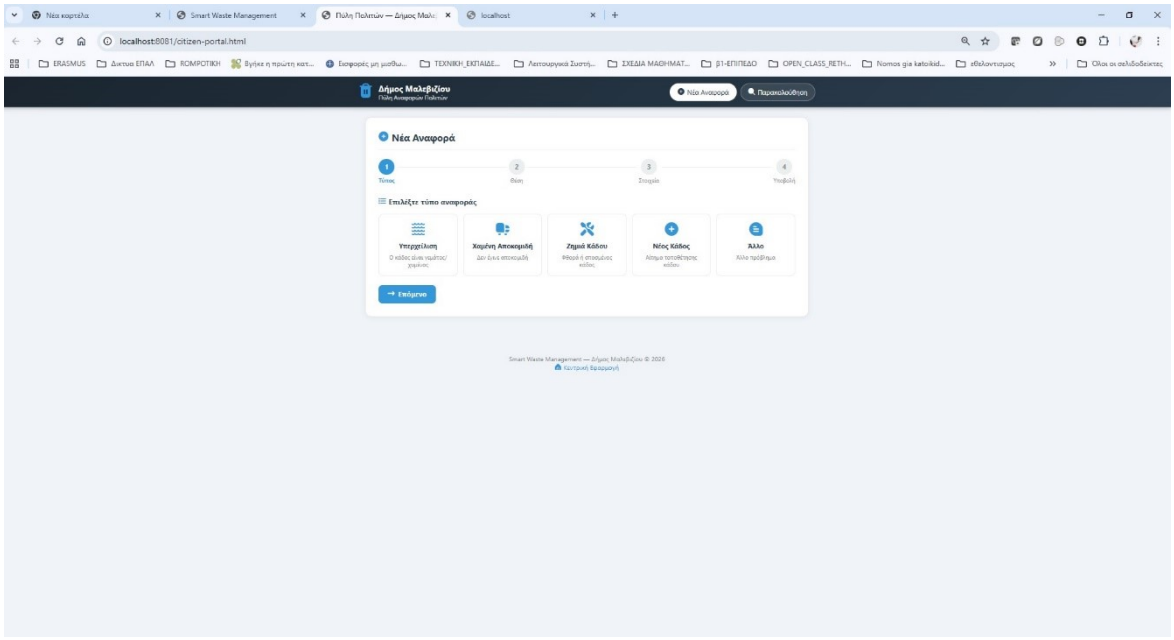
Παρακολούθηση Αναφοράς

Ο πολίτης μπορεί να παρακολουθεί την αναφορά του από το tab «Παρακολούθηση» εισάγοντας τον tracking_code. Εμφανίζεται timeline κατάστασης: Ανοιχτή , Σε Εξέλιξη, Επιλύθηκε.

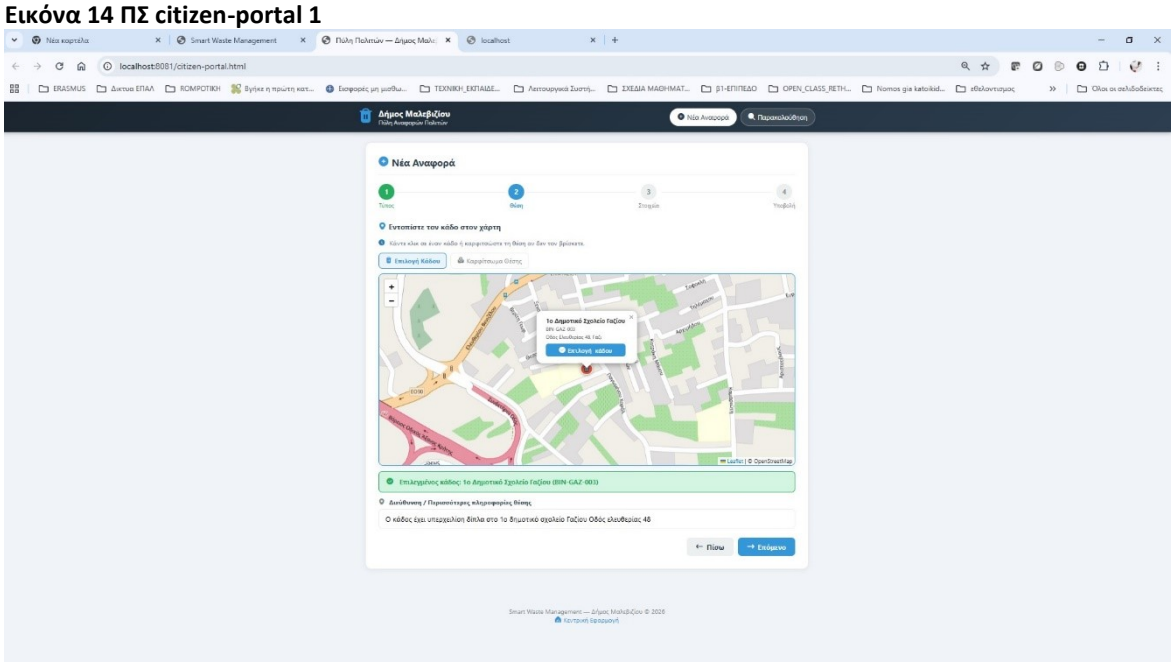
Διαχείριση από Admin

Το κεντρικό dashboard διαθέτει tab «Αναφορές Πολιτών» με πλήρη λίστα, φίλτρα ανά κατάσταση/τύπο, panel λεπτομερειών και δυνατότητα αλλαγής κατάστασης. Το κουμπί

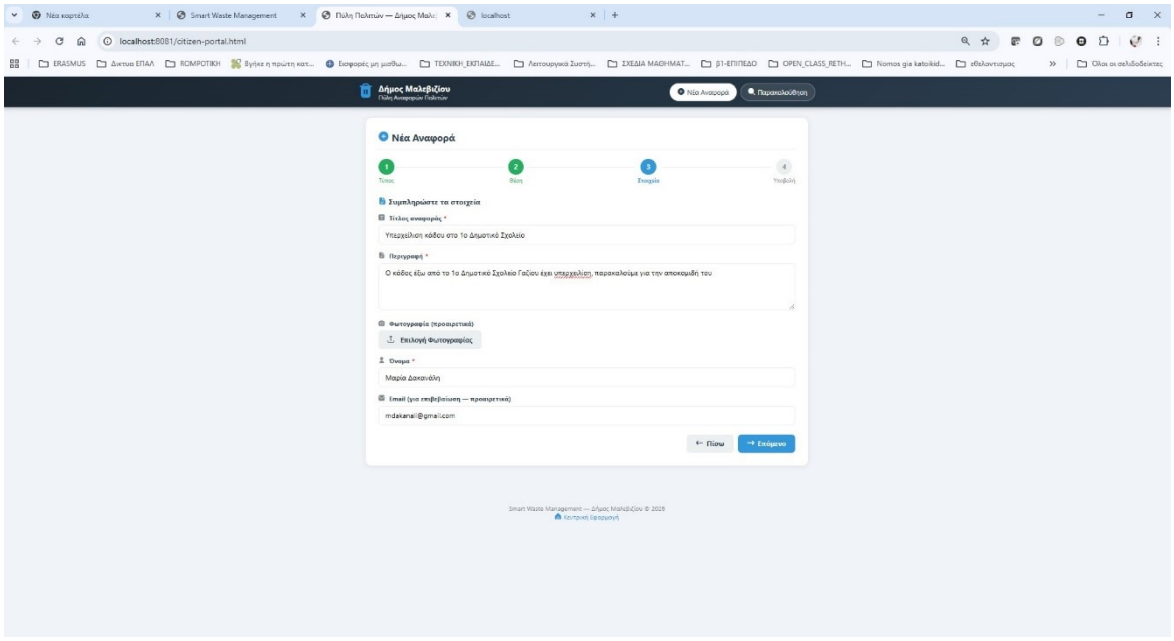
«Εμφάνιση στον χάρτη» ανοίγει modal με Leaflet OpenStreetMap χάρτη στη θέση της αναφοράς.



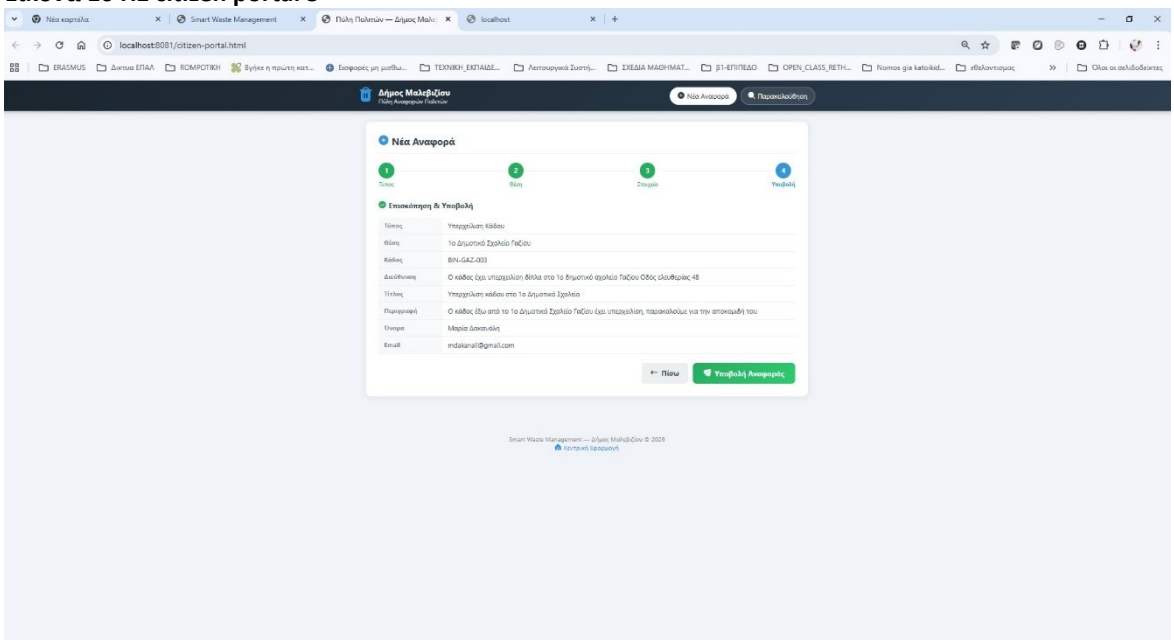
Εικόνα 14 ΠΣ citizen-portal 1



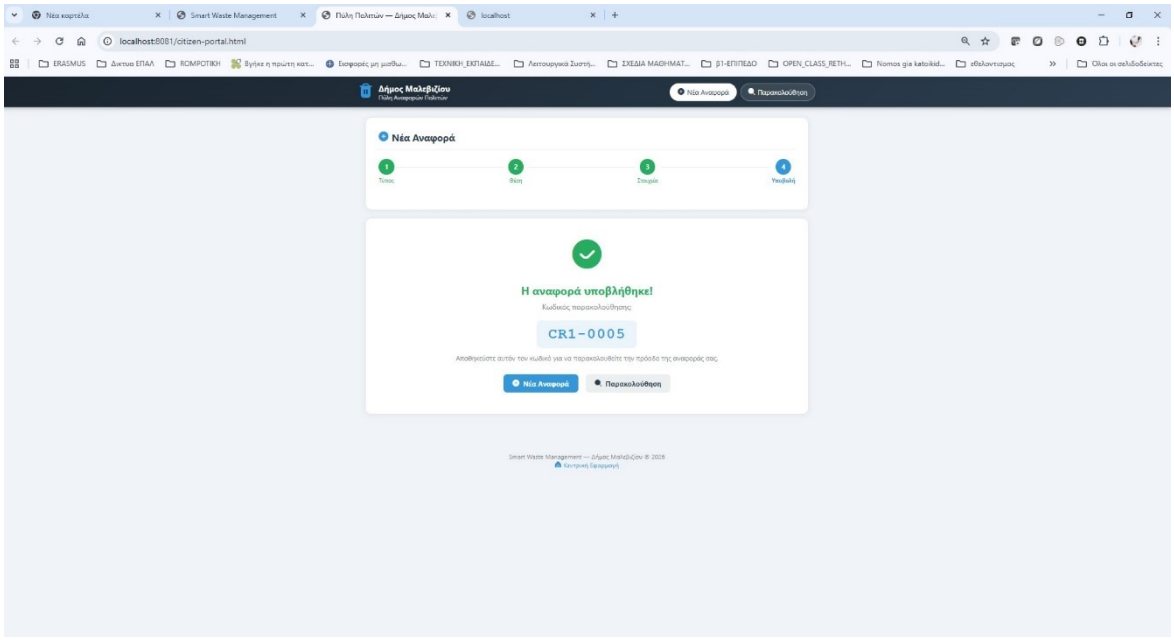
Εικόνα 15 ΠΣ citizen-portal 2



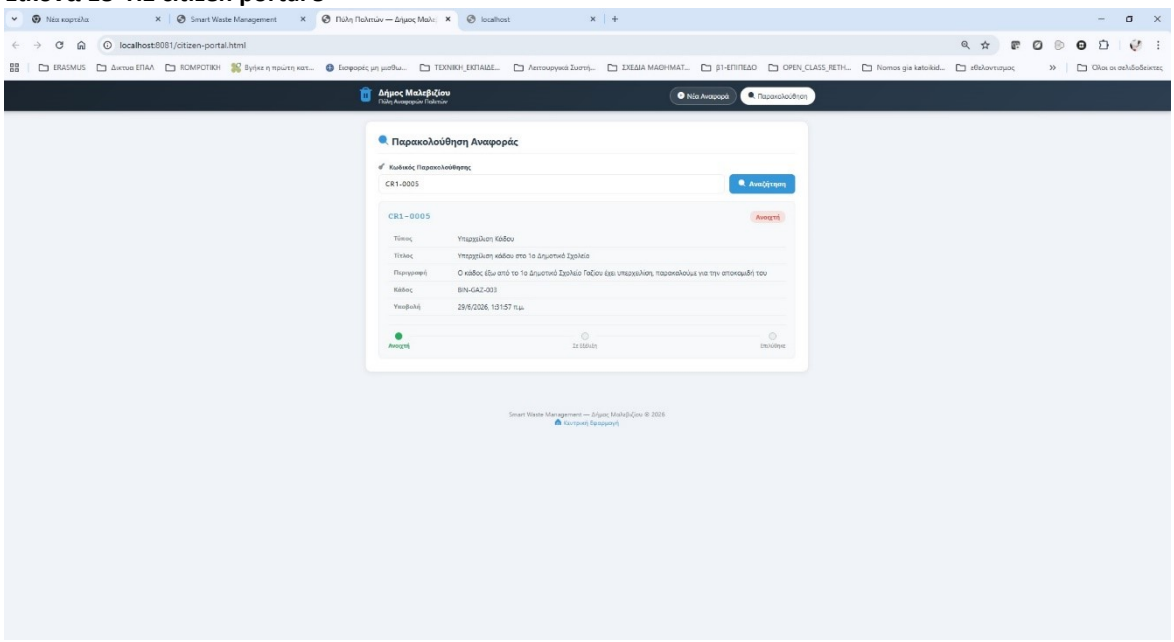
Εικόνα 16 ΠΣ citizen portal 3



Εικόνα 17 ΠΣ citizen portal 4



Εικόνα 18 ΠΣ citizen portal 5



Εικόνα 19 ΠΣ citizen portal 6

6.4. Οδηγίες Εγκατάστασης και Εκτέλεσης

6.4.1. Προαπαιτούμενα

Για την δημιουργία της εφαρμογής χρησιμοποιήσαμε και εγκαταστήσαμε μέσω Poweshell τα εξής προαπαιτούμενα μέσα στο C:\smart-waste-management\backend

- Python 3.12+
- Node.js v18+ και npm

- Docker Desktop

6.4.2. Εγκατάσταση

Βήμα 1: Εκκίνηση Docker (PostgreSQL και Redis)

```
cd C:\smart-waste-management
docker compose up -d timescaledb redis
```

Βήμα 2: Εγκατάσταση Backend

```
cd backend
python -m venv venv
venv\Scripts\activate
pip install -r requirements.txt
pip install osmnx networkx
```

Βήμα 3: Εγκατάσταση Frontend

```
cd .\frontend
npm install
```

6.5. Εκτέλεση Συστήματος

Για την εκτέλεση του συστήματος απαιτούνται 5 ξεχωριστά terminal παράθυρα:

Terminal	Εντολή	Αποτέλεσμα
1 Backend	cd backend venv\Scripts\activate python -m uvicorn app.main:app --reload --port 8000	API server στο http://localhost:8000
2 Frontend	cd frontend npm start	Dashboard στο http://localhost:3000

Terminal	Εντολή	Αποτέλεσμα
3 IoT Sim	python iot_simulator.py	Προσομοίωση 35 αισθητήρων
4 Driver PWA	cd driver-app python -m http.server 8080	PWA οδηγού στο http://localhost:8080
5 Citizen Portal	cd citizen-portal python -m http.server 8081	Πύλη πολιτών στο http://localhost:8081

Table 18 Πίνακας Εκτέλεσης του ΠΣ

6.6. Προεπιλεγμένοι Χρήστες

Οι χρήστες που έχουμε δημιουργήσει για το σύστημα μας είναι τρεις (3).

Username	Password	Ρόλος	Δικαιώματα
admin	Admin123	Admin	Πλήρης πρόσβαση σε όλες τις λειτουργίες Διαχειριστής του πληροφοριακού συστήματος
operator1	operator123	Operator	Προβολή & διαχείριση χωρίς admin
driver1	driver123	Driver	Πρόσβαση μόνο στο PWA οδηγού

Table 19 Χρήστες του ΠΣ Smart Waste Management System & CollectionGarbageTruck

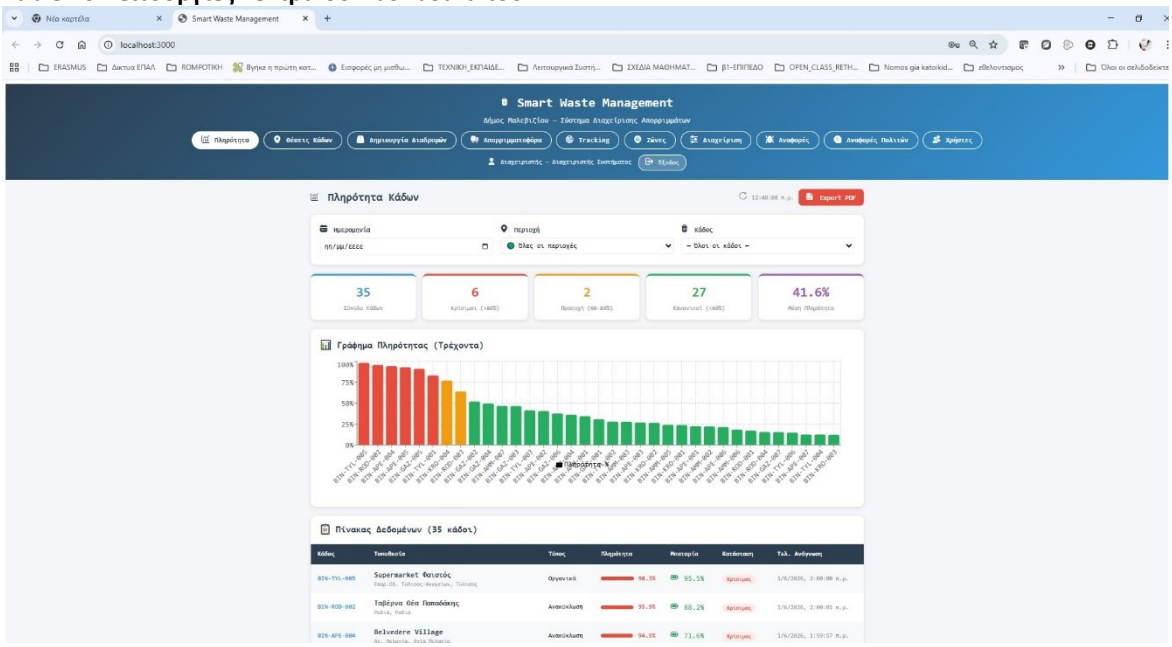
6.7. Λειτουργίες Κεντρικού Dashboard

Το κεντρικό dashboard (http://localhost:3000) παρέχει τις παρακάτω καρτέλες:

Καρτέλα	Λειτουργίες
Πληρότητα	Ιστόγραμμα πληρότητας κάδων, φίλτρα ανά ζώνη/τύπο, export PDF, real-time ανανέωση

Καρτέλα	Λειτουργίες
Θέσεις Κάδων	Leaflet χάρτης με 35 κάδους χρωματισμένους ανά πληρότητα, popup με στοιχεία, εστίαση από αναφορές πολιτών
Δημιουργία Διαδρομών	ΠΧ2: επιλογή ζώνης/threshold/οδηγού, TSP αλγόριθμος, αποθήκευση, χάρτης διαδρομής, ιστορικό
Απορριμματοφόρα	Λίστα 11 οχημάτων, κατάσταση, στοιχεία οδηγού, χωρητικότητα
Tracking	Real-time χάρτης θέσεων οχημάτων GPS, ιστορικό κινήσεων
Ζώνες	Στατιστικά 6 ζωνών, κρίσιμοι κάδοι, διαθέσιμα οχήματα
Διαχείριση	Smart dispatch: κατάσταση ζωνών, ειδοποιήσεις, χειρισμός βλαβών, ΧΥΤΑ
Αναφορές	Bug reports και τεχνικές αναφορές συστήματος
Αναφορές Πολιτών	ΠΧ3: λίστα αναφορών, φίλτρα, λεπτομέρειες, αλλαγή κατάστασης, modal χάρτη
Χρήστες	Δημιουργία/απενεργοποίηση χρηστών (μόνο admin)

Table 20 Λειτουργίες κεντρικού Dashboard του ΠΣ



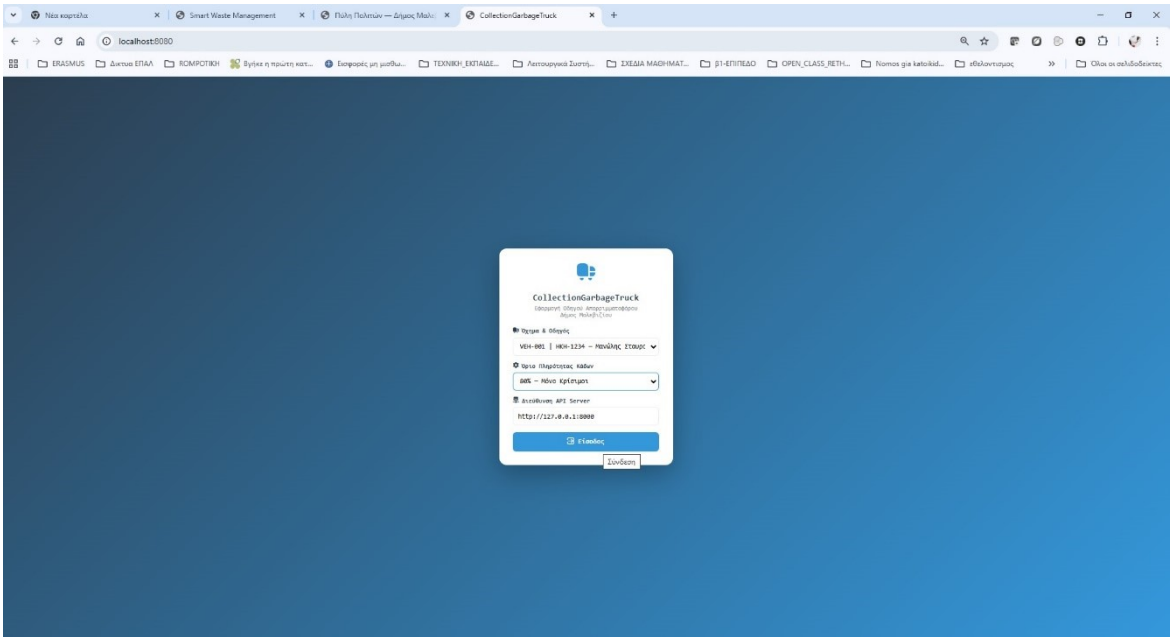
Εικόνα 20 ΠΣ Διαχείρισης Smart Waste Management

6.8. CollectionGarbageTruck - Εφαρμογή Οδηγού Οχήματος Απορριμματοφόρου

Η εφαρμογή οδηγού (http://localhost:8080) είναι Progressive Web App προσβάσιμη από κινητά τηλέφωνα. Παρέχει τις παρακάτω λειτουργίες:

6.8.1. Σύνδεση

- Επιλογή οχήματος ανά ζώνη από dropdown (GAZ: 5 οχήματα, AMM: 5, APE/TYL/KRO/ROD: 1)
- Ορισμός κατωφλίου πληρότητας και διεύθυνσης API server
- Αποθήκευση state σε sessionStorage με 8-ωρη ισχύ (βάρδια)



Εικόνα 21 ΠΣ CollectionGarbageTruck για οδηγούς

6.8.2. Καρτέλες

- Κάδοι: φόρτωση κάδων ζώνης, καταγραφή αποκομιδής, εμφάνιση πληρότητας οχήματος
- Χάρτης: Leaflet χάρτης με κάδους ζώνης, κατάσταση αποκομιδής
- Ειδοποιήσεις: ειδοποιήσεις ζώνης από κεντρικό σύστημα (route_assigned, zone_change)
- GPS: εκκίνηση/διακοπή GPS tracking, αποστολή θέσης, αναφορά βλάβης/ΧΥΤΑ

6.8.3. Διαχείριση Χωρητικότητας

Κάθε κάδος έχει χωρητικοτητα 1.100 λίτρα που αντιστοιχεί σε 1,1m³. Η εφαρμογή παρακολουθεί την πληρότητα του οχήματος και αποστέλλει αυτόματη ειδοποίηση ΧΥΤΑ όταν η πληρότητα υπερβεί το 85%.

Πληρότητα οχήματος = (κάδος που άδειασε * 1,1m³) / capacity_m3 * 100%

6.9. Ασφάλεια και Προσβασιμότητα

6.9.1. Ασφάλεια

- JWT authentication με configurable expiry (8h/30d)
- bcrypt password hashing (cost factor 12)
- Captcha για αποτροπή brute force
- Password reset tokens με 2-ωρη λήξη και single-use
- CORS configuration για production
- SQL injection protection μέσω SQLAlchemy parameterized queries
- Input validation μέσω Pydantic models

6.9.2. Προσβασιμότητα (WCAG)

- aria-label, aria-required, aria-live σε όλα τα interactive elements
- htmlFor/id σύνδεση labels-inputs
- semantic HTML: header, nav, main, footer, section
- aria-current="page" στο active navigation item
- aria-hidden="true" σε decorative icons
- role="alert" και aria-live="polite" σε error messages
- Keyboard navigation υποστήριξη
- inline styles όλα σε ξεχωριστά .css αρχεία

6.10. Συμπεράσματα και Μελλοντικές Επεκτάσεις

6.10.1. Συμπεράσματα

Το Smart Waste Management System για τον Δήμο Μαλεβιζίου υλοποιεί με επιτυχία ένα ολοκληρωμένο σύστημα έξυπνης διαχείρισης απορριμμάτων που καλύπτει πλήρως τις τρεις καθορισμένες περιπτώσεις χρήσης. Το σύστημα συνδυάζει σύγχρονες τεχνολογίες web

development, IoT προσομοίωση και αλγορίθμους βελτιστοποίησης σε μια αρχιτεκτονικά αρμονική λύση.

Ο αλγόριθμος Nearest Neighbor TSP παρέχει αξιόπιστες βέλτιστες διαδρομές για τις μικρές αποστάσεις του Δήμου Μαλεβιζίου, ενώ το auto-fallback σύστημα (50% threshold) εξασφαλίζει τη λειτουργία ακόμα και σε συνθήκες χαμηλής πληρότητας. Η πύλη πολιτών δημιουργεί κανάλι αμφίδρομης επικοινωνίας μεταξύ δημοτών και Δήμου.

6.10.2. Μελλοντικές Επεκτάσεις

- Ενσωμάτωση πραγματικών IoT αισθητήρων (αντί simulator)
- Machine Learning για πρόβλεψη πληρότητας κάδων
- Ενσωμάτωση OSMnx με τοπικό PBF αρχείο για offline routing
- Mobile app (React Native) για τους οδηγούς
- Ενσωμάτωση με σύστημα GIS Δήμου
- Dashboard analytics με ιστορικά στατιστικά και προβλέψεις
- Push notifications για real-time ειδοποιήσεις οδηγών
- Εναλλακτική Ροή 2 ΠΧ3: αυτόματη προσυμπλήρωση για εγγεγραμμένους πολίτες

Βιβλιογραφία

EU — Waste Framework Directive & EU waste law summary. Environment+1

Εθνικό Σχέδιο Διαχείρισης Αποβλήτων (ΕΣΔΑ 2020–2030). Ανοικτή Διακυβέρνηση

European Environment Agency — Greece municipal waste factsheet (2025 update / 2022 data). eea.europa.eu

ELSTAT — Waste statistics 2022 (προσωρινά στοιχεία). ΕΛΣΤΑΤ

European Commission staff working document / country fiche for Greece (SWD 2023) — αναλύει κύριες προκλήσεις. EUR-Lex

WWF / NGO αναλύσεις για την ελληνική διαχείριση αστικών αποβλήτων (σύντομη ανασκόπηση προβλημάτων υποδομών/δεδομένων). WWF Ελλάς

Assessing the readiness of Greek municipalities to manage bioplastics in municipal waste management streams - SpringerLink

Xu, L., He, W. & Li, S. (2014). Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics*, Vol. 10, No. 4, pp. 2233-2243.

Yang, Y., Wu, L., Yin, G., Li, L. and Zhao, H. (2017). A Survey on Security and Privacy Issues in Internet-of-Things, *IEEE Internet of Things Journal*, Vol. 4, No. 5, pp. 1250-1258.

Albino, V., Berardi, U., & Dangelico, R. M. (2015). Smart cities: Definitions, dimensions, performance, and initiatives. Journal of Urban Technology, 22(1), 3–21.

Komninos, N. (2013). Intelligent Cities: Innovation, Knowledge Systems and Digital Spaces. Routledge.

Caragliu, A., Del Bo, C., & Nijkamp, P. (2011). Smart cities in Europe. Journal of Urban Technology, 18(2), 65–82.

Batty, M. (2013). The New Science of Cities. MIT Press.

European Commission. (2020). European Smart Cities Model.

European Innovation Partnership on Smart Cities and Communities (EIP-SCC). (2019). Strategic Implementation Plan.

European Environment Agency (EEA). (2021). Digitalisation and the Urban Environment in Europe.

Allesch, A., & Brunner, P. H. (2014). Assessment methods for solid waste management: A literature review. Waste Management & Research, 32(6), 461–473.

Akhtar, A., Hannan, M., Basri, H., et al. (2017). Solid waste management using predictive system and IoT: A literature review. IEEE Access, 6, 1–16.

Suresh, M., & Manoharan, S. (2019). IoT-based real-time waste management system. Sensors and Actuators A, 295, 110–122.

Longo, F., Nicoletti, L., & Padovano, A. (2019). Smart waste management in cities: A review. Waste Management, 87, 741–754.

Enevo (2018). Smart Waste Data Solutions Report.

Bigbelly Inc. (2019). Smart Waste and Recycling System Whitepaper.

SmartBin. (2020). IoT Waste Management Technologies Overview.

Barcelona City Council. (2020). Barcelona Smart Waste Collection Strategy.

Amsterdam Smart City Program (ASC). (2021). Circular Economy & Smart Waste Projects.

Copenhagen Solutions Lab. (2020). IoT and Low-Power Networks for Urban Services.

Δήμος Τρικκαίων (2021). Ψηφιακός Μετασχηματισμός και Έξυπνες Υπηρεσίες.

ΙΤΕ – Ίδρυμα Τεχνολογίας και Έρευνας (2020). Έξυπνες Πόλεις και Ψηφιακές Υποδομές στην Ελλάδα.

ΣΕΒ – Σύνδεσμος Επιχειρήσεων & Βιομηχανιών (2021). Έξυπνες Πόλεις: Προοπτικές και Προκλήσεις στην Ελλάδα.

Νομοθεσία & Πολιτικές (Ελλάδα):

Ν. 4685/2020 – Εκσυγχρονισμός περιβαλλοντικής νομοθεσίας.

Ν. 4819/2021 – Ολοκληρωμένο πλαίσιο διαχείρισης αποβλήτων – Εναρμόνιση με οδηγία (ΕΕ) 2018/851.

ΕΣΔΑ 2020–2030 (Εθνικό Σχέδιο Διαχείρισης Αποβλήτων).

ΠΕΣΔΑ Κρήτης (2022).

Bibri, S. E. (2018). Smart sustainable cities: An integrated approach to urban sustainability and smart city technologies. Sustainable Cities and Society, 38, 510–527.

Neirotti, P., De Marco, A., et al. (2014). Current trends in Smart City initiatives: Some stylised facts. Cities, 38, 25–36.

Kourtit, K., Nijkamp, P. (2012). Smart Cities in Perspective. International Journal of Sustainable Development.

https://www.cn.ntua.gr/sites/default/files/2023-02/LoRaWAN.pdf

https://malevizi.gov.gr/wp-content/uploads/2018/05/ep1.pdf

https://www.esdak.gr/assets/uploads/files/%CE%A4%CE%A3%CE%94-%CE%94%CE%AE%CE%BC%CE%BF%CF%85-%CE%9C%CE%91%CE%9B%CE%95%CE%92%CE%99%CE%96%CE%99%CE%9F%CE%A5.pdf

https://malevizi.gov.gr/wp-content/uploads/2018/05/ep1.pdf

https://energy.ec.europa.eu/topics/clean-energy-transition/energy-and-smart-cities_en?prefLang=el

https://www.postgresql.org/docs/current/tutorial-views.html

https://www.postgresql.org/docs/18/plpython-data.html#PLPYTHON-DATA-TYPE-MAPPING

https://www.postgresql.org/docs/18/plpython-database.html#PLPYTHON-DATABASE-ACCESS-FUNCS

https://www.postgresql.org/docs/18/protocol-flow.html#PROTOCOL-FLOW-EXT-QUERY

https://cdn-uicons.flaticon.com/2.6.0/uicons-solid-rounded/css/uicons-solid-rounded.css

Docker Desktop

https://www.docker.com/products/docker-desktop/

PostgreSQL

https://www.postgresql.org/about/

TimescaleDB

https://timescaledb.org/

Complete Installation, Configuration, and Implementation Guide for TimescaleDB

https://timescaledb.org/complete-installation-configuration-and-implementation-guide-for-timescaledb/

pgAdmin

https://www.pgadmin.org/download/

PostgreSQL - Tutorial

https://www.w3schools.com/postgresql/index.php

ngrok

https://dashboard.ngrok.com/login

Wokwi

https://wokwi.com

Παράρτημα Α: «Κώδικας του backend του ΠΣ Smart Waste Management System»

Backend\app\main.py

```
"""
Smart Waste Management System — Δήμος Μαλεβιζίου
================================================
FastAPI Application Entry Point

Αρχιτεκτονική:
    - FastAPI + SQLAlchemy async
    - PostgreSQL / TimescaleDB (port 5433)
    - Redis (port 6379)
    - JWT Authentication
"""

from contextlib import asynccontextmanager

from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

from app.core.config import settings
from app.core.database import check_database_health, create_tables
from app.api.routes import (
    auth,
    bins,
    citizen_portal,
    fillness,
    optimization,
    reports,
    sensors,
    smart_dispatch,
    vehicles,
    zones,
)


# ---------------------------------------------------------------------------
# Lifespan — εκτελείται κατά την εκκίνηση και τον τερματισμό
# ---------------------------------------------------------------------------

@asynccontextmanager
async def lifespan(app: FastAPI):
    """Startup / shutdown hooks."""
```

```
    print(f"🔧 Environment : {settings.ENVIRONMENT}")
    print(f"🗄️  Database    : {settings.DATABASE_URL}")
    print(f"🐛 Debug mode  : {settings.DEBUG}")

    await create_tables()
    await check_database_health()

    yield  # ← εφαρμογή τρέχει εδώ

    print("👋 Shutting down...")


# ---------------------------------------------------------------------------
--
# Application factory
# ---------------------------------------------------------------------------
--

def create_app() -> FastAPI:
    """Δημιουργία και ρύθμιση του FastAPI instance."""

    app = FastAPI(
        title="Smart Waste Management API",
        description="Σύστημα Διαχείρισης Απορριμμάτων — Δήμος Μαλεβιζίου",
        version="1.0.0",
        docs_url="/docs",
        redoc_url="/redoc",
        lifespan=lifespan,
    )

    # --- CORS ---
    app.add_middleware(
        CORSMiddleware,
        allow_origins=settings.ALLOWED_ORIGINS,
        allow_credentials=True,
        allow_methods=["*"],
        allow_headers=["*"],
    )

    # --- Routers ---
    _register_routers(app)

    return app


def _register_routers(app: FastAPI) -> None:
    """Καταχώρηση όλων των API routers."""
```

```
    prefix = "/api/v1"
    tags_map = {
        auth:           "Auth",
        bins:           "Bins",
        citizen_portal: "Citizen",
        fillness:       "Fillness",
        optimization:   "Optimization",
        reports:        "Reports",
        sensors:        "Sensors",
        smart_dispatch: "Dispatch",
        vehicles:       "Vehicles",
        zones:          "Zones",
    }
    for module, tag in tags_map.items():
        app.include_router(module.router, prefix=prefix, tags=[tag])


# ------------------------------------------------------------------------
--
# Entry point
# ------------------------------------------------------------------------
--

app = create_app()
```

backend\app\api_init_.py

```
"""
API endpoints for Smart Waste Management System
"""

# This file makes the api directory a Python package
# Αυτό το αρχειο κατασκευάζει τον φάκελο api ως Python package (Πακέτο)
# που περιέχει αρχεία κώδικα που ανήκουν στην ίδια ομάδα api
```

backend\app\api\iot_endpoints.py

```
"""
app/api/iot_endpoints.py
========================
IoT endpoints για λήψη δεδομένων από αισθητήρες (ESP32 / Wokwi simulator).

Διαφορά από sensors.py:
    - sensors.py  → απλή αποστολή μέτρησης (χρησιμοποιείται από IoT
simulator)
    - iot_endpoints.py → πλούσιο IoT API με bulk, health check, device
status
```

```
Endpoints:
    POST /iot/sensor-data        → Αποστολή μέτρησης από αισθητήρα
    POST /iot/sensor-data/bulk   → Μαζική αποστολή μετρήσεων
    GET  /iot/bins/{bin_id}/status → Κατάσταση IoT συσκευής
    GET  /iot/health             → Health check IoT συστήματος
"""

import logging
from datetime import datetime, timedelta
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from pydantic import BaseModel, Field
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter(prefix="/iot", tags=["IoT"])

# Κατώφλια device status (λεπτά από τελευταία μέτρηση)
DEVICE_OFFLINE_MINUTES = 60
DEVICE_WARNING_MINUTES = 30


# ---------------------------------------------------------------------------
--
# Schemas
# ---------------------------------------------------------------------------
--

class SensorDataPayload(BaseModel):
    """Payload μέτρησης από IoT συσκευή (ESP32 / Wokwi)."""
    bin_id:           str
    fill_level:       float          = Field(ge=0, le=100)
    battery_level:    Optional[float] = Field(None, ge=0, le=100)
    distance_cm:      Optional[float] = Field(None, ge=0)
    temperature:      Optional[float] = None
    humidity:         Optional[float] = Field(None, ge=0, le=100)
    signal_strength:  Optional[float] = None
    latitude:         Optional[float] = None
    longitude:        Optional[float] = None
    device_type:      Optional[str]   = "wokwi"
    firmware_version: Optional[str]   = None
    timestamp:        Optional[datetime] = None
```

```
class SensorDataResponse(BaseModel):
    """Response αποστολής μέτρησης."""
    status:     str
    message:    str
    bin_id:     str
    timestamp:  datetime
    alert_created: Optional[str] = None


# ---------------------------------------------------------------------------
--
# Helpers
# ---------------------------------------------------------------------------
--

def _device_status(minutes_since_last: int) -> str:
    """Κατηγοριοποίηση κατάστασης IoT συσκευής βάσει χρόνου."""
    if minutes_since_last > DEVICE_OFFLINE_MINUTES:
        return "offline"
    if minutes_since_last > DEVICE_WARNING_MINUTES:
        return "warning"
    return "online"


async def _store_reading(
    db:      AsyncSession,
    payload: SensorDataPayload,
) -> str:
    """
    Αποθήκευση μέτρησης στο TimescaleDB και ενημέρωση GPS αν δόθηκαν
συντεταγμένες.

    Returns:
        alert_type αν δημιουργήθηκε ειδοποίηση, αλλιώς None.
    """
    # Έλεγχος ύπαρξης κάδου
    result = await db.execute(
        text("SELECT bin_id FROM bins WHERE bin_id = :bin_id"),
        {"bin_id": payload.bin_id},
    )
    if not result.fetchone():
        raise HTTPException(404, f"Κάδος {payload.bin_id} δεν βρέθηκε.")

    # Αποθήκευση στο TimescaleDB
    await db.execute(
        text("""
            INSERT INTO sensor_data
```

```
                (time, bin_id, fill_level, battery_level, temperature)
            VALUES
                (:time, :bin_id, :fill, :battery, :temp)
        """),
        {
            "time":    payload.timestamp or datetime.now(),
            "bin_id":  payload.bin_id,
            "fill":    payload.fill_level,
            "battery": payload.battery_level,
            "temp":    payload.temperature,
        },
    )

    # Ενημέρωση GPS αν δόθηκαν συντεταγμένες
    if payload.latitude and payload.longitude:
        await db.execute(
            text("UPDATE bins SET latitude = :lat, longitude = :lon WHERE
bin_id = :bin_id"),
            {"lat": payload.latitude, "lon": payload.longitude, "bin_id":
payload.bin_id},
        )

    # Ειδοποίηση αν fill_level ≥ 80%
    alert_type = None
    if payload.fill_level >= 90:
        alert_type = "critical"
    elif payload.fill_level >= 80:
        alert_type = "warning"

    if alert_type:
        # Έλεγχος αν υπάρχει ήδη ενεργή ειδοποίηση
        existing = await db.execute(
            text("""
                SELECT id FROM alerts
                WHERE bin_id = :bin_id
                  AND alert_type = :alert_type
                  AND is_acknowledged = FALSE
                LIMIT 1
            """),
            {"bin_id": payload.bin_id, "alert_type": f"bin_{alert_type}"},
        )
        if not existing.fetchone():
            await db.execute(
                text("""
                    INSERT INTO alerts (bin_id, alert_type, severity,
message, is_acknowledged)
```

```
                    VALUES (:bin_id, :alert_type, :severity, :message,
FALSE)
                """),
                {
                    "bin_id":     payload.bin_id,
                    "alert_type": f"bin_{alert_type}",
                    "severity":   alert_type,
                    "message":    (
                        f"Κάδος {payload.bin_id}: πληρότητα "
                        f"{payload.fill_level:.1f}% — απαιτείται
αποκομιδή."
                    ),
                },
            )

    await db.commit()
    return alert_type


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.post("/sensor-data", response_model=SensorDataResponse)
async def receive_sensor_data(
    payload: SensorDataPayload,
    db:      AsyncSession = Depends(get_db),
) -> SensorDataResponse:
    """
    Λήψη μέτρησης από IoT αισθητήρα (ESP32 / Wokwi).
    Αποθηκεύει στο TimescaleDB και δημιουργεί ειδοποίηση αν χρειάζεται.
    """
    logger.info(f"IoT reading: {payload.bin_id} @ {payload.fill_level}%")

    alert = await _store_reading(db, payload)

    return SensorDataResponse(
        status="success",
        message="Μέτρηση αποθηκεύτηκε επιτυχώς.",
        bin_id=payload.bin_id,
        timestamp=payload.timestamp or datetime.now(),
        alert_created=alert,
    )
```

```python
@router.post("/sensor-data/bulk")
async def receive_bulk_sensor_data(
    payloads: list[SensorDataPayload],
    db:       AsyncSession = Depends(get_db),
) -> dict:
    """
    Μαζική αποστολή μετρήσεων από πολλούς αισθητήρες.
    Κάθε μέτρηση επεξεργάζεται ανεξάρτητα.
    """
    logger.info(f"Bulk IoT: {len(payloads)} readings")

    results = []
    for payload in payloads:
        try:
            alert = await _store_reading(db, payload)
            results.append({
                "bin_id":  payload.bin_id,
                "status":  "success",
                "alert":   alert,
            })
        except Exception as exc:
            logger.error(f"Bulk error {payload.bin_id}: {exc}")
            results.append({
                "bin_id":  payload.bin_id,
                "status":  "error",
                "message": str(exc),
            })

    success = sum(1 for r in results if r["status"] == "success")
    return {
        "status":    "success",
        "total":     len(results),
        "succeeded": success,
        "failed":    len(results) - success,
        "results":   results,
    }


@router.get("/bins/{bin_id}/status")
async def get_bin_iot_status(
    bin_id: str,
    db:     AsyncSession = Depends(get_db),
) -> dict:
    """
    Κατάσταση IoT συσκευής για συγκεκριμένο κάδο.
    Εμφανίζει online/warning/offline βάσει χρόνου τελευταίας μέτρησης.
    """
```

```
    # Τελευταία μέτρηση
    result = await db.execute(
        text("""
            SELECT time, fill_level, battery_level,
                   signal_strength, temperature
            FROM sensor_data
            WHERE bin_id = :bin_id
            ORDER BY time DESC
            LIMIT 1
        """),
        {"bin_id": bin_id},
    )
    row = result.mappings().first()

    if not row:
        return {
            "bin_id":        bin_id,
            "device_status": "no_data",
            "message":       "Δεν υπάρχουν δεδομένα αισθητήρα.",
            "last_seen":     None,
        }

    # Χρόνος από τελευταία μέτρηση
    minutes_ago = int((datetime.now() -
row["time"].replace(tzinfo=None)).total_seconds() / 60)

    return {
        "bin_id":        bin_id,
        "device_status": _device_status(minutes_ago),
        "last_reading": {
            "timestamp":       str(row["time"]),
            "fill_level":      float(row["fill_level"]      or 0),
            "battery_level":   float(row["battery_level"]   or 0) if
row["battery_level"]   else None,
            "signal_strength": float(row["signal_strength"] or 0) if
row["signal_strength"] else None,
            "temperature":     float(row["temperature"]     or 0) if
row["temperature"]     else None,
        },
        "minutes_since_last_reading": minutes_ago,
    }


@router.get("/health")
async def iot_health_check(db: AsyncSession = Depends(get_db)) -> dict:
    """
    Health check του IoT συστήματος.
```

```
    Ελέγχει μετρήσεις τελευταίας ώρας και κάλυψη κάδων.
    """
    # Μετρήσεις τελευταίας ώρας
    readings_result = await db.execute(text("""
        SELECT COUNT(*) AS total FROM sensor_data
        WHERE time >= NOW() - INTERVAL '1 hour'
    """))
    readings_count = int(readings_result.scalar() or 0)

    # Σύνολο ενεργών κάδων
    bins_result = await db.execute(
        text("SELECT COUNT(*) FROM bins WHERE status = 'active'")
    )
    active_bins = int(bins_result.scalar() or 0)

    # Κάδοι με πρόσφατα δεδομένα
    covered_result = await db.execute(text("""
        SELECT COUNT(DISTINCT bin_id) FROM sensor_data
        WHERE time >= NOW() - INTERVAL '1 hour'
    """))
    covered_bins = int(covered_result.scalar() or 0)

    coverage_pct = round((covered_bins / active_bins) * 100, 1) if
active_bins > 0 else 0.0

    return {
        "status":    "healthy",
        "timestamp": datetime.now().isoformat(),
        "metrics": {
            "active_bins":              active_bins,
            "bins_with_recent_data":    covered_bins,
            "recent_readings_count":    readings_count,
            "data_coverage_percentage": coverage_pct,
        },
    }
```

Backend\api\routes\auth.py

```
"""
app/api/routes/auth.py
======================
ΠΧ1 — Πιστοποίηση Χρήστη

Endpoints:
    GET  /auth/captcha                → Δημιουργία captcha
    POST /auth/login                  → Σύνδεση + JWT
    POST /auth/password-reset-request   → Αίτημα επαναφοράς κωδικού
    GET  /auth/password-reset-validate  → Επαλήθευση token
```

```
    POST /auth/password-reset-confirm   → Ορισμός νέου κωδικού
    GET  /auth/users                    → Λίστα χρηστών (admin)
    POST /auth/users                    → Δημιουργία χρήστη
    DEL  /auth/users/{id}               → Απενεργοποίηση χρήστη
"""

import random
import secrets
import smtplib
import uuid
from datetime import datetime, timedelta
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

from fastapi import APIRouter, Depends, HTTPException
from pydantic import BaseModel
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.config import settings
from app.core.database import get_db
from app.core.security import create_access_token, get_password_hash,
verify_password

router = APIRouter()

# In-memory captcha store {captcha_id: correct_answer}
# Σε production: χρησιμοποίησε Redis
_captcha_store: dict[str, int] = {}


# ------------------------------------------------------------------------
--
# Schemas (Pydantic)
# ------------------------------------------------------------------------
--

class LoginRequest(BaseModel):
    username: str
    password: str
    captcha_id: str
    captcha_answer: int
    remember_me: bool = False


class CreateUserRequest(BaseModel):
    username: str
```

```
    email: str
    password: str
    first_name: str
    last_name: str
    role: str = "operator"


class PasswordResetRequest(BaseModel):
    username: str


class PasswordResetConfirm(BaseModel):
    token: str
    new_password: str
    confirm_password: str


# ------------------------------------------------------------------------
--
# Email helper
# ------------------------------------------------------------------------
--

async def _send_email(to: str, subject: str, html_body: str) -> bool:
    """
    Αποστολή email μέσω SMTP.
    Επιστρέφει False αν δεν έχει ρυθμιστεί SMTP.
    """
    if not settings.SMTP_USER or not settings.SMTP_PASS:
        return False

    msg = MIMEMultipart("alternative")
    msg["Subject"] = subject
    msg["From"]    = f"{settings.SYSTEM_NAME} <{settings.SMTP_USER}>"
    msg["To"]      = to

    # Wrapper HTML template
    html = f"""
    <html><body style="font-family:Arial,sans-serif;max-
width:600px;margin:0 auto;">
      <div style="background:linear-gradient(135deg,#2c3e50,#3498db);
                  padding:20px;border-radius:10px 10px 0 0;text-
align:center;">
        <h1 style="color:white;margin:0;">Smart Waste Management</h1>
        <p style="color:rgba(255,255,255,0.8);margin:4px 0 0;">Δήμος
Μαλεβιζίου</p>
      </div>
```

```
      <div style="background:#f8f9fa;padding:30px;border-radius:0 0 10px
10px;">
        {html_body}
      </div>
      <p style="text-align:center;color:#7f8c8d;font-size:0.8rem;margin-
top:20px;">
        2026 Smart Waste Management — Δήμος Μαλεβιζίου
      </p>
    </body></html>
    """
    msg.attach(MIMEText(html, "html"))

    try:
        with smtplib.SMTP(settings.SMTP_HOST, settings.SMTP_PORT) as
server:
            server.starttls()
            server.login(settings.SMTP_USER, settings.SMTP_PASS)
            server.sendmail(settings.SMTP_USER, to, msg.as_string())
        return True
    except Exception as exc:
        print(f"[Email Error] {exc}")
        return False


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.get("/auth/captcha")
async def get_captcha() -> dict:
    """
    Δημιουργία αριθμητικού captcha.
    Επιστρέφει captcha_id και ερώτηση (π.χ. "5 + 3 = ?").
    """
    a, b       = random.randint(1, 20), random.randint(1, 20)
    captcha_id = str(uuid.uuid4())
    _captcha_store[captcha_id] = a + b
    return {"captcha_id": captcha_id, "question": f"{a} + {b} = ?"}


@router.post("/auth/login")
async def login(
    req: LoginRequest,
    db: AsyncSession = Depends(get_db),
) -> dict:
```

```
    """
    ΠΧ1 – Σύνδεση χρήστη.

    Βήματα:
        1. Επαλήθευση captcha
        2. Αναζήτηση χρήστη στη DB
        3. Επαλήθευση κωδικού (bcrypt)
        4. Έκδοση JWT token
    """
    # 1. Captcha
    correct = _captcha_store.pop(req.captcha_id, None)
    if correct is None or req.captcha_answer != correct:
        raise HTTPException(400, "Λάθος απάντηση captcha.")

    # 2. Αναζήτηση χρήστη
    result = await db.execute(
        text("SELECT id, username, password_hash, role, full_name,
is_active, email "
             "FROM users WHERE username = :u"),
        {"u": req.username},
    )
    user = result.mappings().first()

    # 3. Επαλήθευση
    if not user or not verify_password(req.password,
user["password_hash"]):
        raise HTTPException(401, "Λάθος username ή password.")
    if not user["is_active"]:
        raise HTTPException(403, "Ο λογαριασμός είναι ανενεργός.")

    # Ενημέρωση last_login
    await db.execute(
        text("UPDATE users SET last_login = NOW() WHERE id = :id"),
        {"id": user["id"]},
    )
    await db.commit()

    # 4. JWT – remember_me: 30 ημέρες / normal: 8 ώρες
    expire_minutes = 60 * 24 * 30 if req.remember_me else 60 * 8
    token = create_access_token(
        {"sub": user["username"], "role": user["role"]},
        expires_delta=timedelta(minutes=expire_minutes),
    )

    return {
        "access_token":    token,
        "token_type":      "bearer",
```

```
        "username":        user["username"],
        "role":            user["role"],
        "full_name":       user["full_name"],
        "remember_me":     req.remember_me,
        "expires_minutes": expire_minutes,
    }


@router.post("/auth/password-reset-request")
async def password_reset_request(
    req: PasswordResetRequest,
    db: AsyncSession = Depends(get_db),
) -> dict:
    """
    ΠΧ1 – Αίτημα επαναφοράς κωδικού μέσω username.
    Για λόγους ασφαλείας επιστρέφει πάντα το ίδιο μήνυμα.
    """
    generic_response = {
        "status": "success",
        "message": "Αν το username υπάρχει, θα λάβετε email με οδηγίες.",
    }

    result = await db.execute(
        text("SELECT id, username, email, full_name, is_active "
             "FROM users WHERE username = :u"),
        {"u": req.username},
    )
    user = result.mappings().first()

    if not user or not user["is_active"]:
        return generic_response

    # Δημιουργία token
    token      = secrets.token_urlsafe(32)
    expires_at = datetime.utcnow() + timedelta(hours=2)

    # Ακύρωση παλιών tokens
    await db.execute(
        text("UPDATE password_reset_tokens SET used = TRUE "
             "WHERE username = :u AND used = FALSE"),
        {"u": req.username},
    )

    # Αποθήκευση νέου token
    await db.execute(
        text("INSERT INTO password_reset_tokens (username, email, token,
expires_at) "
```

```
            "VALUES (:username, :email, :token, :expires_at)"),
        {"username": user["username"], "email": user["email"],
         "token": token, "expires_at": expires_at},
    )
    await db.commit()

    # Αποστολή email
    reset_url  = f"{settings.FRONTEND_URL}/reset-password?token={token}"
    email_body = f"""
        <h2 style="color:#2c3e50;">Επαναφορά Κωδικού Πρόσβασης</h2>
        <p>Γεια σας <strong>{user['full_name']}</strong>,</p>
        <p>Λάβαμε αίτημα επαναφοράς κωδικού για τον λογαριασμό
           <strong>{user['username']}</strong>.</p>
        <div style="text-align:center;margin:30px 0;">
            <a href="{reset_url}"
               style="background:#3498db;color:white;padding:12px 30px;
                      border-radius:8px;text-decoration:none;font-
weight:bold;">
                Ορισμός Νέου Κωδικού
            </a>
        </div>
        <p style="color:#7f8c8d;font-size:0.85rem;">
            Ο σύνδεσμος ισχύει για <strong>2 ώρες</strong>.
        </p>
    """
    email_sent = await _send_email(
        user["email"],
        f"[{settings.SYSTEM_NAME}] Επαναφορά Κωδικού",
        email_body,
    )

    return {**generic_response, "email_sent": email_sent}


@router.get("/auth/password-reset-validate/{token}")
async def validate_reset_token(
    token: str,
    db: AsyncSession = Depends(get_db),
) -> dict:
    """Επαλήθευση ότι το reset token είναι έγκυρο και μη-ληγμένο."""
    result = await db.execute(
        text("SELECT username, expires_at, used FROM password_reset_tokens
"
             "WHERE token = :t"),
        {"t": token},
    )
    row = result.mappings().first()
```

```
    if not row:
        return {"valid": False, "reason": "Μη έγκυρος σύνδεσμος."}
    if row["used"]:
        return {"valid": False, "reason": "Ο σύνδεσμος έχει ήδη
χρησιμοποιηθεί."}
    if datetime.utcnow() > row["expires_at"].replace(tzinfo=None):
        return {"valid": False, "reason": "Ο σύνδεσμος έχει λήξει."}

    return {"valid": True, "username": row["username"]}


@router.post("/auth/password-reset-confirm")
async def password_reset_confirm(
    req: PasswordResetConfirm,
    db: AsyncSession = Depends(get_db),
) -> dict:
    """ΠΧ1 – Ορισμός νέου κωδικού με έγκυρο reset token."""
    if req.new_password != req.confirm_password:
        raise HTTPException(400, "Οι κωδικοί δεν ταιριάζουν.")
    if len(req.new_password) < 6:
        raise HTTPException(400, "Ο κωδικός πρέπει να έχει τουλάχιστον 6
χαρακτήρες.")

    result = await db.execute(
        text("SELECT username, email, expires_at, used FROM
password_reset_tokens "
             "WHERE token = :t"),
        {"t": req.token},
    )
    row = result.mappings().first()

    if not row:
        raise HTTPException(400, "Μη έγκυρος σύνδεσμος.")
    if row["used"]:
        raise HTTPException(400, "Ο σύνδεσμος έχει ήδη χρησιμοποιηθεί.")
    if datetime.utcnow() > row["expires_at"].replace(tzinfo=None):
        raise HTTPException(400, "Ο σύνδεσμος έχει λήξει.")

    # Ενημέρωση κωδικού
    await db.execute(
        text("UPDATE users SET password_hash = :h, updated_at = NOW() "
             "WHERE username = :u"),
        {"h": get_password_hash(req.new_password), "u": row["username"]},
    )
    # Ακύρωση token
    await db.execute(
```

```
        text("UPDATE password_reset_tokens SET used = TRUE WHERE token =
:t"),
        {"t": req.token},
    )
    await db.commit()

    # Email επιβεβαίωσης
    await _send_email(
        row["email"],
        f"[{settings.SYSTEM_NAME}] Ο κωδικός σας άλλαξε",
        f"""
        <h2 style="color:#27ae60;">Ο κωδικός άλλαξε επιτυχώς!</h2>
        <p>Ο κωδικός για τον λογαριασμό
           <strong>{row['username']}</strong> άλλαξε επιτυχώς.</p>
        <p style="color:#e74c3c;font-size:0.85rem;">
            Αν δεν κάνατε αυτή την αλλαγή, επικοινωνήστε αμέσως
            με τον διαχειριστή.
        </p>
        """,
    )

    return {"status": "success", "message": "Ο κωδικός άλλαξε επιτυχώς!"}


# ---------------------------------------------------------------------------
--
# User management (admin only)
# ---------------------------------------------------------------------------
--

@router.get("/auth/users")
async def get_users(db: AsyncSession = Depends(get_db)) -> dict:
    """Λίστα χρηστών συστήματος."""
    result = await db.execute(
        text("SELECT id, username, email, full_name, role, is_active, "
             "created_at, last_login FROM users ORDER BY id")
    )
    return {"users": [dict(r) for r in result.mappings().all()]}


@router.post("/auth/users")
async def create_user(
    req: CreateUserRequest,
    db: AsyncSession = Depends(get_db),
) -> dict:
    """Δημιουργία νέου χρήστη."""
    await db.execute(
```

```
        text("INSERT INTO users (username, email, password_hash, full_name,
role, is_active) "
             "VALUES (:username, :email, :password_hash, :full_name, :role,
true)"),
        {
            "username":      req.username,
            "email":         req.email,
            "password_hash": get_password_hash(req.password),
            "full_name":     f"{req.first_name} {req.last_name}",
            "role":          req.role,
        },
    )
    await db.commit()
    return {"message": f"Ο χρήστης {req.username} δημιουργήθηκε επιτυχώς."}


@router.delete("/auth/users/{user_id}")
async def deactivate_user(
    user_id: int,
    db: AsyncSession = Depends(get_db),
) -> dict:
    """Soft-delete χρήστη (is_active = false)."""
    await db.execute(
        text("UPDATE users SET is_active = false WHERE id = :id"),
        {"id": user_id},
    )
    await db.commit()
    return {"message": "Ο χρήστης απενεργοποιήθηκε."}
```

backend\app\api\routes\bins.py

```
"""
app/api/routes/bins.py
======================
Διαχείριση έξυπνων κάδων απορριμμάτων.

Endpoints:
    GET  /bins              → Λίστα κάδων με τρέχουσα κατάσταση
    GET  /bins/{bin_id}     → Λεπτομέρειες συγκεκριμένου κάδου
    POST /bins/{bin_id}/collect → Καταγραφή αποκομιδής
"""

import logging
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
```

```
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()


# ---------------------------------------------------------------------------
--
# Helpers
# ---------------------------------------------------------------------------
--

def _build_zone_filter(zone_code: Optional[str]) -> tuple[str, dict]:
    """Δημιουργία SQL WHERE clause για φίλτρο ζώνης."""
    if zone_code and zone_code != "ALL":
        return "AND zone_code = :zone_code", {"zone_code": zone_code}
    return "", {}


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.get("/bins")
async def get_bins(
    zone_code:   Optional[str] = None,
    fill_status: Optional[str] = None,
    db:          AsyncSession  = Depends(get_db),
) -> dict:
    """
    Λίστα όλων των κάδων με τρέχουσα κατάσταση από το view
bin_status_summary.

    Query params:
        zone_code:   Φίλτρο ζώνης (GAZ/AMM/APE/TYL/KRO/ROD)
        fill_status: Φίλτρο κατάστασης (critical/warning/normal)
    """
    where_clauses = ["1=1"]
    params: dict  = {}

    if zone_code and zone_code != "ALL":
        where_clauses.append("zone_code = :zone_code")
```

```
        params["zone_code"] = zone_code

    if fill_status:
        where_clauses.append("fill_status = :fill_status")
        params["fill_status"] = fill_status

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                bin_id, bin_code, location_name, address,
                bin_type, status, capacity_liters,
                latitude, longitude,
                zone_name, zone_code,
                current_fill_level, current_battery_level,
                current_temperature, last_reading_time,
                fill_status, active_alerts_count
            FROM bin_status_summary
            WHERE {where}
            ORDER BY current_fill_level DESC
        """),
        params,
    )

    bins = []
    for row in result.mappings().all():
        bins.append({
            "bin_id":               row["bin_id"],
            "bin_code":             row["bin_code"],
            "location_name":        row["location_name"],
            "address":              row["address"],
            "bin_type":             row["bin_type"],
            "status":               row["status"],
            "capacity_liters":      row["capacity_liters"],
            "latitude":             float(row["latitude"] or 0),
            "longitude":            float(row["longitude"] or 0),
            "zone_name":            row["zone_name"],
            "zone_code":            row["zone_code"],
            "current_fill_level":   float(row["current_fill_level"] or 0),
            "current_battery_level": float(row["current_battery_level"] or
0),
            "current_temperature":  float(row["current_temperature"] or
0),
            "last_reading_time":    str(row["last_reading_time"]) if
row["last_reading_time"] else None,
            "fill_status":          row["fill_status"],
```

```
            "active_alerts_count":   int(row["active_alerts_count"] or 0),
        })

    return {
        "status": "success",
        "total":  len(bins),
        "bins":   bins,
    }


@router.get("/bins/{bin_id}")
async def get_bin(
    bin_id: str,
    db:     AsyncSession = Depends(get_db),
) -> dict:
    """Λεπτομέρειες συγκεκριμένου κάδου με ιστορικό μετρήσεων."""
    # Βασικά στοιχεία
    result = await db.execute(
        text("""
            SELECT *
            FROM bin_status_summary
            WHERE bin_id = :bin_id
        """),
        {"bin_id": bin_id},
    )
    bin_data = result.mappings().first()

    if not bin_data:
        raise HTTPException(404, f"Κάδος {bin_id} δεν βρέθηκε.")

    # Τελευταίες 24 μετρήσεις
    history_result = await db.execute(
        text("""
            SELECT time, fill_level, battery_level, temperature
            FROM sensor_data
            WHERE bin_id = :bin_id
            ORDER BY time DESC
            LIMIT 24
        """),
        {"bin_id": bin_id},
    )
    history = [dict(r) for r in history_result.mappings().all()]

    return {
        "status":  "success",
        "bin":     dict(bin_data),
        "history": history,
```

```
    }


@router.post("/bins/{bin_id}/collect")
async def collect_bin(
    bin_id: str,
    db:     AsyncSession = Depends(get_db),
) -> dict:
    """
    Καταγραφή αποκομιδής κάδου — reset fill_level στο 0%.

    Εισάγει νέα μέτρηση στο sensor_data με fill_level=0
    ώστε να ενημερωθεί το bin_status_summary view.
    """
    # Έλεγχος ύπαρξης κάδου
    result = await db.execute(
        text("SELECT bin_id, location_name FROM bins WHERE bin_id =
:bin_id"),
        {"bin_id": bin_id},
    )
    bin_row = result.mappings().first()

    if not bin_row:
        raise HTTPException(404, f"Κάδος {bin_id} δεν βρέθηκε.")

    # Εισαγωγή μέτρησης αποκομιδής
    await db.execute(
        text("""
            INSERT INTO sensor_data (time, bin_id, fill_level,
battery_level)
            VALUES (NOW(), :bin_id, 0.0, (
                SELECT COALESCE(battery_level, 100)
                FROM sensor_data
                WHERE bin_id = :bin_id
                ORDER BY time DESC
                LIMIT 1
            ))
        """),
        {"bin_id": bin_id},
    )
    await db.commit()

    logger.info(f"Bin {bin_id} collected — fill reset to 0%")

    return {
        "status":       "success",
        "bin_id":       bin_id,
```

```
        "location":     bin_row["location_name"],
        "message":      f"Κάδος {bin_id} αποκομίστηκε επιτυχώς.",
        "new_fill":     0,
    }
```

backend\app\api\routes\citizen_portal.py

```
"""
app/api/routes/citizen_portal.py
==================================
ΠΧ3 — Πύλη Αναφορών Πολιτών

Endpoints:
    GET  /citizen/bins                          → Κάδοι για τον χάρτη πολιτών
    POST /citizen/reports                       → Υποβολή αναφοράς
(multipart)
    GET  /citizen/reports/{tracking_code}       → Παρακολούθηση αναφοράς
    GET  /citizen/reports                       → Λίστα αναφορών (admin)
    PUT  /citizen/reports/{id}                  → Ενημέρωση κατάστασης (admin)
"""

import base64
import logging
import secrets
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from typing import Optional

from fastapi import APIRouter, Depends, File, Form, HTTPException,
UploadFile
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.config import settings
from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()

# Έγκυροι τύποι αναφοράς
REPORT_TYPES = {"overflow", "missed", "damage", "new_bin", "other"}

# Ετικέτες τύπων αναφοράς
TYPE_LABELS: dict[str, str] = {
    "overflow": "Υπερχείλιση Κάδου",
    "missed":   "Χαμένη Αποκομιδή",
    "damage":   "Ζημιά Κάδου",
```

```
    "new_bin":  "Αίτημα Νέου Κάδου",
    "other":    "Άλλο",
}

# Ετικέτες καταστάσεων αναφοράς
STATUS_LABELS: dict[str, str] = {
    "open":        "Ανοιχτή",
    "in_progress": "Σε Εξέλιξη",
    "resolved":    "Επιλύθηκε",
    "closed":      "Κλειστή",
}

# Max μέγεθος φωτογραφίας (bytes)
MAX_PHOTO_SIZE = 5 * 1024 * 1024  # 5 MB


# ---------------------------------------------------------------------------
--
# Helpers
# ---------------------------------------------------------------------------
--

async def _generate_tracking_code(db: AsyncSession) -> str:
    """
    Παραγωγή μοναδικού tracking code μορφής CR1-XXXX (αύξων αριθμός).
    """
    result = await db.execute(text("""
        SELECT COALESCE(
            MAX(CAST(SPLIT_PART(tracking_code, '-', 2) AS INTEGER)),
            0
        ) + 1 AS next_num
        FROM citizen_reports
        WHERE tracking_code LIKE 'CR1-%'
    """))
    row      = result.mappings().first()
    next_num = int(row["next_num"] or 1)
    return f"CR1-{next_num:04d}"


async def _send_confirmation_email(
    to_email:     str,
    citizen_name: str,
    tracking_code: str,
    report_type:  str,
    description:  str,
) -> bool:
```

```
    """Αποστολή email επιβεβαίωσης στον πολίτη μετά την υποβολή
αναφοράς."""
    if not settings.SMTP_USER or not settings.SMTP_PASS:
        logger.info(f"[DEV] Email for {to_email} — tracking:
{tracking_code}")
        return False

    type_label = TYPE_LABELS.get(report_type, report_type)

    msg = MIMEMultipart("alternative")
    msg["Subject"] = f"[{settings.SYSTEM_NAME}] Αναφορά {tracking_code} —
Λήψη"
    msg["From"]    = f"{settings.SYSTEM_NAME} <{settings.SMTP_USER}>"
    msg["To"]      = to_email

    html = f"""
    <html><body style="font-family:Arial,sans-serif;max-
width:600px;margin:0 auto;">
      <div style="background:linear-gradient(135deg,#2c3e50,#3498db);
                  padding:20px;border-radius:10px 10px 0 0;text-
align:center;">
        <h1 style="color:white;margin:0;">Smart Waste Management</h1>
        <p style="color:rgba(255,255,255,0.8);margin:4px 0 0;">Δήμος
Μαλεβιζίου</p>
      </div>
      <div style="background:#f8f9fa;padding:30px;border-radius:0 0 10px
10px;">
        <h2 style="color:#2c3e50;">Η αναφορά σας ελήφθη!</h2>
        <p>Αγαπητέ/ή <strong>{citizen_name}</strong>,</p>

        <div style="background:white;border-radius:8px;padding:20px;
                    border-left:4px solid #3498db;margin:20px 0;">
          <table style="width:100%;border-collapse:collapse;">
            <tr>
              <td style="padding:6px 0;color:#7f8c8d;font-
weight:bold;width:140px;">Κωδικός:</td>
              <td style="padding:6px 0;font-size:1.2rem;font-
weight:bold;color:#3498db;">
                {tracking_code}
              </td>
            </tr>
            <tr>
              <td style="padding:6px 0;color:#7f8c8d;font-
weight:bold;">Τύπος:</td>
              <td style="padding:6px 0;">{type_label}</td>
            </tr>
            <tr>
```

```
            <td style="padding:6px 0;color:#7f8c8d;font-
weight:bold;">Περιγραφή:</td>
            <td style="padding:6px 0;">{description[:200]}</td>
          </tr>
        </table>
      </div>

      <p style="color:#7f8c8d;font-size:0.85rem;">
        Χρησιμοποιήστε τον κωδικό <strong>{tracking_code}</strong>
        για να παρακολουθείτε την κατάσταση της αναφοράς σας.<br>
        Αναμένετε απάντηση εντός 2-3 εργάσιμων ημερών.
      </p>
    </div>
  </body></html>
  """
  msg.attach(MIMEText(html, "html"))

  try:
      with smtplib.SMTP(settings.SMTP_HOST, settings.SMTP_PORT) as
server:
          server.starttls()
          server.login(settings.SMTP_USER, settings.SMTP_PASS)
          server.sendmail(settings.SMTP_USER, to_email, msg.as_string())
      return True
  except Exception as exc:
      logger.error(f"Email error: {exc}")
      return False


# ---------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------
--

@router.get("/citizen/bins")
async def get_bins_for_citizens(db: AsyncSession = Depends(get_db)) ->
dict:
  """
  Κάδοι για εμφάνιση στον Leaflet χάρτη του citizen portal.
  Δεν απαιτεί authentication.
  """
  result = await db.execute(text("""
      SELECT
          bin_id, location_name, address,
          latitude, longitude, bin_type,
          current_fill_level, fill_status,
```

```
            zone_name, zone_code
        FROM bin_status_summary
        ORDER BY bin_id
    """))

    bins = []
    for row in result.mappings().all():
        bins.append({
            "bin_id":             row["bin_id"],
            "location_name":      row["location_name"],
            "address":            row["address"],
            "latitude":           float(row["latitude"]  or 0),
            "longitude":          float(row["longitude"] or 0),
            "bin_type":           row["bin_type"],
            "current_fill_level": float(row["current_fill_level"] or 0),
            "fill_status":        row["fill_status"],
            "zone_name":          row["zone_name"],
            "zone_code":          row["zone_code"],
        })

    return {"status": "success", "total": len(bins), "bins": bins}


@router.post("/citizen/reports")
async def submit_citizen_report(
    report_type:     str            = Form(...),
    title:           str            = Form(...),
    description:     str            = Form(...),
    latitude:        float          = Form(...),
    longitude:       float          = Form(...),
    citizen_name:    str            = Form(...),
    citizen_email:   Optional[str]  = Form(None),
    bin_id:          Optional[str]  = Form(None),
    address_hint:    Optional[str]  = Form(None),
    is_pin_location: bool           = Form(False),
    priority:        str            = Form("medium"),
    photo:           Optional[UploadFile] = File(None),
    db:              AsyncSession   = Depends(get_db),
) -> dict:
    """
    ΠΧ3 — Υποβολή αναφοράς πολίτη.

    Βήμα 7: Λήψη και αποθήκευση αναφοράς
    Βήμα 8: Παραγωγή tracking_code (CR1-XXXX)
    Βήμα 9: Email επιβεβαίωσης (αν δόθηκε email)
    """
    # Validation
```

```
    if report_type not in REPORT_TYPES:
        raise HTTPException(400, f"Μη έγκυρος τύπος αναφοράς:
{report_type}")

    # Βήμα 8: Tracking code
    tracking_code = await _generate_tracking_code(db)

    # Φωτογραφία (base64) — Βήμα 6 ΠΧ3
    image_data = None
    if photo and photo.filename:
        contents = await photo.read()
        if len(contents) > MAX_PHOTO_SIZE:
            raise HTTPException(400, "Η φωτογραφία δεν πρέπει να υπερβαίνει
τα 5MB.")
        ext        = photo.filename.rsplit(".", 1)[-1].lower()
        b64        = base64.b64encode(contents).decode("utf-8")
        image_data = f"{b64}"

    # Αποθήκευση
    result = await db.execute(
        text("""
            INSERT INTO citizen_reports (
                report_type, title, description,
                latitude, longitude,
                citizen_name, citizen_email,
                bin_id, address_hint, is_pin_location,
                tracking_code, status, priority,
                image_url, created_at
            ) VALUES (
                :report_type, :title, :description,
                :latitude, :longitude,
                :citizen_name, :citizen_email,
                :bin_id, :address_hint, :is_pin_location,
                :tracking_code, 'open', :priority,
                :image_url, NOW()
            )
            RETURNING id, created_at
        """),
        {
            "report_type":    report_type,
            "title":          title,
            "description":    description,
            "latitude":       latitude,
            "longitude":      longitude,
            "citizen_name":   citizen_name,
            "citizen_email":  citizen_email,
            "bin_id":         bin_id,
```

```
            "address_hint":   address_hint,
            "is_pin_location": is_pin_location,
            "tracking_code":  tracking_code,
            "priority":       priority,
            "image_url":      image_data,
        },
    )
    row = result.mappings().first()
    await db.commit()

    # Βήμα 9: Email
    email_sent = False
    if citizen_email:
        email_sent = await _send_confirmation_email(
            citizen_email, citizen_name,
            tracking_code, report_type, description,
        )

    return {
        "status":        "success",
        "tracking_code": tracking_code,
        "report_id":     row["id"],
        "created_at":    str(row["created_at"]),
        "email_sent":    email_sent,
        "message":       f"Αναφορά υποβλήθηκε επιτυχώς! Κωδικός:
{tracking_code}",
    }


@router.get("/citizen/reports/{tracking_code}")
async def get_report_by_tracking(
    tracking_code: str,
    db:            AsyncSession = Depends(get_db),
) -> dict:
    """Παρακολούθηση κατάστασης αναφοράς με tracking_code (ΠΧ3)."""
    result = await db.execute(
        text("""
            SELECT
                id, report_type, title, description,
                latitude, longitude, status, priority,
                citizen_name, bin_id, address_hint,
                is_pin_location, tracking_code,
                created_at, resolved_at, image_url
            FROM citizen_reports
            WHERE tracking_code = :tc
        """),
        {"tc": tracking_code},
```

```
    )
    row = result.mappings().first()

    if not row:
        raise HTTPException(404, "Δεν βρέθηκε αναφορά με αυτόν τον
κωδικό.")

    return {
        "status": "success",
        "report": {
            "id":              row["id"],
            "tracking_code":   row["tracking_code"],
            "report_type":     row["report_type"],
            "type_label":      TYPE_LABELS.get(row["report_type"],
row["report_type"]),
            "title":           row["title"],
            "description":     row["description"],
            "latitude":        row["latitude"],
            "longitude":       row["longitude"],
            "status":          row["status"],
            "status_label":    STATUS_LABELS.get(row["status"],
row["status"]),
            "priority":        row["priority"],
            "citizen_name":    row["citizen_name"],
            "bin_id":          row["bin_id"],
            "address_hint":    row["address_hint"],
            "is_pin_location": row["is_pin_location"],
            "created_at":      str(row["created_at"]),
            "resolved_at":     str(row["resolved_at"]) if
row["resolved_at"] else None,
            "has_photo":       row["image_url"] is not None,
        },
    }


@router.get("/citizen/reports")
async def get_all_reports(
    status:      Optional[str] = None,
    report_type: Optional[str] = None,
    db:          AsyncSession  = Depends(get_db),
) -> dict:
    """Λίστα αναφορών για το κεντρικό dashboard (admin)."""
    where_clauses = ["1=1"]
    params: dict  = {}

    if status and status != "ALL":
        where_clauses.append("status = :status")
```

```
        params["status"] = status

    if report_type and report_type != "ALL":
        where_clauses.append("report_type = :report_type")
        params["report_type"] = report_type

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                id, report_type, title, description,
                latitude, longitude, status, priority,
                citizen_name, citizen_email, bin_id,
                tracking_code, is_pin_location,
                created_at, resolved_at
            FROM citizen_reports
            WHERE {where}
            ORDER BY
                CASE priority
                    WHEN 'critical' THEN 1
                    WHEN 'high'     THEN 2
                    WHEN 'medium'   THEN 3
                    ELSE 4
                END,
                created_at DESC
            LIMIT 100
        """),
        params,
    )

    reports = [dict(r) for r in result.mappings().all()]
    return {"status": "success", "total": len(reports), "reports": reports}


@router.put("/citizen/reports/{report_id}")
async def update_report_status(
    report_id: int,
    data:      dict,
    db:        AsyncSession = Depends(get_db),
) -> dict:
    """Ενημέρωση κατάστασης αναφοράς από τον διαχειριστή."""
    allowed_fields = {"status", "priority"}
    updates = {k: v for k, v in data.items() if k in allowed_fields}

    if not updates:
        raise HTTPException(400, "Δεν δόθηκαν έγκυρα στοιχεία.")
```

```
    set_parts = [f"{k} = :{k}" for k in updates]

    # Αυτόματη καταγραφή resolved_at
    if data.get("status") in ("resolved", "closed"):
        set_parts.append("resolved_at = NOW()")

    updates["report_id"] = report_id

    await db.execute(
        text(f"UPDATE citizen_reports SET {', '.join(set_parts)} WHERE id =
:report_id"),
        updates,
    )
    await db.commit()

    return {
        "status":  "success",
        "message": "Η αναφορά ενημερώθηκε.",
        "updated": {k: v for k, v in updates.items() if k != "report_id"},
    }
```

backend\app\api\routes\fillness.py

```
"""
app/api/routes/fillness.py
==========================
Δεδομένα πληρότητας κάδων για γραφήματα και αναφορές.

Endpoints:
    GET /fillness          → Σύνοψη πληρότητας για γράφημα (bar chart)
    GET /fillness/bins     → Λίστα κάδων με φίλτρα κατάστασης
    GET /fillness/stats    → Στατιστικά ανά ζώνη
    GET /fillness/history  → Χρονοσειρά πληρότητας (TimescaleDB)
"""

import logging
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()
```

```
# Κατώφλια πληρότητας
THRESHOLD_CRITICAL = 80.0
THRESHOLD_WARNING  = 60.0


# ----------------------------------------------------------------------------
--
# Helpers
# ----------------------------------------------------------------------------
--

def _fill_status(level: float) -> str:
    """Κατηγοριοποίηση πληρότητας σε critical/warning/normal."""
    if level >= THRESHOLD_CRITICAL:
        return "critical"
    if level >= THRESHOLD_WARNING:
        return "warning"
    return "normal"


def _serialize_bin_row(row: dict) -> dict:
    """Μετατροπή DB row κάδου σε JSON-serializable dict."""
    return {
        "bin_id":               row["bin_id"],
        "location_name":        row["location_name"],
        "address":              row["address"],
        "bin_type":             row["bin_type"],
        "latitude":             float(row["latitude"]  or 0),
        "longitude":            float(row["longitude"] or 0),
        "zone_name":            row["zone_name"],
        "zone_code":            row["zone_code"],
        "current_fill_level":   float(row["current_fill_level"]   or 0),
        "current_battery_level": float(row["current_battery_level"] or 0),
        "fill_status":          row["fill_status"],
        "last_reading_time":    str(row["last_reading_time"]) if
row["last_reading_time"] else None,
        "active_alerts_count":  int(row["active_alerts_count"] or 0),
    }


# ----------------------------------------------------------------------------
--
# Endpoints
# ----------------------------------------------------------------------------
--

@router.get("/fillness")
```

```python
async def get_fillness_summary(
    zone_code: Optional[str] = None,
    db:        AsyncSession  = Depends(get_db),
) -> dict:
    """
    Σύνοψη πληρότητας για bar chart στο dashboard.

    Επιστρέφει λίστα κάδων με fill_level ταξινομημένη φθίνουσα,
    κατάλληλη για απευθείας χρήση σε γράφημα.

    Query params:
        zone_code: Φίλτρο ζώνης (GAZ/AMM/APE/TYL/KRO/ROD ή ALL)
    """
    where_clauses = ["1=1"]
    params: dict  = {}

    if zone_code and zone_code != "ALL":
        where_clauses.append("zone_code = :zone_code")
        params["zone_code"] = zone_code

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                bin_id, location_name, address,
                bin_type, latitude, longitude,
                zone_name, zone_code,
                current_fill_level, current_battery_level,
                fill_status, last_reading_time,
                active_alerts_count
            FROM bin_status_summary
            WHERE {where}
            ORDER BY current_fill_level DESC
        """),
        params,
    )

    bins = [_serialize_bin_row(dict(r)) for r in result.mappings().all()]

    # Υπολογισμός συνολικών στατιστικών
    total    = len(bins)
    critical = sum(1 for b in bins if b["fill_status"] == "critical")
    warning  = sum(1 for b in bins if b["fill_status"] == "warning")
    normal   = sum(1 for b in bins if b["fill_status"] == "normal")
    avg_fill = (
        round(sum(b["current_fill_level"] for b in bins) / total, 1)
```

```
        if total > 0 else 0.0
    )

    return {
        "status": "success",
        "stats": {
            "total":    total,
            "critical": critical,
            "warning":  warning,
            "normal":   normal,
            "avg_fill": avg_fill,
        },
        "bins": bins,
    }


@router.get("/fillness/bins")
async def get_bins_by_status(
    fill_status: Optional[str] = None,
    zone_code:   Optional[str] = None,
    min_fill:    float         = 0.0,
    max_fill:    float         = 100.0,
    db:          AsyncSession  = Depends(get_db),
) -> dict:
    """
    Φιλτραρισμένη λίστα κάδων βάσει κατάστασης πληρότητας.

    Query params:
        fill_status: critical / warning / normal
        zone_code:   Φίλτρο ζώνης
        min_fill:    Ελάχιστη πληρότητα (%)
        max_fill:    Μέγιστη πληρότητα (%)
    """
    if min_fill > max_fill:
        raise HTTPException(400, "min_fill δεν μπορεί να είναι μεγαλύτερο
από max_fill.")

    where_clauses = [
        "current_fill_level >= :min_fill",
        "current_fill_level <= :max_fill",
    ]
    params: dict = {"min_fill": min_fill, "max_fill": max_fill}

    if fill_status and fill_status != "ALL":
        where_clauses.append("fill_status = :fill_status")
        params["fill_status"] = fill_status
```

```
    if zone_code and zone_code != "ALL":
        where_clauses.append("zone_code = :zone_code")
        params["zone_code"] = zone_code

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                bin_id, location_name, address,
                bin_type, latitude, longitude,
                zone_name, zone_code,
                current_fill_level, current_battery_level,
                fill_status, last_reading_time,
                active_alerts_count
            FROM bin_status_summary
            WHERE {where}
            ORDER BY current_fill_level DESC
        """),
        params,
    )

    bins = [_serialize_bin_row(dict(r)) for r in result.mappings().all()]

    return {
        "status":  "success",
        "total":   len(bins),
        "filters": {
            "fill_status": fill_status,
            "zone_code":   zone_code,
            "min_fill":    min_fill,
            "max_fill":    max_fill,
        },
        "bins": bins,
    }


@router.get("/fillness/stats")
async def get_fillness_stats_by_zone(
    db: AsyncSession = Depends(get_db),
) -> dict:
    """
    Στατιστικά πληρότητας ανά ζώνη.
    Χρησιμοποιείται στο tab Ζώνες του dashboard.
    """
    result = await db.execute(text("""
        SELECT
```

```
            zone_code,
            zone_name,
            COUNT(*)                                                    AS
total_bins,
            COUNT(CASE WHEN fill_status = 'critical' THEN 1 END)       AS
critical_bins,
            COUNT(CASE WHEN fill_status = 'warning'  THEN 1 END)       AS
warning_bins,
            COUNT(CASE WHEN fill_status = 'normal'   THEN 1 END)       AS
normal_bins,
            ROUND(AVG(current_fill_level)::numeric, 1)                 AS
avg_fill_level,
            MAX(current_fill_level)                                     AS
max_fill_level,
            COUNT(CASE WHEN active_alerts_count > 0 THEN 1 END)        AS
bins_with_alerts
        FROM bin_status_summary
        WHERE zone_code IS NOT NULL
        GROUP BY zone_code, zone_name
        ORDER BY zone_code
    """))

    zones = []
    for row in result.mappings().all():
        zones.append({
            "zone_code":       row["zone_code"],
            "zone_name":       row["zone_name"],
            "total_bins":      int(row["total_bins"]       or 0),
            "critical_bins":   int(row["critical_bins"]    or 0),
            "warning_bins":    int(row["warning_bins"]     or 0),
            "normal_bins":     int(row["normal_bins"]      or 0),
            "avg_fill_level":  float(row["avg_fill_level"] or 0),
            "max_fill_level":  float(row["max_fill_level"] or 0),
            "bins_with_alerts": int(row["bins_with_alerts"] or 0),
        })

    return {
        "status": "success",
        "total":  len(zones),
        "zones":  zones,
    }


@router.get("/fillness/history/{bin_id}")
async def get_bin_fill_history(
    bin_id: str,
    hours:  int          = 24,
```

```
    db:     AsyncSession = Depends(get_db),
) -> dict:
    """
    Χρονοσειρά πληρότητας κάδου από TimescaleDB.

    Path params:
        bin_id: Αναγνωριστικό κάδου
    Query params:
        hours:  Ώρες ιστορικού (default: 24, max: 168 = 7 ημέρες)
    """
    if hours > 168:
        raise HTTPException(400, "Μέγιστο ιστορικό: 168 ώρες (7 ημέρες).")

    # Έλεγχος ύπαρξης κάδου
    exists = await db.execute(
        text("SELECT 1 FROM bins WHERE bin_id = :bin_id"),
        {"bin_id": bin_id},
    )
    if not exists.fetchone():
        raise HTTPException(404, f"Κάδος {bin_id} δεν βρέθηκε.")

    result = await db.execute(
        text("""
            SELECT
                time,
                fill_level,
                battery_level,
                temperature
            FROM sensor_data
            WHERE bin_id = :bin_id
              AND time >= NOW() - INTERVAL ':hours hours'
            ORDER BY time ASC
        """).bindparams(hours=hours),
        {"bin_id": bin_id},
    )

    history = []
    for row in result.mappings().all():
        history.append({
            "time":          str(row["time"]),
            "fill_level":    float(row["fill_level"]    or 0),
            "battery_level": float(row["battery_level"] or 0),
            "temperature":   float(row["temperature"]   or 0) if
row["temperature"] else None,
        })

    return {
```

```
        "status":   "success",
        "bin_id":   bin_id,
        "hours":    hours,
        "readings": len(history),
        "history":  history,
    }
```

backend\app\api\routes\optimization.py

```
"""
app/api/routes/optimization.py
================================
ΠΧ2 — Βελτιστοποίηση Διαδρομής Αποκομιδής

Αλγόριθμος: Nearest Neighbor TSP Heuristic
Αποστάσεις: Haversine (Great-Circle Distance)

Endpoints:
    GET /optimization/route          → Υπολογισμός βέλτιστης διαδρομής
    GET /optimization/routes-history → Ιστορικό αποθηκευμένων διαδρομών
    GET /optimization/osm-status     → Κατάσταση OSMnx cache
"""

import math
import logging
from datetime import date, datetime
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = APIRouter()
router = APIRouter()

# ---------------------------------------------------------------------------
--
# Σταθερές
# ---------------------------------------------------------------------------
--

# Depot — αφετηρία διαδρομής (Δημαρχείο Μαλεβιζίου)
DEPOT = {
    "name":      "Δημαρχείο Μαλεβιζίου",
    "latitude":  35.3259723,
    "longitude": 25.0655098,
```

```
}

# Κατώφλι αυτόματου fallback (Εναλλακτική Ροή 1)
MIN_BINS_FOR_TSP   = 3
FALLBACK_THRESHOLD = 50.0


# ----------------------------------------------------------------------
--
# Αλγόριθμος
# ----------------------------------------------------------------------
--

def haversine(lat1: float, lon1: float, lat2: float, lon2: float) -> float:
    """
    Υπολογισμός αεροπορικής απόστασης (km) μεταξύ δύο σημείων.
    Κατάλληλο για αποστάσεις <15km με ακρίβεια ~99.7%.
    """
    R = 6371
    lat1, lon1, lat2, lon2 = map(math.radians, [lat1, lon1, lat2, lon2])
    dlat, dlon = lat2 - lat1, lon2 - lon1
    a = math.sin(dlat / 2) ** 2 + math.cos(lat1) * math.cos(lat2) *
math.sin(dlon / 2) ** 2
    return R * 2 * math.asin(math.sqrt(a))


def nearest_neighbor_tsp(bins: list[dict], depot: dict) -> list[dict]:
    """
    Nearest Neighbor Heuristic για TSP.

    Βήματα:
        1. Ξεκινά από το depot
        2. Επιλέγει τον κοντινότερο αδεύτερο κάδο
           (score = distance / fill_pct, ώστε γεμάτοι κοντινοί κάδοι να
έχουν προτεραιότητα)
        3. Επαναλαμβάνει μέχρι να επισκεφθεί όλους

    Πολυπλοκότητα: O(N²)
    """
    if not bins:
        return []

    unvisited = bins.copy()
    route     = []
    cur_lat, cur_lon = depot["latitude"], depot["longitude"]

    while unvisited:
```

```
        best, best_score = None, float("inf")

        for b in unvisited:
            dist  = haversine(cur_lat, cur_lon, float(b["latitude"]),
float(b["longitude"]))
            fill  = float(b["current_fill_level"]) / 100
            score = dist / max(fill, 0.01)  # αποφυγή division by zero

            if score < best_score:
                best_score = score
                best       = b

        route.append(best)
        cur_lat, cur_lon = float(best["latitude"]),
float(best["longitude"])
        unvisited.remove(best)

    return route


def add_distance_info(route: list[dict], depot: dict) -> tuple[list[dict],
float]:
    """
    Εμπλουτισμός διαδρομής με πληροφορίες απόστασης ανά στάση.

    Returns:
        (enriched_route, total_distance_km)
    """
    total = 0.0
    prev_lat, prev_lon = depot["latitude"], depot["longitude"]
    enriched = []

    for i, stop in enumerate(route):
        dist   = haversine(prev_lat, prev_lon, float(stop["latitude"]),
float(stop["longitude"]))
        total += dist

        enriched.append({
            **stop,
            "stop_number":           i + 1,
            "distance_from_prev_km": round(dist, 2),
            "cumulative_distance_km": round(total, 2),
            "current_fill_level":    float(stop["current_fill_level"]),
            "current_battery_level": float(stop["current_battery_level"]),
            "latitude":              float(stop["latitude"]),
            "longitude":             float(stop["longitude"]),
        })
```

```
        prev_lat, prev_lon = float(stop["latitude"]),
float(stop["longitude"])

    # Προσθήκη επιστροφής στο depot
    return_dist = haversine(prev_lat, prev_lon, depot["latitude"],
depot["longitude"])
    total      += return_dist

    return enriched, round(total, 2)


# ---------------------------------------------------------------------------
--
# DB helpers
# ---------------------------------------------------------------------------
--

async def _fetch_bins(
    db: AsyncSession,
    zone_code: Optional[str],
    threshold: float,
) -> list[dict]:
    """Φόρτωση κάδων από DB βάσει ζώνης και κατωφλίου πληρότητας."""
    where  = "WHERE current_fill_level >= :threshold"
    params: dict = {"threshold": threshold}

    if zone_code and zone_code != "ALL":
        where += " AND zone_code = :zone_code"
        params["zone_code"] = zone_code

    result = await db.execute(
        text(f"""
            SELECT bin_id, location_name, address,
                   latitude, longitude,
                   current_fill_level, current_battery_level,
                   bin_type, fill_status, zone_code, zone_name
            FROM bin_status_summary
            {where}
            ORDER BY current_fill_level DESC
        """),
        params,
    )
    return [dict(r) for r in result.mappings().all()]


async def _save_route(
```

```
    db: AsyncSession,
    route: list[dict],
    zone_code: Optional[str],
    vehicle_id: Optional[str],
    total_km: float,
    threshold: float,
) -> Optional[int]:
    """Αποθήκευση διαδρομής στο collection_routes +
collection_route_bins."""
    try:
        # Εύρεση driver_name αν δόθηκε vehicle_id
        driver_name = None
        if vehicle_id:
            res = await db.execute(
                text("SELECT driver_name FROM vehicles WHERE vehicle_id =
:v"),
                {"v": vehicle_id},
            )
            row = res.mappings().first()
            if row:
                driver_name = row["driver_name"]

        route_name = (
            f"Διαδρομή {zone_code or 'Γενική'} — "
            f"{datetime.now().strftime('%d/%m/%Y %H:%M')}"
        )

        # Εισαγωγή route header
        res = await db.execute(
            text("""
                INSERT INTO collection_routes
                    (route_name, description, driver_name, vehicle_id,
                     scheduled_date, estimated_duration, status)
                VALUES
                    (:name, :desc, :driver, :vehicle,
                     :sched, :duration, 'scheduled')
                RETURNING id
            """),
            {
                "name":     route_name,
                "desc":     f"Αυτόματη βελτιστοποίηση | Ζώνη: {zone_code or
'ALL'} | "
                            f"Threshold: {threshold}% | {len(route)} κάδοι
| {total_km} km",
                "driver":   driver_name,
                "vehicle":  vehicle_id,
                "sched":    date.today(),
```

```
                "duration": int(total_km * 3),  # εκτίμηση: 3 λεπτά/km
            },
        )
        route_id = res.mappings().first()["id"]

        # Εισαγωγή stops
        for i, stop in enumerate(route):
            await db.execute(
                text("""
                    INSERT INTO collection_route_bins
                        (route_id, bin_id, sequence_order,
fill_level_at_collection)
                    VALUES (:rid, :bid, :seq, :fill)
                    ON CONFLICT (route_id, sequence_order) DO NOTHING
                """),
                {
                    "rid":  route_id,
                    "bid":  stop["bin_id"],
                    "seq":  i + 1,
                    "fill": stop["current_fill_level"],
                },
            )

        await db.commit()
        return route_id

    except Exception as exc:
        await db.rollback()
        logging.getLogger(__name__).error(f"Save route error: {exc}")
        return None


async def _notify_drivers(
    db: AsyncSession,
    vehicle_id: Optional[str],
    route_id: int,
    route: list[dict],
    zone_code: Optional[str],
) -> None:
    """
    ΠΧ2 Βήμα 9 — Ειδοποίηση οδηγών ζώνης μέσω vehicle_alerts.

    Λογική:
        - vehicle_id δόθηκε → ειδοποίηση μόνο αυτού
        - zone_code μόνο    → ειδοποίηση ΟΛΩΝ ενεργών οδηγών ζώνης
    """
    try:
```

```
        # Zone name για το μήνυμα
        zone_name = zone_code or "Γενική"
        if zone_code:
            res = await db.execute(
                text("SELECT zone_name FROM zones WHERE zone_code = :c"),
                {"c": zone_code},
            )
            row = res.mappings().first()
            if row:
                zone_name = row["zone_name"]

        # Εύρεση target οχημάτων
        if vehicle_id:
            res = await db.execute(
                text("SELECT vehicle_id, driver_name FROM vehicles "
                     "WHERE vehicle_id = :v AND status = 'active'"),
                {"v": vehicle_id},
            )
        else:
            res = await db.execute(
                text("""
                    SELECT v.vehicle_id, v.driver_name
                    FROM vehicles v
                    JOIN zones z ON v.zone_id = z.id
                    WHERE z.zone_code = :c AND v.status = 'active'
                    ORDER BY v.vehicle_id
                """),
                {"c": zone_code},
            )

        targets = res.mappings().all()
        if not targets:
            return

        first_stop = route[0]
        for veh in targets:
            msg = (
                f"Νέα διαδρομή #{route_id} για {veh['driver_name']} "
                f"({veh['vehicle_id']})! Ζώνη: {zone_name} | "
                f"{len(route)} κάδοι | Πρώτος:
{first_stop['location_name']} "
                f"({first_stop['current_fill_level']}%)."
            )
            await db.execute(
                text("""
                    INSERT INTO vehicle_alerts
                        (alert_type, vehicle_id, message, zone_code,
```

```
                        driver_name, initiated_by, status)
                    VALUES
                        ('route_assigned', :vid, :msg, :zone, :driver,
'system', 'active')
                """),
                {
                    "vid":    veh["vehicle_id"],
                    "msg":    msg,
                    "zone":   zone_code,
                    "driver": veh["driver_name"],
                },
            )

        await db.commit()

    except Exception as exc:
        logging.getLogger(__name__).error(f"Notify drivers error: {exc}")


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.get("/optimization/route")
async def get_optimized_route(
    threshold:  float           = 70.0,
    zone_code:  Optional[str]   = None,
    vehicle_id: Optional[str]   = None,
    save_route: bool            = False,
    db:         AsyncSession    = Depends(get_db),
) -> dict:
    """
    ΠΧ2 – Υπολογισμός βέλτιστης διαδρομής αποκομιδής.

    Query params:
        threshold:  Κατώφλι πληρότητας (0-100). Default: 70
        zone_code:  Φίλτρο ζώνης (GAZ/AMM/APE/TYL/KRO/ROD/ALL)
        vehicle_id: Αποστολή ειδοποίησης σε συγκεκριμένο οδηγό
        save_route: Αποθήκευση στο collection_routes
    """
    try:
        # Βήμα 3: Φόρτωση κάδων
        bins          = await _fetch_bins(db, zone_code, threshold)
        fallback_used = False
```

```
        # Εναλλακτική Ροή 1: Auto-fallback αν <3 κάδοι
        if len(bins) < MIN_BINS_FOR_TSP:
            logging.getLogger(__name__).info(
                f"<{MIN_BINS_FOR_TSP} κάδοι @ {threshold}% → fallback
{FALLBACK_THRESHOLD}%"
            )
            bins          = await _fetch_bins(db, zone_code,
FALLBACK_THRESHOLD)
            threshold     = FALLBACK_THRESHOLD
            fallback_used = True

        if not bins:
            return {
                "status":   "no_bins",
                "message":  f"Δεν υπάρχουν κάδοι ≥{threshold}% στη ζώνη
{zone_code or 'ALL'}",
                "route":    [],
                "total_bins": 0,
                "total_distance_km": 0,
            }

        # Βήμα 6: TSP
        optimized_route             = nearest_neighbor_tsp(bins, DEPOT)
        route_with_dist, total_km   = add_distance_info(optimized_route,
DEPOT)

        # Βήμα 7: Αποθήκευση
        route_id = None
        if save_route:
            route_id = await _save_route(
                db, route_with_dist, zone_code, vehicle_id, total_km,
threshold
            )

        # Βήμα 9: Ειδοποίηση οδηγών
        if save_route and route_id and zone_code and zone_code != "ALL":
            await _notify_drivers(db, vehicle_id, route_id,
route_with_dist, zone_code)

        return {
            "status":             "success",
            "algorithm":          "Nearest Neighbor TSP Heuristic",
            "distance_mode":      "haversine",
            "threshold":          threshold,
            "fallback_used":      fallback_used,
            "zone_code":          zone_code or "ALL",
            "total_bins":         len(optimized_route),
```

```
            "total_distance_km":   total_km,
            "estimated_minutes":   int(total_km * 3),
            "depot":               DEPOT,
            "route":               route_with_dist,
            "route_id":            route_id,
            "saved":               save_route,
        }

    except Exception as exc:
        logging.getLogger(__name__).error(f"Route optimization error:
{exc}")
        raise HTTPException(500, str(exc))


@router.get("/optimization/routes-history")
async def get_routes_history(
    zone_code: Optional[str] = None,
    db:        AsyncSession  = Depends(get_db),
) -> dict:
    """Ιστορικό αποθηκευμένων διαδρομών."""
    where  = "WHERE 1=1"
    params: dict = {}

    if zone_code and zone_code != "ALL":
        where += " AND description LIKE :zone"
        params["zone"] = f"%{zone_code}%"

    result = await db.execute(
        text(f"""
            SELECT id, route_name, description, driver_name,
                   vehicle_id, scheduled_date, estimated_duration,
                   status, created_at
            FROM collection_routes
            {where}
            ORDER BY created_at DESC
            LIMIT 20
        """),
        params,
    )
    routes = [dict(r) for r in result.mappings().all()]
    return {"status": "success", "total": len(routes), "routes": routes}


@router.get("/optimization/osm-status")
async def get_osm_status() -> dict:
    """Κατάσταση OSMnx — διαθεσιμότητα βιβλιοθήκης."""
    try:
```

```
        import osmnx as ox
        return {
            "osmnx_available": True,
            "osmnx_version":   ox.__version__,
            "note": "OSMnx εγκατεστημένο αλλά Overpass API απαιτεί
σύνδεση.",
        }
    except ImportError:
        return {
            "osmnx_available": False,
            "note": "Χρησιμοποιείται Haversine fallback.",
        }
```

Backend\app\api\routes\reports.py

```
"""
app/api/routes/reports.py
=========================
Αναφορές συστήματος, bug reports και στατιστικά.

Endpoints:
    GET  /reports              → Λίστα αναφορών συστήματος
    POST /reports              → Υποβολή νέας αναφοράς/bug report
    GET  /reports/{id}         → Λεπτομέρειες αναφοράς
    PUT  /reports/{id}/resolve → Επίλυση αναφοράς
    GET  /reports/stats        → Στατιστικά συστήματος
"""

import logging
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from pydantic import BaseModel
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()

# Έγκυροι τύποι αναφοράς
REPORT_TYPES = {
    "bug",
    "feature_request",
    "maintenance",
    "other",
```

```
}

# Ετικέτες κατάστασης
STATUS_LABELS: dict[str, str] = {
    "open":        "Ανοιχτή",
    "in_progress": "Σε Εξέλιξη",
    "resolved":    "Επιλύθηκε",
    "closed":      "Κλειστή",
}


# ---------------------------------------------------------------------------
--
# Schemas
# ---------------------------------------------------------------------------
--

class ReportCreate(BaseModel):
    """Δημιουργία νέας αναφοράς / bug report."""
    title:       str
    description: str
    report_type: str = "bug"
    priority:    str = "medium"
    submitted_by: Optional[str] = None


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.get("/reports")
async def get_reports(
    report_type: Optional[str] = None,
    status:      Optional[str] = None,
    priority:    Optional[str] = None,
    db:          AsyncSession  = Depends(get_db),
) -> dict:
    """
    Λίστα αναφορών συστήματος με φίλτρα.

    Query params:
        report_type: bug / feature_request / maintenance / other
        status:      open / in_progress / resolved / closed
        priority:    low / medium / high / critical
    """
```

```
    where_clauses = ["1=1"]
    params: dict  = {}

    if report_type and report_type != "ALL":
        where_clauses.append("report_type = :report_type")
        params["report_type"] = report_type

    if status and status != "ALL":
        where_clauses.append("status = :status")
        params["status"] = status

    if priority and priority != "ALL":
        where_clauses.append("priority = :priority")
        params["priority"] = priority

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                id, title, description, report_type,
                status, priority, submitted_by,
                created_at, resolved_at, resolution_notes
            FROM bug_reports
            WHERE {where}
            ORDER BY
                CASE priority
                    WHEN 'critical' THEN 1
                    WHEN 'high'     THEN 2
                    WHEN 'medium'   THEN 3
                    ELSE 4
                END,
                created_at DESC
            LIMIT 100
        """),
        params,
    )

    reports = []
    for row in result.mappings().all():
        reports.append({
            "id":              row["id"],
            "title":           row["title"],
            "description":     row["description"],
            "report_type":     row["report_type"],
            "status":          row["status"],
```

```
            "status_label":     STATUS_LABELS.get(row["status"],
row["status"]),
            "priority":         row["priority"],
            "submitted_by":     row["submitted_by"],
            "created_at":       str(row["created_at"]),
            "resolved_at":      str(row["resolved_at"]) if
row["resolved_at"] else None,
            "resolution_notes": row["resolution_notes"],
        })

    return {
        "status":  "success",
        "total":   len(reports),
        "reports": reports,
    }


@router.post("/reports")
async def create_report(
    req: ReportCreate,
    db:  AsyncSession = Depends(get_db),
) -> dict:
    """Υποβολή νέας αναφοράς / bug report."""
    if req.report_type not in REPORT_TYPES:
        raise HTTPException(400, f"Μη έγκυρος τύπος: {req.report_type}")

    result = await db.execute(
        text("""
            INSERT INTO bug_reports
                (title, description, report_type, priority, submitted_by,
status)
            VALUES
                (:title, :description, :report_type, :priority,
:submitted_by, 'open')
            RETURNING id, created_at
        """),
        {
            "title":        req.title,
            "description":  req.description,
            "report_type":  req.report_type,
            "priority":     req.priority,
            "submitted_by": req.submitted_by,
        },
    )
    row = result.mappings().first()
    await db.commit()
```

```
    logger.info(f"New report #{row['id']}: {req.title}
({req.report_type})")

    return {
        "status":     "success",
        "id":         row["id"],
        "created_at": str(row["created_at"]),
        "message":    f"Αναφορά #{row['id']} υποβλήθηκε επιτυχώς.",
    }


@router.get("/reports/stats")
async def get_system_stats(db: AsyncSession = Depends(get_db)) -> dict:
    """
    Στατιστικά ολόκληρου του συστήματος.
    Χρησιμοποιείται στο overview section του dashboard.
    """
    # Στατιστικά κάδων
    bins_result = await db.execute(text("""
        SELECT
            COUNT(*)                                                AS
total_bins,
            COUNT(CASE WHEN fill_status = 'critical' THEN 1 END)  AS
critical_bins,
            COUNT(CASE WHEN fill_status = 'warning'  THEN 1 END)  AS
warning_bins,
            ROUND(AVG(current_fill_level)::numeric, 1)            AS
avg_fill
        FROM bin_status_summary
    """))
    bins_row = bins_result.mappings().first()

    # Στατιστικά οχημάτων
    vehicles_result = await db.execute(text("""
        SELECT
            COUNT(*)                                                AS
total,
            COUNT(CASE WHEN status = 'active'      THEN 1 END)   AS active,
            COUNT(CASE WHEN status = 'maintenance' THEN 1 END)   AS
maintenance,
            COUNT(CASE WHEN status = 'inactive'    THEN 1 END)   AS
inactive
        FROM vehicles
    """))
    vehicles_row = vehicles_result.mappings().first()

    # Στατιστικά αναφορών πολιτών
```

```
    citizen_result = await db.execute(text("""
        SELECT
            COUNT(*)                                                AS total,
            COUNT(CASE WHEN status = 'open'        THEN 1 END)  AS open,
            COUNT(CASE WHEN status = 'in_progress' THEN 1 END)  AS
in_progress,
            COUNT(CASE WHEN status = 'resolved'    THEN 1 END)  AS resolved
        FROM citizen_reports
    """))
    citizen_row = citizen_result.mappings().first()

    # Στατιστικά ειδοποιήσεων (τελευταίες 24 ώρες)
    alerts_result = await db.execute(text("""
        SELECT COUNT(*) AS total
        FROM vehicle_alerts
        WHERE created_at >= NOW() - INTERVAL '24 hours'
          AND status = 'active'
    """))
    alerts_row = alerts_result.mappings().first()

    # Διαδρομές σήμερα
    routes_result = await db.execute(text("""
        SELECT COUNT(*) AS total
        FROM collection_routes
        WHERE scheduled_date = CURRENT_DATE
    """))
    routes_row = routes_result.mappings().first()

    return {
        "status": "success",
        "bins": {
            "total":         int(bins_row["total_bins"]    or 0),
            "critical":      int(bins_row["critical_bins"] or 0),
            "warning":       int(bins_row["warning_bins"]  or 0),
            "avg_fill":      float(bins_row["avg_fill"]    or 0),
        },
        "vehicles": {
            "total":       int(vehicles_row["total"]       or 0),
            "active":      int(vehicles_row["active"]      or 0),
            "maintenance": int(vehicles_row["maintenance"] or 0),
            "inactive":    int(vehicles_row["inactive"]    or 0),
        },
        "citizen_reports": {
            "total":       int(citizen_row["total"]       or 0),
            "open":        int(citizen_row["open"]        or 0),
            "in_progress": int(citizen_row["in_progress"] or 0),
            "resolved":    int(citizen_row["resolved"]    or 0),
```

```
        },
        "alerts_last_24h": int(alerts_row["total"]  or 0),
        "routes_today":    int(routes_row["total"]  or 0),
    }


@router.get("/reports/{report_id}")
async def get_report(
    report_id: int,
    db:        AsyncSession = Depends(get_db),
) -> dict:
    """Λεπτομέρειες συγκεκριμένης αναφοράς."""
    result = await db.execute(
        text("""
            SELECT
                id, title, description, report_type,
                status, priority, submitted_by,
                created_at, resolved_at, resolution_notes
            FROM bug_reports
            WHERE id = :id
        """),
        {"id": report_id},
    )
    row = result.mappings().first()

    if not row:
        raise HTTPException(404, f"Αναφορά #{report_id} δεν βρέθηκε.")

    return {
        "status": "success",
        "report": {
            "id":               row["id"],
            "title":            row["title"],
            "description":      row["description"],
            "report_type":      row["report_type"],
            "status":           row["status"],
            "status_label":     STATUS_LABELS.get(row["status"],
row["status"]),
            "priority":         row["priority"],
            "submitted_by":     row["submitted_by"],
            "created_at":       str(row["created_at"]),
            "resolved_at":      str(row["resolved_at"]) if
row["resolved_at"] else None,
            "resolution_notes": row["resolution_notes"],
        },
    }
```

```
@router.put("/reports/{report_id}/resolve")
async def resolve_report(
    report_id: int,
    data:      dict,
    db:        AsyncSession = Depends(get_db),
) -> dict:
    """
    Επίλυση αναφοράς.

    Body:
        resolution_notes: str – Σημειώσεις επίλυσης (προαιρετικό)
        status: str – Νέα κατάσταση (resolved / closed)
    """
    new_status = data.get("status", "resolved")
    if new_status not in ("resolved", "closed"):
        raise HTTPException(400, "Status πρέπει να είναι 'resolved' ή
'closed'.")

    await db.execute(
        text("""
            UPDATE bug_reports
            SET status           = :status,
                resolved_at      = NOW(),
                resolution_notes = :notes
            WHERE id = :id
        """),
        {
            "status": new_status,
            "notes":  data.get("resolution_notes"),
            "id":     report_id,
        },
    )
    await db.commit()

    return {
        "status":  "success",
        "message": f"Αναφορά #{report_id} επιλύθηκε.",
    }
```

backend\app\api\routes\sensors.py

```
"""
app/api/routes/sensors.py
==========================
IoT αισθητήρες – λήψη μετρήσεων και ειδοποιήσεων.

Endpoints:
```

```
    POST /sensors/readings → Αποστολή μέτρησης από αισθητήρα (IoT)
    GET  /sensors/alerts   → Ενεργές ειδοποιήσεις κάδων
"""

import logging
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from pydantic import BaseModel, Field
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()

# Κατώφλια ειδοποίησης (%)
ALERT_CRITICAL = 90.0
ALERT_WARNING  = 80.0


# ---------------------------------------------------------------------------
--
# Schemas
# ---------------------------------------------------------------------------
--

class SensorReading(BaseModel):
    """Μέτρηση IoT αισθητήρα κάδου."""
    bin_id:        str
    fill_level:    float = Field(ge=0, le=100, description="Πληρότητα 0-
100%")
    battery_level: float = Field(ge=0, le=100, description="Μπαταρία 0-
100%")
    temperature:   Optional[float] = None
    latitude:      Optional[float] = None
    longitude:     Optional[float] = None


# ---------------------------------------------------------------------------
--
# Helpers
# ---------------------------------------------------------------------------
--

async def _create_alert_if_needed(
```

```
    db:        AsyncSession,
    bin_id:    str,
    fill_level: float,
) -> Optional[str]:
    """
    Δημιουργία ειδοποίησης αν η πληρότητα υπερβαίνει το κατώφλι.

    Returns:
        Τύπος ειδοποίησης ('critical'/'warning') ή None.
    """
    if fill_level < ALERT_WARNING:
        return None

    severity = "critical" if fill_level >= ALERT_CRITICAL else "warning"
    alert_type = "bin_critical" if severity == "critical" else
"bin_warning"

    # Έλεγχος αν υπάρχει ήδη ενεργή ειδοποίηση για αυτόν τον κάδο
    existing = await db.execute(
        text("""
            SELECT id FROM alerts
            WHERE bin_id = :bin_id
              AND alert_type = :alert_type
              AND is_acknowledged = FALSE
            LIMIT 1
        """),
        {"bin_id": bin_id, "alert_type": alert_type},
    )
    if existing.fetchone():
        return None  # Υπάρχει ήδη ειδοποίηση

    # Δημιουργία νέας ειδοποίησης
    await db.execute(
        text("""
            INSERT INTO alerts (bin_id, alert_type, severity, message,
is_acknowledged)
            VALUES (:bin_id, :alert_type, :severity, :message, FALSE)
        """),
        {
            "bin_id":     bin_id,
            "alert_type": alert_type,
            "severity":   severity,
            "message":    f"Κάδος {bin_id}: πληρότητα {fill_level:.1f}% –
απαιτείται αποκομιδή.",
        },
    )
    logger.info(f"Alert created: {bin_id} @ {fill_level}% ({severity})")
```

```
    return severity


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.post("/sensors/readings")
async def receive_sensor_reading(
    reading: SensorReading,
    db:      AsyncSession = Depends(get_db),
) -> dict:
    """
    Λήψη μέτρησης από IoT αισθητήρα (ESP32 / Simulator).

    Αποθηκεύει στο TimescaleDB hypertable sensor_data.
    Ενημερώνει GPS αν δόθηκαν συντεταγμένες.
    Δημιουργεί ειδοποίηση αν fill_level ≥ 80%.
    """
    # Έλεγχος ύπαρξης κάδου
    result = await db.execute(
        text("SELECT bin_id FROM bins WHERE bin_id = :bin_id"),
        {"bin_id": reading.bin_id},
    )
    if not result.fetchone():
        raise HTTPException(404, f"Κάδος {reading.bin_id} δεν βρέθηκε.")

    # Αποθήκευση στο TimescaleDB
    await db.execute(
        text("""
            INSERT INTO sensor_data
                (time, bin_id, fill_level, battery_level, temperature)
            VALUES
                (NOW(), :bin_id, :fill, :battery, :temp)
        """),
        {
            "bin_id":  reading.bin_id,
            "fill":    reading.fill_level,
            "battery": reading.battery_level,
            "temp":    reading.temperature,
        },
    )

    # Ενημέρωση GPS αν δόθηκαν συντεταγμένες
    if reading.latitude and reading.longitude:
```

```
        await db.execute(
            text("""
                UPDATE bins
                SET latitude = :lat, longitude = :lon
                WHERE bin_id = :bin_id
            """),
            {
                "lat":    reading.latitude,
                "lon":    reading.longitude,
                "bin_id": reading.bin_id,
            },
        )

    # Ειδοποίηση αν χρειάζεται
    alert_created = await _create_alert_if_needed(
        db, reading.bin_id, reading.fill_level
    )

    await db.commit()

    return {
        "status":        "success",
        "bin_id":        reading.bin_id,
        "fill_level":    reading.fill_level,
        "battery_level": reading.battery_level,
        "alert_created": alert_created,
    }


@router.get("/sensors/alerts")
async def get_sensor_alerts(
    zone_code:    Optional[str] = None,
    acknowledged: bool          = False,
    db:           AsyncSession  = Depends(get_db),
) -> dict:
    """
    Ενεργές ειδοποιήσεις κάδων.

    Query params:
        zone_code:    Φίλτρο ζώνης
        acknowledged: True = εμφάνιση επιβεβαιωμένων, False = μόνο ενεργές
    """
    where_clauses = []
    params: dict  = {}

    if not acknowledged:
        where_clauses.append("a.is_acknowledged = FALSE")
```

```
    if zone_code and zone_code != "ALL":
        where_clauses.append("b.zone_code = :zone_code")
        params["zone_code"] = zone_code

    where = ("WHERE " + " AND ".join(where_clauses)) if where_clauses else
""

    result = await db.execute(
        text(f"""
            SELECT
                a.id, a.bin_id, a.alert_type, a.severity,
                a.message, a.is_acknowledged,
                a.created_at, a.acknowledged_at,
                b.location_name, b.zone_code
            FROM alerts a
            JOIN bins b ON a.bin_id = b.bin_id
            {where}
            ORDER BY
                CASE a.severity WHEN 'critical' THEN 1 ELSE 2 END,
                a.created_at DESC
            LIMIT 100
        """),
        params,
    )

    alerts = [dict(r) for r in result.mappings().all()]
    return {
        "status": "success",
        "total":  len(alerts),
        "alerts": alerts,
    }


@router.put("/sensors/alerts/{alert_id}/acknowledge")
async def acknowledge_alert(
    alert_id: int,
    db:       AsyncSession = Depends(get_db),
) -> dict:
    """Επιβεβαίωση (acknowledgement) ειδοποίησης κάδου."""
    await db.execute(
        text("""
            UPDATE alerts
            SET is_acknowledged = TRUE, acknowledged_at = NOW()
            WHERE id = :id
        """),
        {"id": alert_id},
```

```
    )
    await db.commit()
    return {"status": "success", "message": "Ειδοποίηση επιβεβαιώθηκε."}
```

backend\app\api\routes\smart_dispatch.py

```
"""
app/api/routes/smart_dispatch.py
================================
Διαχείριση αποστολής οχημάτων, βλαβών και μεταφορών ΧΥΤΑ.

Endpoints:
    GET  /dispatch/status               → Κατάσταση ζωνών & οχημάτων
    POST /dispatch/driver-action        → Ενέργεια οδηγού
    POST /dispatch/vehicle-breakdown    → Διαχείριση βλάβης
    POST /dispatch/xyta-trip            → Μεταφορά στο ΧΥΤΑ
    GET  /dispatch/alerts               → Ειδοποιήσεις με φίλτρα
    PUT  /dispatch/alerts/{id}/acknowledge → Επιβεβαίωση ειδοποίησης
    GET  /dispatch/xyta-trips           → Ιστορικό ΧΥΤΑ
    GET  /dispatch/zone-vehicles/{zone} → Οχήματα ζώνης
"""

import logging
import math
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()

# ΧΥΤΑ Πέρα Γαλήνων
XYTA = {
    "lat":  35.39236790444294,
    "lon":  24.921436396669584,
    "name": "ΧΥΤΑ Πέρα Γαλήνων",
}

# Ετικέτες ενεργειών οδηγού
ACTION_LABELS: dict[str, str] = {
    "bin_collected":      "Αποκομιδή Κάδου",
    "vehicle_full":       "Όχημα Γεμάτο",
    "vehicle_restored":   "Επαναφορά σε Ενεργό",
```

```
    "breakdown":           "Βλάβη",
    "maintenance":         "Συντήρηση",
    "xyta_arrived":        "Άφιξη ΧΥΤΑ",
    "xyta_trip":           "Μεταφορά ΧΥΤΑ",
    "zone_change":         "Αλλαγή Ζώνης",
    "zone_change_request": "Αίτημα Αλλαγής Ζώνης",
}


# ---------------------------------------------------------------------------
--
# Helpers
# ---------------------------------------------------------------------------
--

def haversine(lat1: float, lon1: float, lat2: float, lon2: float) -> float:
    """Αεροπορική απόσταση σε km."""
    R    = 6371
    dlat = math.radians(lat2 - lat1)
    dlon = math.radians(lon2 - lon1)
    a    = (math.sin(dlat / 2) ** 2
            + math.cos(math.radians(lat1))
            * math.cos(math.radians(lat2))
            * math.sin(dlon / 2) ** 2)
    return R * 2 * math.asin(math.sqrt(a))


async def _insert_vehicle_alert(
    db:         AsyncSession,
    alert_type: str,
    vehicle_id: str,
    message:    str,
    zone_code:  Optional[str]  = None,
    driver_name: Optional[str] = None,
    fill_pct:   float          = 0,
    bins_count: int            = 0,
    notes:      Optional[str]  = None,
    initiated_by: str          = "driver",
    nearest_vehicle_id: Optional[str] = None,
    suggested_zone_code: Optional[str] = None,
) -> None:
    """Κεντρική συνάρτηση εισαγωγής vehicle_alert."""
    await db.execute(
        text("""
            INSERT INTO vehicle_alerts (
                alert_type, vehicle_id, message, zone_code, driver_name,
                fill_level_pct, collected_bins_count, notes,
```

```
                initiated_by, status,
                nearest_vehicle_id, suggested_zone_code
            ) VALUES (
                :alert_type, :vehicle_id, :message, :zone_code,
:driver_name,
                :fill_pct, :bins_count, :notes,
                :initiated_by, 'active',
                :nearest_vid, :suggested_zone
            )
        """),
        {
            "alert_type":   alert_type,
            "vehicle_id":   vehicle_id,
            "message":      message,
            "zone_code":    zone_code,
            "driver_name":  driver_name,
            "fill_pct":     fill_pct,
            "bins_count":   bins_count,
            "notes":        notes,
            "initiated_by": initiated_by,
            "nearest_vid":  nearest_vehicle_id,
            "suggested_zone": suggested_zone_code,
        },
    )


async def _find_nearest_vehicle(
    db:        AsyncSession,
    from_lat:  float,
    from_lon:  float,
    exclude_zone: str,
) -> Optional[dict]:
    """Εύρεση κοντινότερου ενεργού οχήματος εκτός της τρέχουσας ζώνης."""
    result = await db.execute(
        text("""
            SELECT v.vehicle_id, v.plate_number, v.driver_name,
                   v.latitude, v.longitude,
                   z.zone_code, z.zone_name, z.center_lat, z.center_lon
            FROM vehicles v
            JOIN zones z ON v.zone_id = z.id
            WHERE v.status = 'active'
              AND z.zone_code != :zone
              AND z.zone_code != 'XYTA'
        """),
        {"zone": exclude_zone},
    )
```

```
    nearest, min_dist = None, float("inf")
    for veh in result.mappings().all():
        veh_lat = float(veh["latitude"]   or veh["center_lat"])
        veh_lon = float(veh["longitude"]  or veh["center_lon"])
        dist    = haversine(from_lat, from_lon, veh_lat, veh_lon)
        if dist < min_dist:
            min_dist = dist
            nearest  = {
                "vehicle_id":  veh["vehicle_id"],
                "driver_name": veh["driver_name"],
                "zone_code":   veh["zone_code"],
                "zone_name":   veh["zone_name"],
                "distance_km": round(dist, 1),
                "latitude":    veh_lat,
                "longitude":   veh_lon,
            }

    return nearest


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.get("/dispatch/status")
async def get_dispatch_status(db: AsyncSession = Depends(get_db)) -> dict:
    """
    Τρέχουσα κατάσταση διάταξης — ζώνες, οχήματα, αποστάσεις ΧΥΤΑ.
    """
    result = await db.execute(text("""
        SELECT
            z.zone_code, z.zone_name,
            z.center_lat, z.center_lon,
            COUNT(v.id)                                              AS
total_vehicles,
            COUNT(CASE WHEN v.status = 'active'      THEN 1 END)    AS
active_vehicles,
            COUNT(CASE WHEN v.status = 'maintenance' THEN 1 END)    AS
maintenance_vehicles,
            COUNT(CASE WHEN v.status = 'inactive'    THEN 1 END)    AS
inactive_vehicles,
            json_agg(json_build_object(
                'vehicle_id',    v.vehicle_id,
                'plate_number',  v.plate_number,
                'driver_name',   v.driver_name,
```

```
                'status',        v.status,
                'fuel_level',    v.fuel_level,
                'latitude',      v.latitude,
                'longitude',     v.longitude,
                'vehicle_type',  v.vehicle_type,
                'capacity_tons', v.capacity_tons,
                'capacity_m3',   v.capacity_m3
            ) ORDER BY v.vehicle_id) AS vehicles
        FROM zones z
        LEFT JOIN vehicles v ON v.zone_id = z.id
        WHERE z.zone_code != 'XYTA'
        GROUP BY z.zone_code, z.zone_name, z.center_lat, z.center_lon
        ORDER BY z.zone_code
    """))

    zones = []
    for row in result.mappings().all():
        clat = float(row["center_lat"]) if row["center_lat"] else None
        clon = float(row["center_lon"]) if row["center_lon"] else None

        zones.append({
            "zone_code":           row["zone_code"],
            "zone_name":           row["zone_name"],
            "center_lat":          clat,
            "center_lon":          clon,
            "total_vehicles":      int(row["total_vehicles"]      or 0),
            "active_vehicles":     int(row["active_vehicles"]     or 0),
            "maintenance_vehicles": int(row["maintenance_vehicles"] or 0),
            "inactive_vehicles":   int(row["inactive_vehicles"]   or 0),
            "vehicles":            row["vehicles"] or [],
        })

    # Αποστάσεις ζωνών → XYTA
    xyta_distances = {
        z["zone_code"]: round(haversine(
            z["center_lat"], z["center_lon"], XYTA["lat"], XYTA["lon"]
        ), 1)
        for z in zones
        if z["center_lat"] and z["center_lon"]
    }

    return {
        "status": "success",
        "zones":  zones,
        "xyta": {
            "name":                XYTA["name"],
            "latitude":            XYTA["lat"],
```

```
            "longitude":             XYTA["lon"],
            "distances_from_zones":  xyta_distances,
        },
    }


@router.post("/dispatch/driver-action")
async def driver_action(
    data: dict,
    db:   AsyncSession = Depends(get_db),
) -> dict:
    """
    Καταγραφή ενέργειας οδηγού.

    action_type: bin_collected | vehicle_full | breakdown |
                 maintenance | xyta_arrived | zone_change_request |
vehicle_restored
    """
    vehicle_id     = data.get("vehicle_id")
    action_type    = data.get("action_type")
    driver_name    = data.get("driver_name", "")
    zone_code      = data.get("zone_code", "")
    collected_bins = data.get("collected_bins_count", 0)
    fill_pct       = data.get("fill_level_pct", 0)
    notes          = data.get("notes", "")
    bin_id         = data.get("bin_id", "")

    # Δυναμικό μήνυμα ανά τύπο ενέργειας
    messages: dict[str, str] = {
        "bin_collected": (
            f"Ο οδηγός {driver_name} ({vehicle_id}) άδειασε κάδο "
            f"{bin_id} στη ζώνη {zone_code}. Σύνολο: {collected_bins}
κάδοι."
        ),
        "vehicle_full": (
            f"Το όχημα {vehicle_id} ({driver_name}) είναι "
            f"{fill_pct:.1f}% γεμάτο ({collected_bins} κάδοι). Κατευθύνεται
στο XYTA."
        ),
        "breakdown": (
            f"ΒΛΑΒΗ! Το όχημα {vehicle_id} ({driver_name}) "
            f"παρουσίασε βλάβη στη ζώνη {zone_code}. {notes}"
        ),
        "maintenance": (
            f"Το όχημα {vehicle_id} ({driver_name}) πάει για "
            f"συντήρηση. Ζώνη: {zone_code}. {notes}"
        ),
```

```
        "xyta_arrived": (
            f"Το όχημα {vehicle_id} ({driver_name}) έφτασε στο ΧΥΤΑ και
εκφόρτωσε."
        ),
        "zone_change_request": (
            f"Ο οδηγός {driver_name} ({vehicle_id}) ζητά αλλαγή ζώνης από
{zone_code}. {notes}"
        ),
        "vehicle_restored": (
            f"Το όχημα {vehicle_id} ({driver_name}) επαναφέρθηκε σε ενεργό
"
            f"στη ζώνη {zone_code}."
        ),
    }
    message = messages.get(action_type, f"Ενέργεια {action_type} από
{vehicle_id}")

    await _insert_vehicle_alert(
        db,
        alert_type=action_type,
        vehicle_id=vehicle_id,
        message=message,
        zone_code=zone_code,
        driver_name=driver_name,
        fill_pct=fill_pct,
        bins_count=collected_bins,
        notes=notes,
        initiated_by="driver",
    )

    nearest = None

    # Βλάβη / Συντήρηση → ανεύρεση κοντινότερου + ειδοποίηση
    if action_type in ("breakdown", "maintenance"):
        new_status = "maintenance" if action_type == "maintenance" else
"inactive"
        await db.execute(
            text("UPDATE vehicles SET status = :s WHERE vehicle_id = :v"),
            {"s": new_status, "v": vehicle_id},
        )

        # Θέση από ζώνη
        zone_res = await db.execute(
            text("SELECT center_lat, center_lon FROM zones WHERE zone_code
= :c"),
            {"c": zone_code},
        )
```

```
        zone_row = zone_res.mappings().first()
        if zone_row:
            nearest = await _find_nearest_vehicle(
                db,
                float(zone_row["center_lat"]),
                float(zone_row["center_lon"]),
                exclude_zone=zone_code,
            )
            if nearest:
                await _insert_vehicle_alert(
                    db,
                    alert_type="zone_change",
                    vehicle_id=nearest["vehicle_id"],
                    message=(
                        f"Καλείσαι να καλύψεις τη ζώνη {zone_code} "
                        f"λόγω {action_type} του {vehicle_id}
({driver_name}). "
                        f"Απόσταση: {nearest['distance_km']} km."
                    ),
                    zone_code=zone_code,
                    initiated_by="system",
                    nearest_vehicle_id=vehicle_id,
                    suggested_zone_code=nearest["zone_code"],
                )

    # ΧΥΤΑ άφιξη → επαναφορά οχήματος σε active
    if action_type == "xyta_arrived":
        await db.execute(
            text("""
                UPDATE xyta_trips
                SET status = 'arrived', arrived_at = NOW()
                WHERE vehicle_id = :v AND status = 'en_route'
            """),
            {"v": vehicle_id},
        )
        await db.execute(
            text("UPDATE vehicles SET status = 'active' WHERE vehicle_id =
:v"),
            {"v": vehicle_id},
        )

    await db.commit()

    return {
        "status":          "success",
        "action_type":     action_type,
        "action_label":    ACTION_LABELS.get(action_type, action_type),
```

```
        "message":         message,
        "nearest_vehicle": nearest,
    }


@router.post("/dispatch/vehicle-breakdown")
async def handle_vehicle_breakdown(
    data: dict,
    db:   AsyncSession = Depends(get_db),
) -> dict:
    """
    Διαχείριση βλάβης από κεντρικό σύστημα (διαχειριστής).
    Υπολογίζει κοντινότερο διαθέσιμο όχημα.
    """
    vehicle_id = data.get("vehicle_id")
    reason     = data.get("reason", "breakdown")

    # Στοιχεία βλαβερού οχήματος
    result = await db.execute(
        text("""
            SELECT v.vehicle_id, v.plate_number, v.driver_name,
                   v.latitude, v.longitude, v.fuel_level,
                   z.zone_code, z.zone_name, z.center_lat, z.center_lon
            FROM vehicles v
            JOIN zones z ON v.zone_id = z.id
            WHERE v.vehicle_id = :v
        """),
        {"v": vehicle_id},
    )
    vehicle = result.mappings().first()
    if not vehicle:
        raise HTTPException(404, f"Όχημα {vehicle_id} δεν βρέθηκε.")

    veh_lat = float(vehicle["latitude"]  or vehicle["center_lat"])
    veh_lon = float(vehicle["longitude"] or vehicle["center_lon"])

    nearest     = await _find_nearest_vehicle(db, veh_lat, veh_lon,
vehicle["zone_code"])
    xyta_dist   = haversine(veh_lat, veh_lon, XYTA["lat"], XYTA["lon"])

    # Alert για βλάβη
    alert_msg = (
        f"Βλάβη οχήματος {vehicle_id} ({vehicle['driver_name']}) "
        f"στη ζώνη {vehicle['zone_name']}. Λόγος: {reason}."
    )
    if nearest:
        alert_msg += (
```

```
            f" Κοντινότερο: {nearest['vehicle_id']} "
            f"({nearest['driver_name']}) — {nearest['distance_km']} km."
        )

    await _insert_vehicle_alert(
        db,
        alert_type=reason,
        vehicle_id=vehicle_id,
        message=alert_msg,
        initiated_by="system",
        nearest_vehicle_id=nearest["vehicle_id"] if nearest else None,
        suggested_zone_code=vehicle["zone_code"],
    )

    # Αλλαγή status
    new_status = "maintenance" if reason == "maintenance" else "inactive"
    await db.execute(
        text("UPDATE vehicles SET status = :s WHERE vehicle_id = :v"),
        {"s": new_status, "v": vehicle_id},
    )
    await db.commit()

    return {
        "status": "success",
        "breakdown_vehicle": {
            "vehicle_id":  vehicle_id,
            "driver_name": vehicle["driver_name"],
            "zone":        vehicle["zone_name"],
            "reason":      reason,
            "new_status":  new_status,
        },
        "nearest_vehicle": nearest,
        "xyta_route": {
            "name":              XYTA["name"],
            "latitude":          XYTA["lat"],
            "longitude":         XYTA["lon"],
            "distance_km":       round(xyta_dist, 1),
            "estimated_minutes": int(xyta_dist * 2.5),
        },
        "message": alert_msg,
    }


@router.post("/dispatch/xyta-trip")
async def register_xyta_trip(
    data: dict,
    db:   AsyncSession = Depends(get_db),
```

```
) -> dict:
    """Καταγραφή μεταφοράς απορριμμάτων στο XYTA."""
    vehicle_id     = data.get("vehicle_id")
    reason         = data.get("reason", "full")
    collected_bins = data.get("collected_bins_count", 0)
    fill_pct       = data.get("fill_level_pct", 0)

    result = await db.execute(
        text("""
            SELECT v.vehicle_id, v.driver_name,
                   v.latitude, v.longitude,
                   z.zone_code, z.zone_name, z.center_lat, z.center_lon
            FROM vehicles v
            JOIN zones z ON v.zone_id = z.id
            WHERE v.vehicle_id = :v
        """),
        {"v": vehicle_id},
    )
    vehicle = result.mappings().first()
    if not vehicle:
        raise HTTPException(404, "Όχημα δεν βρέθηκε.")

    dep_lat  = float(vehicle["latitude"]  or vehicle["center_lat"])
    dep_lon  = float(vehicle["longitude"] or vehicle["center_lon"])
    distance = haversine(dep_lat, dep_lon, XYTA["lat"], XYTA["lon"])

    # Εισαγωγή trip
    await db.execute(
        text("""
            INSERT INTO xyta_trips
                (vehicle_id, driver_name, departure_lat, departure_lon,
                 departure_zone, reason, status,
                 collected_bins_count, vehicle_fill_pct)
            VALUES
                (:vid, :driver, :lat, :lon,
                 :zone, :reason, 'en_route', :bins, :fill)
        """),
        {
            "vid":    vehicle_id,
            "driver": vehicle["driver_name"],
            "lat":    dep_lat,
            "lon":    dep_lon,
            "zone":   vehicle["zone_code"],
            "reason": reason,
            "bins":   collected_bins,
            "fill":   fill_pct,
        },
```

```
    )

    # Alert
    msg = (
        f"Το όχημα {vehicle_id} ({vehicle['driver_name']}) κατευθύνεται στο
XYTA. "
        f"Πληρότητα: {fill_pct:.1f}% ({collected_bins} κάδοι). "
        f"Απόσταση: {round(distance, 1)} km."
    )
    await _insert_vehicle_alert(
        db,
        alert_type="xyta_trip",
        vehicle_id=vehicle_id,
        message=msg,
        zone_code=vehicle["zone_code"],
        driver_name=vehicle["driver_name"],
        fill_pct=fill_pct,
        bins_count=collected_bins,
        initiated_by="driver",
    )

    await db.commit()

    return {
        "status":      "success",
        "vehicle_id":  vehicle_id,
        "driver_name": vehicle["driver_name"],
        "from_zone":   vehicle["zone_name"],
        "xyta": {
            "name":              XYTA["name"],
            "latitude":          XYTA["lat"],
            "longitude":         XYTA["lon"],
            "distance_km":       round(distance, 1),
            "estimated_minutes": int(distance * 2.5),
        },
    }


@router.get("/dispatch/alerts")
async def get_vehicle_alerts(
    vehicle_id:  Optional[str] = None,
    zone_code:   Optional[str] = None,
    initiated_by: Optional[str] = None,
    db:          AsyncSession  = Depends(get_db),
) -> dict:
    """
    Ειδοποιήσεις οχημάτων με φίλτρα.
```

```
    vehicle_id:   Φίλτρο ανά όχημα (ή nearest_vehicle_id)
    zone_code:    Φίλτρο ανά ζώνη (ή suggested_zone_code)
    initiated_by: 'driver' ή 'system'
    """
    where_clauses = ["1=1"]
    params: dict  = {}

    if vehicle_id:
        where_clauses.append(
            "(vehicle_id = :vehicle_id OR nearest_vehicle_id =
:vehicle_id)"
        )
        params["vehicle_id"] = vehicle_id

    if zone_code:
        where_clauses.append(
            "(zone_code = :zone_code OR suggested_zone_code = :zone_code)"
        )
        params["zone_code"] = zone_code

    if initiated_by:
        where_clauses.append("initiated_by = :initiated_by")
        params["initiated_by"] = initiated_by

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                id, alert_type, vehicle_id, message,
                nearest_vehicle_id, suggested_zone_code,
                zone_code, driver_name, fill_level_pct,
                collected_bins_count, notes,
                initiated_by, status,
                created_at, acknowledged_at
            FROM vehicle_alerts
            WHERE {where}
            ORDER BY created_at DESC
            LIMIT 100
        """),
        params,
    )

    alerts = []
    for row in result.mappings().all():
        alerts.append({
```

```
            "id":                   row["id"],
            "alert_type":           row["alert_type"],
            "action_label":         ACTION_LABELS.get(row["alert_type"],
row["alert_type"]),
            "vehicle_id":           row["vehicle_id"],
            "message":              row["message"],
            "nearest_vehicle_id":   row["nearest_vehicle_id"],
            "suggested_zone_code":  row["suggested_zone_code"],
            "zone_code":            row["zone_code"],
            "driver_name":          row["driver_name"],
            "fill_level_pct":       float(row["fill_level_pct"]   or 0),
            "collected_bins_count": int(row["collected_bins_count"] or 0),
            "notes":                row["notes"],
            "initiated_by":         row["initiated_by"],
            "status":               row["status"],
            "created_at":           str(row["created_at"]),
            "acknowledged_at":      str(row["acknowledged_at"]) if
row["acknowledged_at"] else None,
        })

    return {"status": "success", "total": len(alerts), "alerts": alerts}


@router.put("/dispatch/alerts/{alert_id}/acknowledge")
async def acknowledge_vehicle_alert(
    alert_id: int,
    db:       AsyncSession = Depends(get_db),
) -> dict:
    """Επιβεβαίωση ειδοποίησης οχήματος."""
    await db.execute(
        text("""
            UPDATE vehicle_alerts
            SET status = 'acknowledged', acknowledged_at = NOW()
            WHERE id = :id
        """),
        {"id": alert_id},
    )
    await db.commit()
    return {"status": "success", "message": "Ειδοποίηση επιβεβαιώθηκε."}


@router.get("/dispatch/xyta-trips")
async def get_xyta_trips(db: AsyncSession = Depends(get_db)) -> dict:
    """Ιστορικό μεταφορών στο ΧΥΤΑ."""
    result = await db.execute(text("""
        SELECT id, vehicle_id, driver_name, departure_zone,
               reason, status, collected_bins_count,
```

```
                vehicle_fill_pct, departed_at, arrived_at
            FROM xyta_trips
            ORDER BY departed_at DESC
            LIMIT 100
        """))
        trips = [dict(r) for r in result.mappings().all()]
        return {"status": "success", "total": len(trips), "trips": trips}


@router.get("/dispatch/zone-vehicles/{zone_code}")
async def get_zone_vehicles(
    zone_code: str,
    db:        AsyncSession = Depends(get_db),
) -> dict:
    """Οχήματα ζώνης — χρησιμοποιείται από το PWA οδηγού."""
    result = await db.execute(
        text("""
            SELECT v.vehicle_id, v.plate_number, v.driver_name,
                   v.status, v.fuel_level,
                   v.capacity_tons, v.capacity_m3, v.vehicle_type,
                   v.latitude, v.longitude
            FROM vehicles v
            JOIN zones z ON v.zone_id = z.id
            WHERE z.zone_code = :zone_code
            ORDER BY v.vehicle_id
        """),
        {"zone_code": zone_code},
    )

    vehicles = []
    for row in result.mappings().all():
        vehicles.append({
            "vehicle_id":    row["vehicle_id"],
            "plate_number":  row["plate_number"],
            "driver_name":   row["driver_name"],
            "status":        row["status"],
            "fuel_level":    float(row["fuel_level"]    or 0),
            "capacity_tons": float(row["capacity_tons"] or 0),
            "capacity_m3":   float(row["capacity_m3"]   or 0),
            "vehicle_type":  row["vehicle_type"],
        })

    return {
        "status":    "success",
        "zone_code": zone_code,
        "vehicles":  vehicles,
    }
```

backend\app\api\routes\vehicles.py

```
"""
app/api/routes/vehicles.py
==========================
Διαχείριση απορριμματοφόρων οχημάτων και GPS tracking.

Endpoints:
    GET  /vehicles               → Λίστα οχημάτων με κατάσταση
    PUT  /vehicles/{vehicle_id}  → Ενημέρωση οχήματος (status, GPS)
    GET  /tracking/vehicles      → Real-time GPS θέσεις
"""

import logging
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from pydantic import BaseModel
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()

# Έγκυρες καταστάσεις οχήματος
VALID_STATUSES = {"active", "inactive", "maintenance"}


# ---------------------------------------------------------------------------
--
# Schemas
# ---------------------------------------------------------------------------
--

class VehicleUpdate(BaseModel):
    """Ενημέρωση στοιχείων οχήματος — όλα προαιρετικά."""
    status:     Optional[str]   = None  # active / inactive / maintenance
    latitude:   Optional[float] = None
    longitude:  Optional[float] = None
    fuel_level: Optional[float] = None
    driver_name: Optional[str]  = None


# ---------------------------------------------------------------------------
--
# Helpers
```

```
# ------------------------------------------------------------------------
--

def _serialize_vehicle(row: dict) -> dict:
    """Μετατροπή DB row σε JSON-serializable dict."""
    return {
        "vehicle_id":    row["vehicle_id"],
        "plate_number":  row["plate_number"],
        "vehicle_type":  row["vehicle_type"],
        "capacity_tons": float(row["capacity_tons"] or 0),
        "capacity_m3":   float(row["capacity_m3"]   or 0),
        "status":        row["status"],
        "driver_name":   row["driver_name"],
        "zone_id":       row["zone_id"],
        "zone_code":     row.get("zone_code"),
        "zone_name":     row.get("zone_name"),
        "latitude":      float(row["latitude"]  or 0),
        "longitude":     float(row["longitude"] or 0),
        "fuel_level":    float(row["fuel_level"] or 0),
    }


# ------------------------------------------------------------------------
--
# Endpoints
# ------------------------------------------------------------------------
--

@router.get("/vehicles")
async def get_vehicles(
    zone_code: Optional[str] = None,
    status:    Optional[str] = None,
    db:        AsyncSession  = Depends(get_db),
) -> dict:
    """
    Λίστα οχημάτων με join στις ζώνες.

    Query params:
        zone_code: Φίλτρο ζώνης
        status:    Φίλτρο κατάστασης (active/inactive/maintenance)
    """
    where_clauses = ["1=1"]
    params: dict  = {}

    if zone_code and zone_code != "ALL":
        where_clauses.append("z.zone_code = :zone_code")
        params["zone_code"] = zone_code
```

```
    if status:
        where_clauses.append("v.status = :status")
        params["status"] = status

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                v.vehicle_id, v.plate_number, v.vehicle_type,
                v.capacity_tons, v.capacity_m3, v.status,
                v.driver_name, v.zone_id,
                v.latitude, v.longitude, v.fuel_level,
                z.zone_code, z.zone_name
            FROM vehicles v
            LEFT JOIN zones z ON v.zone_id = z.id
            WHERE {where}
            ORDER BY v.vehicle_id
        """),
        params,
    )

    vehicles = [_serialize_vehicle(dict(r)) for r in
result.mappings().all()]

    # Στατιστικά ανά status
    stats = {
        "total":       len(vehicles),
        "active":      sum(1 for v in vehicles if v["status"] == "active"),
        "inactive":    sum(1 for v in vehicles if v["status"] ==
"inactive"),
        "maintenance": sum(1 for v in vehicles if v["status"] ==
"maintenance"),
    }

    return {"status": "success", "stats": stats, "vehicles": vehicles}


@router.put("/vehicles/{vehicle_id}")
async def update_vehicle(
    vehicle_id: str,
    update:     VehicleUpdate,
    db:         AsyncSession = Depends(get_db),
) -> dict:
    """
    Ενημέρωση οχήματος.
```

```
    Χρησιμοποιείται για:
        - Αλλαγή status (active/inactive/maintenance/επαναφορά)
        - GPS location update (PWA οδηγού)
        - Fuel level update
    """
    # Έλεγχος ύπαρξης
    result = await db.execute(
        text("SELECT vehicle_id, status FROM vehicles WHERE vehicle_id =
:v"),
        {"v": vehicle_id},
    )
    existing = result.mappings().first()
    if not existing:
        raise HTTPException(404, f"Όχημα {vehicle_id} δεν βρέθηκε.")

    # Έλεγχος έγκυρου status
    if update.status and update.status not in VALID_STATUSES:
        raise HTTPException(400, f"Μη έγκυρη κατάσταση: {update.status}")

    # Δυναμικό SET clause — μόνο τα πεδία που δόθηκαν
    set_parts  = []
    params: dict = {"vehicle_id": vehicle_id}

    field_map = {
        "status":      update.status,
        "latitude":    update.latitude,
        "longitude":   update.longitude,
        "fuel_level":  update.fuel_level,
        "driver_name": update.driver_name,
    }
    for field, value in field_map.items():
        if value is not None:
            set_parts.append(f"{field} = :{field}")
            params[field] = value

    if not set_parts:
        raise HTTPException(400, "Δεν δόθηκαν στοιχεία για ενημέρωση.")

    await db.execute(
        text(f"UPDATE vehicles SET {', '.join(set_parts)} WHERE vehicle_id
= :vehicle_id"),
        params,
    )
    await db.commit()

    logger.info(f"Vehicle {vehicle_id} updated: {field_map}")
```

```
    return {
        "status":     "success",
        "vehicle_id": vehicle_id,
        "updated":    {k: v for k, v in field_map.items() if v is not
None},
    }


@router.get("/tracking/vehicles")
async def get_vehicle_tracking(
    zone_code: Optional[str] = None,
    db:        AsyncSession  = Depends(get_db),
) -> dict:
    """
    Real-time GPS tracking οχημάτων.
    Επιστρέφει μόνο οχήματα με GPS συντεταγμένες.
    """
    where_clauses = ["v.latitude IS NOT NULL", "v.longitude IS NOT NULL"]
    params: dict  = {}

    if zone_code and zone_code != "ALL":
        where_clauses.append("z.zone_code = :zone_code")
        params["zone_code"] = zone_code

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                v.vehicle_id, v.plate_number, v.driver_name,
                v.status, v.latitude, v.longitude, v.fuel_level,
                v.capacity_m3,
                z.zone_code, z.zone_name
            FROM vehicles v
            LEFT JOIN zones z ON v.zone_id = z.id
            WHERE {where}
            ORDER BY v.vehicle_id
        """),
        params,
    )

    vehicles = []
    for row in result.mappings().all():
        vehicles.append({
            "vehicle_id":  row["vehicle_id"],
            "plate_number": row["plate_number"],
```

```
            "driver_name": row["driver_name"],
            "status":      row["status"],
            "latitude":    float(row["latitude"]),
            "longitude":   float(row["longitude"]),
            "fuel_level":  float(row["fuel_level"] or 0),
            "capacity_m3": float(row["capacity_m3"] or 0),
            "zone_code":   row["zone_code"],
            "zone_name":   row["zone_name"],
        })

    return {
        "status":   "success",
        "total":    len(vehicles),
        "vehicles": vehicles,
    }
```

backend\app\api\routes\zones.py

```
"""
app/api/routes/zones.py
=======================
Διαχείριση γεωγραφικών ζωνών Δήμου Μαλεβιζίου.

Ζώνες: GAZ (Γάζι), AMM (Αμμουδάρα), APE (Αγία Πελαγία),
       TYL (Τύλισος), KRO (Κρουσώνας), ROD (Ροδιά)

Endpoints:
    GET /zones              → Λίστα ζωνών με στατιστικά
    GET /zones/{zone_code}  → Λεπτομέρειες ζώνης
    GET /zones/{zone_code}/bins     → Κάδοι ζώνης
    GET /zones/{zone_code}/vehicles → Οχήματα ζώνης
"""

import logging
from typing import Optional

from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.database import get_db

logger = logging.getLogger(__name__)
router = APIRouter()

# Έγκυροι κωδικοί ζωνών
VALID_ZONE_CODES = {"GAZ", "AMM", "APE", "TYL", "KRO", "ROD"}
```

```
# ---------------------------------------------------------------------------
--
# Helpers
# ---------------------------------------------------------------------------
--

async def _get_zone_or_404(
    db:        AsyncSession,
    zone_code: str,
) -> dict:
    """Επιστρέφει ζώνη ή 404 αν δεν βρεθεί."""
    result = await db.execute(
        text("""
            SELECT id, zone_code, zone_name, center_lat, center_lon
            FROM zones
            WHERE zone_code = :zone_code
        """),
        {"zone_code": zone_code},
    )
    zone = result.mappings().first()
    if not zone:
        raise HTTPException(404, f"Ζώνη {zone_code} δεν βρέθηκε.")
    return dict(zone)


# ---------------------------------------------------------------------------
--
# Endpoints
# ---------------------------------------------------------------------------
--

@router.get("/zones")
async def get_zones(db: AsyncSession = Depends(get_db)) -> dict:
    """
    Λίστα ζωνών με στατιστικά κάδων και οχημάτων.
    Χρησιμοποιείται στο tab Ζώνες του dashboard.
    """
    result = await db.execute(text("""
        SELECT
            z.id,
            z.zone_code,
            z.zone_name,
            z.center_lat,
            z.center_lon,
            -- Στατιστικά κάδων
            COUNT(DISTINCT
b.id)                                        AS total_bins,
```

```
            COUNT(DISTINCT CASE
                WHEN s.fill_status = 'critical' THEN b.id END)           AS
critical_bins,
            COUNT(DISTINCT CASE
                WHEN s.fill_status = 'warning'  THEN b.id END)           AS
warning_bins,
            ROUND(AVG(s.current_fill_level)::numeric, 1)                 AS
avg_fill_level,
            -- Στατιστικά οχημάτων
            COUNT(DISTINCT
v.id)                                          AS total_vehicles,
            COUNT(DISTINCT CASE
                WHEN v.status = 'active'   THEN v.id END)                AS
active_vehicles
        FROM zones z
        LEFT JOIN bins b            ON b.zone_id = z.id
        LEFT JOIN bin_status_summary s ON s.bin_id = b.bin_id
        LEFT JOIN vehicles v        ON v.zone_id = z.id
        WHERE z.zone_code != 'XYTA'
        GROUP BY z.id, z.zone_code, z.zone_name, z.center_lat, z.center_lon
        ORDER BY z.zone_code
    """))

    zones = []
    for row in result.mappings().all():
        zones.append({
            "id":              row["id"],
            "zone_code":       row["zone_code"],
            "zone_name":       row["zone_name"],
            "center_lat":      float(row["center_lat"])    if
row["center_lat"] else None,
            "center_lon":      float(row["center_lon"])    if
row["center_lon"] else None,
            "total_bins":      int(row["total_bins"]       or 0),
            "critical_bins":   int(row["critical_bins"]    or 0),
            "warning_bins":    int(row["warning_bins"]     or 0),
            "avg_fill_level":  float(row["avg_fill_level"] or 0),
            "total_vehicles":  int(row["total_vehicles"]   or 0),
            "active_vehicles": int(row["active_vehicles"]  or 0),
        })

    return {
        "status": "success",
        "total":  len(zones),
        "zones":  zones,
    }
```

```
@router.get("/zones/{zone_code}")
async def get_zone_detail(
    zone_code: str,
    db:        AsyncSession = Depends(get_db),
) -> dict:
    """
    Λεπτομέρειες μιας ζώνης με πλήρη στατιστικά.
    """
    zone = await _get_zone_or_404(db, zone_code.upper())

    # Στατιστικά κάδων ζώνης
    stats_result = await db.execute(
        text("""
            SELECT
                COUNT(*)                                                AS
total_bins,
                COUNT(CASE WHEN fill_status = 'critical' THEN 1 END)  AS
critical_bins,
                COUNT(CASE WHEN fill_status = 'warning'  THEN 1 END)  AS
warning_bins,
                COUNT(CASE WHEN fill_status = 'normal'   THEN 1 END)  AS
normal_bins,
                ROUND(AVG(current_fill_level)::numeric, 1)            AS
avg_fill,
                MAX(current_fill_level)                                AS
max_fill,
                MIN(current_fill_level)                                AS
min_fill
            FROM bin_status_summary
            WHERE zone_code = :zone_code
        """),
        {"zone_code": zone_code},
    )
    stats = stats_result.mappings().first()

    # Στατιστικά οχημάτων ζώνης
    veh_result = await db.execute(
        text("""
            SELECT
                COUNT(*)                                              AS
total,
                COUNT(CASE WHEN status = 'active'      THEN 1 END)  AS
active,
                COUNT(CASE WHEN status = 'maintenance' THEN 1 END)  AS
maintenance,
                COUNT(CASE WHEN status = 'inactive'    THEN 1 END)  AS
inactive
```

```
            FROM vehicles v
            JOIN zones z ON v.zone_id = z.id
            WHERE z.zone_code = :zone_code
        """),
        {"zone_code": zone_code},
    )
    veh_stats = veh_result.mappings().first()

    return {
        "status": "success",
        "zone": {
            **zone,
            "bin_stats": {
                "total":    int(stats["total_bins"]    or 0),
                "critical": int(stats["critical_bins"] or 0),
                "warning":  int(stats["warning_bins"]  or 0),
                "normal":   int(stats["normal_bins"]   or 0),
                "avg_fill": float(stats["avg_fill"]    or 0),
                "max_fill": float(stats["max_fill"]    or 0),
                "min_fill": float(stats["min_fill"]    or 0),
            },
            "vehicle_stats": {
                "total":       int(veh_stats["total"]       or 0),
                "active":      int(veh_stats["active"]      or 0),
                "maintenance": int(veh_stats["maintenance"] or 0),
                "inactive":    int(veh_stats["inactive"]    or 0),
            },
        },
    }


@router.get("/zones/{zone_code}/bins")
async def get_zone_bins(
    zone_code:   str,
    fill_status: Optional[str] = None,
    db:          AsyncSession  = Depends(get_db),
) -> dict:
    """
    Κάδοι συγκεκριμένης ζώνης.

    Query params:
        fill_status: Φίλτρο κατάστασης (critical/warning/normal)
    """
    # Επαλήθευση ύπαρξης ζώνης
    await _get_zone_or_404(db, zone_code.upper())

    where_clauses = ["zone_code = :zone_code"]
```

```
    params: dict  = {"zone_code": zone_code.upper()}

    if fill_status and fill_status != "ALL":
        where_clauses.append("fill_status = :fill_status")
        params["fill_status"] = fill_status

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                bin_id, location_name, address,
                bin_type, latitude, longitude,
                current_fill_level, current_battery_level,
                fill_status, last_reading_time,
                active_alerts_count
            FROM bin_status_summary
            WHERE {where}
            ORDER BY current_fill_level DESC
        """),
        params,
    )

    bins = []
    for row in result.mappings().all():
        bins.append({
            "bin_id":                row["bin_id"],
            "location_name":         row["location_name"],
            "address":               row["address"],
            "bin_type":              row["bin_type"],
            "latitude":              float(row["latitude"]  or 0),
            "longitude":             float(row["longitude"] or 0),
            "current_fill_level":    float(row["current_fill_level"]    or
0),
            "current_battery_level": float(row["current_battery_level"] or
0),
            "fill_status":           row["fill_status"],
            "last_reading_time":     str(row["last_reading_time"]) if
row["last_reading_time"] else None,
            "active_alerts_count":   int(row["active_alerts_count"] or 0),
        })

    return {
        "status":    "success",
        "zone_code": zone_code.upper(),
        "total":     len(bins),
        "bins":      bins,
```

```
    }


@router.get("/zones/{zone_code}/vehicles")
async def get_zone_vehicles(
    zone_code: str,
    status:    Optional[str] = None,
    db:        AsyncSession  = Depends(get_db),
) -> dict:
    """
    Οχήματα συγκεκριμένης ζώνης.

    Query params:
        status: Φίλτρο κατάστασης (active/inactive/maintenance)
    """
    await _get_zone_or_404(db, zone_code.upper())

    where_clauses = ["z.zone_code = :zone_code"]
    params: dict  = {"zone_code": zone_code.upper()}

    if status and status != "ALL":
        where_clauses.append("v.status = :status")
        params["status"] = status

    where = " AND ".join(where_clauses)

    result = await db.execute(
        text(f"""
            SELECT
                v.vehicle_id, v.plate_number, v.driver_name,
                v.vehicle_type, v.status,
                v.capacity_tons, v.capacity_m3,
                v.fuel_level, v.latitude, v.longitude
            FROM vehicles v
            JOIN zones z ON v.zone_id = z.id
            WHERE {where}
            ORDER BY
                CASE v.status
                    WHEN 'active'      THEN 1
                    WHEN 'maintenance' THEN 2
                    ELSE 3
                END,
                v.vehicle_id
        """),
        params,
    )
```

```
    vehicles = []
    for row in result.mappings().all():
        vehicles.append({
            "vehicle_id":    row["vehicle_id"],
            "plate_number":  row["plate_number"],
            "driver_name":   row["driver_name"],
            "vehicle_type":  row["vehicle_type"],
            "status":        row["status"],
            "capacity_tons": float(row["capacity_tons"] or 0),
            "capacity_m3":   float(row["capacity_m3"]   or 0),
            "fuel_level":    float(row["fuel_level"]    or 0),
            "latitude":      float(row["latitude"]  or 0) if
row["latitude"]  else None,
            "longitude":     float(row["longitude"] or 0) if
row["longitude"] else None,
        })

    return {
        "status":    "success",
        "zone_code": zone_code.upper(),
        "total":     len(vehicles),
        "vehicles":  vehicles,
    }
```

backend\.env

```
#
=============================================================================
# Environment
#
=============================================================================
ENVIRONMENT=development
DEBUG=True

#
=============================================================================
# Database - PostgreSQL (Docker)
#
=============================================================================
#DATABASE_URL=postgresql+asyncpg://admin:managewastebins2025!!@localhost:54
32/wastebin_management
DATABASE_URL=postgresql+asyncpg://admin:wastebin2025@localhost:5433/wastebi
n_management
DATABASE_ECHO=False

POSTGRES_HOST=localhost
POSTGRES_PORT=5433
POSTGRES_DB=wastebin_management
```

```
POSTGRES_USER=admin
POSTGRES_PASSWORD=wastebin2025

#
================================================================================
# Redis (Docker) - Σωστό port: 6379 όχι 6380
#
================================================================================
REDIS_URL=redis://:redis123@localhost:6379/0

#
================================================================================
# Security
#
================================================================================
SECRET_KEY=your-super-secret-key-change-in-production-environment
ACCESS_TOKEN_EXPIRE_MINUTES=30
ALGORITHM=HS256

#
================================================================================
# External APIs
#
================================================================================
GOOGLE_MAPS_API_KEY=your-google-maps-api-key
NGROK_TUNNEL_URL=https://your-ngrok-url.ngrok.io

#
================================================================================
# Application Settings
#
================================================================================
ML_MODEL_PATH=./models
ML_MIN_DATA_POINTS=50
ALLOWED_HOSTS=localhost,127.0.0.1,*.ngrok.io
USE_POSTGRES=true
```

backend\Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

# Install system dependencies
RUN apt-get update && apt-get install -y \
    gcc \
    g++ \
    && rm -rf /var/lib/apt/lists/*
```

```
# Copy requirements first for better caching
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy application code
COPY . .

# Create necessary directories
RUN mkdir -p uploads models logs

# Expose port
EXPOSE 8000

# Run the application
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000", "--
reload"]
```

Backend\app\core__init__.py

```
"""
app/core/__init__.py
====================
Package init για το core module.

Σκόπιμα κενό — δεν κάνουμε imports models εδώ, ώστε:
    1. Να μην απαιτείται SQLAlchemy ORM models (χρησιμοποιούμε raw SQL)
    2. Να αποφεύγονται circular imports
    3. Να γίνεται explicit import ό,τι χρειάζεται σε κάθε αρχείο

Χρήση:
    from app.core.config import settings
    from app.core.database import get_db
    from app.core.security import create_access_token
"""
```

Backend\app\core\config.py

```
"""
app/core/config.py
==================
Κεντρικές ρυθμίσεις εφαρμογής μέσω pydantic-settings.
Όλες οι τιμές διαβάζονται από environment variables ή .env αρχείο.
"""

from pydantic_settings import BaseSettings


class Settings(BaseSettings):
    # --- Περιβάλλον ---
```

```
    ENVIRONMENT: str = "development"
    DEBUG: bool = True

    # --- Βάση Δεδομένων ---
    DATABASE_URL: str = (
        "postgresql+asyncpg://admin:wastebin2025@localhost:5433/wastebin_ma
nagement"
    )

    # --- Redis ---
    REDIS_URL: str = "redis://:redis123@localhost:6379/0"

    # --- JWT ---
    SECRET_KEY: str = "your-secret-key-change-in-production"
    ALGORITHM: str = "HS256"
    ACCESS_TOKEN_EXPIRE_MINUTES: int = 480  # 8 ώρες

    # --- SMTP ---
    SMTP_HOST: str = "smtp.gmail.com"
    SMTP_PORT: int = 587
    SMTP_USER: str = ""
    SMTP_PASS: str = ""

    # --- Frontend ---
    FRONTEND_URL: str = "http://localhost:3000"
    ALLOWED_ORIGINS: list[str] = [
        "http://localhost:3000",
        "http://localhost:8080",
        "http://localhost:8081",
    ]

    # --- Σύστημα ---
    SYSTEM_NAME: str = "Smart Waste Management — Δήμος Μαλεβιζίου"

    class Config:
        env_file = ".env"
        env_file_encoding = "utf-8"
        extra = "ignore"  # Αγνόηση παλιών μεταβλητών στο .env


# Singleton instance — import αυτό παντού
settings = Settings()
```

Backend\app\core\database.py

```
"""
app/core/database.py
====================
```

```
Async SQLAlchemy engine, session factory και dependency injection.
"""

from collections.abc import AsyncGenerator

from sqlalchemy.ext.asyncio import (
    AsyncSession,
    async_sessionmaker,
    create_async_engine,
)
from sqlalchemy import text

from app.core.config import settings

# ---------------------------------------------------------------------------
--
# Engine & Session factory
# ---------------------------------------------------------------------------
--

engine = create_async_engine(
    settings.DATABASE_URL,
    echo=settings.DEBUG,       # SQL logging σε development
    pool_pre_ping=True,        # Έλεγχος σύνδεσης πριν χρήση
    pool_size=10,
    max_overflow=20,
)

AsyncSessionLocal = async_sessionmaker(
    bind=engine,
    expire_on_commit=False,
    autoflush=False,
    autocommit=False,
)


# ---------------------------------------------------------------------------
--
# Dependency — χρησιμοποιείται ως Depends(get_db) στα routes
# ---------------------------------------------------------------------------
--

async def get_db() -> AsyncGenerator[AsyncSession, None]:
    """FastAPI dependency που παρέχει async DB session."""
    async with AsyncSessionLocal() as session:
        try:
            yield session
```

```
        except Exception:
            await session.rollback()
            raise
        finally:
            await session.close()


# ------------------------------------------------------------------------
--
# Startup helpers
# ------------------------------------------------------------------------
--

async def create_tables() -> None:
    """Placeholder — migrations γίνονται μέσω SQL scripts."""
    pass


async def check_database_health() -> bool:
    """Έλεγχος σύνδεσης με τη βάση κατά την εκκίνηση."""
    try:
        async with AsyncSessionLocal() as session:
            await session.execute(text("SELECT 1"))
        print(" Database connection OK")
        return True
    except Exception as exc:
        print(f" Database connection FAILED: {exc}")
        return False
```

Backend\app\core\security.py

```
"""
app/core/security.py
====================
JWT token generation/validation και password hashing με bcrypt.
"""

from datetime import datetime, timedelta

from jose import JWTError, jwt
from passlib.context import CryptContext

from app.core.config import settings

# ------------------------------------------------------------------------
--
# Password hashing
```

```
# ------------------------------------------------------------------------
--

_pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")


def verify_password(plain: str, hashed: str) -> bool:
    """Επαλήθευση κωδικού έναντι bcrypt hash."""
    return _pwd_context.verify(plain, hashed)


def get_password_hash(password: str) -> str:
    """Δημιουργία bcrypt hash από plain-text κωδικό."""
    return _pwd_context.hash(password)


# ------------------------------------------------------------------------
--
# JWT tokens
# ------------------------------------------------------------------------
--

def create_access_token(
    data: dict,
    expires_delta: timedelta | None = None,
) -> str:
    """
    Δημιουργία JWT access token.

    Args:
        data:          Payload (π.χ. {"sub": username, "role": role})
        expires_delta: Διάρκεια ισχύος. Default:
ACCESS_TOKEN_EXPIRE_MINUTES.

    Returns:
        Encoded JWT string.
    """
    expire = datetime.utcnow() + (
        expires_delta or
timedelta(minutes=settings.ACCESS_TOKEN_EXPIRE_MINUTES)
    )
    payload = {**data, "exp": expire}
    return jwt.encode(payload, settings.SECRET_KEY,
algorithm=settings.ALGORITHM)


def decode_access_token(token: str) -> dict | None:
    """
```

```
Αποκωδικοποίηση JWT token.

Returns:
    Payload dict ή None αν το token είναι invalid/expired.
"""
try:
    return jwt.decode(
        token,
        settings.SECRET_KEY,
        algorithms=[settings.ALGORITHM],
    )
except JWTError:
    return None
```

Παράρτημα Β: «Κώδικας του frontend του ΠΣ Smart Waste Management System»

frontend\src\App.tsx

```
/**
 * App.tsx
 * ========
 * Κεντρικό component εφαρμογής — διαχειρίζεται authentication,
 * navigation tabs, και routing μεταξύ των views.
 */
import React, { useState, useEffect } from 'react';
import LoginPage from './LoginPage';
import FillnessView from './FillnessView';
import MapView from './MapView';
import RoutesView from './RoutesView';
import VehiclesView from './VehiclesView';
import TrackingView from './TrackingView';
import ZonesView from './ZonesView';
import DispatchView from './DispatchView';
import ReportsView from './ReportsView';
import CitizenReportsView from './CitizenReportsView';
import './App.css';

const API_URL = 'http://localhost:8000';

// ---------------------------------------------------------------------------
// Types
// ---------------------------------------------------------------------------

type ActiveTab =
  | 'fillness'
  | 'map'
  | 'routes'
  | 'vehicles'
  | 'tracking'
  | 'zones'
  | 'dispatch'
  | 'reports'
  | 'citizen'
  | 'users';

interface User {
  username: string;
  role: string;
  fullName: string;
```

```
}

interface NavTab {
  id: ActiveTab;
  icon: string;
  label: string;
  adminOnly?: boolean;
}

interface Bin {
  bin_id: string;
  location_name: string;
  address: string;
  latitude: number;
  longitude: number;
  bin_type: string;
  status: string;
  current_fill_level: number;
  current_battery_level: number;
}

// ---------------------------------------------------------------------
// Constants
// ---------------------------------------------------------------------

const NAV_TABS: NavTab[] = [
  { id: 'fillness', icon: 'fi-sr-chart-histogram', label: 'Πληρότητα' },
  { id: 'map', icon: 'fi-sr-marker', label: 'Θέσεις Κάδων' },
  { id: 'routes', icon: 'fi-sr-book-alt', label: 'Δημιουργία Διαδρομών' },
  { id: 'vehicles', icon: 'fi-sr-truck-side', label: 'Απορριμματοφόρα' },
  { id: 'tracking', icon: 'fi-sr-broadcast-tower', label: 'Tracking' },
  { id: 'zones', icon: 'fi-sr-map', label: 'Ζώνες' },
  { id: 'dispatch', icon: 'fi-sr-settings-sliders', label: 'Διαχείριση' },
  { id: 'reports', icon: 'fi-sr-bug', label: 'Αναφορές' },
  { id: 'citizen', icon: 'fi-sr-comment', label: 'Αναφορές Πολιτών' },
  { id: 'users', icon: 'fi-sr-users', label: 'Χρήστες', adminOnly: true },
];

// ---------------------------------------------------------------------
// Helpers
// ---------------------------------------------------------------------

function clearStoredAuth(): void {
  localStorage.removeItem('token');
  localStorage.removeItem('user');
  sessionStorage.removeItem('token');
}
```

```
// ---------------------------------------------------------------------
// Component
// ---------------------------------------------------------------------

function App() {
  // isLoggedIn ξεκινά false· το LoginPage κάνει auto-login από
localStorage
  // και καλεί onLogin() αν βρει αποθηκευμένο token (βλ. useEffect στο
LoginPage.tsx)
  const [isLoggedIn, setIsLoggedIn] = useState<boolean>(false);
  const [currentUser, setCurrentUser] = useState<User>({ username: '',
role: '', fullName: '' });
  const [activeTab, setActiveTab] = useState<ActiveTab>('fillness');
  const [bins, setBins] = useState<Bin[]>([]);

  // Φόρτωση κάδων μία φορά μετά login (χρησιμοποιείται στο MapView)
  useEffect(() => {
    if (isLoggedIn) {
      fetchBins();
    }
  }, [isLoggedIn]);

  const fetchBins = async (): Promise<void> => {
    try {
      const res = await fetch(`${API_URL}/api/v1/bins`);
      const json = await res.json();
      setBins(json.bins ?? []);
    } catch (error) {
      console.error('Failed to fetch bins:', error);
    }
  };

  const handleLogin = (token: string, username: string, role: string,
fullName: string): void => {
    setCurrentUser({ username, role, fullName });
    setIsLoggedIn(true);
  };

  const handleLogout = (): void => {
    clearStoredAuth();
    setIsLoggedIn(false);
    setCurrentUser({ username: '', role: '', fullName: '' });
  };

  if (!isLoggedIn) {
    return <LoginPage onLogin={handleLogin} />;
```

```
  }

  const visibleTabs = NAV_TABS.filter(
    tab => !tab.adminOnly || currentUser.role === 'admin'
  );

  return (
    <div className="app-container">
      <header className="app-header">
        <h1 className="app-title">
          <i className="fi fi-sr-trash icon-mr8" aria-hidden="true"></i>
          Smart Waste Management
        </h1>
        <p className="app-subtitle">
          Δήμος Μαλεβιζίου — Σύστημα Διαχείρισης Απορριμμάτων
        </p>

        <nav className="app-nav" aria-label="Κύρια πλοήγηση">
          {visibleTabs.map(tab => (
            <button
              key={tab.id}
              onClick={() => setActiveTab(tab.id)}
              className={`app-nav-btn ${activeTab === tab.id ? 'active' :
''}`}
              aria-current={activeTab === tab.id ? 'page' : undefined}
              title={tab.label}
            >
              <i className={`fi ${tab.icon} icon-mr6`} aria-
hidden="true"></i>
              {tab.label}
            </button>
          ))}
        </nav>

        <div className="app-user-bar">
          <span className="app-user-info">
            <i className="fi fi-sr-user icon-mr4" aria-hidden="true"></i>
            {currentUser.fullName || currentUser.username}
            {' — '}
            <span className="app-user-role">
              {currentUser.role === 'admin' ? 'Διαχειριστής' : 'Χειριστής
Συστήματος'}
            </span>
          </span>
          <button
            onClick={handleLogout}
            className="app-logout-btn"
```

```
            aria-label="Αποσύνδεση"
            title="Έξοδος"
          >
            <i className="fi fi-sr-sign-out-alt icon-mr6" aria-
hidden="true"></i>
            Έξοδος
          </button>
        </div>
      </header>

      <main className="app-main">
        {activeTab === 'fillness' && <FillnessView />}
        {activeTab === 'map' && (
          <div className="app-map-wrapper">
            <MapView bins={bins} />
          </div>
        )}
        {activeTab === 'routes' && <RoutesView />}
        {activeTab === 'vehicles' && <VehiclesView />}
        {activeTab === 'tracking' && <TrackingView />}
        {activeTab === 'zones' && <ZonesView />}
        {activeTab === 'dispatch' && <DispatchView />}
        {activeTab === 'reports' && <ReportsView />}
        {activeTab === 'citizen' && <CitizenReportsView />}
        {activeTab === 'users' && currentUser.role === 'admin' && (
          <div className="app-users-placeholder">
            <p>Διαχείριση Χρηστών — Coming soon</p>
          </div>
        )}
      </main>
    </div>
  );
}

export default App;
```

frontend\src\App.css

```
/* ===== App.css ===== */

* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
```

```
  font-family: 'Segoe UI', Arial, sans-serif;
  background-color: #f0f4f8;
}

.icon-mr4 {
  margin-right: 4px;
}

.icon-mr6 {
  margin-right: 6px;
}

.icon-mr8 {
  margin-right: 8px;
}

/* ===== CONTAINER ===== */
.app-container {
  min-height: 100vh;
}

/* ===== HEADER ===== */
.app-header {
  background: linear-gradient(135deg, #1a252f, #2c3e50);
  color: white;
  padding: 16px 24px;
}

.app-title {
  font-size: 1.6rem;
  display: flex;
  align-items: center;
  margin-bottom: 2px;
}

.app-subtitle {
  font-size: 0.85rem;
  opacity: 0.75;
  margin-bottom: 14px;
}

/* ===== NAVIGATION ===== */
.app-nav {
  display: flex;
  gap: 8px;
  flex-wrap: wrap;
  margin-bottom: 12px;
```

```
}

.app-nav-btn {
  padding: 8px 14px;
  background: rgba(255, 255, 255, 0.08);
  color: white;
  border: 1px solid rgba(255, 255, 255, 0.2);
  border-radius: 20px;
  cursor: pointer;
  font-size: 0.85rem;
  font-family: inherit;
  display: flex;
  align-items: center;
  transition: background-color 0.2s, border-color 0.2s;
}

.app-nav-btn:hover {
  background: rgba(255, 255, 255, 0.18);
}

.app-nav-btn.active {
  background: white;
  color: #2c3e50;
  border-color: white;
  font-weight: bold;
}

/* ===== USER BAR ===== */
.app-user-bar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding-top: 10px;
  border-top: 1px solid rgba(255, 255, 255, 0.15);
}

.app-user-info {
  font-size: 0.85rem;
  opacity: 0.9;
  display: flex;
  align-items: center;
}

.app-user-role {
  font-weight: bold;
}
```

```
.app-logout-btn {
  padding: 6px 14px;
  background: rgba(231, 76, 60, 0.2);
  color: white;
  border: 1px solid rgba(231, 76, 60, 0.5);
  border-radius: 8px;
  cursor: pointer;
  font-size: 0.85rem;
  font-family: inherit;
  display: flex;
  align-items: center;
  transition: background-color 0.2s;
}

.app-logout-btn:hover {
  background: rgba(231, 76, 60, 0.4);
}

/* ===== MAIN CONTENT ===== */
.app-main {
  padding: 0;
}

.app-map-wrapper {
  width: 100%;
}

.app-users-placeholder {
  padding: 60px;
  text-align: center;
  color: #7f8c8d;
}

/* ===== RESPONSIVE ===== */
@media (max-width: 768px) {
  .app-nav {
    overflow-x: auto;
    flex-wrap: nowrap;
  }

  .app-user-bar {
    flex-direction: column;
    gap: 8px;
    align-items: flex-start;
  }
}
```

frontend\src\LoginPage.tsx

```
/**
 * LoginPage.tsx
 * ==============
 * ΠΧ1 – Πιστοποίηση Χρήστη.
 *
 * Καταστάσεις (PageMode):
 *   login          → Βασική φόρμα σύνδεσης (username + password + captcha)
 *   forgot         → Αίτημα ανάκτησης κωδικού (username)
 *   reset-sent     → Επιβεβαίωση αποστολής email
 *   reset-password → Ορισμός νέου κωδικού (από link email)
 *   reset-success  → Επιβεβαίωση επιτυχούς αλλαγής
 */
import React, { useState, useEffect } from 'react';
import './LoginPage.css';

const API_URL = 'http://localhost:8000';

interface LoginPageProps {
  onLogin: (token: string, username: string, role: string, fullName:
string) => void;
}

type PageMode = 'login' | 'forgot' | 'reset-sent' | 'reset-password' |
'reset-success';

interface PasswordStrength {
  label: string;
}

// ------------------------------------------------------------------------
---
// Helpers
// ------------------------------------------------------------------------
---

/** Αξιολόγηση ισχύος κωδικού – επιστρέφει label + CSS class suffix. */
function getPasswordStrength(pwd: string): PasswordStrength | null {
  if (pwd.length === 0) return null;
  if (pwd.length < 6) return { label: 'Πολύ αδύναμος' };
  if (pwd.length < 8) return { label: 'Αδύναμος' };
  if (!/[A-Z]/.test(pwd) || !/[0-9]/.test(pwd)) {
    return { label: 'Μέτριος' };
  }
  if (pwd.length >= 10) return { label: 'Ισχυρός' };
  return { label: 'Καλός' };
}
```

```
/** Μετατροπή label σε CSS-safe slug (π.χ. "Πολύ αδύναμος" → "poly-
adynamos" δεν χρειάζεται — χρησιμοποιούμε λατινικά keys). */
function strengthSlug(label: string): string {
  const map: Record<string, string> = {
    'Πολύ αδύναμος': 'very-weak',
    'Αδύναμος': 'weak',
    'Μέτριος': 'medium',
    'Καλός': 'good',
    'Ισχυρός': 'strong',
  };
  return map[label] ?? 'medium';
}

// ---------------------------------------------------------------------
---
// Component
// ---------------------------------------------------------------------
---

function LoginPage({ onLogin }: LoginPageProps) {
  const [mode, setMode] = useState<PageMode>('login');

  // --- Login ---
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [rememberMe, setRememberMe] = useState(false);
  const [showPassword, setShowPassword] = useState(false);

  // --- Captcha ---
  const [captchaId, setCaptchaId] = useState('');
  const [captchaQuestion, setCaptchaQuestion] = useState('');
  const [captchaAnswer, setCaptchaAnswer] = useState('');

  // --- Forgot password ---
  const [resetUsername, setResetUsername] = useState('');
  const [devResetUrl, setDevResetUrl] = useState('');

  // --- Reset password ---
  const [resetToken, setResetToken] = useState('');
  const [resetTokenUsername, setResetTokenUsername] = useState('');
  const [newPassword, setNewPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');
  const [showNewPassword, setShowNewPassword] = useState(false);
  const [showConfirmPassword, setShowConfirmPassword] = useState(false);

  // --- UI state ---
  const [loading, setLoading] = useState(false);
```

```
  const [error, setError] = useState('');
  const [success, setSuccess] = useState('');

  // ---------------------------------------------------------------------
--
  // Effects
  // ---------------------------------------------------------------------
--

  useEffect(() => {
    fetchCaptcha();
    checkUrlResetToken();
    attemptAutoLogin();
  }, []);

  /** Έλεγχος αν το URL περιέχει ?token=... (link από email reset). */
  function checkUrlResetToken(): void {
    const params = new URLSearchParams(window.location.search);
    const token = params.get('token');
    if (token) validateResetToken(token);
  }

  /** Auto-login από αποθηκευμένο localStorage token (αν υπάρχει). */
  function attemptAutoLogin(): void {
    const savedToken = localStorage.getItem('token');
    const savedUser = localStorage.getItem('user');
    if (!savedToken || !savedUser) return;

    try {
      const user = JSON.parse(savedUser);
      onLogin(savedToken, user.username, user.role, user.fullName);
    } catch {
      localStorage.removeItem('token');
      localStorage.removeItem('user');
    }
  }

  // ---------------------------------------------------------------------
--
  // API calls
  // ---------------------------------------------------------------------
--

  const fetchCaptcha = async (): Promise<void> => {
    try {
      const res = await fetch(`${API_URL}/api/v1/auth/captcha`);
      const data = await res.json();
```

```
      setCaptchaId(data.captcha_id);
      setCaptchaQuestion(data.question);
      setCaptchaAnswer('');
    } catch {
      setError('Αδυναμία σύνδεσης με τον server.');
    }
  };

  const handleLogin = async (e: React.FormEvent): Promise<void> => {
    e.preventDefault();
    setError('');
    setSuccess('');

    if (!username || !password) {
      setError('Συμπληρώστε username και password.');
      return;
    }
    if (!captchaAnswer) {
      setError('Συμπληρώστε το captcha.');
      return;
    }

    setLoading(true);
    try {
      const res = await fetch(`${API_URL}/api/v1/auth/login`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          username,
          password,
          captcha_id: captchaId,
          captcha_answer: parseInt(captchaAnswer, 10),
          remember_me: rememberMe,
        }),
      });
      const data = await res.json();

      if (!res.ok) {
        setError(data.detail || 'Σφάλμα σύνδεσης.');
        fetchCaptcha();
        return;
      }

      // Αποθήκευση session — localStorage (μόνιμη) ή sessionStorage
(προσωρινή)
      const userPayload = JSON.stringify({
        username: data.username,
```

```
        role: data.role,
        fullName: data.full_name,
      });

      if (rememberMe) {
        localStorage.setItem('token', data.access_token);
        localStorage.setItem('user', userPayload);
      } else {
        sessionStorage.setItem('token', data.access_token);
        localStorage.removeItem('token');
        localStorage.removeItem('user');
      }

      onLogin(data.access_token, data.username, data.role, data.full_name);
    } catch {
      setError('Αδυναμία σύνδεσης με τον server.');
      fetchCaptcha();
    } finally {
      setLoading(false);
    }
  };

  const handleForgotPassword = async (e: React.FormEvent): Promise<void> =>
{
    e.preventDefault();
    setError('');
    setSuccess('');

    if (!resetUsername) {
      setError('Εισάγετε το username σας.');
      return;
    }

    setLoading(true);
    try {
      const res = await fetch(`${API_URL}/api/v1/auth/password-reset-
request`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username: resetUsername }),
      });
      const data = await res.json();

      if (res.ok) {
        setMode('reset-sent');
        if (data.dev_reset_url) setDevResetUrl(data.dev_reset_url);
      } else {
```

```
        setError(data.detail || 'Σφάλμα αποστολής.');
      }
    } catch {
      setError('Αδυναμία σύνδεσης με τον server.');
    } finally {
      setLoading(false);
    }
  };

  const validateResetToken = async (token: string): Promise<void> => {
    try {
      const res = await fetch(`${API_URL}/api/v1/auth/password-reset-
validate/${token}`);
      const data = await res.json();

      if (data.valid) {
        setResetToken(token);
        setResetTokenUsername(data.username);
        setMode('reset-password');
        window.history.replaceState({}, '', window.location.pathname);
      } else {
        setError(`Ο σύνδεσμος δεν είναι έγκυρος: ${data.reason}`);
        setMode('login');
      }
    } catch {
      setError('Αδυναμία επαλήθευσης συνδέσμου.');
    }
  };

  const handleResetPassword = async (e: React.FormEvent): Promise<void> =>
{
    e.preventDefault();
    setError('');
    setSuccess('');

    if (!newPassword || !confirmPassword) {
      setError('Συμπληρώστε και τα δύο πεδία.');
      return;
    }
    if (newPassword !== confirmPassword) {
      setError('Οι κωδικοί δεν ταιριάζουν.');
      return;
    }
    if (newPassword.length < 6) {
      setError('Ο κωδικός πρέπει να έχει τουλάχιστον 6 χαρακτήρες.');
      return;
    }
```

```
    setLoading(true);
    try {
      const res = await fetch(`${API_URL}/api/v1/auth/password-reset-
confirm`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          token: resetToken,
          new_password: newPassword,
          confirm_password: confirmPassword,
        }),
      });
      const data = await res.json();

      if (res.ok) {
        setMode('reset-success');
      } else {
        setError(data.detail || 'Σφάλμα αλλαγής κωδικού.');
      }
    } catch {
      setError('Αδυναμία σύνδεσης με τον server.');
    } finally {
      setLoading(false);
    }
  };

  const resetToLoginMode = (): void => {
    setMode('login');
    setError('');
    setSuccess('');
  };

  const pwdStrength = getPasswordStrength(newPassword);
  const passwordsMatch = confirmPassword.length > 0 && newPassword ===
confirmPassword;
  const passwordsMismatch = confirmPassword.length > 0 && newPassword !==
confirmPassword;

  // ------------------------------------------------------------------
--
  // Render
  // ------------------------------------------------------------------
--

  return (
    <div className="login-container">
```

```
      <div className="login-bg-circle1" aria-hidden="true"></div>
      <div className="login-bg-circle2" aria-hidden="true"></div>
      <div className="login-bg-circle3" aria-hidden="true"></div>

      <div className="login-card" role="main">
        <div className="login-logo-area">
          <div className="login-logo-icon" aria-hidden="true">
            <i className="fi fi-sr-trash icon-logo"></i>
          </div>
          <h1 className="login-title">Smart Waste Management</h1>
          <p className="login-subtitle">Δήμος Μαλεβιζίου</p>
        </div>

        {error && (
          <div className="login-alert-error" role="alert" aria-
live="polite">
            <i className="fi fi-sr-cross-circle icon-mr8" aria-
hidden="true"></i>
            {error}
          </div>
        )}
        {success && (
          <div className="login-alert-success" role="alert" aria-
live="polite">
            <i className="fi fi-sr-check-circle icon-mr8" aria-
hidden="true"></i>
            {success}
          </div>
        )}

        {/* ===== LOGIN ===== */}
        {mode === 'login' && (
          <form onSubmit={handleLogin} noValidate>
            <div className="login-section-title">
              <i className="fi fi-sr-lock icon-mr8 icon-blue" aria-
hidden="true"></i>
              Πιστοποίηση Χρήστη
            </div>

            <div className="login-field">
              <label htmlFor="login-username" className="login-label">
                <i className="fi fi-sr-user icon-mr6 icon-gray" aria-
hidden="true"></i>
                Όνομα Χρήστη
              </label>
              <input
                id="login-username"
```

```
                type="text"
                value={username}
                onChange={e => setUsername(e.target.value)}
                placeholder="Εισάγετε username"
                className="login-input"
                autoComplete="username"
                aria-required="true"
                title="Εισάγετε το όνομα χρήστη σας"
              />
            </div>

            <div className="login-field">
              <label htmlFor="login-password" className="login-label">
                <i className="fi fi-sr-key icon-mr6 icon-gray" aria-
hidden="true"></i>
                Κωδικός Πρόσβασης
              </label>
              <div className="login-input-wrapper">
                <input
                  id="login-password"
                  type={showPassword ? 'text' : 'password'}
                  value={password}
                  onChange={e => setPassword(e.target.value)}
                  placeholder="Εισάγετε κωδικό"
                  className="login-input login-input-password"
                  autoComplete="current-password"
                  aria-required="true"
                  title="Εισάγετε τον κωδικό πρόσβασής σας"
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="login-eye-btn"
                  aria-label={showPassword ? 'Απόκρυψη κωδικού' : 'Εμφάνιση
κωδικού'}
                  title={showPassword ? 'Απόκρυψη κωδικού' : 'Εμφάνιση
κωδικού'}
                >
                  <i className={`fi ${showPassword ? 'fi-sr-eye-crossed' :
'fi-sr-eye'}`} aria-hidden="true"></i>
                </button>
              </div>
            </div>

            <div className="login-remember-row">
              <label className="login-checkbox-label" htmlFor="remember-me-
checkbox">
```

```
                <input
                  type="checkbox"
                  id="remember-me-checkbox"
                  checked={rememberMe}
                  onChange={e => setRememberMe(e.target.checked)}
                  className="login-checkbox-native"
                  title="Θυμήσου με"
                />
                <div
                  className={`login-checkbox ${rememberMe ? 'checked' :
''}`}
                  aria-hidden="true"
                  onClick={() => setRememberMe(!rememberMe)}
                >
                  {rememberMe && <i className="fi fi-sr-check icon-check-
sm" aria-hidden="true"></i>}
                </div>
                <span className="login-checkbox-text" onClick={() =>
setRememberMe(!rememberMe)}>
                  Θυμήσου με
                </span>
              </label>
              <button
                type="button"
                onClick={() => { setMode('forgot'); setError('');
setSuccess(''); }}
                className="login-link-btn"
                title="Ανάκτηση κωδικού πρόσβασης"
              >
                Ξεχάσατε τον κωδικό;
              </button>
            </div>

            <div className="login-captcha-box" role="group" aria-
labelledby="captcha-label">
              <div className="login-captcha-header">
                <i className="fi fi-sr-shield-check icon-mr6 icon-green"
aria-hidden="true"></i>
                <span id="captcha-label" className="login-captcha-label">
                  Επαλήθευση ασφαλείας
                </span>
                <button
                  type="button"
                  onClick={fetchCaptcha}
                  className="login-refresh-btn"
                  aria-label="Ανανέωση captcha"
                  title="Ανανέωση captcha"
```

```
                >
                  <i className="fi fi-sr-rotate-right" aria-
hidden="true"></i>
                </button>
              </div>
              <div className="login-captcha-row">
                <div className="login-captcha-question" aria-
label={`Captcha: ${captchaQuestion}`}>
                  {captchaQuestion}
                </div>
                <input
                  type="number"
                  value={captchaAnswer}
                  onChange={e => setCaptchaAnswer(e.target.value)}
                  placeholder="="
                  className="login-captcha-input"
                  aria-label="Απάντηση captcha"
                  title="Εισάγετε την απάντηση του captcha"
                  aria-required="true"
                />
              </div>
            </div>

            {rememberMe && (
              <div className="login-remember-info" role="note">
                <i className="fi fi-sr-info icon-mr6 icon-blue2" aria-
hidden="true"></i>
                Θα παραμείνετε συνδεδεμένος για 30 ημέρες.
              </div>
            )}

            <button type="submit" disabled={loading} className="login-
submit-btn" title="Σύνδεση στο σύστημα">
              {loading ? (
                <><i className="fi fi-sr-rotate-right icon-mr8" aria-
hidden="true"></i>Σύνδεση...</>
              ) : (
                <><i className="fi fi-sr-sign-in-alt icon-mr8" aria-
hidden="true"></i>Σύνδεση</>
              )}
            </button>
          </form>
        )}

        {/* ===== FORGOT PASSWORD ===== */}
        {mode === 'forgot' && (
          <form onSubmit={handleForgotPassword} noValidate>
```

```
            <div className="login-section-title">
              <i className="fi fi-sr-key icon-mr8 icon-orange" aria-
hidden="true"></i>
              Ανάκτηση Κωδικού
            </div>
            <p className="login-info-text">
              Εισάγετε το <strong>όνομα χρήστη</strong> σας και θα σας
στείλουμε
              email με οδηγίες για επαναφορά του κωδικού.
            </p>

            <div className="login-field">
              <label htmlFor="reset-username" className="login-label">
                <i className="fi fi-sr-user icon-mr6 icon-gray" aria-
hidden="true"></i>
                Όνομα Χρήστη (Username)
              </label>
              <input
                id="reset-username"
                type="text"
                value={resetUsername}
                onChange={e => setResetUsername(e.target.value)}
                placeholder="Εισάγετε το username σας"
                className="login-input"
                autoFocus
                aria-required="true"
                title="Εισάγετε το όνομα χρήστη σας"
              />
            </div>

            <button type="submit" disabled={loading} className="login-
submit-btn" title="Αποστολή email ανάκτησης κωδικού">
              {loading ? (
                <><i className="fi fi-sr-rotate-right icon-mr8" aria-
hidden="true"></i>Αποστολή...</>
              ) : (
                <><i className="fi fi-sr-envelope icon-mr8" aria-
hidden="true"></i>Αποστολή Email</>
              )}
            </button>

            <button
              type="button"
              onClick={resetToLoginMode}
              className="login-back-btn"
              title="Επιστροφή στη σύνδεση"
            >
```

```
              <i className="fi fi-sr-arrow-left icon-mr6" aria-
hidden="true"></i>
              Επιστροφή στη Σύνδεση
            </button>
          </form>
        )}

        {/* ===== RESET SENT ===== */}
        {mode === 'reset-sent' && (
          <div className="login-text-center" role="status" aria-
live="polite">
            <div className="login-section-title login-section-title-
center">
              <i className="fi fi-sr-envelope icon-mr8 icon-green" aria-
hidden="true"></i>
              Email Απεστάλη!
            </div>
            <div className="login-success-icon" aria-hidden="true">
              <i className="fi fi-sr-envelope"></i>
            </div>
            <p className="login-info-text">
              Αν το username <strong>"{resetUsername}"</strong> υπάρχει στο
σύστημα,
              θα λάβετε email με σύνδεσμο για επαναφορά κωδικού.
            </p>
            <p className="login-hint-text">
              Ο σύνδεσμος ισχύει για <strong>2 ώρες</strong>.
              Ελέγξτε και τον φάκελο spam.
            </p>

            {devResetUrl && (
              <div className="login-dev-box">
                <div className="login-dev-title">
                  <i className="fi fi-sr-tools" aria-hidden="true"></i>
                  Dev Mode — SMTP δεν έχει ρυθμιστεί
                </div>
                <div className="login-dev-text">Χρησιμοποιήστε αυτόν τον
σύνδεσμο για δοκιμή:</div>
                <button
                  onClick={() =>
validateResetToken(devResetUrl.split('token=')[1])}
                  className="login-dev-btn"
                  title="Προχώρηση σε reset κωδικού (dev mode)"
                >
                  <i className="fi fi-sr-key icon-mr6" aria-
hidden="true"></i>
                  Προχώρηση σε Reset (Dev)
```

```
              </button>
            </div>
          )}

          <button
            onClick={() => { setMode('login'); setError('');
setDevResetUrl(''); }}
            className="login-submit-btn"
            title="Επιστροφή στη σύνδεση"
          >
            <i className="fi fi-sr-sign-in-alt icon-mr8" aria-
hidden="true"></i>
            Επιστροφή στη Σύνδεση
          </button>
        </div>
      )}

      {/* ===== RESET PASSWORD ===== */}
      {mode === 'reset-password' && (
        <form onSubmit={handleResetPassword} noValidate>
          <div className="login-section-title">
            <i className="fi fi-sr-lock icon-mr8 icon-blue" aria-
hidden="true"></i>
            Ορισμός Νέου Κωδικού
          </div>

          {resetTokenUsername && (
            <div className="login-remember-info" role="note">
              <i className="fi fi-sr-user icon-mr6" aria-
hidden="true"></i>
              Χρήστης: <strong>{resetTokenUsername}</strong>
            </div>
          )}

          <div className="login-field">
            <label htmlFor="new-password" className="login-label">
              <i className="fi fi-sr-key icon-mr6 icon-gray" aria-
hidden="true"></i>
              Νέος Κωδικός
            </label>
            <div className="login-input-wrapper">
              <input
                id="new-password"
                type={showNewPassword ? 'text' : 'password'}
                value={newPassword}
                onChange={e => setNewPassword(e.target.value)}
                placeholder="Εισάγετε νέο κωδικό"
```

```
                  className="login-input login-input-password"
                  autoFocus
                  autoComplete="new-password"
                  aria-required="true"
                  title="Εισάγετε τον νέο κωδικό πρόσβασης"
                />
                <button
                  type="button"
                  onClick={() => setShowNewPassword(!showNewPassword)}
                  className="login-eye-btn"
                  aria-label={showNewPassword ? 'Απόκρυψη νέου κωδικού' :
'Εμφάνιση νέου κωδικού'}
                  title={showNewPassword ? 'Απόκρυψη κωδικού' : 'Εμφάνιση
κωδικού'}
                >
                  <i className={`fi ${showNewPassword ? 'fi-sr-eye-crossed'
: 'fi-sr-eye'}`} aria-hidden="true"></i>
                </button>
              </div>
              {pwdStrength && (
                <div className="login-pwd-strength" aria-label={`Ισχύς
κωδικού: ${pwdStrength.label}`}>
                  <div className="login-pwd-strength-row">
                    <span className="icon-gray">Ισχύς κωδικού:</span>
                    <span className={`login-pwd-label login-pwd-label-
${strengthSlug(pwdStrength.label)}`}>
                      {pwdStrength.label}
                    </span>
                  </div>
                  <div className="login-pwd-strength-bar-bg" title={`Ισχύς
κωδικού: ${pwdStrength.label}`}>
                    <div
                      className={`login-pwd-strength-bar login-pwd-bar-
${strengthSlug(pwdStrength.label)}`}
                    />
                  </div>
                </div>
              )}
            </div>

            <div className="login-field">
              <label htmlFor="confirm-password" className="login-label">
                <i className="fi fi-sr-key icon-mr6 icon-gray" aria-
hidden="true"></i>
                Επιβεβαίωση Νέου Κωδικού
              </label>
              <div className="login-input-wrapper">
```

```
                <input
                  id="confirm-password"
                  type={showConfirmPassword ? 'text' : 'password'}
                  value={confirmPassword}
                  onChange={e => setConfirmPassword(e.target.value)}
                  placeholder="Επαναλάβετε τον κωδικό"
                  className={`login-input login-input-password
${passwordsMismatch ? 'login-input-confirm-nomatch' :
                      passwordsMatch ? 'login-input-confirm-match' : ''
                    }`}
                  autoComplete="new-password"
                  aria-required="true"
                  title="Επαναλάβετε τον νέο κωδικό πρόσβασης"
                  aria-describedby="confirm-password-error"
                />
                <button
                  type="button"
                  onClick={() =>
setShowConfirmPassword(!showConfirmPassword)}
                  className="login-eye-btn"
                  aria-label={showConfirmPassword ? 'Απόκρυψη επιβεβαίωσης
κωδικού' : 'Εμφάνιση επιβεβαίωσης κωδικού'}
                  title={showConfirmPassword ? 'Απόκρυψη κωδικού' :
'Εμφάνιση κωδικού'}
                >
                  <i className={`fi ${showConfirmPassword ? 'fi-sr-eye-
crossed' : 'fi-sr-eye'}`} aria-hidden="true"></i>
                </button>
              </div>
              {passwordsMismatch && (
                <div id="confirm-password-error" className="login-match-
error" role="alert">
                  <i className="fi fi-sr-cross-circle" aria-
hidden="true"></i>
                  Οι κωδικοί δεν ταιριάζουν
                </div>
              )}
              {passwordsMatch && (
                <div className="login-match-ok" role="status">
                  <i className="fi fi-sr-check-circle" aria-
hidden="true"></i>
                  Οι κωδικοί ταιριάζουν
                </div>
              )}
            </div>
```

```
            <button type="submit" disabled={loading} className="login-
submit-btn" title="Αποθήκευση νέου κωδικού">
              {loading ? (
                <><i className="fi fi-sr-rotate-right icon-mr8" aria-
hidden="true"></i>Αποθήκευση...</>
              ) : (
                <><i className="fi fi-sr-check-circle icon-mr8" aria-
hidden="true"></i>Αλλαγή Κωδικού</>
              )}
            </button>
          </form>
        )}

        {/* ===== RESET SUCCESS ===== */}
        {mode === 'reset-success' && (
          <div className="login-text-center" role="status" aria-
live="polite">
            <div className="login-section-title login-section-title-
center">
              <i className="fi fi-sr-check-circle icon-mr8 icon-green"
aria-hidden="true"></i>
              Επιτυχής Αλλαγή!
            </div>
            <div className="login-success-icon" aria-hidden="true">
              <i className="fi fi-sr-check-circle"></i>
            </div>
            <p className="login-info-text">
              Ο κωδικός σας άλλαξε επιτυχώς!
              Μπορείτε τώρα να συνδεθείτε με τον νέο σας κωδικό.
            </p>
            <button
              onClick={() => {
                setMode('login');
                setNewPassword('');
                setConfirmPassword('');
                setResetToken('');
                setResetTokenUsername('');
                fetchCaptcha();
              }}
              className="login-submit-btn"
              title="Μετάβαση στη σύνδεση"
            >
              <i className="fi fi-sr-sign-in-alt icon-mr8" aria-
hidden="true"></i>
              Μετάβαση στη Σύνδεση
            </button>
          </div>
```

```
        )}

        <div className="login-footer" role="contentinfo">
          Smart Waste Management v1.0 — Δήμος Μαλεβιζίου
        </div>
      </div>
    </div>
  );
}

export default LoginPage;
```

frontend\src\LoginPage.css

```
/* ===== LoginPage.css ===== */

/* Icon utility classes */
.icon-mr8 { margin-right: 8px; }
.icon-mr6 { margin-right: 6px; }
.icon-blue { color: #3498db; }
.icon-blue2 { color: #2980b9; }
.icon-green { color: #27ae60; }
.icon-orange { color: #f39c12; }
.icon-gray { color: #7f8c8d; }
.icon-logo { font-size: 1.8rem; color: white; }
.icon-check-sm { font-size: 0.7rem; color: white; }

/* Text center utility */
.login-text-center { text-align: center; }

/* Section title centered */
.login-section-title-center { justify-content: center; }

/* Captcha label */
.login-captcha-label {
  font-size: 0.85rem;
  color: #2c3e50;
  font-weight: bold;
}

/* Hint text */
.login-hint-text {
  font-size: 0.85rem;
  color: #7f8c8d;
  margin-bottom: 20px;
}
```

```
/* Password strength label colors */
.login-pwd-label { font-weight: bold; }
.login-pwd-label-πολύ-αδύναμος { color: #e74c3c; }
.login-pwd-label-αδύναμος { color: #f39c12; }
.login-pwd-label-μέτριος { color: #f39c12; }
.login-pwd-label-καλός { color: #2ecc71; }
.login-pwd-label-ισχυρός { color: #27ae60; }

/* Password strength bar widths */
.login-pwd-bar-πολύ-αδύναμος { width: 20%; background-color: #e74c3c; }
.login-pwd-bar-αδύναμος { width: 40%; background-color: #f39c12; }
.login-pwd-bar-μέτριος { width: 60%; background-color: #f39c12; }
.login-pwd-bar-καλός { width: 80%; background-color: #2ecc71; }
.login-pwd-bar-ισχυρός { width: 100%; background-color: #27ae60; }


.login-container {
  min-height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  background: linear-gradient(135deg, #1a252f 0%, #2c3e50 50%, #1a3a4a
100%);
  padding: 20px;
  position: relative;
  overflow: hidden;
}

/* Background decorative circles */
.login-bg-circle1 {
  position: absolute;
  top: -100px;
  right: -100px;
  width: 400px;
  height: 400px;
  border-radius: 50%;
  background: rgba(52, 152, 219, 0.1);
  pointer-events: none;
}

.login-bg-circle2 {
  position: absolute;
  bottom: -150px;
  left: -150px;
  width: 500px;
  height: 500px;
  border-radius: 50%;
```

```
  background: rgba(39, 174, 96, 0.08);
  pointer-events: none;
}

.login-bg-circle3 {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 600px;
  height: 600px;
  border-radius: 50%;
  background: rgba(255, 255, 255, 0.02);
  pointer-events: none;
}

/* Card */
.login-card {
  background-color: white;
  border-radius: 16px;
  padding: 36px;
  width: 100%;
  max-width: 440px;
  box-shadow: 0 25px 80px rgba(0, 0, 0, 0.4);
  position: relative;
  z-index: 1;
}

/* Logo area */
.login-logo-area {
  text-align: center;
  margin-bottom: 28px;
}

.login-logo-icon {
  width: 64px;
  height: 64px;
  background: linear-gradient(135deg, #2c3e50, #3498db);
  border-radius: 16px;
  display: flex;
  align-items: center;
  justify-content: center;
  margin: 0 auto 12px;
  box-shadow: 0 8px 24px rgba(52, 152, 219, 0.3);
}

.login-title {
```

```
  font-size: 1.3rem;
  font-weight: bold;
  color: #2c3e50;
  margin: 0 0 4px;
}

.login-subtitle {
  font-size: 0.85rem;
  color: #7f8c8d;
  margin: 0;
}

/* Section title */
.login-section-title {
  font-size: 1rem;
  font-weight: bold;
  color: #2c3e50;
  margin-bottom: 20px;
  display: flex;
  align-items: center;
  padding-bottom: 10px;
  border-bottom: 2px solid #ecf0f1;
}

/* Fields */
.login-field {
  margin-bottom: 16px;
}

.login-label {
  display: flex;
  align-items: center;
  margin-bottom: 6px;
  font-weight: bold;
  font-size: 0.87rem;
  color: #2c3e50;
}

.login-input-wrapper {
  position: relative;
}

.login-input {
  width: 100%;
  padding: 11px 14px;
  border: 2px solid #ecf0f1;
  border-radius: 8px;
```

```
  font-size: 0.95rem;
  outline: none;
  transition: border-color 0.2s;
  box-sizing: border-box;
  font-family: inherit;
}

.login-input:focus {
  border-color: #3498db;
}

.login-input-password {
  padding-right: 44px;
}

.login-input-confirm-match {
  border-color: #27ae60;
}

.login-input-confirm-nomatch {
  border-color: #e74c3c;
}

/* Eye button */
.login-eye-btn {
  position: absolute;
  right: 12px;
  top: 50%;
  transform: translateY(-50%);
  background: none;
  border: none;
  cursor: pointer;
  color: #7f8c8d;
  font-size: 1rem;
  padding: 4px;
}

/* Remember me row */
.login-remember-row {
  display: flex;
  align-items: center;
  justify-content: space-between;
  margin-bottom: 16px;
}

.login-checkbox-label {
  display: flex;
```

```
  align-items: center;
  gap: 8px;
  cursor: pointer;
}

/* Native checkbox hidden (χρησιμοποιείται για accessibility) */
.login-checkbox-native {
  position: absolute;
  opacity: 0;
  width: 0;
  height: 0;
  margin: 0;
}


.login-checkbox {
  width: 20px;
  height: 20px;
  border: 2px solid #bdc3c7;
  border-radius: 5px;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: all 0.2s;
  flex-shrink: 0;
}

.login-checkbox.checked {
  background-color: #3498db;
  border-color: #3498db;
}

.login-checkbox-text {
  font-size: 0.9rem;
  color: #2c3e50;
  cursor: pointer;
}

.login-link-btn {
  background: none;
  border: none;
  color: #3498db;
  cursor: pointer;
  font-size: 0.87rem;
  font-weight: bold;
  text-decoration: underline;
```

```
  padding: 0;
}

/* Captcha */
.login-captcha-box {
  background-color: #f8f9fa;
  border-radius: 8px;
  padding: 12px 14px;
  margin-bottom: 16px;
  border: 1px solid #ecf0f1;
}

.login-captcha-header {
  display: flex;
  align-items: center;
  margin-bottom: 8px;
}

.login-captcha-question {
  background-color: #2c3e50;
  color: white;
  padding: 8px 16px;
  border-radius: 8px;
  font-weight: bold;
  font-size: 1rem;
  letter-spacing: 1px;
  flex: 1;
}

.login-captcha-row {
  display: flex;
  gap: 10px;
  align-items: center;
}

.login-captcha-input {
  width: 70px;
  padding: 8px;
  border: 2px solid #3498db;
  border-radius: 8px;
  text-align: center;
  font-size: 1.1rem;
  font-weight: bold;
  outline: none;
}

.login-refresh-btn {
```

```
  background: none;
  border: none;
  cursor: pointer;
  color: #3498db;
  font-size: 0.9rem;
  margin-left: auto;
  padding: 2px 6px;
}

/* Info boxes */
.login-remember-info {
  background-color: #ebf5fb;
  color: #2980b9;
  padding: 8px 12px;
  border-radius: 6px;
  font-size: 0.82rem;
  margin-bottom: 16px;
  display: flex;
  align-items: center;
}

/* Submit button */
.login-submit-btn {
  width: 100%;
  padding: 12px;
  background: linear-gradient(135deg, #2c3e50, #3498db);
  color: white;
  border: none;
  border-radius: 8px;
  font-size: 1rem;
  font-weight: bold;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  box-shadow: 0 4px 15px rgba(52, 152, 219, 0.3);
  margin-bottom: 12px;
  font-family: inherit;
}

.login-submit-btn:disabled {
  opacity: 0.7;
  cursor: not-allowed;
}

.login-back-btn {
  width: 100%;
```

```
  padding: 10px;
  background-color: #ecf0f1;
  color: #2c3e50;
  border: none;
  border-radius: 8px;
  font-size: 0.9rem;
  font-weight: bold;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  font-family: inherit;
}

/* Alerts */
.login-alert-error {
  background-color: #fdecea;
  color: #e74c3c;
  padding: 10px 14px;
  border-radius: 8px;
  margin-bottom: 16px;
  font-size: 0.87rem;
  display: flex;
  align-items: center;
  border: 1px solid #f5c6cb;
}

.login-alert-success {
  background-color: #d5f5e3;
  color: #27ae60;
  padding: 10px 14px;
  border-radius: 8px;
  margin-bottom: 16px;
  font-size: 0.87rem;
  display: flex;
  align-items: center;
  border: 1px solid #a9dfbf;
}

/* Info text */
.login-info-text {
  font-size: 0.87rem;
  color: #5d6d7e;
  line-height: 1.6;
  margin-bottom: 20px;
  text-align: center;
}
```

```
/* Password strength */
.login-pwd-strength {
  margin-top: 6px;
}

.login-pwd-strength-row {
  display: flex;
  justify-content: space-between;
  font-size: 0.75rem;
  margin-bottom: 3px;
}

.login-pwd-strength-bar-bg {
  background-color: #ecf0f1;
  border-radius: 10px;
  height: 5px;
}

.login-pwd-strength-bar {
  height: 100%;
  border-radius: 10px;
  transition: width 0.3s;
}

/* Match messages */
.login-match-ok {
  font-size: 0.78rem;
  color: #27ae60;
  margin-top: 4px;
  display: flex;
  align-items: center;
  gap: 4px;
}

.login-match-error {
  font-size: 0.78rem;
  color: #e74c3c;
  margin-top: 4px;
  display: flex;
  align-items: center;
  gap: 4px;
}

/* Dev mode box */
.login-dev-box {
  background-color: #fef9e7;
```

```
  border: 1px solid #f39c12;
  border-radius: 8px;
  padding: 12px;
  margin-bottom: 16px;
  text-align: left;
}

.login-dev-title {
  font-size: 0.8rem;
  color: #f39c12;
  font-weight: bold;
  margin-bottom: 6px;
  display: flex;
  align-items: center;
  gap: 6px;
}

.login-dev-text {
  font-size: 0.75rem;
  color: #2c3e50;
  margin-bottom: 8px;
}

.login-dev-btn {
  width: 100%;
  padding: 8px;
  background-color: #f39c12;
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-weight: bold;
  font-size: 0.85rem;
  font-family: inherit;
}

/* Success icon */
.login-success-icon {
  font-size: 3rem;
  margin: 20px 0;
  text-align: center;
}

/* Footer */
.login-footer {
  text-align: center;
  font-size: 0.75rem;
```

```
  color: #bdc3c7;
  margin-top: 20px;
  padding-top: 16px;
  border-top: 1px solid #ecf0f1;
}
/* Password strength bar – πλάτος ανά επίπεδο ισχύος (αντί για inline
style) */
.login-pwd-bar-very-weak { width: 20%;  background-color: #e74c3c; }
.login-pwd-bar-weak      { width: 40%;  background-color: #f39c12; }
.login-pwd-bar-medium    { width: 60%;  background-color: #f39c12; }
.login-pwd-bar-good      { width: 80%;  background-color: #2ecc71; }
.login-pwd-bar-strong    { width: 100%; background-color: #27ae60; }

.login-pwd-label-very-weak { color: #e74c3c; }
.login-pwd-label-weak      { color: #f39c12; }
.login-pwd-label-medium    { color: #f39c12; }
.login-pwd-label-good      { color: #2ecc71; }
.login-pwd-label-strong    { color: #27ae60; }
```

Frontend\src\ZonesView.tsx

```
/**
 * ZonesView.tsx
 * =============
 * Στατιστικά ανά ζώνη του δήμου. Δύο προβολές:
 *   table → bar chart σύγκρισης + κάρτες ζωνών. Όταν επιλεγεί ζώνη ανοίγει
 *           panel με τις λεπτομέρειες και τη λίστα των κάδων της.
 *   map   → χάρτης με τα όρια (Circle) και τα κέντρα των ζωνών.
 *
 * Τα δεδομένα ανανεώνονται μόνα τους κάθε 30s.
 *
 * Styling: όλα τα χρώματα ορίζονται από CSS classes (zone-XXX / fill-* /
stat-*).
 * Το μόνο δυναμικό μέγεθος είναι το πλάτος της μπάρας πληρότητας, που το
θέτουμε
 * ως CSS variable μέσω ref (widthRef) ώστε να μη χρειάζεται inline style.
 */
import React, { useState, useEffect } from 'react';
import {
  BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Cell,
} from 'recharts';
import { MapContainer, TileLayer, Marker, Popup, Circle } from 'react-
leaflet';
import L from 'leaflet';
import 'leaflet/dist/leaflet.css';
import './ZonesView.css';
```

```
const API_URL = 'http://localhost:8000';
const REFRESH_INTERVAL_MS = 30000;

// ------------------------------------------------------------------------
---
// Types
// ------------------------------------------------------------------------
---

interface Zone {
  zone_id: number;
  zone_code: string;
  zone_name: string;
  total_bins: number;
  critical_bins: number;
  warning_bins: number;
  normal_bins: number;
  avg_fill_level: number;
  total_vehicles: number;
}

interface ZoneBin {
  bin_id: string;
  location_name: string;
  current_fill_level: number;
  fill_status: string;
  current_battery_level: number;
  latitude: number;
  longitude: number;
  active_alerts_count: number;
}

// ------------------------------------------------------------------------
---
// Constants
// ------------------------------------------------------------------------
---

// Σταθερό χρώμα ανά κωδικό ζώνης — χρειάζεται ως prop στο recharts/leaflet
// (το ίδιο χρώμα ορίζεται και ως CSS class zone-XXX για το υπόλοιπο
styling).
const ZONE_COLORS: Record<string, string> = {
  GAZ: '#3498db',
  AMM: '#27ae60',
  APE: '#e74c3c',
  TYL: '#f39c12',
  KRO: '#9b59b6',
```

```
  ROD: '#1abc9c',
};

// Γεωγραφικό κέντρο κάθε ζώνης (για τα markers στον χάρτη).
const ZONE_CENTERS: Record<string, [number, number]> = {
  GAZ: [35.3259, 25.0659],
  AMM: [35.3370, 25.0800],
  APE: [35.4050, 25.0185],
  TYL: [35.2963, 25.0184],
  KRO: [35.2300, 24.9815],
  ROD: [35.3596, 25.0278],
};

const MAP_CENTER: [number, number] = [35.33, 25.04];
const DEFAULT_ZONE_COLOR = '#3498db';

// ------------------------------------------------------------------------
---
// Helpers
// ------------------------------------------------------------------------
---

function zoneColor(code: string): string {
  return ZONE_COLORS[code] ?? DEFAULT_ZONE_COLOR;
}

/** CSS class με το χρώμα πληρότητας: κόκκινο >=80, πορτοκαλί >=60, αλλιώς
πράσινο. */
function fillClass(level: number): string {
  if (level >= 80) return 'fill-critical';
  if (level >= 60) return 'fill-warning';
  return 'fill-ok';
}

/** Ref callback που θέτει το πλάτος της μπάρας ως CSS variable — χωρίς
inline style. */
function widthRef(level: number) {
  return (el: HTMLDivElement | null): void => {
    if (el) el.style.setProperty('--fill-width', `${level}%`);
  };
}

// ------------------------------------------------------------------------
---
// Component
// ------------------------------------------------------------------------
---
```

```
function ZonesView() {
  const [zones, setZones] = useState<Zone[]>([]);
  const [selectedZone, setSelectedZone] = useState<Zone | null>(null);
  const [zoneBins, setZoneBins] = useState<ZoneBin[]>([]);
  const [loading, setLoading] = useState(true);
  const [loadingBins, setLoadingBins] = useState(false);
  const [showMap, setShowMap] = useState(false);
  const [lastUpdated, setLastUpdated] = useState<Date | null>(null);

  // Αρχική φόρτωση + auto-refresh κάθε 30s
  useEffect(() => {
    fetchZones();
    const interval = setInterval(fetchZones, REFRESH_INTERVAL_MS);
    return () => clearInterval(interval);
  }, []);

  // ------------------------------------------------------------------------
--
  // API calls
  // ------------------------------------------------------------------------
--

  const fetchZones = async (): Promise<void> => {
    try {
      const res = await fetch(`${API_URL}/api/v1/zones`);
      const json = await res.json();
      setZones(json.zones ?? []);
      setLastUpdated(new Date());
    } catch (error) {
      console.error('Failed to fetch zones:', error);
    } finally {
      setLoading(false);
    }
  };

  const fetchZoneBins = async (zoneCode: string): Promise<void> => {
    setLoadingBins(true);
    try {
      const res = await fetch(`${API_URL}/api/v1/zones/${zoneCode}/bins`);
      const json = await res.json();
      setZoneBins(json.bins ?? []);
    } catch (error) {
      console.error('Failed to fetch zone bins:', error);
    } finally {
      setLoadingBins(false);
    }
```

```
  };

  // ------------------------------------------------------------------------
--
  // Handlers
  // ------------------------------------------------------------------------
--

  const handleZoneClick = (zone: Zone): void => {
    setSelectedZone(zone);
    fetchZoneBins(zone.zone_code);
    setShowMap(false);
  };

  // ------------------------------------------------------------------------
--
  // Derived data
  // ------------------------------------------------------------------------
--

  const totalStats = {
    bins: zones.reduce((sum, z) => sum + z.total_bins, 0),
    critical: zones.reduce((sum, z) => sum + z.critical_bins, 0),
    warning: zones.reduce((sum, z) => sum + z.warning_bins, 0),
    vehicles: zones.reduce((sum, z) => sum + z.total_vehicles, 0),
    avgFill: zones.length > 0
      ? (zones.reduce((sum, z) => sum + z.avg_fill_level, 0) /
zones.length).toFixed(1)
      : '0',
  };

  const summaryCards = [
    { label: 'Σύνολο Κάδων', value: totalStats.bins, cls: 'stat-blue',
icon: 'fi-sr-trash' },
    { label: 'Κρίσιμοι', value: totalStats.critical, cls: 'stat-red', icon:
'fi-sr-triangle-warning' },
    { label: 'Προσοχή', value: totalStats.warning, cls: 'stat-orange',
icon: 'fi-sr-triangle-warning' },
    { label: 'Μέση Πληρότητα', value: `${totalStats.avgFill}%`, cls: 'stat-
purple', icon: 'fi-sr-chart-histogram' },
    { label: 'Οχήματα', value: totalStats.vehicles, cls: 'stat-green',
icon: 'fi-sr-truck-side' },
  ];

  const detailStats = selectedZone ? [
    { label: 'Σύνολο Κάδων', value: selectedZone.total_bins, cls: 'stat-
blue' },
```

```
    { label: 'Κρίσιμοι', value: selectedZone.critical_bins, cls: 'stat-red'
},
    { label: 'Προσοχή', value: selectedZone.warning_bins, cls: 'stat-
orange' },
    { label: 'Κανονικοί', value: selectedZone.normal_bins, cls: 'stat-
green' },
    { label: 'Μέση Πληρότητα', value: `${selectedZone.avg_fill_level}%`,
cls: 'stat-purple' },
    { label: 'Οχήματα', value: selectedZone.total_vehicles, cls: 'stat-
teal' },
  ] : [];

  // ------------------------------------------------------------------
--
  // Render
  // ------------------------------------------------------------------
--

  return (
    <div className="zones-container">

      {/* Header */}
      <div className="zones-header">
        <h2 className="zones-title">
          <i className="fi fi-sr-map-marker-home" aria-hidden="true"></i>
          Στατιστικά Ζωνών
        </h2>
        <div className="zones-header-actions">
          {lastUpdated && (
            <span className="zones-last-updated">
              <i className="fi fi-sr-rotate-right icon-mr4" aria-
hidden="true"></i>
              {lastUpdated.toLocaleTimeString('el-GR')}
            </span>
          )}
          <button
            onClick={() => setShowMap(!showMap)}
            className={`zones-toggle-btn ${showMap ? 'active' : ''}`}
            title={showMap ? 'Εμφάνιση πίνακα στατιστικών' : 'Εμφάνιση
χάρτη ζωνών'}
          >
            <i className={`fi ${showMap ? 'fi-sr-list' : 'fi-sr-map'}`}
aria-hidden="true"></i>
            {showMap ? 'Προβολή Πίνακα' : 'Προβολή Χάρτη'}
          </button>
        </div>
      </div>
```

```
      {/* Συνολικά stats */}
      <div className="zones-summary-grid">
        {summaryCards.map(card => (
          <div key={card.label} className={`zones-stat-card ${card.cls}`}>
            <i className={`fi ${card.icon} zones-stat-icon`} aria-
hidden="true"></i>
            <div className="zones-stat-value">{card.value}</div>
            <div className="zones-stat-label">{card.label}</div>
          </div>
        ))}
      </div>

      {loading ? (
        <div className="zones-loading" role="status" aria-live="polite">
          <i className="fi fi-sr-rotate-right zones-loading-icon" aria-
hidden="true"></i>
        </div>
      ) : showMap ? (

        /* ===== Προβολή χάρτη ===== */
        <div className="zones-map-wrap">
          <MapContainer center={MAP_CENTER} zoom={11} className="zones-
map">
            <TileLayer
              attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>'
              url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
            />
            {zones.map(zone => {
              const center = ZONE_CENTERS[zone.zone_code];
              if (!center) return null;
              const color = zoneColor(zone.zone_code);

              return (
                <React.Fragment key={zone.zone_code}>
                  <Circle
                    center={center}
                    radius={3000}
                    color={color}
                    fillColor={color}
                    fillOpacity={0.15}
                    weight={2}
                  />
                  <Marker
                    position={center}
                    icon={L.divIcon({
```

```
                    html: `<div class="zones-map-label zone-
${zone.zone_code}">
                      ${zone.zone_name}<br/>
                      <small>${zone.total_bins} κάδοι |
${zone.avg_fill_level}%</small>
                    </div>`,
                    className: '',
                    iconSize: [120, 50],
                    iconAnchor: [60, 25],
                  })}
                >
                  <Popup>
                    <div className="zones-popup">
                      <h4 className="zones-popup-
title">{zone.zone_name}</h4>
                      <p className="zones-popup-row">
                        <i className="fi fi-sr-trash icon-mr6" aria-
hidden="true"></i>
                        Κάδοι: {zone.total_bins}
                      </p>
                      <p className="zones-popup-row critical">
                        <i className="fi fi-sr-triangle-warning icon-mr6"
aria-hidden="true"></i>
                        Κρίσιμοι: {zone.critical_bins}
                      </p>
                      <p className="zones-popup-row warning">
                        <i className="fi fi-sr-triangle-warning icon-mr6"
aria-hidden="true"></i>
                        Προσοχή: {zone.warning_bins}
                      </p>
                      <p className="zones-popup-row">
                        <i className="fi fi-sr-chart-histogram icon-mr6"
aria-hidden="true"></i>
                        Μέση πληρότητα: {zone.avg_fill_level}%
                      </p>
                      <p className="zones-popup-row">
                        <i className="fi fi-sr-truck-side icon-mr6" aria-
hidden="true"></i>
                        Οχήματα: {zone.total_vehicles}
                      </p>
                    </div>
                  </Popup>
                </Marker>
              </React.Fragment>
            );
          })}
        </MapContainer>
```

```
        </div>

      ) : (

        /* ===== Προβολή πίνακα ===== */
        <div className={`zones-layout ${selectedZone ? 'with-detail' :
''}`}>

          {/* Chart + κάρτες ζωνών */}
          <div>
            <div className="zones-panel">
              <h3 className="zones-panel-title">
                <i className="fi fi-sr-chart-histogram" aria-
hidden="true"></i>
                Σύγκριση Ζωνών
              </h3>
              <ResponsiveContainer width="100%" height={250}>
                <BarChart data={zones} margin={{ top: 5, right: 20, left:
10, bottom: 5 }}>
                  <CartesianGrid strokeDasharray="3 3" />
                  <XAxis dataKey="zone_name" tick={{ fontSize: 11 }} />
                  <YAxis />
                  <Tooltip
                    formatter={(val: number, name: string) =>
                      [val, name === 'avg_fill_level' ? 'Μέση Πληρότητα %'
: name]
                    }
                  />
                  <Legend />
                  <Bar dataKey="total_bins" name="Κάδοι" radius={[4, 4, 0,
0]}>
                    {zones.map(z => (
                      <Cell key={z.zone_code} fill={zoneColor(z.zone_code)}
/>
                    ))}
                  </Bar>
                  <Bar dataKey="critical_bins" name="Κρίσιμοι"
fill="#e74c3c" radius={[4, 4, 0, 0]} />
                </BarChart>
              </ResponsiveContainer>
            </div>

            <div className="zones-cards-grid">
              {zones.map(zone => {
                const isSelected = selectedZone?.zone_code ===
zone.zone_code;
                return (
```

```
                <div
                  key={zone.zone_code}
                  onClick={() => handleZoneClick(zone)}
                  onKeyDown={e => { if (e.key === 'Enter')
handleZoneClick(zone); }}
                  className={`zones-card zone-${zone.zone_code}
${isSelected ? 'selected' : ''}`}
                  role="button"
                  tabIndex={0}
                  title={`Λεπτομέρειες ζώνης ${zone.zone_name}`}
                >
                  <div className="zones-card-header">
                    <h4 className="zones-card-name">{zone.zone_name}</h4>
                    <span className="zones-card-
badge">{zone.zone_code}</span>
                  </div>

                  {/* Μπάρα μέσης πληρότητας */}
                  <div className={`zones-fill
${fillClass(zone.avg_fill_level)}`} ref={widthRef(zone.avg_fill_level)}>
                    <div className="zones-fill-header">
                      <span>Μέση Πληρότητα</span>
                      <span className="zones-fill-
pct">{zone.avg_fill_level}%</span>
                    </div>
                    <div className="zones-fill-bar-bg">
                      <div className="zones-fill-bar" />
                    </div>
                  </div>

                  <div className="zones-card-stats">
                    <div className="zones-card-stat critical">
                      <div className="zones-card-stat-
value">{zone.critical_bins}</div>
                      <div className="zones-card-stat-
label">Κρίσιμοι</div>
                    </div>
                    <div className="zones-card-stat bins">
                      <div className="zones-card-stat-
value">{zone.total_bins}</div>
                      <div className="zones-card-stat-label">Κάδοι</div>
                    </div>
                    <div className="zones-card-stat vehicles">
                      <div className="zones-card-stat-
value">{zone.total_vehicles}</div>
                      <div className="zones-card-stat-
label">Οχήματα</div>
```

```
                    </div>
                  </div>
                </div>
              );
            })}
          </div>
        </div>

        {/* Panel λεπτομερειών επιλεγμένης ζώνης */}
        {selectedZone && (
          <div>
            <div className={`zones-detail-card zone-
${selectedZone.zone_code}`}>
              <div className="zones-detail-header">
                <h3 className="zones-detail-title">
                  <i className="fi fi-sr-marker zones-detail-marker"
aria-hidden="true"></i>
                  {selectedZone.zone_name}
                </h3>
                <button
                  onClick={() => setSelectedZone(null)}
                  className="zones-close-btn"
                  aria-label="Κλείσιμο λεπτομερειών"
                  title="Κλείσιμο"
                >
                  <i className="fi fi-sr-cross" aria-hidden="true"></i>
                </button>
              </div>

              <div className="zones-detail-grid">
                {detailStats.map(stat => (
                  <div key={stat.label} className={`zones-detail-stat
${stat.cls}`}>
                    <div className="zones-detail-stat-
value">{stat.value}</div>
                    <div className="zones-detail-stat-
label">{stat.label}</div>
                  </div>
                ))}
              </div>
            </div>

            {/* Λίστα κάδων της ζώνης */}
            <div className={`zones-bins-panel zone-
${selectedZone.zone_code}`}>
              <div className="zones-bins-header">
```

```
                  <i className="fi fi-sr-trash zones-bins-icon" aria-
hidden="true"></i>
                  <h4 className="zones-bins-title">Κάδοι Ζώνης
({zoneBins.length})</h4>
                </div>

                {loadingBins ? (
                  <div className="zones-loading" role="status" aria-
live="polite">
                    <i className="fi fi-sr-rotate-right zones-loading-icon"
aria-hidden="true"></i>
                  </div>
                ) : (
                  <div className="zones-bins-list">
                    {zoneBins.map(bin => (
                      <div key={bin.bin_id} className="zones-bin-row">
                        <div>
                          <div className="zones-bin-id">
                            {bin.active_alerts_count > 0 && (
                              <i className="fi fi-sr-triangle-warning
zones-bin-alert" aria-hidden="true"></i>
                            )}
                            {bin.bin_id}
                          </div>
                          <div className="zones-bin-
location">{bin.location_name}</div>
                        </div>
                        <div
                          className={`zones-bin-right
${fillClass(bin.current_fill_level)}`}
                          ref={widthRef(bin.current_fill_level)}
                        >
                          <div className="zones-bin-
pct">{bin.current_fill_level}%</div>
                          <div className="zones-bin-bar-bg">
                            <div className="zones-bin-bar" />
                          </div>
                        </div>
                      </div>
                    ))}
                  </div>
                )}
              </div>
            </div>
          )}
        </div>
      )}
```

```
    </div>
  );
}

export default ZonesView;
```

Frontend\src\ZonesView.css

```
/* ===== ZonesView.css ===== */

.icon-mr4 {
   margin-right: 4px;
}

.icon-mr6 {
   margin-right: 6px;
}

.icon-mr8 {
   margin-right: 8px;
}

/*
================================================================================
  ΧΡΩΜΑΤΑ ΜΕΣΩ CSS VARIABLES (αντί για inline styles)
  - zone-XXX  -> χρώμα + ανοιχτή απόχρωση κάθε ζώνης
  - fill-*    -> χρώμα μπάρας ανάλογα με πληρότητα
  - stat-*    -> χρώμα κάρτας στατιστικού
  Το πλάτος της μπάρας (--fill-width) ορίζεται από JS μέσω ref, γιατί είναι
  συνεχής τιμή και δεν γίνεται class.
  ================================================================================ */

.zone-GAZ {
   --zone-color: #3498db;
   --zone-tint: #eaf4fb;
}

.zone-AMM {
   --zone-color: #27ae60;
   --zone-tint: #e7f7ee;
}

.zone-APE {
   --zone-color: #e74c3c;
   --zone-tint: #fdeceb;
}
```

```
.zone-TYL {
   --zone-color: #f39c12;
   --zone-tint: #fef4e1;
}

.zone-KRO {
   --zone-color: #9b59b6;
   --zone-tint: #f3ebf7;
}

.zone-ROD {
   --zone-color: #1abc9c;
   --zone-tint: #e6f9f5;
}

.fill-ok {
   --fill-color: #27ae60;
}

.fill-warning {
   --fill-color: #f39c12;
}

.fill-critical {
   --fill-color: #e74c3c;
}

.stat-blue {
   --stat-color: #3498db;
}

.stat-red {
   --stat-color: #e74c3c;
}

.stat-orange {
   --stat-color: #f39c12;
}

.stat-purple {
   --stat-color: #9b59b6;
}

.stat-green {
   --stat-color: #27ae60;
}

.stat-teal {
   --stat-color: #1abc9c;
```

```
}

/* ===== CONTAINER ===== */
.zones-container {
    padding: 20px;
    max-width: 1400px;
    margin: 0 auto;
}

/* ===== HEADER ===== */
.zones-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

.zones-title {
    margin: 0;
    color: #2c3e50;
    display: flex;
    align-items: center;
    gap: 8px;
}

.zones-header-actions {
    display: flex;
    gap: 10px;
    align-items: center;
}

.zones-last-updated {
    font-size: 0.8rem;
    color: #7f8c8d;
    display: flex;
    align-items: center;
}

.zones-toggle-btn {
    padding: 8px 16px;
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-weight: bold;
    font-family: inherit;
    display: flex;
    align-items: center;
```

```
    gap: 6px;
    transition: background-color 0.2s;
}

/* Όταν είμαστε σε προβολή χάρτη, το κουμπί δείχνει "γυρίστε σε πίνακα" */
.zones-toggle-btn.active {
    background-color: #2c3e50;
}

/* ===== SUMMARY STATS ===== */
.zones-summary-grid {
    display: grid;
    grid-template-columns: repeat(5, 1fr);
    gap: 15px;
    margin-bottom: 20px;
}

.zones-stat-card {
    background-color: white;
    border-radius: 10px;
    padding: 15px;
    text-align: center;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    border-top: 4px solid var(--stat-color, #3498db);
}

.zones-stat-icon {
    color: var(--stat-color, #3498db);
    font-size: 1.2rem;
}

.zones-stat-value {
    font-size: 1.8rem;
    font-weight: bold;
    color: var(--stat-color, #3498db);
}

.zones-stat-label {
    font-size: 0.8rem;
    color: #7f8c8d;
    margin-top: 4px;
}

/* ===== LOADING ===== */
.zones-loading {
    text-align: center;
    padding: 40px;
}
```

```
.zones-loading-icon {
    font-size: 2rem;
    color: #3498db;
}

/* ===== MAP ===== */
.zones-map-wrap {
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    height: 500px;
}

.zones-map {
    height: 100%;
    width: 100%;
}

/* Ετικέτα ζώνης στον χάρτη (Leaflet divIcon). Το χρώμα έρχεται από την κλάση zone-XXX. */
.zones-map-label {
    background: var(--zone-color, #3498db);
    color: white;
    padding: 6px 10px;
    border-radius: 8px;
    font-weight: bold;
    font-size: 0.8rem;
    white-space: nowrap;
    text-align: center;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
}

/* Popup χάρτη */
.zones-popup {
    font-family: 'Consolas', monospace;
    min-width: 180px;
}

.zones-popup-title {
    margin: 0 0 8px;
    color: #2c3e50;
}

.zones-popup-row {
    margin: 3px 0;
    font-size: 0.85rem;
    display: flex;
    align-items: center;
}
```

```
.zones-popup-row.critical {
    color: #e74c3c;
}

.zones-popup-row.warning {
    color: #f39c12;
}

/* ===== LAYOUT (προβολή πίνακα) ===== */
.zones-layout {
    display: grid;
    grid-template-columns: 1fr;
    gap: 20px;
}

.zones-layout.with-detail {
    grid-template-columns: 1fr 1fr;
}

/* ===== PANELS / CHART ===== */
.zones-panel {
    background-color: white;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
}

.zones-panel-title {
    margin: 0 0 15px;
    color: #2c3e50;
    display: flex;
    align-items: center;
    gap: 8px;
}

/* ===== ZONE CARDS ===== */
.zones-cards-grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 12px;
}

.zones-card {
    background-color: white;
    border-radius: 10px;
    padding: 15px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
```

```
    cursor: pointer;
    transition: border-color 0.2s, box-shadow 0.2s;
    /* λεπτό περίγραμμα παντού + χρωματιστό accent αριστερά */
    border: 1px solid #ecf0f1;
    border-left: 4px solid var(--zone-color, #3498db);
}

.zones-card:hover {
    box-shadow: 0 4px 14px rgba(0, 0, 0, 0.15);
}

/* Επιλεγμένη ζώνη: ολόκληρο περίγραμμα στο χρώμα της */
.zones-card.selected {
    border: 2px solid var(--zone-color, #3498db);
}

.zones-card-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 10px;
}

.zones-card-name {
    margin: 0;
    color: #2c3e50;
    font-size: 1rem;
}

.zones-card-badge {
    background-color: var(--zone-tint, #eaf4fb);
    color: var(--zone-color, #3498db);
    padding: 2px 8px;
    border-radius: 10px;
    font-size: 0.75rem;
    font-weight: bold;
}

/* ===== FILL BAR (κοινό σε κάρτες & λίστα κάδων) ===== */
.zones-fill {
    margin-bottom: 8px;
}

.zones-fill-header {
    display: flex;
    justify-content: space-between;
    font-size: 0.75rem;
    color: #7f8c8d;
    margin-bottom: 3px;
```

```
}

.zones-fill-pct {
   font-weight: bold;
   color: var(--fill-color, #27ae60);
}

.zones-fill-bar-bg {
   background-color: #ecf0f1;
   border-radius: 10px;
   height: 6px;
}

.zones-fill-bar {
   height: 100%;
   border-radius: 10px;
   background-color: var(--fill-color, #27ae60);
   width: var(--fill-width, 0%);
   transition: width 0.3s;
}

/* mini stats μέσα στην κάρτα */
.zones-card-stats {
   display: grid;
   grid-template-columns: repeat(3, 1fr);
   gap: 6px;
   font-size: 0.75rem;
   text-align: center;
}

.zones-card-stat {
   border-radius: 6px;
   padding: 4px;
}

.zones-card-stat-value {
   font-weight: bold;
}

.zones-card-stat-label {
   color: #7f8c8d;
}

.zones-card-stat.critical {
   background-color: #fdecea;
}

.zones-card-stat.bins {
   background-color: #d5f5e3;
```

```
}

.zones-card-stat.vehicles {
    background-color: #ebf5fb;
}

.zones-card-stat.critical .zones-card-stat-value {
    color: #e74c3c;
}

.zones-card-stat.bins .zones-card-stat-value {
    color: #27ae60;
}

.zones-card-stat.vehicles .zones-card-stat-value {
    color: #3498db;
}

/* ===== DETAIL PANEL ===== */
.zones-detail-card {
    background-color: white;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    margin-bottom: 15px;
    border-top: 4px solid var(--zone-color, #3498db);
}

.zones-detail-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 15px;
}

.zones-detail-title {
    margin: 0;
    color: #2c3e50;
    display: flex;
    align-items: center;
    gap: 8px;
}

.zones-detail-marker {
    color: var(--zone-color, #3498db);
}

.zones-close-btn {
    background: none;
```

```
    border: none;
    cursor: pointer;
    font-size: 1.2rem;
    color: #7f8c8d;
    padding: 0;
}

.zones-detail-grid {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 10px;
    margin-bottom: 15px;
}

.zones-detail-stat {
    background-color: #f8f9fa;
    border-radius: 8px;
    padding: 10px;
    text-align: center;
}

.zones-detail-stat-value {
    font-size: 1.4rem;
    font-weight: bold;
    color: var(--stat-color, #3498db);
}

.zones-detail-stat-label {
    font-size: 0.75rem;
    color: #7f8c8d;
}

/* ===== BIN LIST ===== */
.zones-bins-panel {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    overflow: hidden;
}

.zones-bins-header {
    padding: 12px 20px;
    border-bottom: 1px solid #ecf0f1;
    display: flex;
    align-items: center;
    gap: 8px;
}

.zones-bins-icon {
```

```
    color: var(--zone-color, #3498db);
}

.zones-bins-title {
    margin: 0;
    color: #2c3e50;
}

.zones-bins-list {
    max-height: 400px;
    overflow-y: auto;
}

.zones-bin-row {
    padding: 10px 15px;
    border-bottom: 1px solid #f0f0f0;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

/* ζέβρα: γκρι φόντο στις μονές γραμμές (όπως το αρχικό idx % 2 === 0) */
.zones-bin-row:nth-child(odd) {
    background-color: #f8f9fa;
}

.zones-bin-id {
    font-weight: bold;
    font-size: 0.85rem;
    color: #3498db;
    display: flex;
    align-items: center;
}

.zones-bin-alert {
    color: #e74c3c;
    margin-right: 4px;
}

.zones-bin-location {
    font-size: 0.75rem;
    color: #7f8c8d;
}

.zones-bin-right {
    text-align: right;
}

.zones-bin-pct {
```

```
    font-weight: bold;
    font-size: 0.9rem;
    color: var(--fill-color, #27ae60);
}

.zones-bin-bar-bg {
    width: 60px;
    background-color: #ecf0f1;
    border-radius: 10px;
    height: 5px;
    margin-top: 3px;
}

.zones-bin-bar {
    height: 100%;
    border-radius: 10px;
    background-color: var(--fill-color, #27ae60);
    width: var(--fill-width, 0%);
}

/* ===== RESPONSIVE ===== */
@media (max-width: 1024px) {
    .zones-layout.with-detail {
        grid-template-columns: 1fr;
    }
}

@media (max-width: 768px) {
    .zones-summary-grid {
        grid-template-columns: repeat(2, 1fr);
    }

    .zones-cards-grid {
        grid-template-columns: 1fr;
    }

    .zones-header {
        flex-direction: column;
        align-items: flex-start;
        gap: 12px;
    }
}
```

Frontend\src\MapView.tsx

```
/**
 * MapView.tsx
 * ===========
 * Χάρτης (Leaflet) με τους κάδους ως χρωματιστά markers ανάλογα με την
 * πληρότητά τους. Προαιρετικά δέχεται focusPoint (π.χ. από αναφορά πολίτη)
 * για να κεντράρει αυτόματα σε ένα σημείο και να δείξει ειδικό pin.
```

```
 *
 * Styling: το χρώμα της μπάρας πληρότητας έρχεται από CSS class (fill-*),
 * ενώ το πλάτος της μπαίνει ως CSS variable μέσω ref (widthRef) — έτσι
 * αποφεύγουμε εντελώς τα inline styles στο JSX.
 */
import React, { useEffect } from 'react';
import { MapContainer, TileLayer, CircleMarker, Popup, Marker, useMap } from 'react-
leaflet';
import L from 'leaflet';
import 'leaflet/dist/leaflet.css';
import './MapView.css';

// ---------------------------------------------------------------------------
// Types
// ---------------------------------------------------------------------------

interface Bin {
  bin_id: string;
  location_name: string;
  address: string;
  latitude: number;
  longitude: number;
  bin_type: string;
  status: string;
  current_fill_level: number;
  current_battery_level: number;
}

interface Props {
  bins: Bin[];
  focusPoint?: { lat: number; lon: number } | null;
  onFocusConsumed?: () => void;
}

// ---------------------------------------------------------------------------
// Constants
// ---------------------------------------------------------------------------

const MAP_CENTER: [number, number] = [35.35, 25.07];

// Custom pin για τη θέση μιας αναφοράς πολίτη.
const reportIcon = L.divIcon({
  html: `<div class="map-report-pin" aria-label="Θέση αναφοράς πολίτη">
          <i class="fi fi-sr-map-marker"></i>
        </div>`,
  className: '',
  iconSize: [38, 38],
  iconAnchor: [19, 38],
});
```

```
// ---------------------------------------------------------------------------
// Helpers
// ---------------------------------------------------------------------------

/** Χρώμα marker ανάλογα με την πληρότητα: κόκκινο >=80, πορτοκαλί >=60, αλλιώς πράσινο. */
function getFillColor(level: number): string {
  if (level >= 80) return '#e74c3c';
  if (level >= 60) return '#f39c12';
  return '#27ae60';
}

/** CSS class με το χρώμα πληρότητας (για τη μπάρα στο popup). */
function fillClass(level: number): string {
  if (level >= 80) return 'fill-critical';
  if (level >= 60) return 'fill-warning';
  return 'fill-ok';
}

/** Ref callback που θέτει το πλάτος της μπάρας ως CSS variable — χωρίς inline style. */
function widthRef(level: number) {
  return (el: HTMLDivElement | null): void => {
    if (el) el.style.setProperty('--fill-width', `${level}%`);
  };
}

// ---------------------------------------------------------------------------
// Sub-components
// ---------------------------------------------------------------------------

/** Κεντράρει τον χάρτη στο focusPoint όταν αυτό αλλάξει. */
function MapFocusHandler({
  focusPoint,
  onFocusConsumed,
}: {
  focusPoint: { lat: number; lon: number } | null | undefined;
  onFocusConsumed?: () => void;
}) {
  const map = useMap();

  useEffect(() => {
    if (!focusPoint) return;
    map.setView([focusPoint.lat, focusPoint.lon], 17, { animate: true });
    onFocusConsumed?.();
  }, [focusPoint]);

  return null;
}
```

```
// ---------------------------------------------------------------------------
// Component
// ---------------------------------------------------------------------------

function MapView({ bins, focusPoint, onFocusConsumed }: Props) {
  return (
    <div className="mapview-container">
      <MapContainer center={MAP_CENTER} zoom={12} className="mapview-map">
        <TileLayer
          attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>'
          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
        />

        {/* Κεντράρισμα στην αναφορά πολίτη */}
        <MapFocusHandler focusPoint={focusPoint}
onFocusConsumed={onFocusConsumed} />

        {/* Marker της αναφοράς */}
        {focusPoint && (
          <Marker position={[focusPoint.lat, focusPoint.lon]} icon={reportIcon}>
            <Popup>
              <div className="mapview-popup">
                <h4>
                  <i className="fi fi-sr-comment icon-mr6" aria-hidden="true"></i>
                  Αναφορά Πολίτη
                </h4>
                <p>
                  <i className="fi fi-sr-marker icon-mr4" aria-hidden="true"></i>
                  {focusPoint.lat.toFixed(5)}, {focusPoint.lon.toFixed(5)}
                </p>
              </div>
            </Popup>
          </Marker>
        )}

        {/* Markers κάδων */}
        {bins.map(bin => (
          <CircleMarker
            key={bin.bin_id}
            center={[bin.latitude, bin.longitude]}
            radius={12}
            fillColor={getFillColor(bin.current_fill_level)}
            color="white"
            weight={2}
            fillOpacity={0.9}
          >
            <Popup>
```

```
          <div className="mapview-popup">
            <h3 className="mapview-popup-title">{bin.location_name}</h3>
            <p className="mapview-popup-row">
              <i className="fi fi-sr-marker icon-mr4" aria-hidden="true"></i>
              {bin.address}
            </p>
            <p className="mapview-popup-row">
              <i className="fi fi-sr-trash icon-mr4" aria-hidden="true"></i>
              {bin.bin_id}
            </p>

            <div
              className={`mapview-fill-bar-wrap ${fillClass(bin.current_fill_level)}`}
              ref={widthRef(bin.current_fill_level)}
            >
              <div className="mapview-fill-bar-header">
                <span>Πληρότητα</span>
                <span className="mapview-fill-pct">{bin.current_fill_level}%</span>
              </div>
              <div className="mapview-fill-bar-bg">
                <div className="mapview-fill-bar" />
              </div>
            </div>

            <p className="mapview-popup-row">
              <i className="fi fi-sr-battery-full icon-mr4" aria-hidden="true"></i>
              Battery: {bin.current_battery_level}%
            </p>
            <p className="mapview-popup-row">
              <i className="fi fi-sr-check-circle icon-mr4" aria-hidden="true"></i>
              {bin.status === 'active' ? 'Ενεργός' : 'Ανενεργός'}
            </p>
          </div>
        </Popup>
      </CircleMarker>
    ))}
  </MapContainer>

  {/* Legend */}
  <div className="mapview-legend" aria-label="Υπόμνημα χάρτη">
    <div className="mapview-legend-title">Υπόμνημα</div>
    <div className="mapview-legend-row">
      <div className="mapview-legend-dot mapview-legend-critical" aria-
hidden="true"></div>
      <span>Κρίσιμος (&gt;80%)</span>
    </div>
    <div className="mapview-legend-row">
      <div className="mapview-legend-dot mapview-legend-warning" aria-
hidden="true"></div>
```

```
        <span>Προσοχή (60-80%)</span>
       </div>
       <div className="mapview-legend-row">
        <div className="mapview-legend-dot mapview-legend-ok" aria-
hidden="true"></div>
        <span>Κανονικός (&lt;60%)</span>
       </div>
       {focusPoint && (
        <div className="mapview-legend-row">
         <div className="mapview-legend-report" aria-hidden="true">
          <i className="fi fi-sr-map-marker"></i>
         </div>
         <span>Αναφορά Πολίτη</span>
        </div>
       )}
     </div>
   </div>
  );
}

export default MapView;
```

Frontend\src\MapView.css

```
/* ===== MapView.css ===== */

.icon-mr6 {
  margin-right: 6px;
}

.icon-mr4 {
  margin-right: 4px;
}

/* Χρώμα πληρότητας ως CSS variable (το πλάτος ορίζεται από JS μέσω ref) */
.fill-ok {
  --fill-color: #27ae60;
}

.fill-warning {
  --fill-color: #f39c12;
}

.fill-critical {
  --fill-color: #e74c3c;
}

/* Container */
.mapview-container {
  position: relative;
```

```
  height: calc(100vh - 140px);
  width: 100%;
}

/* Map */
.mapview-map {
  height: 100%;
  width: 100%;
}

/* Report pin marker */
.map-report-pin {
  width: 38px;
  height: 38px;
  background: linear-gradient(135deg, #8e44ad, #9b59b6);
  border-radius: 50% 50% 50% 0;
  transform: rotate(-45deg);
  border: 3px solid white;
  box-shadow: 0 3px 12px rgba(142, 68, 173, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  animation: pinPulse 1.5s infinite;
}

.map-report-pin i {
  transform: rotate(45deg);
  color: white;
  font-size: 1rem;
}

@keyframes pinPulse {

  0%,
  100% {
    box-shadow: 0 3px 12px rgba(142, 68, 173, 0.5);
  }

  50% {
    box-shadow: 0 3px 20px rgba(142, 68, 173, 0.9);
  }
}

/* Popup */
.mapview-popup {
  min-width: 200px;
  font-family: 'Consolas', monospace;
}
```

```
.mapview-popup-title {
  margin: 0 0 8px 0;
  color: #2c3e50;
  font-size: 0.95rem;
}

.mapview-popup h4 {
  margin: 0 0 6px;
  color: #8e44ad;
  font-size: 0.9rem;
  display: flex;
  align-items: center;
}

.mapview-popup-row {
  margin: 4px 0;
  font-size: 0.85rem;
  color: #7f8c8d;
  display: flex;
  align-items: center;
}

/* Fill bar — χρώμα & πλάτος έρχονται ως CSS variables (δυναμικά, ανά κάδο) */
.mapview-fill-bar-wrap {
  margin: 10px 0;
}

.mapview-fill-bar-header {
  display: flex;
  justify-content: space-between;
  margin-bottom: 4px;
  font-size: 0.85rem;
}

.mapview-fill-pct {
  font-weight: bold;
  color: var(--fill-color, #27ae60);
}

.mapview-fill-bar-bg {
  background-color: #ecf0f1;
  border-radius: 10px;
  height: 8px;
}

.mapview-fill-bar {
  height: 100%;
  border-radius: 10px;
  transition: width 0.3s;
```

```
  background-color: var(--fill-color, #27ae60);
  width: var(--fill-width, 0%);
}

/* Legend */
.mapview-legend {
  position: absolute;
  bottom: 30px;
  right: 10px;
  background-color: white;
  padding: 12px 16px;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
  z-index: 1000;
  font-size: 0.85rem;
}

.mapview-legend-title {
  font-weight: bold;
  color: #2c3e50;
  margin-bottom: 10px;
}

.mapview-legend-row {
  display: flex;
  align-items: center;
  gap: 8px;
  margin-bottom: 6px;
}

.mapview-legend-dot {
  width: 14px;
  height: 14px;
  border-radius: 50%;
  flex-shrink: 0;
}

.mapview-legend-critical {
  background-color: #e74c3c;
}

.mapview-legend-warning {
  background-color: #f39c12;
}

.mapview-legend-ok {
  background-color: #27ae60;
}
```

```
.mapview-legend-report {
  width: 14px;
  height: 14px;
  display: flex;
  align-items: center;
  justify-content: center;
  color: #8e44ad;
  font-size: 0.9rem;
  flex-shrink: 0;
}
```

Frontend\src\DispatchView.tsx

```
/**
 * DispatchView.tsx
 * ================
 * Κέντρο διαχείρισης στόλου & ΧΥΤΑ. Περιλαμβάνει:
 *   - Διαχείριση βλάβης/συντήρησης οχήματος + αυτόματη εύρεση κοντινότερου
 *   - Καταγραφή μεταφοράς στο ΧΥΤΑ
 *   - Επαναφορά οχήματος σε ενεργό
 *   - Τρεις καρτέλες: χάρτης ζωνών, ειδοποιήσεις (με φίλτρα), ιστορικό ΧΥΤΑ
 *
 * Δομή αρχείου:
 *   Constants → Types → Helpers → υπο-components (ZoneSidebarCard,
DispatchMapTab,
 *   AlertCard, DispatchAlertsTab, DispatchXytaTab) → κύριο component DispatchView.
 * Το DispatchView κρατά το state, τα fetch/handlers και αποδίδει την ενεργή καρτέλα.
 *
 * Styling: όλα τα δυναμικά χρώματα ορίζονται από CSS classes — χωρίς inline styles.
 */
import React, { useState, useEffect } from 'react';
import { MapContainer, TileLayer, Marker, Popup, Polyline, Circle } from 'react-leaflet';
import L from 'leaflet';
import 'leaflet/dist/leaflet.css';
import './DispatchView.css';

// ---------------------------------------------------------------------------
// Constants
// ---------------------------------------------------------------------------

const API_URL = 'http://localhost:8000';
const REFRESH_INTERVAL_MS = 30000;

const XYTA = { lat: 35.39236790444294, lon: 24.921436396669584, name: 'ΧΥΤΑ Πέρα
Γαλήνων' };

// Γεωγραφικό κέντρο κάθε ζώνης (το χρώμα ορίζεται ως CSS class zone-XXX).
const ZONE_CENTERS: Record<string, [number, number]> = {
  GAZ: [35.3259, 25.0659], AMM: [35.3370, 25.0800],
  APE: [35.4050, 25.0185], TYL: [35.2963, 25.0184],
```

```
  KRO: [35.2300, 24.9815], ROD: [35.3596, 25.0278],
};

// Χρώμα ζώνης — χρειάζεται ως prop στα Leaflet paths (Circle/Polyline).
// Για το υπόλοιπο styling χρησιμοποιούμε τις CSS classes zone-XXX.
const ZONE_COLORS: Record<string, string> = {
  GAZ: '#3498db', AMM: '#27ae60', APE: '#e74c3c',
  TYL: '#f39c12', KRO: '#9b59b6', ROD: '#1abc9c',
};
const DEFAULT_ZONE_COLOR = '#3498db';

// ---------------------------------------------------------------------
// Types
// ---------------------------------------------------------------------

interface Vehicle {
  vehicle_id: string;
  plate_number?: string;
  driver_name?: string;
  vehicle_type?: string;
  capacity_tons?: number;
  capacity_m3?: number;
  status: string;
  fuel_level?: number;
  latitude?: string;
  longitude?: string;
  zone_code?: string;
  zone_name?: string;
}

interface Zone {
  zone_code: string;
  zone_name: string;
  active_vehicles?: number;
  total_vehicles?: number;
  vehicles?: Vehicle[];
}

interface DispatchData {
  zones?: Zone[];
  xyta?: { distances_from_zones?: Record<string, number> };
}

interface Alert {
  id: number;
  initiated_by: string;
  alert_type: string;
  vehicle_id: string;
  zone_code?: string;
```

```
  status: string;
  created_at: string;
  message?: string;
  collected_bins_count?: number;
  fill_level_pct?: number;
  driver_name?: string;
  nearest_vehicle_id?: string;
  notes?: string;
}

interface XytaTrip {
  id: number;
  vehicle_id: string;
  driver_name?: string;
  departure_zone: string;
  reason: string;
  collected_bins_count?: number;
  vehicle_fill_pct?: string;
  status: string;
  departed_at: string;
}

interface BreakdownResult {
  breakdown_vehicle?: { vehicle_id?: string; driver_name?: string; zone?: string };
  nearest_vehicle?: { vehicle_id?: string; driver_name?: string; zone_name?: string;
distance_km?: number };
  xyta_route?: { distance_km?: number; estimated_minutes?: number };
  xyta?: { distance_km?: number };
}

type MessageType = 'success' | 'error';
type AlertFilter = 'all' | 'driver' | 'system';

// ---------------------------------------------------------------------------
// Helpers
// ---------------------------------------------------------------------------

/** Κλειδί κατάστασης οχήματος για CSS classes (active / maintenance / inactive). */
function statusKey(status: string): 'active' | 'maintenance' | 'inactive' {
  if (status === 'active') return 'active';
  if (status === 'maintenance') return 'maintenance';
  return 'inactive';
}

/** Εικονίδιο ανάλογα με την κατάσταση οχήματος. */
function statusIcon(status: string): string {
  if (status === 'active') return 'fi-sr-check-circle';
  if (status === 'maintenance') return 'fi-sr-tools';
  return 'fi-sr-cross-circle';
```

```
}

/** Χρώμα αριστερού accent της κάρτας ειδοποίησης ανάλογα με προέλευση/τύπο. */
function alertBorderClass(alert: Alert): string {
  if (alert.initiated_by === 'driver') return 'alert-border-driver';
  if (alert.alert_type === 'breakdown') return 'alert-border-red';
  if (alert.alert_type === 'maintenance') return 'alert-border-orange';
  return 'alert-border-green';
}

interface ActionLabelInfo {
  icon: string;
  label: string;
}

/** Εικονίδιο + ετικέτα ανά τύπο ενέργειας (αντικαθιστά τα παλιά emoji-strings). */
function getActionLabel(type: string): ActionLabelInfo {
  const labels: Record<string, ActionLabelInfo> = {
    bin_collected: { icon: 'fi-sr-check-circle', label: 'Αποκομιδή Κάδου' },
    vehicle_full: { icon: 'fi-sr-trash', label: 'Όχημα Γεμάτο' },
    vehicle_restored: { icon: 'fi-sr-check-circle', label: 'Επαναφορά σε Ενεργό' },
    breakdown: { icon: 'fi-sr-tools', label: 'Βλάβη' },
    maintenance: { icon: 'fi-sr-tools', label: 'Συντήρηση' },
    xyta_arrived: { icon: 'fi-sr-road', label: 'Άφιξη ΧΥΤΑ' },
    xyta_trip: { icon: 'fi-sr-truck-side', label: 'Μεταφορά ΧΥΤΑ' },
    zone_change: { icon: 'fi-sr-marker', label: 'Αλλαγή Ζώνης' },
    zone_change_request: { icon: 'fi-sr-marker', label: 'Αίτημα Ζώνης' },
    full: { icon: 'fi-sr-trash', label: 'Γεμάτο' },
  };
  return labels[type] ?? { icon: 'fi-sr-info', label: type };
}

/** Μικρό presentational component: εικονίδιο + ετικέτα ενέργειας. */
function ActionLabel({ type }: { type: string }) {
  const { icon, label } = getActionLabel(type);
  return (
    <>
      <i className={`fi ${icon}`} aria-hidden="true"></i> {label}
    </>
  );
}

//
================================================================
===========
// Υπο-components καρτελών
//
================================================================
===========
```

```
// --- Κάρτα ζώνης στο sidebar του χάρτη ------------------------------------

interface ZoneSidebarCardProps {
  zone: Zone;
  showXyta: boolean;
  xytaDistanceKm?: number;
  onSelectVehicle: (vehicleId: string) => void;
}

function ZoneSidebarCard({ zone, showXyta, xytaDistanceKm, onSelectVehicle }:
ZoneSidebarCardProps) {
  return (
    <div className={`dispatch-zone-card zone-${zone.zone_code}`}>
      <div className="dispatch-zone-header">
        <h4 className="dispatch-zone-title">{zone.zone_name}</h4>
        <span className="dispatch-zone-badge">
          {zone.active_vehicles}/{zone.total_vehicles} ενεργά
        </span>
      </div>
      <div className="dispatch-vehicle-chips">
        {zone.vehicles?.map(v => (
          <div
            key={v.vehicle_id}
            title={`${v.driver_name} | ${v.status}`}
            className={`dispatch-vehicle-chip veh-${statusKey(v.status)}`}
            onClick={() => onSelectVehicle(v.vehicle_id)}
            role="button"
            tabIndex={0}
            aria-label={`Επιλογή ${v.vehicle_id}`}
            onKeyDown={e => { if (e.key === 'Enter') onSelectVehicle(v.vehicle_id); }}
          >
            <i className={`fi ${statusIcon(v.status)}`} aria-hidden="true"></i>
            {v.vehicle_id}
          </div>
        ))}
      </div>
      {showXyta && (
        <div className="dispatch-zone-xyta">
          <i className="fi fi-sr-road" aria-hidden="true"></i>
          XYTA: {xytaDistanceKm} km
        </div>
      )}
    </div>
  );
}

// --- Καρτέλα χάρτη (sidebar ζωνών + Leaflet map) --------------------------
```

```
interface DispatchMapTabProps {
  dispatchData: DispatchData | null;
  onSelectVehicle: (vehicleId: string) => void;
  onXytaTrip: (vehicleId: string) => void;
}

function DispatchMapTab({ dispatchData, onSelectVehicle, onXytaTrip }:
DispatchMapTabProps) {
  const xytaDistances = dispatchData?.xyta?.distances_from_zones;

  return (
    <div className="dispatch-map-layout">
      <div className="dispatch-zone-sidebar" aria-label="Ζώνες">
        {dispatchData?.zones?.map(zone => (
          <ZoneSidebarCard
            key={zone.zone_code}
            zone={zone}
            showXyta={!!xytaDistances}
            xytaDistanceKm={xytaDistances?.[zone.zone_code]}
            onSelectVehicle={onSelectVehicle}
          />
        ))}
      </div>

      <div className="dispatch-map-wrap">
        <MapContainer center={[35.33, 25.00]} zoom={10} className="dispatch-map">
          <TileLayer
            attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>'
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          />
          <Marker
            position={[XYTA.lat, XYTA.lon]}
            icon={L.divIcon({
              html: `<div class="dispatch-xyta-marker">XYTA Πέρα Γαλήνων</div>`,
              className: '',
              iconSize: [170, 34],
              iconAnchor: [85, 17],
            })}
          >
            <Popup>
              <div className="dispatch-popup">
                <h4 className="dispatch-popup-xyta-title"><i className="fi fi-sr-road icon-
mr6" aria-hidden="true"></i>{XYTA.name}</h4>
                <p className="dispatch-popup-sub">Χώρος Υγειονομικής Ταφής</p>
                <p className="dispatch-popup-coords">{XYTA.lat.toFixed(5)},
{XYTA.lon.toFixed(5)}</p>
              </div>
            </Popup>
```

```
        </Marker>

        {dispatchData?.zones?.map(zone => {
          const center = ZONE_CENTERS[zone.zone_code];
          if (!center) return null;
          const color = ZONE_COLORS[zone.zone_code] || DEFAULT_ZONE_COLOR;
          return (
            <React.Fragment key={zone.zone_code}>
              <Circle center={center} radius={2500} color={color} fillColor={color}
fillOpacity={0.08} weight={2} />
              {zone.vehicles?.map(v => {
                // Αν λείπουν συντεταγμένες, μικρή τυχαία μετατόπιση γύρω από το κέντρο
της ζώνης.
                const vLat = v.latitude ? parseFloat(v.latitude) : center[0] + (Math.random() -
0.5) * 0.01;
                const vLon = v.longitude ? parseFloat(v.longitude) : center[1] +
(Math.random() - 0.5) * 0.01;
                return (
                  <Marker
                    key={v.vehicle_id}
                    position={[vLat, vLon]}
                    icon={L.divIcon({
                      html: `<div class="dispatch-vehicle-marker veh-marker-
${statusKey(v.status)}"><i class="fi fi-sr-truck-side"></i></div>`,
                      className: '',
                      iconSize: [32, 32],
                      iconAnchor: [16, 16],
                    })}
                  >
                    <Popup>
                      <div className="dispatch-popup">
                        <h4 className="dispatch-popup-title"><i className="fi fi-sr-truck-side
icon-mr6" aria-hidden="true"></i>{v.vehicle_id}</h4>
                        <table className="dispatch-popup-table">
                          <tbody>
                            {([
                              ['Πινακίδα', v.plate_number],
                              ['Οδηγός', v.driver_name],
                              ['Τύπος', v.vehicle_type],
                              ['Χωρητικότητα', `${v.capacity_tons}τ / ${v.capacity_m3}m³`],
                              ['Κατάσταση', <><i className={`fi ${statusIcon(v.status)} ${v.status
=== 'active' ? 'icon-green' : v.status === 'maintenance' ? 'icon-orange' : 'icon-red'}`} aria-
hidden="true"></i> {v.status === 'active' ? 'Ενεργό' : v.status === 'maintenance' ?
'Συντήρηση' : 'Ανενεργό'}</>],
                              ['Καύσιμα', `${v.fuel_level}%`],
                            ] as [string, React.ReactNode][]).map(([lbl, val]) => (
                              <tr key={lbl}><td>{lbl}</td><td>{val}</td></tr>
                            ))}
                          </tbody>
```

```
                    </table>
                    <button
                      onClick={() => onXytaTrip(v.vehicle_id)}
                      className="dispatch-popup-xyta-btn"
                      aria-label={`${v.vehicle_id} προς ΧΥΤΑ`}
                    >
                      <i className="fi fi-sr-road icon-mr4" aria-hidden="true"></i> Προς
ΧΥΤΑ
                    </button>
                  </div>
                </Popup>
              </Marker>
            );
          })}
          <Polyline positions={[center, [XYTA.lat, XYTA.lon]]} color={color}
weight={1} dashArray="5,10" opacity={0.4} />
        </React.Fragment>
      );
    })}
  </MapContainer>
 </div>
</div>
);
}

// --- Κάρτα μεμονωμένης ειδοποίησης ----------------------------------------

interface AlertCardProps {
 alert: Alert;
 onAcknowledge: (alertId: number) => void;
}

function AlertCard({ alert, onAcknowledge }: AlertCardProps) {
 const isDriver = alert.initiated_by === 'driver';
 const bins = alert.collected_bins_count ?? 0;
 const fill = alert.fill_level_pct ?? 0;

 return (
  <div className={`dispatch-alert-card ${alertBorderClass(alert)}`}>
   <div className="dispatch-alert-inner">
    <div className="dispatch-alert-body">
     <div className="dispatch-alert-badges">
      <span className={`dispatch-badge ${isDriver ? 'dispatch-badge-from-driver' :
'dispatch-badge-from-system'}`}>
       <i className={`fi ${isDriver ? 'fi-sr-user' : 'fi-sr-settings'}`} aria-
hidden="true"></i>
       {isDriver ? 'Από Οδηγό' : 'Από Σύστημα'}
      </span>
```

```
          <span className="dispatch-badge dispatch-badge-action-type"><ActionLabel
type={alert.alert_type} /></span>
          <span className="dispatch-badge-vehicle">
            <i className="fi fi-sr-truck-side" aria-hidden="true"></i> {alert.vehicle_id}
          </span>
          {alert.zone_code && (
            <span className={`dispatch-badge-zone dispatch-badge zone-
${alert.zone_code}`}>
              {alert.zone_code}
            </span>
          )}
          <span className={`dispatch-badge ${alert.status === 'active' ? 'dispatch-badge-
active' : 'dispatch-badge-acknowledged'}`}>
            <i className={`fi ${alert.status === 'active' ? 'fi-sr-triangle-warning' : 'fi-sr-
check-circle'}`} aria-hidden="true"></i>
            {alert.status === 'active' ? 'Ενεργή' : 'Επιβεβαιωμένη'}
          </span>
          <span className="dispatch-badge-time">
            <i className="fi fi-sr-calendar" aria-hidden="true"></i>
            {new Date(alert.created_at).toLocaleString('el-GR')}
          </span>
        </div>
        <p className="dispatch-alert-message">{alert.message}</p>
        {isDriver && (bins > 0 || fill > 0) && (
          <div className="dispatch-alert-stats">
            {bins > 0 && (
              <span className="dispatch-alert-stat-bins">
                <i className="fi fi-sr-trash" aria-hidden="true"></i> {bins} κάδοι
              </span>
            )}
            {fill > 0 && (
              <span className={`dispatch-alert-stat-fill ${fill >= 85 ? 'high' : 'med'}`}>
                <i className="fi fi-sr-truck-side" aria-hidden="true"></i> {fill.toFixed(1)}%
γεμάτο
              </span>
            )}
            {alert.driver_name && (
              <span className="dispatch-alert-stat-driver">
                <i className="fi fi-sr-user" aria-hidden="true"></i> {alert.driver_name}
              </span>
            )}
          </div>
        )}
        {alert.nearest_vehicle_id && (
          <p className="dispatch-alert-nearest">
            <i className="fi fi-sr-truck-side" aria-hidden="true"></i> Εναλλακτικό:
{alert.nearest_vehicle_id}
          </p>
        )}
```

```
        {alert.notes && (
          <p className="dispatch-alert-notes">
            <i className="fi fi-sr-document icon-mr4" aria-hidden="true"></i>{alert.notes}
          </p>
        )}
      </div>
      {alert.status === 'active' && (
        <button
          onClick={() => onAcknowledge(alert.id)}
          className="dispatch-acknowledge-btn"
          aria-label={`Επιβεβαίωση #${alert.id}`}
        >
          <i className="fi fi-sr-check-circle" aria-hidden="true"></i> Επιβεβαίωση
        </button>
      )}
    </div>
  </div>
 );
}

// --- Καρτέλα ειδοποιήσεων (φίλτρα + λίστα) ---------------------------------

interface DispatchAlertsTabProps {
 alerts: Alert[];
 alertFilter: AlertFilter;
 driverAlerts: number;
 systemAlerts: number;
 onFilterChange: (filter: AlertFilter) => void;
 onAcknowledge: (alertId: number) => void;
}

function DispatchAlertsTab({ alerts, alertFilter, driverAlerts, systemAlerts,
onFilterChange, onAcknowledge }: DispatchAlertsTabProps) {
 const filteredAlerts = alertFilter === 'all' ? alerts : alerts.filter(a => a.initiated_by ===
alertFilter);

 const filters: { id: AlertFilter; label: string }[] = [
  { id: 'all', label: `Όλες (${alerts.length})` },
  { id: 'driver', label: `Από Οδηγούς (${driverAlerts})` },
  { id: 'system', label: `Από Σύστημα (${systemAlerts})` },
 ];

 return (
  <div>
   <div className="dispatch-alert-filters" role="group" aria-label="Φίλτρα">
    <span className="dispatch-alert-filter-label">
     <i className="fi fi-sr-filter" aria-hidden="true"></i> Φίλτρο:
    </span>
    {filters.map(f => (
```

```
        <button
          key={f.id}
          onClick={() => onFilterChange(f.id)}
          className={`dispatch-filter-btn filter-${f.id} ${alertFilter === f.id ? 'active' : ''}`}
        >
          {f.label}
        </button>
      ))}
    </div>

    {filteredAlerts.length === 0 ? (
      <div className="dispatch-empty" role="status">
        <i className="fi fi-sr-check-circle dispatch-empty-icon" aria-hidden="true"></i>
        <h3 className="dispatch-empty-title">Δεν υπάρχουν ειδοποιήσεις!</h3>
      </div>
    ) : (
      <div className="dispatch-alerts-list">
        {filteredAlerts.map(alert => (
          <AlertCard key={alert.id} alert={alert} onAcknowledge={onAcknowledge} />
        ))}
      </div>
    )}
  </div>
 );
}

// --- Καρτέλα ιστορικού ΧΥΤΑ (πίνακας) ------------------------------------

interface DispatchXytaTabProps {
 xytaTrips: XytaTrip[];
}

function DispatchXytaTab({ xytaTrips }: DispatchXytaTabProps) {
 const columns = ['#', 'Όχημα', 'Οδηγός', 'Ζώνη', 'Λόγος', 'Κάδοι', 'Πληρότητα',
'Κατάσταση', 'Αναχώρηση'];

 return (
  <div className="dispatch-xyta-panel">
   <div className="dispatch-xyta-header">
    <i className="fi fi-sr-road icon-purple" aria-hidden="true"></i>
    <h3 className="dispatch-xyta-title">Ιστορικό Μεταφορών στο ΧΥΤΑ
({xytaTrips.length})</h3>
   </div>
   {xytaTrips.length === 0 ? (
    <div className="dispatch-xyta-empty" role="status">
     <i className="fi fi-sr-road dispatch-xyta-empty-icon" aria-hidden="true"></i>
     <p>Δεν υπάρχουν καταγεγραμμένες μεταφορές</p>
    </div>
   ) : (
```

```
      <div className="dispatch-xyta-table-wrap">
        <table className="dispatch-xyta-table" aria-label="Μεταφορές XYTA">
          <thead>
            <tr>{columns.map(h => <th key={h}>{h}</th>)}</tr>
          </thead>
          <tbody>
            {xytaTrips.map(trip => (
              <tr key={trip.id}>
                <td className="dispatch-xyta-id">#{trip.id}</td>
                <td className="dispatch-xyta-vehicle">{trip.vehicle_id}</td>
                <td className="dispatch-xyta-driver"><i className="fi fi-sr-user icon-gray"
aria-hidden="true"></i>{trip.driver_name}</td>
                <td><span className={`dispatch-xyta-zone-badge zone-
${trip.departure_zone}`}>{trip.departure_zone}</span></td>
                <td><ActionLabel type={trip.reason} /></td>
                <td><strong className="icon-blue">{trip.collected_bins_count ||
0}</strong><span className="icon-gray"> κάδοι</span></td>
                <td>{trip.vehicle_fill_pct ? <span
className={parseFloat(trip.vehicle_fill_pct) >= 85 ? 'dispatch-xyta-fill-high' : 'dispatch-
xyta-fill-med'}>{parseFloat(trip.vehicle_fill_pct).toFixed(1)}%</span> : '—'}</td>
                <td><span className={`dispatch-xyta-status-badge ${trip.status ===
'en_route' ? 'dispatch-xyta-status-enroute' : 'dispatch-xyta-status-arrived'}`}><i
className={`fi ${trip.status === 'en_route' ? 'fi-sr-road' : 'fi-sr-check-circle'}`} aria-
hidden="true"></i>{trip.status === 'en_route' ? 'Εν πλω' : 'Έφτασε'}</span></td>
                <td className="dispatch-xyta-time"><i className="fi fi-sr-calendar icon-
mr4" aria-hidden="true"></i>{new Date(trip.departed_at).toLocaleString('el-GR')}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}
  </div>
 );
}

//
============================================================================
===========
// Κύριο component
//
============================================================================
===========

function DispatchView() {
 const [dispatchData, setDispatchData] = useState<DispatchData | null>(null);
 const [alerts, setAlerts] = useState<Alert[]>([]);
 const [xytaTrips, setXytaTrips] = useState<XytaTrip[]>([]);
 const [loading, setLoading] = useState(true);
```

```
  const [activeTab, setActiveTab] = useState<'map' | 'alerts' | 'xyta'>('map');
  const [selectedVehicle, setSelectedVehicle] = useState<string>('');
  const [breakdownReason, setBreakdownReason] = useState<string>('breakdown');
  const [processing, setProcessing] = useState(false);
  const [message, setMessage] = useState('');
  const [messageType, setMessageType] = useState<MessageType>('success');
  const [breakdownResult, setBreakdownResult] = useState<BreakdownResult |
null>(null);
  const [alertFilter, setAlertFilter] = useState<AlertFilter>('all');

  useEffect(() => {
    fetchAll();
    const interval = setInterval(fetchAll, REFRESH_INTERVAL_MS);
    return () => clearInterval(interval);
  }, []);

  // ---------------------------------------------------------------------
  // API calls
  // ---------------------------------------------------------------------

  const fetchAll = async (): Promise<void> => {
    try {
      const [dispRes, alertRes, xytaRes] = await Promise.all([
        fetch(`${API_URL}/api/v1/dispatch/status`),
        fetch(`${API_URL}/api/v1/dispatch/alerts`),
        fetch(`${API_URL}/api/v1/dispatch/xyta-trips`),
      ]);
      const [disp, alert, xyta] = await Promise.all([dispRes.json(), alertRes.json(),
xytaRes.json()]);
      setDispatchData(disp);
      setAlerts(alert.alerts ?? []);
      setXytaTrips(xyta.trips ?? []);
    } catch (error) {
      console.error('Failed to fetch dispatch data:', error);
    } finally {
      setLoading(false);
    }
  };

  const handleBreakdown = async (): Promise<void> => {
    if (!selectedVehicle) { showMessage('Επιλέξτε όχημα!', 'error'); return; }
    setProcessing(true);
    try {
      const res = await fetch(`${API_URL}/api/v1/dispatch/vehicle-breakdown`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ vehicle_id: selectedVehicle, reason: breakdownReason }),
      });
      const json = await res.json();
```

```
      if (res.ok) {
        setBreakdownResult(json);
        showMessage(`Ειδοποίηση στάλθηκε! Κοντινότερο:
${json.nearest_vehicle?.vehicle_id || '—'}`, 'success');
        fetchAll();
      } else {
        showMessage(json.detail ?? 'Σφάλμα', 'error');
      }
    } catch (error) {
      console.error('Breakdown request failed:', error);
      showMessage('Σφάλμα σύνδεσης', 'error');
    }
    setProcessing(false);
  };

  const handleXytaTrip = async (vehicleId: string): Promise<void> => {
    setProcessing(true);
    try {
      const res = await fetch(`${API_URL}/api/v1/dispatch/xyta-trip`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ vehicle_id: vehicleId, reason: 'full' }),
      });
      const json = await res.json();
      if (res.ok) {
        showMessage(`Καταγράφηκε μεταφορά στο XYTA (${json.xyta?.distance_km}
km)`, 'success');
        fetchAll();
      }
    } catch (error) {
      console.error('XYTA trip request failed:', error);
      showMessage('Σφάλμα', 'error');
    }
    setProcessing(false);
  };

  const handleRestoreVehicle = async (): Promise<void> => {
    if (!selectedVehicle) { showMessage('Επιλέξτε όχημα!', 'error'); return; }
    const vehicle = allVehicles.find(v => v.vehicle_id === selectedVehicle);
    if (vehicle?.status === 'active') { showMessage('Το όχημα είναι ήδη ενεργό!', 'error');
return; }
    setProcessing(true);
    try {
      const res = await fetch(`${API_URL}/api/v1/vehicles/${selectedVehicle}`, {
        method: 'PUT',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ status: 'active' }),
      });
      if (res.ok) {
```

```
      // Καταγράφουμε και την ενέργεια στο ιστορικό του dispatcher
      await fetch(`${API_URL}/api/v1/dispatch/driver-action`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          vehicle_id: selectedVehicle,
          action_type: 'vehicle_restored',
          driver_name: vehicle?.driver_name || '',
          zone_code: vehicle?.zone_code || '',
          notes: 'Επαναφορά οχήματος σε ενεργό από διαχειριστή',
        }),
      });
      showMessage(`Το όχημα ${selectedVehicle} επαναφέρθηκε σε ενεργό!`, 'success');
      setBreakdownResult(null);
      fetchAll();
    } else {
      const json = await res.json();
      showMessage(json.detail || 'Σφάλμα επαναφοράς', 'error');
    }
  } catch (error) {
    console.error('Restore request failed:', error);
    showMessage('Σφάλμα σύνδεσης', 'error');
  }
  setProcessing(false);
 };

 const handleAcknowledge = async (alertId: number): Promise<void> => {
  await fetch(`${API_URL}/api/v1/dispatch/alerts/${alertId}/acknowledge`, { method:
'PUT' });
  fetchAll();
 };

 const showMessage = (msg: string, type: MessageType): void => {
  setMessage(msg);
  setMessageType(type);
  setTimeout(() => setMessage(''), 5000);
 };

 // ---------------------------------------------------------------------
 // Derived data
 // ---------------------------------------------------------------------

 const allVehicles: Vehicle[] = dispatchData?.zones?.flatMap(z =>
  (z.vehicles ?? []).map(v => ({ ...v, zone_name: z.zone_name, zone_code: z.zone_code
}))
 ) ?? [];

 const activeAlerts = alerts.filter(a => a.status === 'active').length;
 const driverAlerts = alerts.filter(a => a.initiated_by === 'driver').length;
```

```
  const systemAlerts = alerts.filter(a => a.initiated_by === 'system').length;
  const selectedStatus = allVehicles.find(v => v.vehicle_id === selectedVehicle)?.status;
  const isAlreadyActive = selectedStatus === 'active';

  const statCards = [
    { label: 'Σύνολο Ειδοποιήσεων', value: alerts.length, cls: 'stat-color-blue', icon: 'fi-sr-
bell' },
    { label: 'Ενεργές', value: activeAlerts, cls: 'stat-color-red', icon: 'fi-sr-triangle-warning'
},
    { label: 'Από Οδηγούς', value: driverAlerts, cls: 'stat-color-navy', icon: 'fi-sr-user' },
    { label: 'Μεταφορές XYTA', value: xytaTrips.length, cls: 'stat-color-purple', icon: 'fi-sr-
road' },
  ];

  const tabs = [
    { id: 'map' as const, icon: 'fi-sr-map', label: 'Χάρτης Ζωνών' },
    { id: 'alerts' as const, icon: 'fi-sr-bell', label: `Ειδοποιήσεις (${alerts.length})` },
    { id: 'xyta' as const, icon: 'fi-sr-road', label: `XYTA Μεταφορές (${xytaTrips.length})`
},
  ];

  // ---------------------------------------------------------------------
  // Render
  // ---------------------------------------------------------------------

  return (
    <div className="dispatch-container">

      {/* Header */}
      <div className="dispatch-header">
        <h2 className="dispatch-title">
          <i className="fi fi-sr-settings-sliders" aria-hidden="true"></i>
          Διαχείριση Οχημάτων &amp;amp; XYTA
        </h2>
        <div className="dispatch-header-badges">
          {activeAlerts > 0 && (
            <span className="dispatch-badge-alert">
              <i className="fi fi-sr-triangle-warning" aria-hidden="true"></i>
              {activeAlerts} ενεργές
            </span>
          )}
          {driverAlerts > 0 && (
            <span className="dispatch-badge-driver">
              <i className="fi fi-sr-user" aria-hidden="true"></i>
              {driverAlerts} από οδηγούς
            </span>
          )}
        </div>
      </div>
```

```
    {/* Message */}
    {message && (
     <div
      className={`dispatch-message ${messageType === 'success' ? 'dispatch-message-
success' : 'dispatch-message-error'}`}
      role="alert"
      aria-live="polite"
     >
      <i className={`fi ${messageType === 'success' ? 'fi-sr-check-circle' : 'fi-sr-cross-
circle'}`} aria-hidden="true"></i>
       {message}
     </div>
    )}

    {/* Stats */}
    <div className="dispatch-stats-grid" aria-label="Στατιστικά">
     {statCards.map(s => (
      <div key={s.label} className={`dispatch-stat-card ${s.cls}`}>
       <i className={`fi ${s.icon} dispatch-stat-icon`} aria-hidden="true"></i>
       <div className="dispatch-stat-value">{s.value}</div>
       <div className="dispatch-stat-label">{s.label}</div>
      </div>
     ))}
    </div>

    {/* Breakdown Panel */}
    <div className="dispatch-breakdown-panel">
     <h3 className="dispatch-breakdown-title">
      <i className="fi fi-sr-tools icon-red" aria-hidden="true"></i>
      Διαχείριση Βλάβης / Εκτός Λειτουργίας
     </h3>
     <div className="dispatch-breakdown-grid">
      <div>
       <label htmlFor="vehicle-select" className="dispatch-control-label">
        <i className="fi fi-sr-truck-side" aria-hidden="true"></i> Επιλογή Οχήματος
       </label>
       <select
        id="vehicle-select"
        value={selectedVehicle}
        onChange={e => setSelectedVehicle(e.target.value)}
        className="dispatch-select"
        aria-label="Επιλογή οχήματος"
        title="Επιλέξτε όχημα"
       >
        <option value="">— Επιλέξτε όχημα —</option>
        {allVehicles.map(v => (
         <option key={v.vehicle_id} value={v.vehicle_id}>
```

```
              {v.vehicle_id} | {v.plate_number} — {v.driver_name} ({v.zone_name})
[{v.status}]
            </option>
          ))}
        </select>
      </div>
      <div>
        <label htmlFor="reason-select" className="dispatch-control-label">
          <i className="fi fi-sr-triangle-warning" aria-hidden="true"></i> Λόγος
        </label>
        <select
          id="reason-select"
          value={breakdownReason}
          onChange={e => setBreakdownReason(e.target.value)}
          className="dispatch-select"
          aria-label="Λόγος βλάβης"
          title="Επιλέξτε λόγο"
        >
          <option value="breakdown">Βλάβη</option>
          <option value="maintenance">Συντήρηση</option>
          <option value="full">Γεμάτο — ΧΥΤΑ</option>
        </select>
      </div>
      <button
        onClick={handleBreakdown}
        disabled={processing}
        className="dispatch-btn dispatch-btn-red"
        aria-label="Ειδοποίηση βλάβης"
        title="Αποστολή ειδοποίησης"
      >
        <i className="fi fi-sr-bell" aria-hidden="true"></i> Ειδοποίηση
      </button>
      <button
        onClick={() => selectedVehicle && handleXytaTrip(selectedVehicle)}
        disabled={processing || !selectedVehicle}
        className="dispatch-btn dispatch-btn-purple"
        aria-label="Μεταφορά προς ΧΥΤΑ"
        title="Προς ΧΥΤΑ"
      >
        <i className="fi fi-sr-road" aria-hidden="true"></i> Προς ΧΥΤΑ
      </button>
    </div>

    {/* Κουμπί Επαναφοράς — εκτός grid */}
    <button
      onClick={handleRestoreVehicle}
      disabled={processing || !selectedVehicle || isAlreadyActive}
      className={`dispatch-restore-btn ${isAlreadyActive || !selectedVehicle ? 'active-
vehicle' : 'inactive-vehicle'}`}
```

```
        aria-label="Επαναφορά οχήματος σε ενεργό"
        title={isAlreadyActive ? 'Το όχημα είναι ήδη ενεργό' : 'Επαναφορά σε ενεργό μετά από βλάβη/συντήρηση'}
      >
        <i className="fi fi-sr-check-circle" aria-hidden="true"></i>
        Επαναφορά σε Ενεργό
      </button>

      {/* Breakdown Result */}
      {breakdownResult && (
        <div className="dispatch-breakdown-result">
          <div className="dispatch-result-grid">
            <div className="dispatch-result-card dispatch-result-card-red">
              <div className="dispatch-result-card-label">
                <i className="fi fi-sr-cross-circle icon-red" aria-hidden="true"></i> Βλάβη
              </div>
              <div className="dispatch-result-card-title">{breakdownResult.breakdown_vehicle?.vehicle_id}</div>
              <div className="dispatch-result-card-sub">{breakdownResult.breakdown_vehicle?.driver_name}</div>
              <div className="dispatch-result-card-detail-red">{breakdownResult.breakdown_vehicle?.zone}</div>
            </div>
            {breakdownResult.nearest_vehicle && (
              <div className="dispatch-result-card dispatch-result-card-green">
                <div className="dispatch-result-card-label">
                  <i className="fi fi-sr-truck-side icon-green" aria-hidden="true"></i> Κοντινότερο
                </div>
                <div className="dispatch-result-card-title">{breakdownResult.nearest_vehicle.vehicle_id}</div>
                <div className="dispatch-result-card-sub">{breakdownResult.nearest_vehicle.driver_name}</div>
                <div className="dispatch-result-card-detail-green">
                  {breakdownResult.nearest_vehicle.zone_name} — {breakdownResult.nearest_vehicle.distance_km} km
                </div>
              </div>
            )}
            <div className="dispatch-result-card dispatch-result-card-blue">
              <div className="dispatch-result-card-label">
                <i className="fi fi-sr-road icon-blue" aria-hidden="true"></i> ΧΥΤΑ
              </div>
              <div className="dispatch-result-card-title">{XYTA.name}</div>
              <div className="dispatch-result-card-detail-blue">{breakdownResult.xyta_route?.distance_km} km</div>
              <div className="dispatch-result-card-detail-gray">~{breakdownResult.xyta_route?.estimated_minutes} λεπτά</div>
            </div>
```

```
          </div>
        </div>
      )}
    </div>

    {/* Tabs */}
    <div className="dispatch-tabs" aria-label="Καρτέλες προβολής">
      {tabs.map(tab => (
        <button
          key={tab.id}
          onClick={() => setActiveTab(tab.id)}
          className={`dispatch-tab-btn ${activeTab === tab.id ? 'active' : 'inactive'}`}
        >
          <i className={`fi ${tab.icon}`} aria-hidden="true"></i>
          {tab.label}
        </button>
      ))}
    </div>

    {/* Ενεργή καρτέλα */}
    {loading ? (
      <div className="dispatch-loading" role="status">
        <i className="fi fi-sr-rotate-right dispatch-loading-icon" aria-hidden="true"></i>
      </div>
    ) : activeTab === 'map' ? (
      <DispatchMapTab
        dispatchData={dispatchData}
        onSelectVehicle={setSelectedVehicle}
        onXytaTrip={handleXytaTrip}
      />
    ) : activeTab === 'alerts' ? (
      <DispatchAlertsTab
        alerts={alerts}
        alertFilter={alertFilter}
        driverAlerts={driverAlerts}
        systemAlerts={systemAlerts}
        onFilterChange={setAlertFilter}
        onAcknowledge={handleAcknowledge}
      />
    ) : (
      <DispatchXytaTab xytaTrips={xytaTrips} />
    )}
  </div>
 );
}

export default DispatchView;
```

frontend\src\DispatchView.css

```css
/* ===== DispatchView.css ===== */

.icon-mr4 {
  margin-right: 4px;
}

.icon-mr6 {
  margin-right: 6px;
}

.icon-mr8 {
  margin-right: 8px;
}

.icon-blue {
  color: #3498db;
}

.icon-red {
  color: #e74c3c;
}

.icon-green {
  color: #27ae60;
}

.icon-gray {
  color: #7f8c8d;
}

.icon-orange {
  color: #f39c12;
}

.icon-purple {
  color: #8e44ad;
}

/* ===== CONTAINER ===== */
.dispatch-container {
  padding: 20px;
  max-width: 1400px;
  margin: 0 auto;
}

/* ===== HEADER ===== */
.dispatch-header {
```

```
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.dispatch-title {
  margin: 0;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
}

.dispatch-header-badges {
  display: flex;
  gap: 10px;
  align-items: center;
}

.dispatch-badge-alert {
  background-color: #fdecea;
  color: #e74c3c;
  padding: 6px 14px;
  border-radius: 20px;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 6px;
}

.dispatch-badge-driver {
  background-color: #ebf5fb;
  color: #2980b9;
  padding: 6px 14px;
  border-radius: 20px;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 6px;
}

/* ===== MESSAGE ===== */
.dispatch-message {
  padding: 12px 16px;
  border-radius: 8px;
  margin-bottom: 15px;
```

```
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 8px;
}

.dispatch-message-success {
  background-color: #d5f5e3;
  color: #27ae60;
}

.dispatch-message-error {
  background-color: #fdecea;
  color: #e74c3c;
}

/* ===== STATS ROW ===== */
.dispatch-stats-grid {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 15px;
  margin-bottom: 20px;
}

.dispatch-stat-card {
  background-color: white;
  border-radius: 10px;
  padding: 15px;
  text-align: center;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.dispatch-stat-value {
  font-size: 1.8rem;
  font-weight: bold;
}

.dispatch-stat-label {
  font-size: 0.8rem;
  color: #7f8c8d;
  margin-top: 4px;
}

.dispatch-stat-icon {
  font-size: 1.2rem;
}
```

```
.stat-color-blue {
  color: #3498db;
  border-top: 4px solid #3498db;
}

.stat-color-red {
  color: #e74c3c;
  border-top: 4px solid #e74c3c;
}

.stat-color-navy {
  color: #2980b9;
  border-top: 4px solid #2980b9;
}

.stat-color-purple {
  color: #8e44ad;
  border-top: 4px solid #8e44ad;
}

/* ===== BREAKDOWN PANEL ===== */
.dispatch-breakdown-panel {
  background-color: white;
  border-radius: 10px;
  padding: 20px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  margin-bottom: 20px;
  border-left: 4px solid #e74c3c;
}

.dispatch-breakdown-title {
  margin: 0 0 15px;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
}

.dispatch-breakdown-grid {
  display: grid;
  grid-template-columns: 2fr 1fr auto auto;
  gap: 10px;
  align-items: end;
}

.dispatch-control-label {
  display: flex;
```

```
  align-items: center;
  gap: 6px;
  margin-bottom: 6px;
  font-weight: bold;
  font-size: 0.9rem;
}

.dispatch-select {
  width: 100%;
  padding: 10px;
  border: 2px solid #ecf0f1;
  border-radius: 8px;
  font-size: 0.9rem;
  font-family: inherit;
  outline: none;
  transition: border-color 0.2s;
}

.dispatch-select:focus {
  border-color: #3498db;
}

/* ===== BUTTONS ===== */
.dispatch-btn {
  padding: 10px 20px;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 6px;
  height: 44px;
  margin-top: 24px;
  font-family: inherit;
  font-size: 0.9rem;
  transition: opacity 0.2s;
}

.dispatch-btn:disabled {
  opacity: 0.6;
  cursor: not-allowed;
}

.dispatch-btn-red {
  background-color: #e74c3c;
```

```
}

.dispatch-btn-red:not(:disabled):hover {
  background-color: #c0392b;
}

.dispatch-btn-purple {
  background-color: #8e44ad;
}

.dispatch-btn-purple:not(:disabled):hover {
  background-color: #7d3c98;
}

.dispatch-btn-green {
  background-color: #27ae60;
}

.dispatch-btn-green:not(:disabled):hover {
  background-color: #229954;
}

.dispatch-btn-gray {
  background-color: #95a5a6;
  cursor: not-allowed;
}

.dispatch-restore-btn {
  padding: 10px 20px;
  color: white;
  border: none;
  border-radius: 8px;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 6px;
  height: 44px;
  margin-top: 10px;
  font-family: inherit;
  font-size: 0.9rem;
  transition: opacity 0.2s;
}

.dispatch-restore-btn.active-vehicle {
  background-color: #95a5a6;
  cursor: not-allowed;
}
```

```
.dispatch-restore-btn.inactive-vehicle {
  background-color: #27ae60;
  cursor: pointer;
}

.dispatch-restore-btn.inactive-vehicle:hover {
  background-color: #229954;
}

/* ===== BREAKDOWN RESULT ===== */
.dispatch-breakdown-result {
  margin-top: 15px;
  background-color: #f8f9fa;
  border-radius: 8px;
  padding: 15px;
}

.dispatch-result-grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 15px;
}

.dispatch-result-card {
  border-radius: 8px;
  padding: 12px;
}

.dispatch-result-card-red {
  background-color: #fdecea;
}

.dispatch-result-card-green {
  background-color: #d5f5e3;
}

.dispatch-result-card-blue {
  background-color: #ebf5fb;
}

.dispatch-result-card-label {
  font-size: 0.8rem;
  color: #7f8c8d;
  margin-bottom: 4px;
  display: flex;
  align-items: center;
```

```
  gap: 4px;
}

.dispatch-result-card-title {
  font-weight: bold;
  color: #2c3e50;
}

.dispatch-result-card-sub {
  font-size: 0.85rem;
  color: #7f8c8d;
}

.dispatch-result-card-detail-red {
  font-size: 0.8rem;
  color: #e74c3c;
}

.dispatch-result-card-detail-green {
  font-size: 0.8rem;
  color: #27ae60;
}

.dispatch-result-card-detail-blue {
  font-size: 0.85rem;
  color: #3498db;
}

.dispatch-result-card-detail-gray {
  font-size: 0.8rem;
  color: #7f8c8d;
}

/* ===== TABS ===== */
.dispatch-tabs {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
}

.dispatch-tab-btn {
  padding: 8px 16px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  font-size: 0.9rem;
```

```
  display: flex;
  align-items: center;
  gap: 6px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  font-family: inherit;
  transition: all 0.2s;
}

.dispatch-tab-btn.active {
  background-color: #2c3e50;
  color: white;
}

.dispatch-tab-btn.inactive {
  background-color: white;
  color: #2c3e50;
}

/* ===== LOADING ===== */
.dispatch-loading {
  text-align: center;
  padding: 40px;
}

.dispatch-loading-icon {
  font-size: 2rem;
  color: #3498db;
}

/* ===== MAP TAB ===== */
.dispatch-map-layout {
  display: grid;
  grid-template-columns: 300px 1fr;
  gap: 20px;
}

.dispatch-zone-sidebar {
  overflow-y: auto;
  max-height: 600px;
}

.dispatch-zone-card {
  background-color: white;
  border-radius: 10px;
  padding: 15px;
  margin-bottom: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
```

```
  border-left: 4px solid var(--zone-color, #3498db);
}

.dispatch-zone-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 8px;
}

.dispatch-zone-title {
  margin: 0;
  color: #2c3e50;
  font-size: 0.95rem;
}

.dispatch-zone-badge {
  padding: 2px 8px;
  border-radius: 10px;
  font-size: 0.75rem;
  font-weight: bold;
  background-color: var(--zone-tint, #eaf4fb);
  color: var(--zone-color, #3498db);
}

.dispatch-vehicle-chips {
  display: flex;
  gap: 5px;
  flex-wrap: wrap;
  margin-bottom: 6px;
}

.dispatch-vehicle-chip {
  padding: 3px 7px;
  border-radius: 6px;
  font-size: 0.7rem;
  font-weight: bold;
  cursor: pointer;
  display: flex;
  align-items: center;
  gap: 3px;
  border-width: 1px;
  border-style: solid;
  transition: transform 0.1s;
}

.dispatch-vehicle-chip:hover {
```

```
  transform: scale(1.05);
}

.dispatch-zone-xyta {
  font-size: 0.72rem;
  color: #7f8c8d;
  display: flex;
  align-items: center;
  gap: 4px;
}

.dispatch-map-wrap {
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  height: 600px;
}

.dispatch-map {
  height: 100%;
  width: 100%;
}

/* Popup styles */
.dispatch-popup {
  font-family: Consolas, monospace;
  min-width: 200px;
}

.dispatch-popup-title {
  margin: 0 0 8px;
  color: #2c3e50;
  border-bottom: 2px solid #3498db;
  padding-bottom: 6px;
}

.dispatch-popup-table {
  width: 100%;
  font-size: 0.8rem;
  border-collapse: collapse;
}

.dispatch-popup-table tr {
  border-bottom: 1px solid #ecf0f1;
}

.dispatch-popup-table td:first-child {
```

```
  padding: 3px 6px;
  color: #7f8c8d;
  font-weight: bold;
}

.dispatch-popup-table td:last-child {
  padding: 3px 6px;
  color: #2c3e50;
}

.dispatch-popup-xyta-btn {
  margin-top: 8px;
  padding: 5px 10px;
  background-color: #8e44ad;
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-size: 0.75rem;
  width: 100%;
  font-family: Consolas, monospace;
}

/* ===== ALERTS TAB ===== */
.dispatch-alert-filters {
  display: flex;
  gap: 8px;
  margin-bottom: 15px;
  background-color: white;
  padding: 12px;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  align-items: center;
}

.dispatch-alert-filter-label {
  font-weight: bold;
  font-size: 0.9rem;
  display: flex;
  align-items: center;
  gap: 6px;
  margin-right: 8px;
}

.dispatch-filter-btn {
  padding: 6px 14px;
  border: none;
```

```
  border-radius: 20px;
  cursor: pointer;
  font-weight: bold;
  font-size: 0.85rem;
  font-family: inherit;
  transition: all 0.2s;
}

.dispatch-empty {
  text-align: center;
  padding: 60px;
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.dispatch-empty-icon {
  font-size: 3rem;
  color: #27ae60;
  display: block;
  margin-bottom: 10px;
}

.dispatch-empty-title {
  color: #27ae60;
}

.dispatch-alerts-list {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.dispatch-alert-card {
  background-color: white;
  border-radius: 10px;
  padding: 15px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.dispatch-alert-inner {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
}

.dispatch-alert-body {
```

```
  flex: 1;
}

.dispatch-alert-badges {
  display: flex;
  align-items: center;
  gap: 8px;
  margin-bottom: 8px;
  flex-wrap: wrap;
}

.dispatch-badge {
  padding: 2px 8px;
  border-radius: 10px;
  font-size: 0.75rem;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 4px;
}

.dispatch-badge-from-driver {
  background-color: #ebf5fb;
  color: #2980b9;
}

.dispatch-badge-from-system {
  background-color: #fef9e7;
  color: #f39c12;
}

.dispatch-badge-action-type {
  background-color: #f8f9fa;
  color: #2c3e50;
  font-weight: normal;
}

.dispatch-badge-vehicle {
  font-size: 0.75rem;
  color: #7f8c8d;
  display: flex;
  align-items: center;
  gap: 4px;
}

.dispatch-badge-zone {
  padding: 2px 8px;
```

```
  border-radius: 10px;
  font-size: 0.75rem;
  font-weight: bold;
  background-color: var(--zone-tint, #eaf4fb);
  color: var(--zone-color, #3498db);
}

.dispatch-badge-active {
  background-color: #fdecea;
  color: #e74c3c;
}

.dispatch-badge-acknowledged {
  background-color: #d5f5e3;
  color: #27ae60;
}

.dispatch-badge-time {
  font-size: 0.72rem;
  color: #7f8c8d;
  display: flex;
  align-items: center;
  gap: 4px;
}

.dispatch-alert-message {
  margin: 0 0 8px;
  font-size: 0.85rem;
  color: #2c3e50;
  line-height: 1.5;
}

.dispatch-alert-stats {
  display: flex;
  gap: 12px;
  font-size: 0.8rem;
  margin-bottom: 8px;
}

.dispatch-alert-stat-bins {
  display: flex;
  align-items: center;
  gap: 4px;
  color: #3498db;
}

.dispatch-alert-stat-fill {
```

```
  display: flex;
  align-items: center;
  gap: 4px;
}

.dispatch-alert-stat-driver {
  display: flex;
  align-items: center;
  gap: 4px;
  color: #7f8c8d;
}

.dispatch-alert-nearest {
  margin: 0 0 6px;
  font-size: 0.8rem;
  color: #27ae60;
  display: flex;
  align-items: center;
  gap: 4px;
}

.dispatch-alert-notes {
  margin: 0;
  font-size: 0.8rem;
  color: #7f8c8d;
  font-style: italic;
}

.dispatch-acknowledge-btn {
  padding: 6px 12px;
  background-color: #27ae60;
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-size: 0.8rem;
  display: flex;
  align-items: center;
  gap: 4px;
  white-space: nowrap;
  margin-left: 10px;
  font-family: inherit;
  transition: background-color 0.2s;
}

.dispatch-acknowledge-btn:hover {
  background-color: #229954;
```

```
}

/* ===== XYTA TAB ===== */
.dispatch-xyta-panel {
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  overflow: hidden;
}

.dispatch-xyta-header {
  padding: 15px 20px;
  border-bottom: 1px solid #ecf0f1;
  display: flex;
  align-items: center;
  gap: 8px;
}

.dispatch-xyta-title {
  margin: 0;
  color: #2c3e50;
}

.dispatch-xyta-empty {
  text-align: center;
  padding: 40px;
  color: #7f8c8d;
}

.dispatch-xyta-empty-icon {
  font-size: 2rem;
  display: block;
  margin-bottom: 10px;
}

.dispatch-xyta-table-wrap {
  overflow-x: auto;
}

.dispatch-xyta-table {
  width: 100%;
  border-collapse: collapse;
}

.dispatch-xyta-table thead tr {
  background-color: #2c3e50;
  color: white;
```

```
}

.dispatch-xyta-table th {
  padding: 12px 15px;
  text-align: left;
  font-size: 0.85rem;
}

.dispatch-xyta-table tbody tr:nth-child(even) {
  background-color: #f8f9fa;
}

.dispatch-xyta-table tbody tr:nth-child(odd) {
  background-color: white;
}

.dispatch-xyta-table td {
  padding: 10px 15px;
  font-size: 0.85rem;
}

.dispatch-xyta-id {
  color: #7f8c8d;
}

.dispatch-xyta-vehicle {
  font-weight: bold;
  color: #3498db;
}

.dispatch-xyta-driver {
  display: flex;
  align-items: center;
  gap: 4px;
}

.dispatch-xyta-zone-badge {
  padding: 2px 8px;
  border-radius: 10px;
  font-size: 0.8rem;
  font-weight: bold;
  background-color: var(--zone-tint, #eaf4fb);
  color: var(--zone-color, #3498db);
}

.dispatch-xyta-fill-high {
  color: #e74c3c;
```

```
  font-weight: bold;
}

.dispatch-xyta-fill-med {
  color: #f39c12;
  font-weight: bold;
}

.dispatch-xyta-status-badge {
  padding: 3px 8px;
  border-radius: 10px;
  font-size: 0.75rem;
  font-weight: bold;
  display: inline-flex;
  align-items: center;
  gap: 4px;
}

.dispatch-xyta-status-enroute {
  background-color: #ebf5fb;
  color: #2980b9;
}

.dispatch-xyta-status-arrived {
  background-color: #d5f5e3;
  color: #27ae60;
}

.dispatch-xyta-time {
  font-size: 0.8rem;
  color: #7f8c8d;
}

/*
================================================================================
   ΔΥΝΑΜΙΚΑ ΧΡΩΜΑΤΑ ΜΕΣΩ CLASSES (αντικαθιστούν τα παλιά inline styles)
   ================================================================================
=== */

/* Χρώμα + ανοιχτή απόχρωση ανά ζώνη (set σε κάρτα/badge, κληρονομείται από
παιδιά) */
.zone-GAZ {
  --zone-color: #3498db;
  --zone-tint: #eaf4fb;
}

.zone-AMM {
```

```
  --zone-color: #27ae60;
  --zone-tint: #e7f7ee;
}

.zone-APE {
  --zone-color: #e74c3c;
  --zone-tint: #fdeceb;
}

.zone-TYL {
  --zone-color: #f39c12;
  --zone-tint: #fef4e1;
}

.zone-KRO {
  --zone-color: #9b59b6;
  --zone-tint: #f3ebf7;
}

.zone-ROD {
  --zone-color: #1abc9c;
  --zone-tint: #e6f9f5;
}

/* Vehicle chips — χρώμα ανά κατάσταση οχήματος */
.dispatch-vehicle-chip.veh-active {
  background-color: #e7f7ee;
  color: #27ae60;
  border-color: #bfe9d0;
}

.dispatch-vehicle-chip.veh-maintenance {
  background-color: #fef4e1;
  color: #f39c12;
  border-color: #fbe0b3;
}

.dispatch-vehicle-chip.veh-inactive {
  background-color: #fdeceb;
  color: #e74c3c;
  border-color: #f8c9c4;
}

/* Filter buttons — ενεργό χρώμα ανά φίλτρο (ανενεργό = γκρι από .dispatch-
filter-btn) */
.dispatch-filter-btn {
  background-color: #ecf0f1;
```

```
  color: #2c3e50;
}

.dispatch-filter-btn.active.filter-all {
  background-color: #2c3e50;
  color: white;
}

.dispatch-filter-btn.active.filter-driver {
  background-color: #2980b9;
  color: white;
}

.dispatch-filter-btn.active.filter-system {
  background-color: #f39c12;
  color: white;
}

/* Alert card — χρώμα αριστερού accent ανάλογα με προέλευση/τύπο */
.dispatch-alert-card.alert-border-driver {
  border-left: 4px solid #2980b9;
}

.dispatch-alert-card.alert-border-red {
  border-left: 4px solid #e74c3c;
}

.dispatch-alert-card.alert-border-orange {
  border-left: 4px solid #f39c12;
}

.dispatch-alert-card.alert-border-green {
  border-left: 4px solid #27ae60;
}

/* Alert fill stat — χρώμα ανάλογα με πληρότητα οχήματος */
.dispatch-alert-stat-fill.high {
  color: #e74c3c;
}

.dispatch-alert-stat-fill.med {
  color: #f39c12;
}

/* XYTA popup (Leaflet) */
.dispatch-popup-xyta-title {
  margin: 0 0 6px;
```

```
  color: #8e44ad;
}

.dispatch-popup-sub {
  margin: 3px 0;
  font-size: 0.85rem;
}

.dispatch-popup-coords {
  margin: 3px 0;
  font-size: 0.8rem;
  color: #7f8c8d;
}

/* Map markers (Leaflet divIcon) */
.dispatch-xyta-marker {
  background: #8e44ad;
  color: white;
  padding: 6px 10px;
  border-radius: 8px;
  font-weight: bold;
  font-size: 0.75rem;
  white-space: nowrap;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
}

.dispatch-vehicle-marker {
  width: 32px;
  height: 32px;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 15px;
  color: white;
}

.veh-marker-active {
  background: #27ae60;
}

.veh-marker-maintenance {
  background: #f39c12;
}
```

```
.veh-marker-inactive {
  background: #e74c3c;
}

/* ===== RESPONSIVE ===== */
@media (max-width: 1024px) {
  .dispatch-map-layout {
    grid-template-columns: 1fr;
  }

  .dispatch-zone-sidebar {
    max-height: none;
  }
}

@media (max-width: 768px) {
  .dispatch-stats-grid {
    grid-template-columns: repeat(2, 1fr);
  }

  .dispatch-breakdown-grid {
    grid-template-columns: 1fr;
  }

  .dispatch-result-grid {
    grid-template-columns: 1fr;
  }
}
```

Frontend\src\FillnessView.tsx

```
/**
 * FillnessView.tsx
 * ================
 * Επισκόπηση πληρότητας κάδων με φίλτρα (ημερομηνία / περιοχή / κάδος),
 * κάρτες στατιστικών, γράφημα ράβδων, χρονοσειρά ανά ώρα (όταν επιλεγεί
 * ημερομηνία) και αναλυτικούς πίνακες. Υποστηρίζει export σε PDF.
 *
 * Τα δεδομένα ανανεώνονται μόνα τους κάθε 30s.
 *
 * Styling: τα δυναμικά χρώματα ορίζονται από CSS classes (fill-* / stat-*).
 * Το πλάτος της μπάρας πληρότητας μπαίνει ως CSS variable μέσω ref (widthRef).
 */
import React, { useState, useEffect, useRef } from 'react';
import {
  BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip,
  Legend, ResponsiveContainer, Cell, LineChart, Line,
} from 'recharts';
import './FillnessView.css';
```

```
const API_URL = 'http://localhost:8000';
const REFRESH_INTERVAL_MS = 30000;

// ---------------------------------------------------------------------------
// Types
// ---------------------------------------------------------------------------

interface FillnessData {
  bin_id: string;
  location_name: string;
  address: string;
  bin_type: string;
  avg_fill_level: number;
  max_fill_level: number;
  min_fill_level: number;
  avg_battery_level: number;
  reading_count: number;
  last_reading: string | null;
  latitude: number;
  longitude: number;
}

interface BinOption {
  bin_id: string;
  location_name: string;
  area: string;
}

// Σημείο χρονοσειράς: ώρα + δυναμικά κλειδιά ανά κάδο (τιμή πληρότητας).
interface TimePoint {
  time: string;
  [binId: string]: string | number;
}

// ---------------------------------------------------------------------------
// Constants
// ---------------------------------------------------------------------------

const AREAS = [
  { value: 'ALL', label: 'Όλες οι περιοχές' },
  { value: 'GAZ', label: 'Γάζι' },
  { value: 'AMM', label: 'Αμμουδάρα' },
  { value: 'APE', label: 'Αγία Πελαγία' },
  { value: 'TYL', label: 'Τύλισος' },
  { value: 'KRO', label: 'Κρουσώνας' },
  { value: 'ROD', label: 'Ροδιά' },
];
```

```
const LINE_COLORS = ['#e74c3c', '#3498db', '#27ae60', '#f39c12', '#9b59b6', '#1abc9c',
'#e67e22', '#2c3e50', '#16a085', '#d35400'];

// ---------------------------------------------------------------------------
// Helpers
// ---------------------------------------------------------------------------

/** Χρώμα (hex) ανάλογα με την πληρότητα — χρειάζεται ως prop στο recharts Cell. */
function getFillColor(level: number): string {
  if (level >= 80) return '#e74c3c';
  if (level >= 60) return '#f39c12';
  return '#27ae60';
}

/** Κατηγορία πληρότητας για CSS classes (ok / warning / critical). */
function fillBucket(level: number): 'ok' | 'warning' | 'critical' {
  if (level >= 80) return 'critical';
  if (level >= 60) return 'warning';
  return 'ok';
}

/** Ref callback που θέτει το πλάτος της μπάρας ως CSS variable — χωρίς inline style. */
function widthRef(level: number) {
  return (el: HTMLDivElement | null): void => {
    if (el) el.style.setProperty('--fill-width', `${level}%`);
  };
}

function getBinTypeLabel(type: string): string {
  const labels: Record<string, string> = {
    general: 'Γενικά',
    recycling: 'Ανακύκλωση',
    organic: 'Οργανικά',
  };
  return labels[type] || type;
}

// ---------------------------------------------------------------------------
// Component
// ---------------------------------------------------------------------------

function FillnessView() {
  const [data, setData] = useState<FillnessData[]>([]);
  const [bins, setBins] = useState<BinOption[]>([]);
  const [loading, setLoading] = useState(false);
  const [selectedDate, setSelectedDate] = useState('');
  const [selectedArea, setSelectedArea] = useState('ALL');
  const [selectedBin, setSelectedBin] = useState('ALL');
  const [lastUpdated, setLastUpdated] = useState<Date | null>(null);
```

```
const [timeseries, setTimeseries] = useState<TimePoint[]>([]);
const [timeseriesBins, setTimeseriesBins] = useState<Record<string, string>>({});
const intervalRef = useRef<ReturnType<typeof setInterval> | null>(null);

// Λίστα κάδων για το dropdown (μία φορά)
useEffect(() => {
  fetch(`${API_URL}/api/v1/fillness/bins`)
    .then(r => r.json())
    .then(d => setBins(d.bins || []))
    .catch(error => console.error('Failed to fetch bin list:', error));
}, []);

// Δεδομένα + χρονοσειρά, με auto-refresh· ξανατρέχει όταν αλλάζουν τα φίλτρα
useEffect(() => {
  fetchData();
  fetchTimeseries();
  intervalRef.current = setInterval(() => {
    fetchData();
    fetchTimeseries();
  }, REFRESH_INTERVAL_MS);
  return () => {
    if (intervalRef.current) clearInterval(intervalRef.current);
  };
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, [selectedDate, selectedArea, selectedBin]);

// ---------------------------------------------------------------------
// API calls
// ---------------------------------------------------------------------

const fetchData = async (): Promise<void> => {
  setLoading(true);
  try {
    const params = new URLSearchParams();
    if (selectedDate) params.append('date', selectedDate);
    if (selectedArea !== 'ALL') params.append('area', selectedArea);
    if (selectedBin !== 'ALL') params.append('bin_id', selectedBin);

    const res = await fetch(`${API_URL}/api/v1/fillness?${params}`);
    const json = await res.json();
    setData(json.data ?? []);
    setLastUpdated(new Date());
  } catch (error) {
    console.error('Failed to fetch fillness data:', error);
  } finally {
    setLoading(false);
  }
};
```

```
  const fetchTimeseries = async (): Promise<void> => {
    if (!selectedDate) {
      setTimeseries([]);
      setTimeseriesBins({});
      return;
    }
    try {
      const params = new URLSearchParams();
      params.append('date', selectedDate);
      if (selectedArea !== 'ALL') params.append('area', selectedArea);
      if (selectedBin !== 'ALL') params.append('bin_id', selectedBin);

      const res = await fetch(`${API_URL}/api/v1/fillness/timeseries?${params}`);
      const json = await res.json();
      setTimeseries(json.timeline ?? []);
      setTimeseriesBins(json.bins ?? {});
    } catch (error) {
      console.error('Failed to fetch timeseries:', error);
    }
  };

  // ---------------------------------------------------------------------
  // PDF export
  // ---------------------------------------------------------------------

  const exportPDF = async (): Promise<void> => {
    const html2canvas = (await import('html2canvas')).default;
    const jsPDF = (await import('jspdf')).default;

    const element = document.getElementById('fillness-content');
    if (!element) return;

    // Κρύβουμε το κουμπί export όσο τραβάμε το screenshot
    const exportBtn = document.getElementById('export-btn');
    if (exportBtn) exportBtn.style.display = 'none';

    const canvas = await html2canvas(element, { scale: 2, useCORS: true,
backgroundColor: '#f5f6fa' });

    if (exportBtn) exportBtn.style.display = '';

    const imgData = canvas.toDataURL('image/png');
    const pdf = new jsPDF('p', 'mm', 'a4');
    const pdfWidth = pdf.internal.pageSize.getWidth();
    const pdfHeight = (canvas.height * pdfWidth) / canvas.width;
    const pageHeight = pdf.internal.pageSize.getHeight();

    if (pdfHeight <= pageHeight) {
      pdf.addImage(imgData, 'PNG', 0, 0, pdfWidth, pdfHeight);
```

```
    } else {
      // Σπάμε το μεγάλο screenshot σε πολλές σελίδες A4
      let position = 0;
      let remainingHeight = pdfHeight;
      while (remainingHeight > 0) {
        pdf.addImage(imgData, 'PNG', 0, position, pdfWidth, pdfHeight);
        remainingHeight -= pageHeight;
        position -= pageHeight;
        if (remainingHeight > 0) pdf.addPage();
      }
    }

    pdf.save(`πληρότητα_κάδων_${selectedDate || 'τρέχοντα'}_${new
Date().toISOString().slice(0, 10)}.pdf`);
  };

  // ---------------------------------------------------------------------
  // Derived data
  // ---------------------------------------------------------------------

  const filteredBins = selectedArea === 'ALL'
    ? bins
    : bins.filter(b => b.bin_id.startsWith(`BIN-${selectedArea}-`));

  const stats = {
    total: data.length,
    critical: data.filter(d => d.avg_fill_level >= 80).length,
    warning: data.filter(d => d.avg_fill_level >= 60 && d.avg_fill_level < 80).length,
    ok: data.filter(d => d.avg_fill_level < 60).length,
    avgFill: data.length > 0
      ? (data.reduce((s, d) => s + d.avg_fill_level, 0) / data.length).toFixed(1)
      : '0',
  };

  const statCards = [
    { label: 'Σύνολο Κάδων', value: stats.total, cls: 'stat-blue' },
    { label: 'Κρίσιμοι (>80%)', value: stats.critical, cls: 'stat-red' },
    { label: 'Προσοχή (60-80%)', value: stats.warning, cls: 'stat-orange' },
    { label: 'Κανονικοί (<60%)', value: stats.ok, cls: 'stat-green' },
    { label: 'Μέση Πληρότητα', value: `${stats.avgFill}%`, cls: 'stat-purple' },
  ];

  // Στήλες του κύριου πίνακα (τα Max/Min εμφανίζονται μόνο όταν έχει επιλεγεί
ημερομηνία)
  const tableColumns = [
    'Κάδος', 'Τοποθεσία', 'Τύπος',
    selectedDate ? 'Μέση Πληρότητα' : 'Πληρότητα',
    selectedDate ? 'Max' : '',
    selectedDate ? 'Min' : '',
```

```
  'Μπαταρία', 'Κατάσταση', 'Τελ. Ανάγνωση',
 ].filter(h => h !== '');

 // ---------------------------------------------------------------------
 // Render
 // ---------------------------------------------------------------------

 return (
  <div id="fillness-content" className="fillness-container">

   {/* Header */}
   <div className="fillness-header">
    <h2 className="fillness-title">
     <i className="fi fi-sr-chart-histogram icon-mr8" aria-hidden="true"></i>
     Πληρότητα Κάδων
    </h2>
    <div className="fillness-header-actions">
     {lastUpdated && (
      <span className="fillness-last-updated">
       <i className="fi fi-sr-rotate-right icon-mr4" aria-hidden="true"></i>
       {lastUpdated.toLocaleTimeString('el-GR')}
      </span>
     )}
     <button id="export-btn" onClick={exportPDF} className="fillness-export-btn">
      <i className="fi fi-sr-file-pdf icon-mr6" aria-hidden="true"></i>
      Export PDF
     </button>
    </div>
   </div>

   {/* Filters */}
   <div className="fillness-filters">
    <div>
     <label htmlFor="fillness-date" className="fillness-label">
      <i className="fi fi-sr-calendar icon-mr6" aria-hidden="true"></i>
      Ημερομηνία
     </label>
     <input
      id="fillness-date"
      type="date"
      value={selectedDate}
      onChange={e => { setSelectedDate(e.target.value); setSelectedBin('ALL'); }}
      className="fillness-input"
     />
     {selectedDate && (
      <button
       onClick={() => { setSelectedDate(''); setTimeseries([]); setTimeseriesBins({});
}}
       className="fillness-clear-btn"
```

```
      >
        <i className="fi fi-sr-cross icon-mr4" aria-hidden="true"></i>
        Καθαρισμός (τρέχοντα δεδομένα)
      </button>
    )}
  </div>

  <div>
    <label htmlFor="fillness-area" className="fillness-label">
      <i className="fi fi-sr-marker icon-mr6" aria-hidden="true"></i>
      Περιοχή
    </label>
    <select
      id="fillness-area"
      value={selectedArea}
      onChange={e => { setSelectedArea(e.target.value); setSelectedBin('ALL'); }}
      className="fillness-select"
    >
      {AREAS.map(a => (
        <option key={a.value} value={a.value}>{a.label}</option>
      ))}
    </select>
  </div>

  <div>
    <label htmlFor="fillness-bin" className="fillness-label">
      <i className="fi fi-sr-trash icon-mr6" aria-hidden="true"></i>
      Κάδος
    </label>
    <select
      id="fillness-bin"
      value={selectedBin}
      onChange={e => setSelectedBin(e.target.value)}
      className="fillness-select"
    >
      <option value="ALL">— Όλοι οι κάδοι —</option>
      {filteredBins.map(b => (
        <option key={b.bin_id} value={b.bin_id}>
          {b.bin_id} — {b.location_name}
        </option>
      ))}
    </select>
  </div>
</div>

{/* Stats Cards */}
<div className="fillness-stats-grid">
  {statCards.map(s => (
    <div key={s.label} className={`fillness-stat-card ${s.cls}`}>
```

```
          <div className="fillness-stat-value">{s.value}</div>
          <div className="fillness-stat-label">{s.label}</div>
        </div>
      ))}
    </div>

    {loading ? (
     <div className="fillness-loading" role="status" aria-live="polite">
      <i className="fi fi-sr-rotate-right" aria-hidden="true"></i>
      Φόρτωση...
     </div>
    ) : (
     <>
      {/* Bar Chart */}
      <div className="fillness-panel">
       <h3 className="fillness-panel-title">
        <i className="fi fi-sr-chart-histogram icon-mr8" aria-hidden="true"></i>
        Γράφημα Πληρότητας {selectedDate ? `— ${selectedDate} (Μέση Τιμή)` :
'(Τρέχοντα)'}
       </h3>
       <ResponsiveContainer width="100%" height={300}>
        <BarChart data={data} margin={{ top: 5, right: 30, left: 20, bottom: 60 }}>
         <CartesianGrid strokeDasharray="3 3" />
         <XAxis dataKey="bin_id" angle={-45} textAnchor="end" interval={0} tick={{
fontSize: 10 }} />
         <YAxis domain={[0, 100]} tickFormatter={v => `${v}%`} />
         <Tooltip
          formatter={(value: number) => [`${value}%`, 'Πληρότητα']}
          labelFormatter={label => {
           const bin = data.find(d => d.bin_id === label);
           return bin ? `${label} - ${bin.location_name}` : label;
          }}
         />
         <Legend />
         <Bar dataKey="avg_fill_level" name="Πληρότητα %" radius={[4, 4, 0, 0]}>
          {data.map((entry, index) => (
           <Cell key={index} fill={getFillColor(entry.avg_fill_level)} />
          ))}
         </Bar>
        </BarChart>
       </ResponsiveContainer>
      </div>

      {/* Timeseries Line Chart */}
      {selectedDate && timeseries.length > 0 && (
       <div className="fillness-panel">
        <h3 className="fillness-panel-title">
         <i className="fi fi-sr-chart-histogram icon-mr8" aria-hidden="true"></i>
         Χρονοσειρά Πληρότητας — {selectedDate}
```

```
          </h3>
          <ResponsiveContainer width="100%" height={350}>
           <LineChart data={timeseries} margin={{ top: 5, right: 30, left: 20, bottom: 60
}}>
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis
             dataKey="time"
             angle={-45}
             textAnchor="end"
             interval={0}
             tick={{ fontSize: 10 }}
             tickFormatter={v => v.slice(11, 16)}
            />
            <YAxis domain={[0, 100]} tickFormatter={v => `${v}%`} />
            <Tooltip
             formatter={(value: number, name: string) => [`${value}%`,
timeseriesBins[name] || name]}
             labelFormatter={label => `Ώρα: ${label.slice(11, 16)}`}
            />
            <Legend
             formatter={name => timeseriesBins[name] || name}
             wrapperStyle={{ paddingTop: '20px', fontSize: '0.8rem' }}
            />
            {Object.keys(timeseriesBins).map((binId, idx) => (
             <Line
              key={binId}
              type="monotone"
              dataKey={binId}
              stroke={LINE_COLORS[idx % LINE_COLORS.length]}
              strokeWidth={2}
              dot={{ r: 3 }}
              activeDot={{ r: 5 }}
             />
            ))}
           </LineChart>
          </ResponsiveContainer>
         </div>
        )}

        {/* Main Table */}
        <div className="fillness-table-card">
         <div className="fillness-table-header">
          <h3 className="fillness-table-title">
           <i className="fi fi-sr-list icon-mr8" aria-hidden="true"></i>
           Πίνακας Δεδομένων ({data.length} κάδοι)
           {selectedDate && ' — Μέσες Τιμές Ημέρας'}
          </h3>
         </div>
         <div className="fillness-table-wrap">
```

```
          <table className="fillness-table">
            <thead>
              <tr>
                {tableColumns.map(h => <th key={h}>{h}</th>)}
              </tr>
            </thead>
            <tbody>
              {data.map(row => (
                <tr key={row.bin_id}>
                  <td className="fillness-bin-id">{row.bin_id}</td>
                  <td>
                    <div className="fillness-location-name">{row.location_name}</div>
                    <div className="fillness-location-addr">{row.address}</div>
                  </td>
                  <td>{getBinTypeLabel(row.bin_type)}</td>
                  <td>
                    <div className={`fillness-fill-cell fill-${fillBucket(row.avg_fill_level)}`}
ref={widthRef(row.avg_fill_level)}>
                      <div className="fillness-fill-bar-bg">
                        <div className="fillness-fill-bar" />
                      </div>
                      <span className="fillness-fill-pct">{row.avg_fill_level}%</span>
                    </div>
                  </td>
                  {selectedDate && (
                    <>
                      <td className="fillness-max">↑ {row.max_fill_level}%</td>
                      <td className="fillness-min">↓ {row.min_fill_level}%</td>
                    </>
                  )}
                  <td>
                    <span className={`fillness-battery ${row.avg_battery_level > 30 ? 'ok' :
'low'}`}>
                      <i className="fi fi-sr-battery-full icon-mr4" aria-hidden="true"></i>
{row.avg_battery_level}%
                    </span>
                  </td>
                  <td>
                    <span className={`fillness-status-badge
${fillBucket(row.avg_fill_level)}`}>
                      {row.avg_fill_level >= 80 ? 'Κρίσιμος' : row.avg_fill_level >= 60 ?
'Προσοχή' : 'Κανονικός'}
                    </span>
                  </td>
                  <td className="fillness-last-reading">
                    {row.last_reading ? new Date(row.last_reading).toLocaleString('el-GR') :
'—'}
                  </td>
                </tr>
```

```
            ))}
          </tbody>
        </table>
      </div>
    </div>

    {/* Timeseries Table */}
    {selectedDate && timeseries.length > 0 && (
     <div className="fillness-table-card">
      <div className="fillness-table-header">
       <h3 className="fillness-table-title">
        Αναλυτικά Δεδομένα ανά Ώρα ({timeseries.length} ώρες)
       </h3>
      </div>
      <div className="fillness-table-wrap">
       <table className="fillness-table fillness-ts-table">
        <thead>
         <tr>
          <th>Ώρα</th>
          {Object.keys(timeseriesBins).map(binId => (
           <th key={binId}>{binId}</th>
          ))}
         </tr>
         <tr className="fillness-ts-subhead">
          <th></th>
          {Object.values(timeseriesBins).map((name, idx) => (
           <th key={idx}>{name}</th>
          ))}
         </tr>
        </thead>
        <tbody>
         {timeseries.map((row, idx) => (
          <tr key={idx}>
           <td className="fillness-ts-time">{row.time.slice(11, 16)}</td>
           {Object.keys(timeseriesBins).map(binId => {
            const val = row[binId] as number | undefined;
            return (
             <td key={binId}>
              {val !== undefined
               ? <span className={`fillness-ts-val fill-
${fillBucket(val)}`}>{val}%</span>
               : '—'}
             </td>
            );
           })}
          </tr>
         ))}
        </tbody>
       </table>
```

```
            </div>
          </div>
        )}
      </>
    )}
  </div>
 );
}

export default FillnessView;
```

frontend\src\FillnesView.css

```
/* ===== FillnessView.css ===== */

.icon-mr4 {
  margin-right: 4px;
}

.icon-mr6 {
  margin-right: 6px;
}

.icon-mr8 {
  margin-right: 8px;
}

/*
================================================================================
  ΧΡΩΜΑΤΑ ΜΕΣΩ CSS VARIABLES (αντί για inline styles)
  - fill-*  → χρώμα ανάλογα με πληρότητα
  - stat-*  → χρώμα κάρτας στατιστικού
  Το πλάτος της μπάρας (--fill-width) ορίζεται από JS μέσω ref.
  ================================================================================ */

.fill-ok {
  --fill-color: #27ae60;
}

.fill-warning {
  --fill-color: #f39c12;
}

.fill-critical {
  --fill-color: #e74c3c;
}
```

```
.stat-blue {
   --stat-color: #3498db;
}

.stat-red {
   --stat-color: #e74c3c;
}

.stat-orange {
   --stat-color: #f39c12;
}

.stat-green {
   --stat-color: #27ae60;
}

.stat-purple {
   --stat-color: #9b59b6;
}

/* ===== CONTAINER ===== */
.fillness-container {
   padding: 20px;
   max-width: 1200px;
   margin: 0 auto;
}

/* ===== HEADER ===== */
.fillness-header {
   display: flex;
   justify-content: space-between;
   align-items: center;
   margin-bottom: 20px;
}

.fillness-title {
   margin: 0;
   color: #2c3e50;
   display: flex;
   align-items: center;
}

.fillness-header-actions {
   display: flex;
   align-items: center;
   gap: 10px;
}

.fillness-last-updated {
```

```
    font-size: 0.8rem;
    color: #7f8c8d;
    display: flex;
    align-items: center;
}

.fillness-export-btn {
    padding: 8px 16px;
    background-color: #e74c3c;
    color: white;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-weight: bold;
    font-size: 0.9rem;
    font-family: inherit;
    display: flex;
    align-items: center;
    transition: background-color 0.2s;
}

.fillness-export-btn:hover {
    background-color: #c0392b;
}

/* ===== FILTERS ===== */
.fillness-filters {
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 15px;
}

.fillness-label {
    display: flex;
    align-items: center;
    margin-bottom: 6px;
    font-weight: bold;
    font-size: 0.9rem;
    color: #2c3e50;
}

.fillness-input,
.fillness-select {
    width: 100%;
```

```
    padding: 8px;
    border: 2px solid #ecf0f1;
    border-radius: 8px;
    font-size: 0.9rem;
    font-family: inherit;
    box-sizing: border-box;
    outline: none;
    transition: border-color 0.2s;
}

.fillness-input:focus,
.fillness-select:focus {
    border-color: #3498db;
}

.fillness-clear-btn {
    margin-top: 4px;
    background: none;
    border: none;
    color: #e74c3c;
    cursor: pointer;
    font-size: 0.8rem;
    font-family: inherit;
    display: flex;
    align-items: center;
    padding: 0;
}

/* ===== STATS CARDS ===== */
.fillness-stats-grid {
    display: grid;
    grid-template-columns: repeat(5, 1fr);
    gap: 15px;
    margin-bottom: 20px;
}

.fillness-stat-card {
    background-color: white;
    border-radius: 10px;
    padding: 15px;
    text-align: center;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    border-top: 4px solid var(--stat-color, #3498db);
}

.fillness-stat-value {
    font-size: 1.8rem;
    font-weight: bold;
    color: var(--stat-color, #3498db);
```

```
}

.fillness-stat-label {
    font-size: 0.8rem;
    color: #7f8c8d;
    margin-top: 4px;
}

/* ===== LOADING ===== */
.fillness-loading {
    text-align: center;
    padding: 40px;
    font-size: 1.2rem;
    color: #3498db;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 8px;
}

/* ===== PANELS (charts) ===== */
.fillness-panel {
    background-color: white;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
}

.fillness-panel-title {
    margin: 0 0 15px;
    color: #2c3e50;
    display: flex;
    align-items: center;
}

/* ===== TABLE CARD ===== */
.fillness-table-card {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    margin-bottom: 20px;
}

.fillness-table-header {
    padding: 15px 20px;
    border-bottom: 1px solid #ecf0f1;
}
```

```
.fillness-table-title {
    margin: 0;
    color: #2c3e50;
    display: flex;
    align-items: center;
}

.fillness-table-wrap {
    overflow-x: auto;
}

.fillness-table {
    width: 100%;
    border-collapse: collapse;
}

.fillness-table thead tr {
    background-color: #2c3e50;
    color: white;
}

.fillness-table th {
    padding: 12px 15px;
    text-align: left;
    font-size: 0.85rem;
}

.fillness-table td {
    padding: 10px 15px;
    font-size: 0.85rem;
}

.fillness-table tbody tr:nth-child(odd) {
    background-color: #f8f9fa;
}

.fillness-bin-id {
    font-weight: bold;
    color: #3498db;
}

.fillness-location-name {
    font-weight: bold;
    color: #2c3e50;
}

.fillness-location-addr {
    font-size: 0.75rem;
```

```
    color: #7f8c8d;
}

/* Fill bar μέσα στον πίνακα */
.fillness-fill-cell {
    display: flex;
    align-items: center;
    gap: 8px;
}

.fillness-fill-bar-bg {
    flex: 1;
    background-color: #ecf0f1;
    border-radius: 10px;
    height: 8px;
    min-width: 60px;
}

.fillness-fill-bar {
    height: 100%;
    border-radius: 10px;
    background-color: var(--fill-color, #27ae60);
    width: var(--fill-width, 0%);
    transition: width 0.3s;
}

.fillness-fill-pct {
    font-weight: bold;
    color: var(--fill-color, #27ae60);
    font-size: 0.85rem;
    min-width: 40px;
}

/* Max / Min */
.fillness-max {
    color: #e74c3c;
    font-weight: bold;
}

.fillness-min {
    color: #27ae60;
    font-weight: bold;
}

/* Μπαταρία */
.fillness-battery {
    display: flex;
    align-items: center;
}
```

```
.fillness-battery.ok {
    color: #27ae60;
}

.fillness-battery.low {
    color: #e74c3c;
}

/* Status badge */
.fillness-status-badge {
    padding: 3px 10px;
    border-radius: 12px;
    font-size: 0.75rem;
    font-weight: bold;
}

.fillness-status-badge.critical {
    background-color: #fdecea;
    color: #e74c3c;
}

.fillness-status-badge.warning {
    background-color: #fef9e7;
    color: #f39c12;
}

.fillness-status-badge.ok {
    background-color: #d5f5e3;
    color: #27ae60;
}

.fillness-last-reading {
    font-size: 0.8rem;
    color: #7f8c8d;
}

/* ===== TIMESERIES TABLE ===== */
.fillness-ts-table th {
    padding: 10px 15px;
    text-align: center;
    font-size: 0.75rem;
    white-space: nowrap;
}

.fillness-ts-table thead tr:first-child {
    background-color: #2c3e50;
    color: white;
}
```

```
.fillness-ts-table thead tr:first-child th:first-child {
   text-align: left;
}

.fillness-ts-table thead tr.fillness-ts-subhead {
   background-color: #34495e;
   color: #bdc3c7;
}

.fillness-ts-table thead tr.fillness-ts-subhead th {
   padding: 6px 15px;
   font-size: 0.7rem;
   font-weight: normal;
}

.fillness-ts-table td {
   padding: 8px 15px;
   text-align: center;
   font-size: 0.85rem;
}

.fillness-ts-table tbody tr:nth-child(odd) {
   background-color: #f8f9fa;
}

.fillness-ts-time {
   font-weight: bold;
   font-size: 0.85rem;
   white-space: nowrap;
   text-align: left;
}

.fillness-ts-val {
   font-weight: bold;
   color: var(--fill-color, #27ae60);
}

/* ===== RESPONSIVE ===== */
@media (max-width: 768px) {
   .fillness-filters {
      grid-template-columns: 1fr;
   }

   .fillness-stats-grid {
      grid-template-columns: repeat(2, 1fr);
   }

   .fillness-header {
```

```
      flex-direction: column;
      align-items: flex-start;
      gap: 12px;
    }
}
```

Frontend/src/ReportsView.tsx

```
/**
 * ReportsView.tsx
 * ===============
 * Σύστημα αναφοράς προβλημάτων (bug tracker). Επιτρέπει:
 *   - Υποβολή νέας αναφοράς με κατηγορία, προτεραιότητα και προαιρετικό screenshot
 *   - Φιλτράρισμα ανά κατάσταση (ανοικτές / σε εξέλιξη / κλειστές)
 *   - Επέκταση κάρτας για λεπτομέρειες + ενέργειες διαχειριστή (σημειώσεις, αλλαγή κατάστασης)
 *
 * Styling: τα δυναμικά χρώματα (προτεραιότητα, στατιστικά) ορίζονται από CSS classes.
 */
import React, { useState, useEffect, useRef } from 'react';
import './ReportsView.css';

const API_URL = 'http://localhost:8000';

// ---------------------------------------------------------------------------
// Types
// ---------------------------------------------------------------------------

interface Report {
  id: number;
  title: string;
  category: string;
  priority: string;
  description: string;
  screenshot_url: string | null;
  status: string;
  submitted_by: string;
  developer_notes: string | null;
  created_at: string;
  updated_at: string;
  closed_at: string | null;
}

type MessageType = 'success' | 'error';

interface FormState {
  title: string;
  category: string;
  priority: string;
  description: string;
  submitted_by: string;
```

```
  screenshot: File | null;
}

// ---------------------------------------------------------------------
// Constants
// ---------------------------------------------------------------------

const CATEGORIES = [
  { value: 'bug',         label: 'Σφάλμα λογισμικού' },
  { value: 'performance', label: 'Πρόβλημα απόδοσης' },
  { value: 'ui',          label: 'Πρόβλημα εμφάνισης' },
  { value: 'data',        label: 'Πρόβλημα δεδομένων' },
  { value: 'feature',     label: 'Αίτημα νέας λειτουργίας' },
  { value: 'security',    label: 'Θέμα ασφαλείας' },
  { value: 'other',       label: 'Άλλο' },
];

const PRIORITIES = [
  { value: 'critical', label: 'Κρίσιμο' },
  { value: 'high',     label: 'Υψηλό' },
  { value: 'medium',   label: 'Μέτριο' },
  { value: 'low',      label: 'Χαμηλό' },
];

const STATUS_FILTERS = [
  { value: 'ALL',         label: 'Όλες' },
  { value: 'open',        label: 'Ανοικτές' },
  { value: 'in_progress', label: 'Σε εξέλιξη' },
  { value: 'closed',      label: 'Κλειστές' },
];

// ---------------------------------------------------------------------
// Helpers
// ---------------------------------------------------------------------

function getPriorityLabel(priority: string): string {
  return PRIORITIES.find(p => p.value === priority)?.label || priority;
}

function getCategoryLabel(category: string): string {
  return CATEGORIES.find(c => c.value === category)?.label || category;
}

/** Πληροφορίες κατάστασης αναφοράς: CSS class, εικονίδιο, ετικέτα. */
function statusInfo(status: string): { cls: string; icon: string; label: string } {
  if (status === 'open')        return { cls: 'open',        icon: 'fi-sr-folder-open',  label:
'Ανοικτή' };
  if (status === 'in_progress') return { cls: 'in_progress', icon: 'fi-sr-settings',     label: 'Σε
εξέλιξη' };
```

```
  return { cls: 'closed', icon: 'fi-sr-check-circle', label: 'Κλειστή' };
}

// ---------------------------------------------------------------------
// Component
// ---------------------------------------------------------------------

function ReportsView() {
  const [reports, setReports]             = useState<Report[]>([]);
  const [loading, setLoading]             = useState(true);
  const [showForm, setShowForm]           = useState(false);
  const [filterStatus, setFilterStatus]   = useState('ALL');
  const [selectedReport, setSelectedReport] = useState<Report | null>(null);
  const [developerNote, setDeveloperNote] = useState('');
  const [message, setMessage]             = useState('');
  const [messageType, setMessageType]     = useState<MessageType>('success');
  const fileRef = useRef<HTMLInputElement>(null);

  const [form, setForm] = useState<FormState>({
    title: '',
    category: 'bug',
    priority: 'medium',
    description: '',
    submitted_by: 'admin',
    screenshot: null,
  });

  useEffect(() => { fetchReports(); }, [filterStatus]);

  // ---------------------------------------------------------------------
  // API calls
  // ---------------------------------------------------------------------

  const fetchReports = async (): Promise<void> => {
    setLoading(true);
    try {
      const params = filterStatus !== 'ALL' ? `?status=${filterStatus}` : '';
      const res = await fetch(`${API_URL}/api/v1/reports${params}`);
      const json = await res.json();
      setReports(json.reports ?? []);
    } catch (error) {
      console.error('Failed to fetch reports:', error);
    } finally {
      setLoading(false);
    }
  };

  const handleSubmit = async (e: React.FormEvent): Promise<void> => {
    e.preventDefault();
```

```
    try {
      const formData = new FormData();
      formData.append('title', form.title);
      formData.append('category', form.category);
      formData.append('priority', form.priority);
      formData.append('description', form.description);
      formData.append('submitted_by', form.submitted_by);
      if (form.screenshot) formData.append('screenshot', form.screenshot);

      const res = await fetch(`${API_URL}/api/v1/reports`, { method: 'POST', body:
formData });
      const json = await res.json();

      if (res.ok) {
        showMessage(`Η αναφορά στάλθηκε επιτυχώς! (ID: #${json.report_id})`, 'success');
        setShowForm(false);
        setForm({ title: '', category: 'bug', priority: 'medium', description: '', submitted_by:
'admin', screenshot: null });
        if (fileRef.current) fileRef.current.value = '';
        fetchReports();
      } else {
        showMessage(`Σφάλμα: ${json.detail}`, 'error');
      }
    } catch (error) {
      console.error('Failed to submit report:', error);
      showMessage('Αδυναμία σύνδεσης με τον διακομιστή.', 'error');
    }
  };

  const handleUpdateStatus = async (reportId: number, status: string): Promise<void> => {
    try {
      const data: { status: string; developer_notes?: string } = { status };
      if (developerNote) data.developer_notes = developerNote;

      const res = await fetch(`${API_URL}/api/v1/reports/${reportId}`, {
        method: 'PUT',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data),
      });

      if (res.ok) {
        showMessage('Η αναφορά ενημερώθηκε.', 'success');
        setSelectedReport(null);
        setDeveloperNote('');
        fetchReports();
      }
    } catch (error) {
      console.error('Failed to update report:', error);
    }
```

```
};

const showMessage = (msg: string, type: MessageType): void => {
  setMessage(msg);
  setMessageType(type);
  setTimeout(() => setMessage(''), 5000);
};

// ---------------------------------------------------------------------
// Derived data
// ---------------------------------------------------------------------

const stats = {
  total:      reports.length,
  open:       reports.filter(r => r.status === 'open').length,
  inProgress: reports.filter(r => r.status === 'in_progress').length,
  closed:     reports.filter(r => r.status === 'closed').length,
  critical:   reports.filter(r => r.priority === 'critical').length,
};

const statCards = [
  { label: 'Σύνολο',     value: stats.total,      cls: 'stat-blue',   icon: 'fi-sr-list' },
  { label: 'Ανοικτές',   value: stats.open,       cls: 'stat-red',    icon: 'fi-sr-folder-open' },
  { label: 'Σε εξέλιξη', value: stats.inProgress, cls: 'stat-orange', icon: 'fi-sr-settings' },
  { label: 'Κλειστές',   value: stats.closed,     cls: 'stat-green',  icon: 'fi-sr-check-circle' },
  { label: 'Κρίσιμες',   value: stats.critical,   cls: 'stat-purple', icon: 'fi-sr-triangle-warning'
},
];

// ---------------------------------------------------------------------
// Render
// ---------------------------------------------------------------------

return (
  <div className="reports-container">

    {/* Header */}
    <div className="reports-header">
      <h2 className="reports-title">
        <i className="fi fi-sr-bug" aria-hidden="true"></i>
        Αναφορά Προβλήματος
      </h2>
      <button onClick={() => setShowForm(!showForm)} className="reports-new-btn">
        <i className="fi fi-sr-plus" aria-hidden="true"></i>
        Νέα Αναφορά
      </button>
    </div>

    {/* Message */}
```

```
    {message && (
     <div className={`reports-message ${messageType}`} role="alert" aria-
live="polite">
       <i className={`fi ${messageType === 'success' ? 'fi-sr-check-circle' : 'fi-sr-cross-
circle'}`} aria-hidden="true"></i>
       {message}
     </div>
    )}

    {/* Stats */}
    <div className="reports-stats-grid">
     {statCards.map(s => (
      <div key={s.label} className={`reports-stat-card ${s.cls}`}>
       <i className={`fi ${s.icon} reports-stat-icon`} aria-hidden="true"></i>
       <div className="reports-stat-value">{s.value}</div>
       <div className="reports-stat-label">{s.label}</div>
      </div>
     ))}
    </div>

    {/* New Report Form */}
    {showForm && (
     <div className="reports-form-card">
      <h3 className="reports-form-title">
       <i className="fi fi-sr-bug reports-form-title-icon" aria-hidden="true"></i>
       Νέα Αναφορά Προβλήματος
      </h3>
      <form onSubmit={handleSubmit}>
       <div className="reports-form-grid">

        <div className="reports-field-full">
         <label htmlFor="report-title" className="reports-label">
          <i className="fi fi-sr-text" aria-hidden="true"></i>
          Τίτλος Προβλήματος *
         </label>
         <input
          id="report-title"
          type="text"
          value={form.title}
          onChange={e => setForm({ ...form, title: e.target.value })}
          className="reports-input"
          placeholder="Σύντομη περιγραφή του προβλήματος"
          required
         />
        </div>

        <div>
         <label htmlFor="report-category" className="reports-label">
          <i className="fi fi-sr-tags" aria-hidden="true"></i>
```

```
            Κατηγορία *
          </label>
          <select
            id="report-category"
            value={form.category}
            onChange={e => setForm({ ...form, category: e.target.value })}
            className="reports-select"
          >
            {CATEGORIES.map(c => (
              <option key={c.value} value={c.value}>{c.label}</option>
            ))}
          </select>
        </div>

        <div>
          <label htmlFor="report-priority" className="reports-label">
            <i className="fi fi-sr-triangle-warning" aria-hidden="true"></i>
            Προτεραιότητα *
          </label>
          <select
            id="report-priority"
            value={form.priority}
            onChange={e => setForm({ ...form, priority: e.target.value })}
            className={`reports-select reports-select-priority prio-${form.priority}`}
          >
            {PRIORITIES.map(p => (
              <option key={p.value} value={p.value}>{p.label}</option>
            ))}
          </select>
        </div>

        <div className="reports-field-full">
          <label htmlFor="report-description" className="reports-label">
            <i className="fi fi-sr-document" aria-hidden="true"></i>
            Περιγραφή Προβλήματος *
          </label>
          <textarea
            id="report-description"
            value={form.description}
            onChange={e => setForm({ ...form, description: e.target.value })}
            className="reports-textarea"
            placeholder="Περιγράψτε αναλυτικά το πρόβλημα, πότε εμφανίστηκε και πώς μπορεί να αναπαραχθεί..."
            required
          />
        </div>

        <div className="reports-field-full">
          <label className="reports-label">
```

```
            <i className="fi fi-sr-picture" aria-hidden="true"></i>
            Screenshot (προαιρετικό)
          </label>
          <div className="reports-file-drop">
            <input
              ref={fileRef}
              type="file"
              accept="image/*"
              onChange={e => setForm({ ...form, screenshot: e.target.files?.[0] || null })}
              className="reports-file-hidden"
              id="screenshot-input"
            />
            <label htmlFor="screenshot-input" className="reports-file-label">
              {form.screenshot ? (
                <span className="reports-file-chosen">
                  <i className="fi fi-sr-check-circle" aria-hidden="true"></i>
                  {form.screenshot.name}
                </span>
              ) : (
                <span className="reports-file-empty">
                  <i className="fi fi-sr-picture reports-file-empty-icon" aria-
hidden="true"></i>
                  Κλικ για επιλογή αρχείου
                </span>
              )}
            </label>
          </div>
        </div>
      </div>

      <div className="reports-form-actions">
        <button type="button" onClick={() => setShowForm(false)}
className="reports-cancel-btn">
          <i className="fi fi-sr-cross" aria-hidden="true"></i>
          Ακύρωση
        </button>
        <button type="submit" className="reports-submit-btn">
          <i className="fi fi-sr-paper-plane" aria-hidden="true"></i>
          Αποστολή Αναφοράς
        </button>
      </div>
    </form>
  </div>
)}

{/* Filter */}
<div className="reports-filter-bar">
  <label className="reports-filter-label">
    <i className="fi fi-sr-filter" aria-hidden="true"></i>
```

```
        Κατάσταση:
       </label>
       {STATUS_FILTERS.map(f => (
        <button
         key={f.value}
         onClick={() => setFilterStatus(f.value)}
         className={`reports-filter-btn ${filterStatus === f.value ? 'active' : ''}`}
        >
         {f.label}
        </button>
       ))}
       <span className="reports-filter-count">{reports.length} αναφορές</span>
      </div>

      {/* Reports List */}
      {loading ? (
       <div className="reports-loading" role="status" aria-live="polite">
        <i className="fi fi-sr-rotate-right reports-loading-icon" aria-hidden="true"></i>
       </div>
      ) : reports.length === 0 ? (
       <div className="reports-empty" role="status">
        <i className="fi fi-sr-check-circle reports-empty-icon" aria-hidden="true"></i>
        <h3 className="reports-empty-title">Δεν υπάρχουν αναφορές!</h3>
        <p>Το σύστημα λειτουργεί κανονικά.</p>
       </div>
      ) : (
       <div className="reports-list">
        {reports.map(report => {
         const isSelected = selectedReport?.id === report.id;
         const status = statusInfo(report.status);
         return (
          <div
           key={report.id}
           className={`reports-card prio-${report.priority} ${isSelected ? 'selected' : ''}`}
          >
           <div
            className="reports-card-inner"
            onClick={() => setSelectedReport(isSelected ? null : report)}
            role="button"
            tabIndex={0}
            onKeyDown={e => { if (e.key === 'Enter') setSelectedReport(isSelected ? null
: report); }}
            title={isSelected ? 'Σύμπτυξη' : 'Λεπτομέρειες'}
           >
            <div className="reports-card-body">
             <div className="reports-card-badges">
              <span className="reports-id">#{report.id}</span>
              <span className="reports-badge reports-badge-priority">
               <i className="fi fi-sr-triangle-warning icon-mr3" aria-hidden="true"></i>
```

```
                {getPriorityLabel(report.priority)}
               </span>
               <span className="reports-badge reports-badge-category">
                <i className="fi fi-sr-tags icon-mr3" aria-hidden="true"></i>
                {getCategoryLabel(report.category)}
               </span>
               <span className={`reports-badge reports-badge-status ${status.cls}`}>
                <i className={`fi ${status.icon} icon-mr3`} aria-hidden="true"></i>
                {status.label}
               </span>
              </div>

              <h3 className="reports-card-title">{report.title}</h3>
              <p className="reports-card-desc">
               {report.description.length > 150 ? report.description.substring(0, 150) + '...' :
report.description}
              </p>
              <div className="reports-card-meta">
               <span className="reports-meta-item">
                <i className="fi fi-sr-user" aria-hidden="true"></i>
                {report.submitted_by}
               </span>
               <span className="reports-meta-item">
                <i className="fi fi-sr-calendar" aria-hidden="true"></i>
                {new Date(report.created_at).toLocaleString('el-GR')}
               </span>
               {report.closed_at && (
                <span className="reports-meta-item reports-meta-closed">
                 <i className="fi fi-sr-check-circle" aria-hidden="true"></i>
                 Έκλεισε: {new Date(report.closed_at).toLocaleString('el-GR')}
                </span>
               )}
              </div>
             </div>

             {report.screenshot_url && (
              <img src={report.screenshot_url} alt="screenshot" className="reports-card-
thumb" />
             )}
            </div>

            {/* Expanded details */}
            {isSelected && (
             <div className="reports-expanded">
              <p className="reports-full-desc">
               <strong>Πλήρης περιγραφή:</strong><br />
               {report.description}
              </p>
```

```
            {report.screenshot_url && (
              <div className="reports-screenshot-block">
                <strong className="reports-screenshot-label">
                  <i className="fi fi-sr-picture" aria-hidden="true"></i>
                  Screenshot:
                </strong>
                <img src={report.screenshot_url} alt="screenshot" className="reports-
screenshot-img" />
              </div>
            )}

            {report.developer_notes && (
              <div className="reports-devnotes">
                <strong className="reports-devnotes-label">
                  <i className="fi fi-sr-code" aria-hidden="true"></i>
                  Σημειώσεις Developer:
                </strong>
                <p className="reports-devnotes-text">{report.developer_notes}</p>
              </div>
            )}

            {report.status !== 'closed' && (
              <div className="reports-action-box">
                <label className="reports-action-label">
                  <i className="fi fi-sr-code" aria-hidden="true"></i>
                  Σημείωση Developer (προαιρετικό):
                </label>
                <textarea
                  value={developerNote}
                  onChange={e => setDeveloperNote(e.target.value)}
                  placeholder="Προσθέστε σημείωση για την αναφορά..."
                  className="reports-action-textarea"
                />
                <div className="reports-action-buttons">
                  {report.status === 'open' && (
                    <button onClick={() => handleUpdateStatus(report.id, 'in_progress')}
className="reports-btn-progress">
                      <i className="fi fi-sr-settings" aria-hidden="true"></i>
                      Σε εξέλιξη
                    </button>
                  )}
                  <button onClick={() => handleUpdateStatus(report.id, 'closed')}
className="reports-btn-close">
                    <i className="fi fi-sr-check-circle" aria-hidden="true"></i>
                    Κλείσιμο Αναφοράς
                  </button>
                </div>
              </div>
            )}
```

```
              </div>
            )}
          </div>
        );
      })}
    </div>
  )}
 </div>
 );
}

export default ReportsView;
```

frontend/src/ReportsView.css

```
/* ===== ReportsView.css ===== */

.icon-mr3 { margin-right: 3px; }
.icon-mr4 { margin-right: 4px; }
.icon-mr6 { margin-right: 6px; }
.icon-mr8 { margin-right: 8px; }

/*
================================================================
============
  ΔΥΝΑΜΙΚΑ ΧΡΩΜΑΤΑ ΜΕΣΩ CSS VARIABLES
  - prio-*  → χρώμα + ανοιχτή απόχρωση ανά προτεραιότητα
  - stat-*  → χρώμα κάρτας στατιστικού
  ================================================================
============ */

.prio-critical { --prio-color: #e74c3c; --prio-tint: #fdeceb; }
.prio-high     { --prio-color: #e67e22; --prio-tint: #fdf0e6; }
.prio-medium   { --prio-color: #f39c12; --prio-tint: #fef4e1; }
.prio-low      { --prio-color: #27ae60; --prio-tint: #e7f7ee; }

.stat-blue   { --stat-color: #3498db; }
.stat-red    { --stat-color: #e74c3c; }
.stat-orange { --stat-color: #f39c12; }
.stat-green  { --stat-color: #27ae60; }
.stat-purple { --stat-color: #8e44ad; }

/* ===== CONTAINER ===== */
.reports-container {
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
}

/* ===== HEADER ===== */
.reports-header {
```

```
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.reports-title {
  margin: 0;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
}

.reports-new-btn {
  padding: 10px 20px;
  background-color: #e74c3c;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  font-family: inherit;
  display: flex;
  align-items: center;
  gap: 6px;
  transition: background-color 0.2s;
}

.reports-new-btn:hover { background-color: #c0392b; }

/* ===== MESSAGE ===== */
.reports-message {
  padding: 12px 16px;
  border-radius: 8px;
  margin-bottom: 15px;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 8px;
}

.reports-message.success { background-color: #d5f5e3; color: #27ae60; }
.reports-message.error   { background-color: #fdecea; color: #e74c3c; }

/* ===== STATS ===== */
.reports-stats-grid {
  display: grid;
  grid-template-columns: repeat(5, 1fr);
```

```
  gap: 15px;
  margin-bottom: 20px;
}

.reports-stat-card {
  background-color: white;
  border-radius: 10px;
  padding: 15px;
  text-align: center;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  border-top: 4px solid var(--stat-color, #3498db);
}

.reports-stat-icon {
  color: var(--stat-color, #3498db);
  font-size: 1.2rem;
}

.reports-stat-value {
  font-size: 1.8rem;
  font-weight: bold;
  color: var(--stat-color, #3498db);
}

.reports-stat-label {
  font-size: 0.8rem;
  color: #7f8c8d;
  margin-top: 4px;
}

/* ===== FORM ===== */
.reports-form-card {
  background-color: white;
  border-radius: 10px;
  padding: 25px;
  margin-bottom: 20px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  border-left: 4px solid #e74c3c;
}

.reports-form-title {
  margin: 0 0 20px;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
}

.reports-form-title .reports-form-title-icon { color: #e74c3c; }
```

```
.reports-form-grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 15px;
  margin-bottom: 15px;
}

.reports-field-full { grid-column: 1 / -1; }

.reports-label {
  display: flex;
  align-items: center;
  gap: 6px;
  margin-bottom: 6px;
  font-weight: bold;
  font-size: 0.9rem;
}

.reports-input,
.reports-select,
.reports-textarea {
  width: 100%;
  padding: 10px;
  border: 2px solid #ecf0f1;
  border-radius: 8px;
  box-sizing: border-box;
  font-size: 0.95rem;
  font-family: inherit;
  outline: none;
  transition: border-color 0.2s;
}

.reports-input:focus,
.reports-select:focus,
.reports-textarea:focus {
  border-color: #3498db;
}

/* Το select προτεραιότητας παίρνει το χρώμα της επιλεγμένης προτεραιότητας */
.reports-select-priority {
  border-color: var(--prio-color, #ecf0f1);
}

.reports-textarea {
  min-height: 120px;
  resize: vertical;
}
```

```
/* File upload */
.reports-file-drop {
  border: 2px dashed #ecf0f1;
  border-radius: 8px;
  padding: 20px;
  text-align: center;
  background-color: #f8f9fa;
  cursor: pointer;
}

.reports-file-hidden { display: none; }

.reports-file-label { cursor: pointer; }

.reports-file-chosen {
  color: #27ae60;
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 8px;
}

.reports-file-empty { color: #7f8c8d; }

.reports-file-empty-icon {
  font-size: 1.8rem;
  display: block;
  margin-bottom: 8px;
}

.reports-form-actions {
  display: flex;
  gap: 10px;
  justify-content: flex-end;
}

.reports-cancel-btn {
  padding: 10px 20px;
  background-color: #ecf0f1;
  color: #2c3e50;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  font-family: inherit;
  display: flex;
  align-items: center;
  gap: 6px;
}
```

```
.reports-submit-btn {
  padding: 10px 25px;
  background-color: #e74c3c;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  font-family: inherit;
  display: flex;
  align-items: center;
  gap: 6px;
}

.reports-submit-btn:hover { background-color: #c0392b; }

/* ===== FILTER BAR ===== */
.reports-filter-bar {
  background-color: white;
  padding: 15px 20px;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  margin-bottom: 20px;
  display: flex;
  gap: 10px;
  align-items: center;
  flex-wrap: wrap;
}

.reports-filter-label {
  font-weight: bold;
  font-size: 0.9rem;
  display: flex;
  align-items: center;
  gap: 6px;
}

.reports-filter-btn {
  padding: 6px 14px;
  border: none;
  border-radius: 20px;
  cursor: pointer;
  font-weight: bold;
  font-size: 0.85rem;
  font-family: inherit;
  background-color: #ecf0f1;
  color: #2c3e50;
  transition: background-color 0.2s;
```

```
}

.reports-filter-btn.active {
  background-color: #3498db;
  color: white;
}

.reports-filter-count {
  margin-left: auto;
  color: #7f8c8d;
  font-size: 0.85rem;
}

/* ===== LOADING / EMPTY ===== */
.reports-loading {
  text-align: center;
  padding: 40px;
}

.reports-loading-icon {
  font-size: 2rem;
  color: #3498db;
}

.reports-empty {
  text-align: center;
  padding: 60px;
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.reports-empty-icon {
  font-size: 3rem;
  color: #27ae60;
  display: block;
  margin-bottom: 10px;
}

.reports-empty-title { color: #27ae60; }

/* ===== REPORT LIST ===== */
.reports-list {
  display: flex;
  flex-direction: column;
  gap: 12px;
}

.reports-card {
```

```
  background-color: white;
  border-radius: 10px;
  padding: 20px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  cursor: pointer;
  transition: border-color 0.2s, box-shadow 0.2s;
  /* λεπτό περίγραμμα + χρωματιστό accent προτεραιότητας αριστερά */
  border: 1px solid #ecf0f1;
  border-left: 4px solid var(--prio-color, #7f8c8d);
}

.reports-card:hover { box-shadow: 0 4px 14px rgba(0, 0, 0, 0.15); }

.reports-card.selected { border: 2px solid #3498db; }

.reports-card-inner {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
}

.reports-card-body { flex: 1; }

.reports-card-badges {
  display: flex;
  align-items: center;
  gap: 8px;
  margin-bottom: 8px;
  flex-wrap: wrap;
}

.reports-id {
  font-size: 0.75rem;
  color: #7f8c8d;
  font-weight: bold;
}

.reports-badge {
  padding: 2px 8px;
  border-radius: 10px;
  font-size: 0.75rem;
  font-weight: bold;
  display: inline-flex;
  align-items: center;
}

.reports-badge-priority {
  background-color: var(--prio-tint, #f0f0f0);
  color: var(--prio-color, #7f8c8d);
```

```
}

.reports-badge-category {
  background-color: #ecf0f1;
  color: #2c3e50;
  font-weight: normal;
}

.reports-badge-status.open        { background-color: #fdecea; color: #e74c3c; }
.reports-badge-status.in_progress { background-color: #fef9e7; color: #f39c12; }
.reports-badge-status.closed      { background-color: #d5f5e3; color: #27ae60; }

.reports-card-title {
  margin: 0 0 6px;
  color: #2c3e50;
  font-size: 1rem;
}

.reports-card-desc {
  margin: 0 0 8px;
  color: #7f8c8d;
  font-size: 0.85rem;
  line-height: 1.4;
}

.reports-card-meta {
  display: flex;
  gap: 15px;
  font-size: 0.75rem;
  color: #7f8c8d;
  flex-wrap: wrap;
}

.reports-meta-item {
  display: flex;
  align-items: center;
  gap: 4px;
}

.reports-meta-closed { color: #27ae60; }

.reports-card-thumb {
  width: 80px;
  height: 60px;
  object-fit: cover;
  border-radius: 6px;
  margin-left: 15px;
  border: 1px solid #ecf0f1;
}
```

```
/* ===== EXPANDED DETAILS ===== */
.reports-expanded {
  margin-top: 15px;
  border-top: 1px solid #ecf0f1;
  padding-top: 15px;
}

.reports-full-desc {
  margin: 0 0 12px;
  color: #2c3e50;
  font-size: 0.9rem;
  line-height: 1.6;
}

.reports-screenshot-block { margin-bottom: 12px; }

.reports-screenshot-label {
  font-size: 0.85rem;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 6px;
  margin-bottom: 8px;
}

.reports-screenshot-img {
  display: block;
  max-width: 400px;
  max-height: 250px;
  border-radius: 8px;
  border: 1px solid #ecf0f1;
}

.reports-devnotes {
  background-color: #ebf5fb;
  padding: 12px;
  border-radius: 8px;
  margin-bottom: 12px;
}

.reports-devnotes-label {
  font-size: 0.85rem;
  color: #2980b9;
  display: flex;
  align-items: center;
  gap: 6px;
  margin-bottom: 6px;
}
```

```
.reports-devnotes-text {
  margin: 0;
  font-size: 0.85rem;
  color: #2c3e50;
}

.reports-action-box {
  background-color: #f8f9fa;
  padding: 15px;
  border-radius: 8px;
}

.reports-action-label {
  display: flex;
  align-items: center;
  gap: 6px;
  margin-bottom: 8px;
  font-weight: bold;
  font-size: 0.85rem;
}

.reports-action-textarea {
  width: 100%;
  padding: 8px;
  border: 1px solid #ddd;
  border-radius: 6px;
  box-sizing: border-box;
  min-height: 70px;
  resize: vertical;
  margin-bottom: 10px;
  font-family: inherit;
}

.reports-action-buttons {
  display: flex;
  gap: 8px;
}

.reports-btn-progress,
.reports-btn-close {
  padding: 8px 16px;
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-weight: bold;
  font-size: 0.85rem;
  font-family: inherit;
```

```
  display: flex;
  align-items: center;
  gap: 6px;
}

.reports-btn-progress { background-color: #f39c12; }
.reports-btn-close    { background-color: #27ae60; }

/* ===== RESPONSIVE ===== */
@media (max-width: 768px) {
  .reports-stats-grid {
    grid-template-columns: repeat(2, 1fr);
  }
  .reports-form-grid {
    grid-template-columns: 1fr;
  }
  .reports-header {
    flex-direction: column;
    align-items: flex-start;
    gap: 12px;
  }
}
```

Frontend/src/RoutesView.tsx

```
import React, { useState, useEffect } from 'react';
import { MapContainer, TileLayer, CircleMarker, Popup, Polyline } from 'react-leaflet';
import 'leaflet/dist/leaflet.css';
import './RoutesView.css';

const API_URL = 'http://localhost:8000';

interface RouteStop {
  bin_id: string;
  location_name: string;
  address: string;
  latitude: number;
  longitude: number;
  current_fill_level: number;
  current_battery_level: number;
  bin_type: string;
  fill_status: string;
  zone_code: string;
  zone_name: string;
  stop_number: number;
  distance_from_prev_km: number;
  cumulative_distance_km: number;
}

interface RouteData {
```

```
  status: string;
  algorithm: string;
  threshold: number;
  fallback_used: boolean;
  original_threshold: number;
  zone_code: string;
  total_bins: number;
  total_distance_km: number;
  estimated_duration_minutes: number;
  depot: { name: string; latitude: number; longitude: number };
  route: RouteStop[];
  route_id: number | null;
  saved: boolean;
  osm_warning: string | null;
  distance_mode: string;
  osm_nodes: number;
  osm_edges: number;
  message?: string;
}

interface HistoryRoute {
  id: number;
  route_name: string;
  description: string;
  driver_name: string;
  vehicle_id: string;
  scheduled_date: string;
  estimated_duration: number;
  status: string;
  created_at: string;
}

interface Vehicle {
  vehicle_id: string;
  plate_number: string;
  driver_name: string;
  zone_code: string;
  status?: string;
}

function getFillColor(level: number): string {
  if (level >= 80) return '#e74c3c';
  if (level >= 60) return '#f39c12';
  return '#27ae60';
}

/** Κατηγορία πληρότητας για CSS classes (ok / warning / critical). */
function fillBucket(level: number): 'ok' | 'warning' | 'critical' {
  if (level >= 80) return 'critical';
```

```
  if (level >= 60) return 'warning';
  return 'ok';
}

function RoutesView() {
  const [routeData, setRouteData] = useState<RouteData | null>(null);
  const [loading, setLoading] = useState(false);
  const [saving, setSaving] = useState(false);
  const [threshold, setThreshold] = useState(70);
  const [zoneCode, setZoneCode] = useState('ALL');
  const [vehicleId, setVehicleId] = useState('');
  const [activeStop, setActiveStop] = useState<number | null>(null);
  const [vehicles, setVehicles] = useState<Vehicle[]>([]);
  const [history, setHistory] = useState<HistoryRoute[]>([]);
  const [showHistory, setShowHistory] = useState(false);

  useEffect(() => {
    fetchVehicles();
    fetchHistory();
  }, []);

  const fetchVehicles = async (): Promise<void> => {
    try {
      const res = await fetch(`${API_URL}/api/v1/vehicles`);
      const json = await res.json();
      setVehicles((json.vehicles ?? []).filter((v: Vehicle) => v.status === 'active'));
    } catch (error) {
      console.error('Failed to fetch vehicles:', error);
    }
  };

  // Φιλτράρισμα οχημάτων βάσει επιλεγμένης ζώνης
  const filteredVehicles = zoneCode === 'ALL'
    ? vehicles
    : vehicles.filter(v => v.zone_code === zoneCode);

  const fetchHistory = async () => {
    try {
      const res = await fetch(`${API_URL}/api/v1/optimization/routes-history`);
      const json = await res.json();
      setHistory(json.routes ?? []);
    } catch (error) {
      console.error('Failed to fetch route history:', error);
    }
  };

  const fetchRoute = async () => {
    setLoading(true);
    setRouteData(null);
```

```
  try {
    const params = new URLSearchParams({
      threshold: threshold.toString(),
      ...(zoneCode !== 'ALL' && { zone_code: zoneCode }),
      ...(vehicleId && { vehicle_id: vehicleId }),
    });
    const res = await fetch(`${API_URL}/api/v1/optimization/route?${params}`);
    const data = await res.json();
    setRouteData(data);
  } catch (error) {
    console.error('Failed to fetch route:', error);
  }
  setLoading(false);
};

const handleSaveRoute = async () => {
  if (!routeData || routeData.total_bins === 0) return;
  setSaving(true);
  try {
    const params = new URLSearchParams({
      threshold: threshold.toString(),
      save_route: 'true',
      ...(zoneCode !== 'ALL' && { zone_code: zoneCode }),
      ...(vehicleId && { vehicle_id: vehicleId }),
    });
    const res = await fetch(`${API_URL}/api/v1/optimization/route?${params}`);
    const data = await res.json();
    setRouteData(data);
    fetchHistory();
  } catch (error) {
    console.error('Failed to save route:', error);
  }
  setSaving(false);
};

const routePoints: [number, number][] =
  routeData?.depot
    ? [
        [routeData.depot.latitude, routeData.depot.longitude],
        ...routeData.route
          .filter(s => s.latitude && s.longitude)
          .map(s => [s.latitude, s.longitude] as [number, number]),
        [routeData.depot.latitude, routeData.depot.longitude],
      ]
    : [];

const getStatusClass = (status: string) =>
  `routes-status-${status}`;
```

```
 return (
   <div className="routes-container">

     {/* ===== SIDEBAR ===== */}
     <aside className="routes-sidebar" aria-label="Παράθυρο βελτιστοποίησης
διαδρομής">

       {/* Controls */}
       <div className="routes-controls">
         <h3 className="routes-controls-title">
           <i className="fi fi-sr-road icon-mr8 icon-blue" aria-hidden="true"></i>
           Βελτιστοποίηση Διαδρομής
         </h3>

         <div className="routes-controls-row">
           {/* Threshold */}
           <div className="routes-control-group">
             <label htmlFor="threshold-select" className="routes-control-label">
               <i className="fi fi-sr-chart-histogram icon-mr6 icon-gray" aria-
hidden="true"></i>
               Κατώφλι Πληρότητας (fill_threshold)
             </label>
             <select
               id="threshold-select"
               value={threshold}
               onChange={e => setThreshold(Number(e.target.value))}
               className="routes-select"
               aria-label="Κατώφλι πληρότητας"
               title="Επιλέξτε κατώφλι πληρότητας"
             >
               <option value={70}>70% — Default (ΠΧ2)</option>
               <option value={60}>60% — Κρίσιμοι + Προσοχή</option>
               <option value={80}>80% — Μόνο Κρίσιμοι</option>
               <option value={50}>50% — Εκτεταμένη</option>
               <option value={40}>40% — Όλοι σχεδόν</option>
             </select>
           </div>

           {/* Zone */}
           <div className="routes-control-group">
             <label htmlFor="zone-select" className="routes-control-label">
               <i className="fi fi-sr-marker icon-mr6 icon-gray" aria-hidden="true"></i>
               Ζώνη (zone_id)
             </label>
             <select
               id="zone-select"
               value={zoneCode}
               onChange={e => { setZoneCode(e.target.value); setVehicleId(''); }}
               className="routes-select"
```

```
            aria-label="Επιλογή ζώνης"
            title="Επιλέξτε ζώνη"
          >
            <option value="ALL">Όλες οι Ζώνες</option>
            <option value="GAZ">Γάζι (GAZ)</option>
            <option value="AMM">Αμμουδάρα (AMM)</option>
            <option value="APE">Αγία Πελαγία (APE)</option>
            <option value="TYL">Τύλισος (TYL)</option>
            <option value="KRO">Κρουσώνας (KRO)</option>
            <option value="ROD">Ροδιά (ROD)</option>
          </select>
        </div>

        {/* Vehicle / Driver */}
        <div className="routes-control-group">
          <label htmlFor="vehicle-select" className="routes-control-label">
            <i className="fi fi-sr-truck-side icon-mr6 icon-gray" aria-hidden="true"></i>
            Αποστολή σε Οδηγό (προαιρετικό)
          </label>
          <select
            id="vehicle-select"
            value={vehicleId}
            onChange={e => setVehicleId(e.target.value)}
            className="routes-select"
            aria-label="Επιλογή οχήματος για αποστολή"
            title="Επιλέξτε οδηγό για αποστολή διαδρομής"
          >
            <option value="">— Χωρίς αποστολή —</option>
            {filteredVehicles.length === 0 ? (
              <option value="" disabled>
                — Δεν υπάρχουν ενεργά οχήματα στη ζώνη —
              </option>
            ) : (
              filteredVehicles.map(v => (
                <option key={v.vehicle_id} value={v.vehicle_id}>
                  {v.vehicle_id} — {v.driver_name}
                </option>
              ))
            )}
          </select>
        </div>
      </div>

      {/* Algorithm Info */}
      <div className="routes-algo-info" role="note" aria-label="Πληροφορίες
αλγορίθμου">
        <i className="fi fi-sr-info icon-mr6 icon-blue2" aria-hidden="true"></i>
        <span>
          Αλγόριθμος: <strong>Nearest Neighbor TSP</strong>
```

```
          <br/>Αποστάσεις: <strong>Haversine</strong> (αεροπορική)
        </span>
      </div>

      {/* Buttons */}
      <button
        onClick={fetchRoute}
        disabled={loading}
        className="routes-calc-btn"
        aria-label="Υπολογισμός βέλτιστης διαδρομής"
        title="Υπολογισμός διαδρομής"
      >
        <i className="fi fi-sr-rotate-right icon-mr6" aria-hidden="true"></i>
        {loading ? 'Υπολογισμός...' : 'Υπολογισμός Διαδρομής'}
      </button>

      {routeData && routeData.total_bins > 0 && !routeData.saved && (
        <button
          onClick={handleSaveRoute}
          disabled={saving}
          className="routes-save-btn"
          aria-label="Αποθήκευση διαδρομής και αποστολή σε οδηγό"
          title="Αποθήκευση και αποστολή"
        >
          <i className="fi fi-sr-disk icon-mr6" aria-hidden="true"></i>
          {saving ? 'Αποθήκευση...' : 'Αποθήκευση & Αποστολή σε Οδηγό'}
        </button>
      )}
    </div>

    {/* Alerts */}
    {routeData?.fallback_used && (
      <div className="routes-alert routes-alert-fallback" role="alert">
        <i className="fi fi-sr-triangle-warning" aria-hidden="true"></i>
        <span>
          <strong>Auto-fallback:</strong> Λιγότεροι από 3 κάδοι με
{routeData.original_threshold}%.
          Χρησιμοποιήθηκε threshold {routeData.threshold}%.
        </span>
      </div>
    )}

    {routeData?.osm_warning && (
      <div className="routes-alert routes-alert-osm" role="note">
        <i className="fi fi-sr-info" aria-hidden="true"></i>
        <span>{routeData.osm_warning}</span>
      </div>
    )}
```

```
      {routeData?.saved && routeData.route_id && (
       <div className="routes-alert routes-alert-success" role="status">
        <i className="fi fi-sr-check-circle" aria-hidden="true"></i>
        <span>
         Διαδρομή #{routeData.route_id} αποθηκεύτηκε!
         {vehicleId && ` Ειδοποίηση στάλθηκε στον οδηγό (${vehicleId}).`}
        </span>
       </div>
      )}

      {/* Stats */}
      {routeData && routeData.total_bins > 0 && (
       <div className="routes-stats" aria-label="Στατιστικά διαδρομής">
        <div className="routes-stat-card blue">
         <div className="routes-stat-value blue">{routeData.total_bins}</div>
         <div className="routes-stat-label">Κάδοι</div>
        </div>
        <div className="routes-stat-card green">
         <div className="routes-stat-value green">{routeData.total_distance_km}</div>
         <div className="routes-stat-label">Χιλιόμετρα</div>
        </div>
        <div className="routes-stat-card orange">
         <div className="routes-stat-value
orange">{routeData.estimated_duration_minutes}'</div>
         <div className="routes-stat-label">Λεπτά</div>
        </div>
        <div className="routes-stat-card red">
         <div className="routes-stat-value red">{routeData.threshold}%</div>
         <div className="routes-stat-label">Κατώφλι</div>
        </div>
       </div>
      )}

      {/* Route List */}
      {loading ? (
       <div className="routes-loading" role="status" aria-live="polite">
        <i className="fi fi-sr-rotate-right" aria-hidden="true"></i>
        <span>Υπολογισμός βέλτιστης διαδρομής...</span>
       </div>
      ) : routeData?.status === 'no_bins' ? (
       <div className="routes-alert-no-bins" role="status">
        <i className="fi fi-sr-check-circle icon-green" aria-hidden="true"></i>
        <p>{routeData.message}</p>
       </div>
      ) : routeData && routeData.total_bins > 0 ? (
       <div className="routes-list" aria-label="Λίστα στάσεων διαδρομής">
        {/* Depot Start */}
        <div className="routes-depot-item" aria-label="Σημείο εκκίνησης">
         <i className="fi fi-sr-flag-checkered icon-mr6" aria-hidden="true"></i>
```

```
          <strong>Εκκίνηση:</strong> {routeData.depot.name}
        </div>

        {/* Stops */}
        {routeData.route.map((stop, idx) => (
         <div
          key={stop.bin_id}
          onClick={() => setActiveStop(activeStop === idx ? null : idx)}
          className={`routes-stop-item ${activeStop === idx ? 'active' : ''}`}
          role="button"
          tabIndex={0}
          aria-label={`Στάση ${stop.stop_number}: ${stop.location_name},
${stop.current_fill_level}%`}
          onKeyDown={e => e.key === 'Enter' && setActiveStop(activeStop === idx ?
null : idx)}
         >
          <div className="routes-stop-header">
           <div className="routes-stop-left">
            <div
             className={`routes-stop-number fill-
${fillBucket(stop.current_fill_level)}`}
             aria-hidden="true"
            >
             {stop.stop_number}
            </div>
            <div>
             <div className="routes-stop-name">{stop.location_name}</div>
             <div className="routes-stop-id">{stop.bin_id}</div>
             {stop.zone_code && (
              <span className={`routes-zone-badge zone-${stop.zone_code}`}>
               {stop.zone_code}
              </span>
             )}
            </div>
           </div>
           <div className="routes-stop-right">
            <div className={`routes-stop-pct fill-
${fillBucket(stop.current_fill_level)}`}>
             {stop.current_fill_level}%
            </div>
            <div className="routes-stop-dist">
             +{stop.distance_from_prev_km}km
            </div>
           </div>
          </div>
         </div>
        ))}

        {/* Depot End */}
```

```
          <div className="routes-depot-item" aria-label="Σημείο επιστροφής">
            <i className="fi fi-sr-flag-checkered icon-mr6" aria-hidden="true"></i>
            <strong>Επιστροφή:</strong> {routeData.depot.name}
          </div>
        </div>
      ) : null}

      {/* History */}
      {history.length > 0 && (
        <div className="routes-history">
          <h4 className="routes-history-title">
            <i className="fi fi-sr-time-past icon-mr6 icon-gray" aria-hidden="true"></i>
            <button
              onClick={() => setShowHistory(!showHistory)}
              className="routes-history-toggle"
              title="Εμφάνιση ιστορικού διαδρομών"
            >
              Ιστορικό Διαδρομών ({history.length})
              <i className={`fi ${showHistory ? 'fi-sr-angle-up' : 'fi-sr-angle-down'}`} aria-
hidden="true"></i>
            </button>
          </h4>
          {showHistory && history.slice(0, 5).map(r => (
            <div key={r.id} className="routes-history-item">
              <div className="routes-history-name">#{r.id} {r.route_name}</div>
              <div className="routes-history-meta">
                <span>
                  <i className="fi fi-sr-calendar icon-mr4" aria-hidden="true"></i>
                  {new Date(r.created_at).toLocaleDateString('el-GR')}
                </span>
                {r.driver_name && (
                  <span>
                    <i className="fi fi-sr-user icon-mr4" aria-hidden="true"></i>
                    {r.driver_name}
                  </span>
                )}
                <span className={getStatusClass(r.status)}>{r.status}</span>
              </div>
            </div>
          ))}
        </div>
      )}
    </aside>

    {/* ===== MAP ===== */}
    <div className="routes-map-panel" aria-label="Χάρτης διαδρομής">
      <MapContainer
        center={[35.33, 25.06]}
        zoom={11}
```

```
  className="routes-map"
>
  <TileLayer
    attribution="&copy; OpenStreetMap"
    url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
  />

  {routeData && routeData.route.length > 0 && (
    <>
      {/* Route line */}
      <Polyline
        positions={routePoints}
        color="#3498db"
        weight={3}
        dashArray="8, 4"
        opacity={0.8}
      />

      {/* Depot marker */}
      <CircleMarker
        center={[routeData.depot.latitude, routeData.depot.longitude]}
        radius={14}
        fillColor="#2c3e50"
        color="white"
        weight={2}
        fillOpacity={1}
      >
        <Popup>
          <div className="routes-popup">
            <h4>
              <i className="fi fi-sr-flag-checkered icon-mr6" aria-hidden="true"></i>
              {routeData.depot.name}
            </h4>
            <p>
              <i className="fi fi-sr-marker icon-mr4" aria-hidden="true"></i>
              Σημείο εκκίνησης/επιστροφής
            </p>
          </div>
        </Popup>
      </CircleMarker>

      {/* Stop markers */}
      {routeData.route.map((stop, idx) => (
        <CircleMarker
          key={stop.bin_id}
          center={[stop.latitude, stop.longitude]}
          radius={14}
          fillColor={getFillColor(stop.current_fill_level)}
          color={activeStop === idx ? '#2c3e50' : 'white'}
```

```
            weight={activeStop === idx ? 3 : 2}
            fillOpacity={0.9}
            eventHandlers={{ click: () => setActiveStop(idx) }}
          >
            <Popup>
              <div className="routes-popup">
                <h4>
                  #{stop.stop_number} {stop.location_name}
                </h4>
                <p>
                  <i className="fi fi-sr-trash icon-mr4" aria-hidden="true"></i>
                  {stop.bin_id}
                </p>
                <p className={`routes-popup-pct fill-
${fillBucket(stop.current_fill_level)}`}>
                  <i className="fi fi-sr-chart-histogram icon-mr4" aria-hidden="true"></i>
                  Πληρότητα: {stop.current_fill_level}%
                </p>
                <p>
                  <i className="fi fi-sr-road icon-mr4" aria-hidden="true"></i>
                  +{stop.distance_from_prev_km} km από προηγούμενο
                </p>
                <p>
                  <i className="fi fi-sr-map icon-mr4" aria-hidden="true"></i>
                  {stop.cumulative_distance_km} km συνολικά
                </p>
                {stop.zone_code && (
                  <p>
                    <i className="fi fi-sr-marker icon-mr4" aria-hidden="true"></i>
                    Ζώνη: {stop.zone_name} ({stop.zone_code})
                  </p>
                )}
              </div>
            </Popup>
          </CircleMarker>
        ))}
      </>
    )}

    {/* Empty state */}
    {!routeData && !loading && (
     <CircleMarker
       center={[35.3259723, 25.0655098]}
       radius={14}
       fillColor="#2c3e50"
       color="white"
       weight={2}
       fillOpacity={1}
     >
```

```
            <Popup>
              <div className="routes-popup">
                <h4>
                  <i className="fi fi-sr-flag-checkered icon-mr6" aria-hidden="true"></i>
                  Δημαρχείο Μαλεβιζίου
                </h4>
                <p>Σημείο εκκίνησης οχημάτων</p>
              </div>
            </Popup>
          </CircleMarker>
        )}
      </MapContainer>
    </div>
  </div>
 );
}

export default RoutesView;
```

Frontend/src/RoutesView.css

```
/* ===== RoutesView.css ===== */

.routes-container {
  display: grid;
  grid-template-columns: 400px 1fr;
  height: calc(100vh - 140px);
}

/* Sidebar */
.routes-sidebar {
  overflow-y: auto;
  background-color: #f5f6fa;
  border-right: 1px solid #ddd;
  display: flex;
  flex-direction: column;
}

/* Controls panel */
.routes-controls {
  padding: 15px;
  background-color: white;
  border-bottom: 1px solid #ddd;
}

.routes-controls-title {
  margin: 0 0 12px 0;
  color: #2c3e50;
  font-size: 1rem;
  display: flex;
```

```
  align-items: center;
  gap: 8px;
}

.routes-controls-row {
  display: flex;
  flex-direction: column;
  gap: 10px;
  margin-bottom: 10px;
}

.routes-control-group {
  display: flex;
  flex-direction: column;
  gap: 4px;
}

.routes-control-label {
  font-size: 0.82rem;
  color: #7f8c8d;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 5px;
}

.routes-select {
  padding: 7px 10px;
  border-radius: 6px;
  border: 1px solid #ddd;
  font-size: 0.9rem;
  font-family: inherit;
  width: 100%;
}

.routes-select:focus {
  outline: none;
  border-color: #3498db;
}

.routes-input {
  padding: 7px 10px;
  border-radius: 6px;
  border: 1px solid #ddd;
  font-size: 0.9rem;
  font-family: inherit;
  width: 100%;
}
```

```
.routes-input:focus {
  outline: none;
  border-color: #3498db;
}

/* Buttons */
.routes-calc-btn {
  width: 100%;
  padding: 9px;
  background-color: #3498db;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  font-size: 0.9rem;
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 6px;
  font-family: inherit;
  margin-bottom: 6px;
}

.routes-calc-btn:hover { background-color: #2980b9; }
.routes-calc-btn:disabled { opacity: 0.6; cursor: not-allowed; }

.routes-save-btn {
  width: 100%;
  padding: 9px;
  background-color: #27ae60;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  font-size: 0.9rem;
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 6px;
  font-family: inherit;
}

.routes-save-btn:hover { background-color: #229954; }
.routes-save-btn:disabled { opacity: 0.6; cursor: not-allowed; }

/* Stats grid */
.routes-stats {
```

```
  padding: 12px 15px;
  background-color: white;
  border-bottom: 1px solid #ddd;
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 8px;
}

.routes-stat-card {
  text-align: center;
  padding: 10px;
  border-radius: 8px;
}

.routes-stat-card.blue   { background-color: #ebf5fb; }
.routes-stat-card.green  { background-color: #eafaf1; }
.routes-stat-card.orange { background-color: #fef9e7; }
.routes-stat-card.red    { background-color: #fdedec; }

.routes-stat-value {
  font-size: 1.5rem;
  font-weight: bold;
}

.routes-stat-value.blue   { color: #3498db; }
.routes-stat-value.green  { color: #27ae60; }
.routes-stat-value.orange { color: #f39c12; }
.routes-stat-value.red    { color: #e74c3c; }

.routes-stat-label {
  font-size: 0.72rem;
  color: #7f8c8d;
  margin-top: 2px;
}

/* Alerts */
.routes-alert {
  margin: 10px 15px 0;
  padding: 8px 12px;
  border-radius: 8px;
  font-size: 0.82rem;
  display: flex;
  align-items: flex-start;
  gap: 7px;
}

.routes-alert-fallback {
  background-color: #fef9e7;
  color: #f39c12;
```

```
  border: 1px solid #f9ca74;
}

.routes-alert-osm {
  background-color: #ebf5fb;
  color: #2980b9;
  border: 1px solid #aed6f1;
}

.routes-alert-success {
  background-color: #d5f5e3;
  color: #27ae60;
  border: 1px solid #a9dfbf;
}

.routes-alert-no-bins {
  padding: 30px 20px;
  text-align: center;
  color: #27ae60;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 8px;
}

.routes-alert-no-bins i {
  font-size: 2rem;
}

/* Route list */
.routes-list {
  padding: 10px;
  flex: 1;
}

/* Depot item */
.routes-depot-item {
  padding: 10px 14px;
  background-color: #2c3e50;
  color: white;
  border-radius: 8px;
  margin-bottom: 8px;
  font-size: 0.85rem;
  display: flex;
  align-items: center;
  gap: 8px;
}

/* Stop item */
```

```
.routes-stop-item {
  padding: 12px;
  background-color: white;
  border-radius: 8px;
  margin-bottom: 8px;
  cursor: pointer;
  border: 2px solid #ecf0f1;
  transition: all 0.2s;
}

.routes-stop-item:hover { border-color: #aed6f1; }
.routes-stop-item.active {
  background-color: #ebf5fb;
  border-color: #3498db;
}

.routes-stop-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.routes-stop-left {
  display: flex;
  align-items: center;
  gap: 10px;
}

.routes-stop-number {
  width: 28px;
  height: 28px;
  border-radius: 50%;
  color: white;
  display: flex;
  align-items: center;
  justify-content: center;
  font-weight: bold;
  font-size: 0.8rem;
  flex-shrink: 0;
  background-color: var(--fill-color, #3498db);
}

.routes-stop-name {
  font-weight: bold;
  font-size: 0.85rem;
  color: #2c3e50;
}

.routes-stop-id {
```

```
  font-size: 0.72rem;
  color: #7f8c8d;
  margin-top: 2px;
}

.routes-stop-right {
  text-align: right;
}

.routes-stop-pct {
  font-weight: bold;
  font-size: 0.95rem;
  color: var(--fill-color, #2c3e50);
}

.routes-stop-dist {
  font-size: 0.72rem;
  color: #7f8c8d;
  margin-top: 2px;
}

/* Zone badge */
.routes-zone-badge {
  display: inline-block;
  padding: 2px 7px;
  border-radius: 8px;
  font-size: 0.7rem;
  font-weight: bold;
  margin-top: 4px;
  background-color: var(--zone-tint, #eaf4fb);
  color: var(--zone-color, #3498db);
}

/* Loading */
.routes-loading {
  padding: 30px 20px;
  text-align: center;
  color: #7f8c8d;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 8px;
}

.routes-loading i { font-size: 1.8rem; color: #3498db; }

/* Map panel */
.routes-map-panel {
  position: relative;
```

```
}

.routes-map-panel > div {
  height: 100%;
  width: 100%;
}

/* Popup styles */
.routes-popup {
  min-width: 180px;
  font-family: 'Consolas', monospace;
}

.routes-popup h4 {
  margin: 0 0 6px;
  color: #2c3e50;
  font-size: 0.9rem;
}

.routes-popup p {
  margin: 3px 0;
  font-size: 0.82rem;
  color: #555;
  display: flex;
  align-items: center;
  gap: 5px;
}

/* History panel */
.routes-history {
  padding: 15px;
  background-color: white;
  border-top: 1px solid #ddd;
}

.routes-history-title {
  margin: 0 0 10px;
  color: #2c3e50;
  font-size: 0.9rem;
  display: flex;
  align-items: center;
  gap: 6px;
}

.routes-history-item {
  padding: 8px 10px;
  border-radius: 6px;
  background-color: #f8f9fa;
  margin-bottom: 6px;
```

```
  font-size: 0.8rem;
  border-left: 3px solid #3498db;
}

.routes-history-name {
  font-weight: bold;
  color: #2c3e50;
  margin-bottom: 3px;
}

.routes-history-meta {
  color: #7f8c8d;
  display: flex;
  gap: 10px;
  flex-wrap: wrap;
}

/* Status badges */
.routes-status-scheduled  { color: #2980b9; background: #ebf5fb; padding: 1px 6px;
border-radius: 6px; }
.routes-status-completed  { color: #27ae60; background: #d5f5e3; padding: 1px 6px;
border-radius: 6px; }
.routes-status-cancelled  { color: #e74c3c; background: #fdecea; padding: 1px 6px;
border-radius: 6px; }
.routes-status-in_progress { color: #f39c12; background: #fef9e7; padding: 1px 6px;
border-radius: 6px; }

/* Responsive */
@media (max-width: 900px) {
  .routes-container {
    grid-template-columns: 1fr;
    height: auto;
  }
  .routes-map-panel {
    height: 400px;
  }
}

/* Algorithm Info */
.routes-algo-info {
  display: flex;
  align-items: flex-start;
  gap: 8px;
  padding: 8px 10px;
  background-color: #ebf5fb;
  border-radius: 8px;
  font-size: 0.82rem;
  color: #2c3e50;
  border: 1px solid #aed6f1;
```

```
}

.routes-algo-info span {
  line-height: 1.5;
}

/*
================================================================================
  ICON UTILITIES + ΔΥΝΑΜΙΚΑ ΧΡΩΜΑΤΑ ()
  ================================================================================ */

.icon-mr4 { margin-right: 4px; }
.icon-mr6 { margin-right: 6px; }
.icon-mr8 { margin-right: 8px; }
.icon-blue  { color: #3498db; }
.icon-blue2 { color: #2980b9; }
.icon-gray  { color: #7f8c8d; }
.icon-green { color: #27ae60; }

/* Χρώμα πληρότητας ως CSS variable */
.fill-ok       { --fill-color: #27ae60; }
.fill-warning  { --fill-color: #f39c12; }
.fill-critical { --fill-color: #e74c3c; }

/* Χρώμα + ανοιχτή απόχρωση ανά ζώνη */
.zone-GAZ { --zone-color: #3498db; --zone-tint: #eaf4fb; }
.zone-AMM { --zone-color: #27ae60; --zone-tint: #e7f7ee; }
.zone-APE { --zone-color: #e74c3c; --zone-tint: #fdeceb; }
.zone-TYL { --zone-color: #f39c12; --zone-tint: #fef4e1; }
.zone-KRO { --zone-color: #9b59b6; --zone-tint: #f3ebf7; }
.zone-ROD { --zone-color: #1abc9c; --zone-tint: #e6f9f5; }

/* Map fills 100% του panel (αντί για inline style στο MapContainer) */
.routes-map {
  height: 100%;
  width: 100%;
}

/* Κουμπί toggle ιστορικού */
.routes-history-toggle {
  background: none;
  border: none;
  cursor: pointer;
  color: #2c3e50;
  font-weight: bold;
  font-size: 0.9rem;
  font-family: inherit;
```

```
  display: flex;
  align-items: center;
  gap: 4px;
  padding: 0;
}

/* Popup πληρότητα (χρώμα από fill-* class) */
.routes-popup-pct {
  color: var(--fill-color, #2c3e50);
  font-weight: bold;
}
```

Frontend/src/TrackingView.tsx

```
/**
 * TrackingView.tsx
 * ================
 * Ζωντανή παρακολούθηση στόλου σε χάρτη: θέσεις οχημάτων (με χρώμα ανά κατάσταση),
 * κρίσιμοι κάδοι, και sidebar με στατιστικά + λίστα οχημάτων (καύσιμα, κάδοι/χλμ ημέρας).
 *
 * Τα δεδομένα ανανεώνονται μόνα τους κάθε 30s.
 *
 * Styling: τα δυναμικά χρώματα (κατάσταση, καύσιμα) ορίζονται από CSS classes.
 * Το πλάτος της μπάρας καυσίμου μπαίνει ως CSS variable μέσω ref (fuelWidthRef).
 */
import React, { useState, useEffect, useRef } from 'react';
import { MapContainer, TileLayer, Marker, Popup, Circle } from 'react-leaflet';
import L from 'leaflet';
import 'leaflet/dist/leaflet.css';
import './TrackingView.css';

const API_URL = 'http://localhost:8000';
const REFRESH_INTERVAL_MS = 30000;

// ---------------------------------------------------------------------------
// Types
// ---------------------------------------------------------------------------

interface VehicleTracking {
  vehicle_id: string;
  plate_number: string;
  vehicle_type: string;
  status: string;
  driver_name: string;
  latitude: number;
  longitude: number;
  fuel_level: number;
  notes: string;
```

```
  bins_collected_today: number;
  km_today: number;
  last_update: string;
}

interface CriticalBin {
  bin_id: string;
  location_name: string;
  current_fill_level: number;
  latitude: number;
  longitude: number;
}

interface TrackingData {
  vehicles: VehicleTracking[];
  critical_bins: CriticalBin[];
  total_critical: number;
  total_vehicles_active: number;
}

// ---------------------------------------------------------------------
// Helpers
// ---------------------------------------------------------------------

/** Κλειδί κατάστασης οχήματος για CSS classes. */
function statusKey(status: string): 'active' | 'maintenance' | 'inactive' {
  if (status === 'active') return 'active';
  if (status === 'maintenance') return 'maintenance';
  return 'inactive';
}

function statusLabel(status: string): string {
  if (status === 'active') return 'Ενεργό';
  if (status === 'maintenance') return 'Συντήρηση';
  return 'Ανενεργό';
}

/** Κατηγορία στάθμης καυσίμου για CSS classes. */
function fuelBucket(level: number): 'ok' | 'mid' | 'low' {
  if (level > 50) return 'ok';
  if (level > 25) return 'mid';
  return 'low';
}

/** Εικονίδιο τύπου οχήματος (πρέσα ή απορριμματοφόρο). */
function vehicleIconClass(type: string): string {
  return type === 'Πρέσα' ? 'fi-sr-compress' : 'fi-sr-truck-side';
}
```

```
/** Ref callback που θέτει το πλάτος της μπάρας καυσίμου ως CSS variable. */
function fuelWidthRef(level: number) {
  return (el: HTMLDivElement | null): void => {
    if (el) el.style.setProperty('--fuel-width', `${level}%`);
  };
}

// Marker οχήματος (χρώμα/κατάσταση από CSS class status-XXX· pulse στα ενεργά).
function createVehicleIcon(status: string, type: string) {
  return L.divIcon({
    html: `<div class="tracking-vehicle-marker status-${statusKey(status)}"><i class="fi
${vehicleIconClass(type)}"></i></div>`,
    className: '',
    iconSize: [36, 36],
    iconAnchor: [18, 18],
  });
}

// Marker κρίσιμου κάδου (κόκκινο >=90%, αλλιώς πορτοκαλί).
function createCriticalBinIcon(fillLevel: number) {
  const level = fillLevel >= 90 ? 'high' : 'med';
  return L.divIcon({
    html: `<div class="tracking-critical-marker ${level}"><i class="fi fi-sr-triangle-
warning"></i></div>`,
    className: '',
    iconSize: [28, 28],
    iconAnchor: [14, 14],
  });
}

// ---------------------------------------------------------------------------
// Component
// ---------------------------------------------------------------------------

function TrackingView() {
  const [data, setData]                   = useState<TrackingData | null>(null);
  const [loading, setLoading]             = useState(true);
  const [lastUpdated, setLastUpdated]     = useState<Date | null>(null);
  const [selectedVehicle, setSelectedVehicle] = useState<string | null>(null);
  const [showCritical, setShowCritical]   = useState(true);
  const intervalRef = useRef<ReturnType<typeof setInterval> | null>(null);

  useEffect(() => {
    fetchTracking();
    intervalRef.current = setInterval(fetchTracking, REFRESH_INTERVAL_MS);
    return () => { if (intervalRef.current) clearInterval(intervalRef.current); };
  }, []);

  const fetchTracking = async (): Promise<void> => {
```

```
  try {
    const res = await fetch(`${API_URL}/api/v1/tracking/vehicles`);
    const json = await res.json();
    setData(json);
    setLastUpdated(new Date());
  } catch (error) {
    console.error('Failed to fetch tracking data:', error);
  } finally {
    setLoading(false);
  }
};

 if (loading) {
  return (
    <div className="tracking-loading" role="status" aria-live="polite">
      <i className="fi fi-sr-rotate-right tracking-loading-icon" aria-hidden="true"></i>
      <p>Φόρτωση δεδομένων tracking...</p>
    </div>
  );
 }

 if (!data) return null;

 const totalBinsToday = data.vehicles.reduce((s, v) => s + v.bins_collected_today, 0);
 const totalKmToday   = data.vehicles.reduce((s, v) => s + v.km_today, 0);

 const statCards = [
   { label: 'Ενεργά',         value: data.total_vehicles_active,   cls: 'stat-green',  icon: 'fi-sr-
truck-side' },
   { label: 'Κρίσιμοι κάδοι', value: data.total_critical,          cls: 'stat-red',    icon: 'fi-sr-
triangle-warning' },
   { label: 'Κάδοι σήμερα',   value: totalBinsToday,               cls: 'stat-blue',   icon: 'fi-sr-
trash' },
   { label: 'Χλμ σήμερα',     value: `${totalKmToday.toFixed(1)}`, cls: 'stat-purple', icon:
'fi-sr-road' },
 ];

 return (
  <div className="tracking-container">

    {/* ===== SIDEBAR ===== */}
    <div className="tracking-sidebar">

      {/* Header */}
      <div className="tracking-header">
        <div className="tracking-header-row">
          <h3 className="tracking-title">
            <i className="fi fi-sr-radar icon-mr8" aria-hidden="true"></i>
            Tracking Οχημάτων
```

```
          </h3>
          {lastUpdated && (
           <span className="tracking-updated">
            <i className="fi fi-sr-rotate-right icon-mr4" aria-hidden="true"></i>
            {lastUpdated.toLocaleTimeString('el-GR')}
           </span>
          )}
         </div>
        </div>

        {/* Stats */}
        <div className="tracking-stats-grid">
         {statCards.map(s => (
          <div key={s.label} className={`tracking-stat-card ${s.cls}`}>
           <i className={`fi ${s.icon} tracking-stat-icon`} aria-hidden="true"></i>
           <div className="tracking-stat-value">{s.value}</div>
           <div className="tracking-stat-label">{s.label}</div>
          </div>
         ))}
        </div>

        {/* Critical bins toggle */}
        {data.total_critical > 0 && (
         <div className="tracking-critical-box">
          <div className="tracking-critical-head">
           <span className="tracking-critical-title">
            <i className="fi fi-sr-triangle-warning icon-mr6" aria-hidden="true"></i>
            {data.total_critical} Κρίσιμοι κάδοι (&gt;80%)
           </span>
           <button onClick={() => setShowCritical(!showCritical)} className="tracking-
critical-toggle">
            {showCritical ? 'Απόκρυψη' : 'Εμφάνιση'}
           </button>
          </div>
          {showCritical && (
           <div className="tracking-critical-list">
            {data.critical_bins.map(bin => (
             <div key={bin.bin_id} className="tracking-critical-item">
              <span className="tracking-critical-item-name">
               <i className="fi fi-sr-trash icon-mr4" aria-hidden="true"></i>
               {bin.bin_id}
              </span>
              <span className="tracking-critical-pct">{bin.current_fill_level}%</span>
             </div>
            ))}
           </div>
          )}
         </div>
        )}
```

```
      {/* Vehicle list */}
      <div className="tracking-vehicle-list">
       <h4 className="tracking-vehicle-list-title">
        <i className="fi fi-sr-truck-side icon-mr6" aria-hidden="true"></i>
        Οχήματα ({data.vehicles.length})
       </h4>
       {data.vehicles.map(v => (
        <div
         key={v.vehicle_id}
         onClick={() => setSelectedVehicle(selectedVehicle === v.vehicle_id ? null :
v.vehicle_id)}
         onKeyDown={e => { if (e.key === 'Enter') setSelectedVehicle(selectedVehicle
=== v.vehicle_id ? null : v.vehicle_id); }}
         className={`tracking-vehicle-card ${selectedVehicle === v.vehicle_id ?
'selected' : ''}`}
         role="button"
         tabIndex={0}
         aria-label={`Όχημα ${v.vehicle_id}, οδηγός ${v.driver_name}`}
        >
         <div className="tracking-vehicle-head">
          <div className="tracking-vehicle-left">
           <div className={`tracking-status-dot status-${statusKey(v.status)}`} aria-
hidden="true" />
           <div>
            <div className="tracking-vehicle-id">{v.vehicle_id} —
{v.plate_number}</div>
            <div className="tracking-vehicle-driver">
             <i className="fi fi-sr-user icon-mr4" aria-hidden="true"></i>
             {v.driver_name}
            </div>
           </div>
          </div>
          <div className="tracking-vehicle-right">
           <div className="tracking-stat-bins">
            <i className="fi fi-sr-trash icon-mr3" aria-hidden="true"></i>
            {v.bins_collected_today} κάδοι
           </div>
           <div className="tracking-stat-km">
            <i className="fi fi-sr-road icon-mr3" aria-hidden="true"></i>
            {v.km_today} km
           </div>
          </div>
         </div>

         {/* Fuel bar */}
         <div className={`tracking-fuel-row fuel-${fuelBucket(v.fuel_level)}`}>
          <i className="fi fi-sr-gas-pump tracking-fuel-icon" aria-hidden="true"></i>
          <div className="tracking-fuel-bar-bg">
```

```
              <div className="tracking-fuel-bar" ref={fuelWidthRef(v.fuel_level)} />
             </div>
             <span className="tracking-fuel-pct">{v.fuel_level}%</span>
            </div>
           </div>
         ))}
        </div>
      </div>

      {/* ===== MAP ===== */}
      <div className="tracking-map-panel">
        <MapContainer center={[35.35, 25.06]} zoom={11} className="tracking-map">
          <TileLayer
            attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>'
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          />

          {/* Vehicle markers */}
          {data.vehicles.map(v => (
            v.latitude && v.longitude ? (
              <Marker key={v.vehicle_id} position={[v.latitude, v.longitude]}
icon={createVehicleIcon(v.status, v.vehicle_type)}>
                <Popup>
                  <div className="tracking-popup">
                    <h4 className="tracking-popup-title">
                      <i className={`fi ${vehicleIconClass(v.vehicle_type)} icon-mr6`} aria-
hidden="true"></i>
                      {v.vehicle_id}
                    </h4>
                    <table className="tracking-popup-table">
                      <tbody>
                        {([
                          ['Πινακίδα', v.plate_number],
                          ['Οδηγός', v.driver_name],
                          ['Κατάσταση', statusLabel(v.status)],
                          ['Καύσιμα', `${v.fuel_level}%`],
                          ['Κάδοι σήμερα', `${v.bins_collected_today}`],
                          ['Χλμ σήμερα', `${v.km_today} km`],
                          ['Τελ. ενημέρωση', v.last_update],
                        ] as [string, React.ReactNode][]).map(([label, value]) => (
                          <tr key={label}>
                            <td>{label}</td>
                            <td>{value}</td>
                          </tr>
                        ))}
                      </tbody>
                    </table>
                    <div className="tracking-popup-coords">
```

```
                <i className="fi fi-sr-marker icon-mr4" aria-hidden="true"></i>
                {v.latitude.toFixed(4)}, {v.longitude.toFixed(4)}
              </div>
            </div>
          </Popup>
        </Marker>
      ) : null
    ))}

    {/* Critical bin markers */}
    {showCritical && data.critical_bins.map(bin => (
      bin.latitude && bin.longitude ? (
        <React.Fragment key={bin.bin_id}>
          <Marker position={[bin.latitude, bin.longitude]}
icon={createCriticalBinIcon(bin.current_fill_level)}>
            <Popup>
              <div className="tracking-critical-popup">
                <h4>
                  <i className="fi fi-sr-triangle-warning icon-mr6" aria-hidden="true"></i>
                  Κρίσιμος Κάδος
                </h4>
                <p><strong>{bin.bin_id}</strong></p>
                <p>{bin.location_name}</p>
                <p className="tracking-critical-popup-pct">Πληρότητα:
{bin.current_fill_level}%</p>
              </div>
            </Popup>
          </Marker>
          <Circle
            center={[bin.latitude, bin.longitude]}
            radius={200}
            color="#e74c3c"
            fillColor="#e74c3c"
            fillOpacity={0.1}
            weight={1}
          />
        </React.Fragment>
      ) : null
    ))}
  </MapContainer>

  {/* Map Legend */}
  <div className="tracking-legend" aria-label="Υπόμνημα χάρτη">
    <div className="tracking-legend-title">Υπόμνημα</div>
    <div className="tracking-legend-row">
      <div className="tracking-legend-dot active" aria-hidden="true" />
      <span>Ενεργό όχημα</span>
    </div>
    <div className="tracking-legend-row">
```

```
          <div className="tracking-legend-dot maintenance" aria-hidden="true" />
          <span>Συντήρηση</span>
        </div>
        <div className="tracking-legend-row">
          <div className="tracking-legend-dot inactive" aria-hidden="true" />
          <span>Ανενεργό</span>
        </div>
        <div className="tracking-legend-row">
          <i className="fi fi-sr-triangle-warning tracking-legend-warn" aria-
hidden="true"></i>
          <span>Κρίσιμος κάδος</span>
        </div>
      </div>
    </div>
  </div>
 );
}

export default TrackingView;
```

frontend/src/TrackingView.css

```
/* ===== TrackingView.css ===== */

.icon-mr3 { margin-right: 3px; }
.icon-mr4 { margin-right: 4px; }
.icon-mr6 { margin-right: 6px; }
.icon-mr8 { margin-right: 8px; }

/*
================================================================================
  ΔΥΝΑΜΙΚΑ ΧΡΩΜΑΤΑ ΜΕΣΩ CSS VARIABLES
  - status-* → χρώμα κατάστασης οχήματος
  - fuel-*   → χρώμα στάθμης καυσίμου
  - stat-*   → χρώμα κάρτας στατιστικού
  Το πλάτος της μπάρας καυσίμου (--fuel-width) ορίζεται από JS μέσω ref.
  ================================================================================ */

.status-active      { --status-color: #27ae60; }
.status-maintenance { --status-color: #f39c12; }
.status-inactive    { --status-color: #e74c3c; }

.fuel-ok  { --fuel-color: #27ae60; }
.fuel-mid { --fuel-color: #f39c12; }
.fuel-low { --fuel-color: #e74c3c; }

.stat-green  { --stat-color: #27ae60; }
.stat-red    { --stat-color: #e74c3c; }
.stat-blue   { --stat-color: #3498db; }
```

```
.stat-purple { --stat-color: #9b59b6; }

/* ===== LAYOUT ===== */
.tracking-container {
  display: grid;
  grid-template-columns: 360px 1fr;
  height: calc(100vh - 145px);
}

.tracking-sidebar {
  overflow-y: auto;
  background-color: #f5f6fa;
  border-right: 2px solid #ecf0f1;
}

/* ===== SIDEBAR HEADER ===== */
.tracking-header {
  padding: 15px;
  background-color: #2c3e50;
  color: white;
}

.tracking-header-row {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.tracking-title {
  margin: 0;
  font-size: 1rem;
  display: flex;
  align-items: center;
}

.tracking-updated {
  font-size: 0.75rem;
  opacity: 0.8;
  display: flex;
  align-items: center;
}

/* ===== STATS ===== */
.tracking-stats-grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 10px;
  padding: 10px;
}
```

```
.tracking-stat-card {
  background-color: white;
  border-radius: 8px;
  padding: 12px;
  text-align: center;
  box-shadow: 0 1px 4px rgba(0, 0, 0, 0.1);
  border-top: 3px solid var(--stat-color, #3498db);
}

.tracking-stat-icon {
  color: var(--stat-color, #3498db);
  font-size: 1.2rem;
}

.tracking-stat-value {
  font-size: 1.4rem;
  font-weight: bold;
  color: var(--stat-color, #3498db);
}

.tracking-stat-label {
  font-size: 0.75rem;
  color: #7f8c8d;
}

/* ===== CRITICAL BINS BOX ===== */
.tracking-critical-box {
  margin: 0 10px 10px;
  background-color: #fdecea;
  border-radius: 8px;
  padding: 10px;
  border: 1px solid #e74c3c;
}

.tracking-critical-head {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.tracking-critical-title {
  color: #e74c3c;
  font-weight: bold;
  font-size: 0.85rem;
  display: flex;
  align-items: center;
}
```

```
.tracking-critical-toggle {
  padding: 3px 8px;
  background-color: #e74c3c;
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-size: 0.75rem;
  font-family: inherit;
}

.tracking-critical-list {
  margin-top: 8px;
  max-height: 120px;
  overflow-y: auto;
}

.tracking-critical-item {
  font-size: 0.75rem;
  padding: 3px 0;
  border-bottom: 1px solid #fadbd8;
  display: flex;
  justify-content: space-between;
}

.tracking-critical-item-name { color: #2c3e50; }
.tracking-critical-pct { color: #e74c3c; font-weight: bold; }

/* ===== VEHICLE LIST ===== */
.tracking-vehicle-list {
  padding: 0 10px 10px;
}

.tracking-vehicle-list-title {
  margin: 0 0 8px;
  color: #2c3e50;
  font-size: 0.9rem;
  display: flex;
  align-items: center;
}

.tracking-vehicle-card {
  background-color: white;
  border-radius: 8px;
  padding: 10px;
  margin-bottom: 6px;
  cursor: pointer;
  border: 2px solid #ecf0f1;
  transition: border-color 0.2s, background-color 0.2s;
```

```
}

.tracking-vehicle-card.selected {
  background-color: #ebf5fb;
  border-color: #3498db;
}

.tracking-vehicle-head {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.tracking-vehicle-left {
  display: flex;
  align-items: center;
  gap: 8px;
}

.tracking-status-dot {
  width: 10px;
  height: 10px;
  border-radius: 50%;
  flex-shrink: 0;
  background-color: var(--status-color, #7f8c8d);
}

/* Μόνο τα ενεργά οχήματα έχουν φωτεινό «glow» */
.tracking-status-dot.status-active {
  box-shadow: 0 0 6px #27ae60;
}

.tracking-vehicle-id {
  font-weight: bold;
  font-size: 0.85rem;
  color: #2c3e50;
}

.tracking-vehicle-driver {
  font-size: 0.75rem;
  color: #7f8c8d;
}

.tracking-vehicle-right {
  text-align: right;
  font-size: 0.75rem;
}

.tracking-stat-bins { color: #3498db; font-weight: bold; }
```

```
.tracking-stat-km { color: #9b59b6; }

/* Fuel bar */
.tracking-fuel-row {
  margin-top: 6px;
  display: flex;
  align-items: center;
  gap: 6px;
}

.tracking-fuel-icon {
  font-size: 0.75rem;
  color: #7f8c8d;
}

.tracking-fuel-bar-bg {
  flex: 1;
  background-color: #ecf0f1;
  border-radius: 10px;
  height: 5px;
}

.tracking-fuel-bar {
  height: 100%;
  border-radius: 10px;
  background-color: var(--fuel-color, #27ae60);
  width: var(--fuel-width, 0%);
}

.tracking-fuel-pct {
  font-size: 0.7rem;
  color: #7f8c8d;
  min-width: 28px;
}

/* ===== MAP ===== */
.tracking-map-panel {
  position: relative;
}

.tracking-map {
  height: 100%;
  width: 100%;
}

/* Markers (Leaflet divIcon) */
.tracking-vehicle-marker {
  width: 36px;
  height: 36px;
```

```
  border-radius: 50%;
  border: 3px solid white;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.4);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 16px;
  color: white;
  background-color: var(--status-color, #7f8c8d);
}

.tracking-vehicle-marker.status-active {
  animation: trackingPulse 2s infinite;
}

.tracking-critical-marker {
  width: 28px;
  height: 28px;
  border-radius: 6px;
  border: 2px solid white;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 13px;
  color: white;
}

.tracking-critical-marker.high { background-color: #e74c3c; }
.tracking-critical-marker.med  { background-color: #f39c12; }

@keyframes trackingPulse {
  0%   { box-shadow: 0 0 0 0 rgba(39, 174, 96, 0.7); }
  70%  { box-shadow: 0 0 0 10px rgba(39, 174, 96, 0); }
  100% { box-shadow: 0 0 0 0 rgba(39, 174, 96, 0); }
}

/* Popup */
.tracking-popup {
  min-width: 220px;
  font-family: 'Consolas', monospace;
}

.tracking-popup-title {
  margin: 0 0 8px;
  color: #2c3e50;
  border-bottom: 2px solid #3498db;
  padding-bottom: 6px;
  display: flex;
```

```
  align-items: center;
}

.tracking-popup-table {
  width: 100%;
  font-size: 0.8rem;
  border-collapse: collapse;
}

.tracking-popup-table tr { border-bottom: 1px solid #ecf0f1; }
.tracking-popup-table td { padding: 4px 6px; }
.tracking-popup-table td:first-child { color: #7f8c8d; font-weight: bold; }
.tracking-popup-table td:last-child { color: #2c3e50; }

.tracking-popup-coords {
  margin-top: 8px;
  font-size: 0.75rem;
  color: #7f8c8d;
  display: flex;
  align-items: center;
}

.tracking-critical-popup {
  font-family: 'Consolas', monospace;
  font-size: 0.85rem;
}

.tracking-critical-popup h4 {
  margin: 0 0 6px;
  color: #e74c3c;
  display: flex;
  align-items: center;
}

.tracking-critical-popup p { margin: 3px 0; }

.tracking-critical-popup-pct {
  color: #e74c3c;
  font-weight: bold;
  font-size: 1rem;
}

/* ===== LEGEND ===== */
.tracking-legend {
  position: absolute;
  bottom: 20px;
  right: 10px;
  background-color: white;
  padding: 10px 15px;
```

```
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);
  z-index: 1000;
  font-size: 0.8rem;
}

.tracking-legend-title {
  font-weight: bold;
  margin-bottom: 6px;
}

.tracking-legend-row {
  display: flex;
  align-items: center;
  gap: 8px;
  margin-bottom: 4px;
}

.tracking-legend-row:last-child { margin-bottom: 0; }

.tracking-legend-dot {
  width: 12px;
  height: 12px;
  border-radius: 50%;
}

.tracking-legend-dot.active      { background-color: #27ae60; }
.tracking-legend-dot.maintenance { background-color: #f39c12; }
.tracking-legend-dot.inactive    { background-color: #e74c3c; }

.tracking-legend-warn { color: #e74c3c; font-size: 1rem; }

/* ===== LOADING ===== */
.tracking-loading {
  text-align: center;
  padding: 40px;
}

.tracking-loading-icon {
  font-size: 2rem;
  color: #3498db;
}

/* ===== RESPONSIVE ===== */
@media (max-width: 900px) {
  .tracking-container {
    grid-template-columns: 1fr;
    height: auto;
  }
```

```
  .tracking-map-panel {
    height: 400px;
  }
}
```

Frontend/src/VehicleViews.tsx

```
/**
 * VehiclesView.tsx
 * ================
 * Διαχείριση/προβολή στόλου απορριμματοφόρων. Φίλτρα (κατάσταση/τύπος/ζώνη),
 * κάρτες στατιστικών, και εναλλαγή ανάμεσα σε προβολή πίνακα και χάρτη.
 * Επιλογή οχήματος στον πίνακα ανοίγει panel λεπτομερειών.
 *
 * Styling: τα δυναμικά χρώματα (κατάσταση, καύσιμα, ζώνη) ορίζονται από CSS classes.
 * Το πλάτος της μπάρας καυσίμου μπαίνει ως CSS variable μέσω ref (fuelWidthRef).
 */
import React, { useState, useEffect } from 'react';
import { MapContainer, TileLayer, Marker, Popup } from 'react-leaflet';
import L from 'leaflet';
import 'leaflet/dist/leaflet.css';
import './VehiclesView.css';

const API_URL = 'http://localhost:8000';

// ---------------------------------------------------------------------------
// Types
// ---------------------------------------------------------------------------

interface Vehicle {
  id: number;
  vehicle_id: string;
  plate_number: string;
  vehicle_type: string;
  capacity_tons: number;
  capacity_m3: number | null;
  status: string;
  driver_name: string;
  last_used: string | null;
  latitude: number | null;
  longitude: number | null;
  fuel_level: number;
  mileage: number;
  year: number;
  notes: string;
  zone_name: string | null;
  zone_code: string | null;
}

// ---------------------------------------------------------------------------
// Helpers
```

```
// ---------------------------------------------------------------------

/** Κλειδί κατάστασης οχήματος για CSS classes. */
function statusKey(status: string): 'active' | 'maintenance' | 'inactive' {
  if (status === 'active') return 'active';
  if (status === 'maintenance') return 'maintenance';
  return 'inactive';
}

function statusLabel(status: string): string {
  if (status === 'active') return 'Ενεργό';
  if (status === 'maintenance') return 'Συντήρηση';
  return 'Ανενεργό';
}

function statusIcon(status: string): string {
  if (status === 'active') return 'fi-sr-check-circle';
  if (status === 'maintenance') return 'fi-sr-tools';
  return 'fi-sr-cross-circle';
}

/** Κατηγορία στάθμης καυσίμου για CSS classes. */
function fuelBucket(level: number): 'ok' | 'mid' | 'low' {
  if (level >= 50) return 'ok';
  if (level >= 25) return 'mid';
  return 'low';
}

/** Ref callback που θέτει το πλάτος της μπάρας καυσίμου ως CSS variable. */
function fuelWidthRef(level: number) {
  return (el: HTMLDivElement | null): void => {
    if (el) el.style.setProperty('--fuel-width', `${level}%`);
  };
}

// Marker οχήματος (χρώμα κατάστασης από CSS class status-XXX).
function createVehicleIcon(status: string) {
  return L.divIcon({
    html: `<div class="vehicles-marker status-${statusKey(status)}"><i class="fi fi-sr-
truck-side"></i></div>`,
    className: '',
    iconSize: [36, 36],
    iconAnchor: [18, 18],
  });
}

// ---------------------------------------------------------------------
// Component
// ---------------------------------------------------------------------
```

```
function VehiclesView() {
  const [vehicles, setVehicles]           = useState<Vehicle[]>([]);
  const [loading, setLoading]             = useState(true);
  const [filterStatus, setFilterStatus]   = useState('ALL');
  const [filterType, setFilterType]       = useState('ALL');
  const [filterZone, setFilterZone]       = useState('ALL');
  const [selectedVehicle, setSelectedVehicle] = useState<Vehicle | null>(null);
  const [showMap, setShowMap]             = useState(false);

  useEffect(() => {
    fetchVehicles();
  }, [filterStatus, filterType]);

  const fetchVehicles = async (): Promise<void> => {
    setLoading(true);
    try {
      const params = new URLSearchParams();
      if (filterStatus !== 'ALL') params.append('status', filterStatus);
      if (filterType !== 'ALL') params.append('vehicle_type', filterType);

      const res = await fetch(`${API_URL}/api/v1/vehicles?${params}`);
      const json = await res.json();
      setVehicles(json.vehicles ?? []);
    } catch (error) {
      console.error('Failed to fetch vehicles:', error);
    } finally {
      setLoading(false);
    }
  };

  // ---------------------------------------------------------------------
  // Derived data
  // ---------------------------------------------------------------------

  // Το φίλτρο ζώνης εφαρμόζεται client-side (τα υπόλοιπα από το API)
  const filteredVehicles = filterZone === 'ALL'
    ? vehicles
    : vehicles.filter(v => v.zone_code === filterZone);

  const vehiclesWithLocation = filteredVehicles.filter(v => v.latitude && v.longitude);

  const uniqueZones = Array.from(
    new Set(vehicles.map(v => v.zone_code).filter(Boolean))
  ) as string[];

  const stats = {
    total:      filteredVehicles.length,
    active:     filteredVehicles.filter(v => v.status === 'active').length,
```

```
  maintenance: filteredVehicles.filter(v => v.status === 'maintenance').length,
  inactive:    filteredVehicles.filter(v => v.status === 'inactive').length,
  trucks:      filteredVehicles.filter(v => v.vehicle_type === 'Απορριμματοφόρο').length,
  press:       filteredVehicles.filter(v => v.vehicle_type === 'Πρέσα').length,
 };

 const statCards = [
  { label: 'Σύνολο',     value: stats.total,       cls: 'stat-blue',   icon: 'fi-sr-truck-side' },
  { label: 'Ενεργά',     value: stats.active,      cls: 'stat-green',  icon: 'fi-sr-check-circle' },
  { label: 'Συντήρηση',  value: stats.maintenance, cls: 'stat-orange', icon: 'fi-sr-tools' },
  { label: 'Ανενεργά',   value: stats.inactive,    cls: 'stat-red',    icon: 'fi-sr-cross-circle' },
  { label: 'Απορ/φόρα',  value: stats.trucks,      cls: 'stat-navy',   icon: 'fi-sr-truck-side' },
  { label: 'Πρέσες',     value: stats.press,       cls: 'stat-purple', icon: 'fi-sr-compress' },
 ];

 const detailItems = selectedVehicle ? [
  { label: 'Τύπος',        value: selectedVehicle.vehicle_type },
  { label: 'Χωρητικότητα', value:
`${selectedVehicle.capacity_tons}τ${selectedVehicle.capacity_m3 ? ` /
${selectedVehicle.capacity_m3}m³` : ''}` },
  { label: 'Κατάσταση',    value: statusLabel(selectedVehicle.status) },
  { label: 'Οδηγός',       value: selectedVehicle.driver_name },
  { label: 'Ζώνη',         value: selectedVehicle.zone_code || '—' },
  { label: 'Καύσιμα',      value: `${selectedVehicle.fuel_level}%` },
  { label: 'Χιλιόμετρα',   value: `${selectedVehicle.mileage?.toLocaleString('el-GR')} km`
},
  { label: 'Έτος',         value: String(selectedVehicle.year) },
 ] : [];

 // ---------------------------------------------------------------------
 // Render
 // ---------------------------------------------------------------------

 return (
  <div className="vehicles-container">

   {/* Header */}
   <div className="vehicles-header">
    <h2 className="vehicles-title">
     <i className="fi fi-sr-truck-side" aria-hidden="true"></i>
     Προβολή Απορριμματοφόρων
    </h2>
    <button
     onClick={() => setShowMap(!showMap)}
     className={`vehicles-toggle-btn ${showMap ? 'active' : ''}`}
     title={showMap ? 'Εμφάνιση πίνακα' : 'Εμφάνιση χάρτη'}
    >
     <i className={`fi ${showMap ? 'fi-sr-list' : 'fi-sr-map'}`} aria-hidden="true"></i>
     {showMap ? 'Προβολή Πίνακα' : 'Προβολή Χάρτη'}
```

```
      </button>
    </div>

    {/* Stats Cards */}
    <div className="vehicles-stats-grid">
      {statCards.map(s => (
        <div key={s.label} className={`vehicles-stat-card ${s.cls}`}>
          <i className={`fi ${s.icon} vehicles-stat-icon`} aria-hidden="true"></i>
          <div className="vehicles-stat-value">{s.value}</div>
          <div className="vehicles-stat-label">{s.label}</div>
        </div>
      ))}
    </div>

    {/* Filters */}
    <div className="vehicles-filters">
      <div className="vehicles-filter-group">
        <i className="fi fi-sr-settings vehicles-filter-icon" aria-hidden="true"></i>
        <label htmlFor="filter-status" className="vehicles-filter-
label">Κατάσταση:</label>
        <select id="filter-status" value={filterStatus} onChange={e =>
setFilterStatus(e.target.value)} className="vehicles-select">
          <option value="ALL">Όλα</option>
          <option value="active">Ενεργά</option>
          <option value="maintenance">Συντήρηση</option>
          <option value="inactive">Ανενεργά</option>
        </select>
      </div>

      <div className="vehicles-filter-group">
        <i className="fi fi-sr-truck-side vehicles-filter-icon" aria-hidden="true"></i>
        <label htmlFor="filter-type" className="vehicles-filter-label">Τύπος:</label>
        <select id="filter-type" value={filterType} onChange={e =>
setFilterType(e.target.value)} className="vehicles-select">
          <option value="ALL">Όλα</option>
          <option value="Απορριμματοφόρο">Απορριμματοφόρο</option>
          <option value="Πρέσα">Πρέσα</option>
        </select>
      </div>

      <div className="vehicles-filter-group">
        <i className="fi fi-sr-marker vehicles-filter-icon" aria-hidden="true"></i>
        <label htmlFor="filter-zone" className="vehicles-filter-label">Ζώνη:</label>
        <select id="filter-zone" value={filterZone} onChange={e =>
setFilterZone(e.target.value)} className="vehicles-select">
          <option value="ALL">Όλες</option>
          {uniqueZones.map(zc => (
            <option key={zc} value={zc}>{zc}</option>
          ))}
```

```
        </select>
      </div>

      <span className="vehicles-filter-count">{filteredVehicles.length} οχήματα</span>
    </div>

    {loading ? (
      <div className="vehicles-loading" role="status" aria-live="polite">
        <i className="fi fi-sr-rotate-right vehicles-loading-icon" aria-hidden="true"></i>
      </div>
    ) : showMap ? (

      /* Map View */
      <div className="vehicles-map-wrap">
        <MapContainer center={[35.35, 25.06]} zoom={11} className="vehicles-map">
          <TileLayer
            attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>'
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          />
          {vehiclesWithLocation.map(vehicle => (
            <Marker
              key={vehicle.vehicle_id}
              position={[vehicle.latitude!, vehicle.longitude!]}
              icon={createVehicleIcon(vehicle.status)}
            >
              <Popup>
                <div className="vehicles-popup">
                  <h4 className="vehicles-popup-title">
                    <i className="fi fi-sr-truck-side icon-mr6" aria-hidden="true"></i>
                    {vehicle.vehicle_id}
                  </h4>
                  <table className="vehicles-popup-table">
                    <tbody>
                      {([
                        ['Πινακίδα', vehicle.plate_number],
                        ['Τύπος', vehicle.vehicle_type],
                        ['Οδηγός', vehicle.driver_name],
                        ['Ζώνη', vehicle.zone_name || '—'],
                        ['Κατάσταση', statusLabel(vehicle.status)],
                        ['Καύσιμα', `${vehicle.fuel_level}%`],
                        ['Χωρητικότητα', `${vehicle.capacity_tons}τ${vehicle.capacity_m3 ? ` /
${vehicle.capacity_m3}m³` : ''}`],
                      ] as [string, React.ReactNode][]).map(([label, value]) => (
                        <tr key={label}>
                          <td>{label}</td>
                          <td>{value}</td>
                        </tr>
                      ))}
```

```
                </tbody>
              </table>
            </div>
          </Popup>
        </Marker>
      ))}
    </MapContainer>
  </div>

) : (

  /* Table View */
  <div className="vehicles-table-card">
    <div className="vehicles-table-wrap">
      <table className="vehicles-table">
        <thead>
          <tr>
            {['ID', 'Πινακίδα', 'Τύπος', 'Χωρητικότητα', 'Ζώνη', 'Κατάσταση', 'Οδηγός',
'Καύσιμα', 'Χιλιόμετρα', 'Έτος', 'Σημειώσεις'].map(h => (
              <th key={h}>{h}</th>
            ))}
          </tr>
        </thead>
        <tbody>
          {filteredVehicles.map(v => {
            const isSelected = selectedVehicle?.vehicle_id === v.vehicle_id;
            return (
              <tr
                key={v.vehicle_id}
                className={isSelected ? 'selected' : ''}
                onClick={() => setSelectedVehicle(isSelected ? null : v)}
              >
                <td className="vehicles-id">{v.vehicle_id}</td>
                <td className="vehicles-plate">{v.plate_number}</td>
                <td>
                  <i className={`fi ${v.vehicle_type === 'Απορριμματοφόρο' ? 'fi-sr-truck-
side' : 'fi-sr-compress'} icon-mr4`} aria-hidden="true"></i>
                  {v.vehicle_type}
                </td>
                <td>
                  {v.capacity_tons}τ
                  {v.capacity_m3 && <span className="vehicles-capacity-sub"> /
{v.capacity_m3}m³</span>}
                </td>
                <td>
                  {v.zone_code ? (
                    <span className={`vehicles-zone-badge zone-${v.zone_code}`}>
                      <i className="fi fi-sr-marker" aria-hidden="true"></i>
                      {v.zone_code}
```

```
                  </span>
                ) : '—'}
              </td>
              <td>
                <span className={`vehicles-status-badge status-${statusKey(v.status)}`}>
                  <i className={`fi ${statusIcon(v.status)}`} aria-hidden="true"></i>
                  {statusLabel(v.status)}
                </span>
              </td>
              <td>
                <i className="fi fi-sr-user vehicles-driver-icon" aria-hidden="true"></i>
                {v.driver_name}
              </td>
              <td>
                <div className={`vehicles-fuel-cell fuel-${fuelBucket(v.fuel_level)}`}>
                  <div className="vehicles-fuel-bar-bg">
                    <div className="vehicles-fuel-bar" ref={fuelWidthRef(v.fuel_level)}
/>
                  </div>
                  <span className="vehicles-fuel-pct">
                    <i className="fi fi-sr-gas-pump" aria-hidden="true"></i>
                    {v.fuel_level}%
                  </span>
                </div>
              </td>
              <td>{v.mileage?.toLocaleString('el-GR')} km</td>
              <td>{v.year}</td>
              <td className="vehicles-notes">{v.notes}</td>
            </tr>
          );
        })}
      </tbody>
    </table>
  </div>
</div>
)}

{/* Selected Vehicle Detail */}
{selectedVehicle && !showMap && (
  <div className={`vehicles-detail-card status-
${statusKey(selectedVehicle.status)}`}>
    <h3 className="vehicles-detail-title">
      <i className="fi fi-sr-truck-side" aria-hidden="true"></i>
      Λεπτομέρειες — {selectedVehicle.vehicle_id} ({selectedVehicle.plate_number})
    </h3>
    <div className="vehicles-detail-grid">
      {detailItems.map(item => (
        <div key={item.label} className="vehicles-detail-item">
          <div className="vehicles-detail-label">{item.label}</div>
```

```
            <div className="vehicles-detail-value">{item.value}</div>
          </div>
        ))}
      </div>
    </div>
  )}
 </div>
);
}

export default VehiclesView;
```

Frontend/src/VehiclesView.css

```
/* ===== VehiclesView.css ===== */

.icon-mr4 { margin-right: 4px; }
.icon-mr6 { margin-right: 6px; }
.icon-mr8 { margin-right: 8px; }

/*
================================================================================
  ΔΥΝΑΜΙΚΑ ΧΡΩΜΑΤΑ ΜΕΣΩ CSS VARIABLES
  - status-* → χρώμα + απόχρωση κατάστασης οχήματος
  - fuel-*   → χρώμα στάθμης καυσίμου
  - zone-*   → χρώμα + απόχρωση ζώνης
  - stat-*   → χρώμα κάρτας στατιστικού
  Το πλάτος της μπάρας καυσίμου (--fuel-width) ορίζεται από JS μέσω ref.
  ================================================================================ */

.status-active      { --status-color: #27ae60; --status-tint: #d5f5e3; }
.status-maintenance { --status-color: #f39c12; --status-tint: #fef9e7; }
.status-inactive    { --status-color: #e74c3c; --status-tint: #fdecea; }

.fuel-ok  { --fuel-color: #27ae60; }
.fuel-mid { --fuel-color: #f39c12; }
.fuel-low { --fuel-color: #e74c3c; }

.zone-GAZ { --zone-color: #3498db; --zone-tint: #eaf4fb; }
.zone-AMM { --zone-color: #27ae60; --zone-tint: #e7f7ee; }
.zone-APE { --zone-color: #e74c3c; --zone-tint: #fdeceb; }
.zone-TYL { --zone-color: #f39c12; --zone-tint: #fef4e1; }
.zone-KRO { --zone-color: #9b59b6; --zone-tint: #f3ebf7; }
.zone-ROD { --zone-color: #1abc9c; --zone-tint: #e6f9f5; }

.stat-blue   { --stat-color: #3498db; }
.stat-green  { --stat-color: #27ae60; }
.stat-orange { --stat-color: #f39c12; }
```

```
.stat-red    { --stat-color: #e74c3c; }
.stat-navy   { --stat-color: #2c3e50; }
.stat-purple { --stat-color: #8e44ad; }

/* ===== CONTAINER ===== */
.vehicles-container {
  padding: 20px;
  max-width: 1300px;
  margin: 0 auto;
}

/* ===== HEADER ===== */
.vehicles-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.vehicles-title {
  margin: 0;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
}

.vehicles-toggle-btn {
  padding: 8px 16px;
  background-color: #3498db;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
  font-family: inherit;
  display: flex;
  align-items: center;
  gap: 6px;
  transition: background-color 0.2s;
}

.vehicles-toggle-btn.active { background-color: #2c3e50; }

/* ===== STATS ===== */
.vehicles-stats-grid {
  display: grid;
  grid-template-columns: repeat(6, 1fr);
  gap: 15px;
```

```
  margin-bottom: 20px;
}

.vehicles-stat-card {
  background-color: white;
  border-radius: 10px;
  padding: 15px;
  text-align: center;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  border-top: 4px solid var(--stat-color, #3498db);
}

.vehicles-stat-icon {
  color: var(--stat-color, #3498db);
  font-size: 1.1rem;
}

.vehicles-stat-value {
  font-size: 1.8rem;
  font-weight: bold;
  color: var(--stat-color, #3498db);
}

.vehicles-stat-label {
  font-size: 0.75rem;
  color: #7f8c8d;
  margin-top: 4px;
}

/* ===== FILTERS ===== */
.vehicles-filters {
  background-color: white;
  padding: 15px 20px;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  margin-bottom: 20px;
  display: flex;
  gap: 20px;
  align-items: center;
  flex-wrap: wrap;
}

.vehicles-filter-group {
  display: flex;
  align-items: center;
  gap: 8px;
}

.vehicles-filter-icon { color: #7f8c8d; }
```

```
.vehicles-filter-label {
  font-weight: bold;
  font-size: 0.9rem;
}

.vehicles-select {
  padding: 6px 10px;
  border: 2px solid #ecf0f1;
  border-radius: 6px;
  font-size: 0.9rem;
  font-family: inherit;
  outline: none;
  transition: border-color 0.2s;
}

.vehicles-select:focus { border-color: #3498db; }

.vehicles-filter-count {
  margin-left: auto;
  color: #7f8c8d;
  font-size: 0.9rem;
}

/* ===== LOADING ===== */
.vehicles-loading {
  text-align: center;
  padding: 40px;
}

.vehicles-loading-icon {
  font-size: 2rem;
  color: #3498db;
}

/* ===== MAP ===== */
.vehicles-map-wrap {
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  height: 500px;
}

.vehicles-map {
  height: 100%;
  width: 100%;
}

/* Marker (Leaflet divIcon) */
```

```
.vehicles-marker {
  width: 36px;
  height: 36px;
  border-radius: 50%;
  border: 3px solid white;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.4);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 16px;
  color: white;
  background-color: var(--status-color, #7f8c8d);
}

/* Popup */
.vehicles-popup {
  min-width: 220px;
  font-family: 'Consolas', monospace;
}

.vehicles-popup-title {
  margin: 0 0 8px;
  color: #2c3e50;
  border-bottom: 2px solid #3498db;
  padding-bottom: 6px;
  display: flex;
  align-items: center;
}

.vehicles-popup-table {
  width: 100%;
  font-size: 0.8rem;
  border-collapse: collapse;
}

.vehicles-popup-table tr { border-bottom: 1px solid #ecf0f1; }
.vehicles-popup-table td { padding: 4px 6px; }
.vehicles-popup-table td:first-child { color: #7f8c8d; font-weight: bold; }
.vehicles-popup-table td:last-child { color: #2c3e50; }

/* ===== TABLE ===== */
.vehicles-table-card {
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  overflow: hidden;
}

.vehicles-table-wrap {
```

```
  overflow-x: auto;
}

.vehicles-table {
  width: 100%;
  border-collapse: collapse;
}

.vehicles-table thead tr {
  background-color: #2c3e50;
  color: white;
}

.vehicles-table th {
  padding: 12px 15px;
  text-align: left;
  font-size: 0.85rem;
}

.vehicles-table td {
  padding: 12px 15px;
  font-size: 0.85rem;
}

.vehicles-table tbody tr {
  cursor: pointer;
}

.vehicles-table tbody tr:nth-child(odd) { background-color: #f8f9fa; }
.vehicles-table tbody tr.selected { background-color: #ebf5fb; }

.vehicles-id { font-weight: bold; color: #3498db; }
.vehicles-plate { font-weight: bold; }
.vehicles-capacity-sub { color: #7f8c8d; font-size: 0.75rem; }

.vehicles-zone-badge {
  padding: 3px 8px;
  border-radius: 10px;
  font-size: 0.8rem;
  font-weight: bold;
  display: inline-flex;
  align-items: center;
  gap: 4px;
  background-color: var(--zone-tint, #eaf4fb);
  color: var(--zone-color, #3498db);
}

.vehicles-status-badge {
  padding: 3px 10px;
```

```
  border-radius: 12px;
  font-size: 0.75rem;
  font-weight: bold;
  display: inline-flex;
  align-items: center;
  gap: 4px;
  background-color: var(--status-tint, #ecf0f1);
  color: var(--status-color, #7f8c8d);
}

.vehicles-driver-icon { color: #7f8c8d; margin-right: 4px; }

/* Fuel cell */
.vehicles-fuel-cell {
  display: flex;
  align-items: center;
  gap: 6px;
}

.vehicles-fuel-bar-bg {
  flex: 1;
  background-color: #ecf0f1;
  border-radius: 10px;
  height: 6px;
  min-width: 50px;
}

.vehicles-fuel-bar {
  height: 100%;
  border-radius: 10px;
  background-color: var(--fuel-color, #27ae60);
  width: var(--fuel-width, 0%);
}

.vehicles-fuel-pct {
  color: var(--fuel-color, #27ae60);
  font-weight: bold;
  font-size: 0.8rem;
  display: flex;
  align-items: center;
  gap: 3px;
}

.vehicles-notes { color: #7f8c8d; font-size: 0.8rem; }

/* ===== DETAIL CARD ===== */
.vehicles-detail-card {
  background-color: white;
  border-radius: 10px;
```

```
  padding: 20px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  margin-top: 20px;
  border-left: 4px solid var(--status-color, #3498db);
}

.vehicles-detail-title {
  margin: 0 0 15px;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
}

.vehicles-detail-grid {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 15px;
}

.vehicles-detail-item {
  padding: 10px;
  background-color: #f8f9fa;
  border-radius: 8px;
}

.vehicles-detail-label {
  font-size: 0.75rem;
  color: #7f8c8d;
  margin-bottom: 4px;
}

.vehicles-detail-value {
  font-weight: bold;
  font-size: 0.9rem;
  color: #2c3e50;
}

/* ===== RESPONSIVE ===== */
@media (max-width: 1024px) {
  .vehicles-stats-grid {
    grid-template-columns: repeat(3, 1fr);
  }
  .vehicles-detail-grid {
    grid-template-columns: repeat(2, 1fr);
  }
}

@media (max-width: 768px) {
```

```
  .vehicles-stats-grid {
    grid-template-columns: repeat(2, 1fr);
  }
  .vehicles-header {
    flex-direction: column;
    align-items: flex-start;
    gap: 12px;
  }
}
```

Frontend/src/CitizenReportsView.tsx

```
/**
 * CitizenReportsView.tsx
 * ======================
 * Διαχείριση αναφορών πολιτών (μέσω της πύλης πολιτών). Εμφανίζει λίστα αναφορών
 * με φίλτρα (κατάσταση/τύπος), στατιστικά, και panel λεπτομερειών με δυνατότητα
 * αλλαγής κατάστασης. Η θέση κάθε αναφοράς προβάλλεται σε modal χάρτη
(MapModal).
 *
 * Styling: όλα τα χρώματα/καταστάσεις ορίζονται από CSS classes (cr-*).
 */
import React, { useState, useEffect } from 'react';
import { MapContainer, TileLayer, CircleMarker, Marker, Popup } from 'react-leaflet';
import L from 'leaflet';
import 'leaflet/dist/leaflet.css';
import './CitizenReportsView.css';

const API_URL = 'http://localhost:8000';

interface CitizenReport {
  id: number;
  tracking_code: string;
  report_type: string;
  title: string;
  description: string;
  latitude: number;
  longitude: number;
  status: string;
  priority: string;
  citizen_name: string;
  citizen_email: string;
  bin_id: string | null;
  is_pin_location: boolean;
  created_at: string;
  resolved_at: string | null;
}

// ===== MAP MODAL =====
const reportPinIcon = L.divIcon({
```

```
  html: '<div class="cr-map-pin-icon" aria-label="Θέση αναφοράς"><i class="fi fi-sr-map-
marker"></i></div>',
  className: '',
  iconSize: [36, 36],
  iconAnchor: [18, 36],
});

interface MapModalProps {
  lat: number;
  lon: number;
  title: string;
  onClose: () => void;
}

function MapModal({ lat, lon, title, onClose }: MapModalProps) {
  // Κλείσιμο με Escape
  useEffect(() => {
    const handleKey = (e: KeyboardEvent) => {
      if (e.key === 'Escape') onClose();
    };
    window.addEventListener('keydown', handleKey);
    return () => window.removeEventListener('keydown', handleKey);
  }, [onClose]);

  return (
    <div
      className="cr-modal-overlay"
      role="dialog"
      aria-modal="true"
      aria-label="Χάρτης θέσης αναφοράς"
      onClick={e => { if (e.target === e.currentTarget) onClose(); }}
    >
      <div className="cr-modal-box">
        <div className="cr-modal-header">
          <h3 className="cr-modal-title">
            <i className="fi fi-sr-map icon-mr8 icon-blue" aria-hidden="true"></i>
            {title}
          </h3>
          <button
            onClick={onClose}
            className="cr-modal-close"
            aria-label="Κλείσιμο χάρτη"
            title="Κλείσιμο (Esc)"
          >
            <i className="fi fi-sr-cross" aria-hidden="true"></i>
          </button>
        </div>
        <div className="cr-modal-map-wrap">
          <MapContainer
```

```
          center={[lat, lon]}
          zoom={17}
          className="cr-modal-map"
          scrollWheelZoom={true}
        >
          <TileLayer
            attribution="&copy; OpenStreetMap"
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          />
          <Marker position={[lat, lon]} icon={reportPinIcon}>
            <Popup>
              <div className="cr-modal-popup">
                <strong className="cr-modal-popup-title">{title}</strong>
                <br/>
                <small className="cr-modal-popup-coords">
                  {lat.toFixed(6)}, {lon.toFixed(6)}
                </small>
              </div>
            </Popup>
          </Marker>
        </MapContainer>
      </div>
      <div className="cr-modal-footer">
        <span className="cr-modal-coords">
          <i className="fi fi-sr-marker icon-mr4 icon-gray" aria-hidden="true"></i>
          {lat.toFixed(6)}, {lon.toFixed(6)}
        </span>
        <button
          onClick={onClose}
          className="cr-modal-back-btn"
          aria-label="Επιστροφή στις αναφορές πολιτών"
          title="Επιστροφή"
        >
          <i className="fi fi-sr-arrow-left icon-mr6" aria-hidden="true"></i>
          Επιστροφή στις Αναφορές
        </button>
      </div>
    </div>
  </div>
 );
}

const TYPE_LABELS: Record<string, string> = {
  overflow:  'Υπερχείλιση',
  missed:    'Χαμένη Αποκομιδή',
  damage:    'Ζημιά Κάδου',
  new_bin:   'Νέος Κάδος',
  other:     'Άλλο',
};
```

```
const TYPE_ICONS: Record<string, string> = {
  overflow:  'fi-sr-water',
  missed:    'fi-sr-truck-side',
  damage:    'fi-sr-tools',
  new_bin:   'fi-sr-add',
  other:     'fi-sr-comment',
};

const STATUS_LABELS: Record<string, string> = {
  open:        'Ανοιχτή',
  in_progress: 'Σε Εξέλιξη',
  resolved:    'Επιλύθηκε',
  closed:      'Κλειστή',
};

const PRIORITY_LABELS: Record<string, string> = {
  low:      'Χαμηλή',
  medium:   'Μέτρια',
  high:     'Υψηλή',
  critical: 'Κρίσιμη',
};

function CitizenReportsView() {
  const [reports, setReports] = useState<CitizenReport[]>([]);
  const [loading, setLoading] = useState(true);
  const [filterStatus, setFilterStatus] = useState('ALL');
  const [filterType, setFilterType] = useState('ALL');
  const [selectedReport, setSelectedReport] = useState<CitizenReport | null>(null);
  const [message, setMessage] = useState('');
  const [mapModal, setMapModal] = useState<{ lat: number; lon: number; title: string } |
null>(null);

  useEffect(() => { fetchReports(); }, [filterStatus, filterType]);

  const fetchReports = async (): Promise<void> => {
    setLoading(true);
    try {
      const params = new URLSearchParams();
      if (filterStatus !== 'ALL') params.append('status', filterStatus);
      if (filterType !== 'ALL') params.append('report_type', filterType);
      const res = await fetch(`${API_URL}/api/v1/citizen/reports?${params}`);
      const json = await res.json();
      setReports(json.reports ?? []);
    } catch (error) {
      console.error('Failed to fetch citizen reports:', error);
    }
    setLoading(false);
  };
```

```
const updateStatus = async (reportId: number, newStatus: string): Promise<void> => {
  try {
    const res = await fetch(`${API_URL}/api/v1/citizen/reports/${reportId}`, {
      method: 'PUT',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ status: newStatus }),
    });
    if (res.ok) {
      setMessage(`Η κατάσταση ενημερώθηκε σε "${STATUS_LABELS[newStatus]}".`);
      fetchReports();
      if (selectedReport?.id === reportId) {
        setSelectedReport({ ...selectedReport, status: newStatus });
      }
      setTimeout(() => setMessage(''), 3000);
    }
  } catch (error) {
    console.error('Failed to update report status:', error);
  }
};

const getStatusClass = (status: string) => `cr-status-${status}`;
const getPriorityClass = (priority: string) => `cr-priority-${priority}`;

// Stats
const stats = {
  total:       reports.length,
  open:        reports.filter(r => r.status === 'open').length,
  in_progress: reports.filter(r => r.status === 'in_progress').length,
  resolved:    reports.filter(r => r.status === 'resolved').length,
};

return (
  <>
  <div className="cr-container">

    {/* Header + Stats */}
    <div className="cr-header">
      <div className="cr-header-top">
        <h2 className="cr-title">
          <i className="fi fi-sr-comment icon-mr8 icon-blue" aria-hidden="true"></i>
          Αναφορές Πολιτών
        </h2>
        <div className="cr-header-actions">
          <a
            href="http://localhost:8081/citizen-portal.html"
            target="_blank"
            rel="noopener noreferrer"
            className="cr-portal-link"
```

```
        aria-label="Άνοιγμα πύλης πολιτών"
        title="Πύλη Πολιτών"
      >
        <i className="fi fi-sr-globe icon-mr6" aria-hidden="true"></i>
        Πύλη Πολιτών
      </a>
      <button
        onClick={fetchReports}
        className="cr-refresh-btn"
        aria-label="Ανανέωση αναφορών"
        title="Ανανέωση"
      >
        <i className="fi fi-sr-rotate-right" aria-hidden="true"></i>
      </button>
    </div>
  </div>

  {/* Stats row */}
  <div className="cr-stats" aria-label="Στατιστικά αναφορών">
    <div className="cr-stat-card">
      <div className="cr-stat-val cr-stat-blue">{stats.total}</div>
      <div className="cr-stat-label">Σύνολο</div>
    </div>
    <div className="cr-stat-card">
      <div className="cr-stat-val cr-stat-red">{stats.open}</div>
      <div className="cr-stat-label">Ανοιχτές</div>
    </div>
    <div className="cr-stat-card">
      <div className="cr-stat-val cr-stat-orange">{stats.in_progress}</div>
      <div className="cr-stat-label">Σε Εξέλιξη</div>
    </div>
    <div className="cr-stat-card">
      <div className="cr-stat-val cr-stat-green">{stats.resolved}</div>
      <div className="cr-stat-label">Επιλύθηκαν</div>
    </div>
  </div>

  {/* Filters */}
  <div className="cr-filters" aria-label="Φίλτρα αναφορών">
    <div className="cr-filter-group">
      <label htmlFor="filter-status" className="cr-filter-label">
        <i className="fi fi-sr-settings icon-mr4 icon-gray" aria-hidden="true"></i>
        Κατάσταση
      </label>
      <select
        id="filter-status"
        value={filterStatus}
        onChange={e => setFilterStatus(e.target.value)}
        className="cr-select"
```

```
        aria-label="Φίλτρο κατάστασης"
        title="Φίλτρο κατάστασης"
      >
        <option value="ALL">Όλες</option>
        <option value="open">Ανοιχτές</option>
        <option value="in_progress">Σε Εξέλιξη</option>
        <option value="resolved">Επιλυμένες</option>
        <option value="closed">Κλειστές</option>
      </select>
    </div>
    <div className="cr-filter-group">
      <label htmlFor="filter-type" className="cr-filter-label">
        <i className="fi fi-sr-list icon-mr4 icon-gray" aria-hidden="true"></i>
        Τύπος
      </label>
      <select
        id="filter-type"
        value={filterType}
        onChange={e => setFilterType(e.target.value)}
        className="cr-select"
        aria-label="Φίλτρο τύπου αναφοράς"
        title="Φίλτρο τύπου"
      >
        <option value="ALL">Όλοι</option>
        <option value="overflow">Υπερχείλιση</option>
        <option value="missed">Χαμένη Αποκομιδή</option>
        <option value="damage">Ζημιά Κάδου</option>
        <option value="new_bin">Νέος Κάδος</option>
        <option value="other">Άλλο</option>
      </select>
    </div>
  </div>
</div>

{/* Message */}
{message && (
  <div className="cr-message" role="status" aria-live="polite">
    <i className="fi fi-sr-check-circle icon-mr6 icon-green" aria-hidden="true"></i>
    {message}
  </div>
)}

{/* Content */}
<div className="cr-content">

  {/* Reports List */}
  <div className="cr-list" aria-label="Λίστα αναφορών πολιτών">
    {loading ? (
      <div className="cr-loading" role="status">
```

```
          <i className="fi fi-sr-rotate-right icon-blue" aria-hidden="true"></i>
          <span>Φόρτωση...</span>
         </div>
        ) : reports.length === 0 ? (
         <div className="cr-empty">
          <i className="fi fi-sr-check-circle" aria-hidden="true"></i>
          <p>Δεν υπάρχουν αναφορές με τα επιλεγμένα φίλτρα.</p>
         </div>
        ) : (
         reports.map(report => (
          <div
           key={report.id}
           className={`cr-report-item ${selectedReport?.id === report.id ? 'selected' :
"}`}
           onClick={() => setSelectedReport(report)}
           role="button"
           tabIndex={0}
           aria-label={`Αναφορά ${report.tracking_code}: ${report.title}`}
           onKeyDown={e => e.key === 'Enter' && setSelectedReport(report)}
          >
           <div className="cr-report-header">
            <div className="cr-report-left">
             <div className="cr-type-icon">
              <i className={`fi ${TYPE_ICONS[report.report_type] || 'fi-sr-comment'}`}
aria-hidden="true"></i>
             </div>
             <div>
              <div className="cr-report-title">{report.title}</div>
              <div className="cr-report-meta">
               <span className="cr-tracking">{report.tracking_code}</span>
               <span className="cr-dot" aria-hidden="true">·</span>
               <span>{TYPE_LABELS[report.report_type] || report.report_type}</span>
               <span className="cr-dot" aria-hidden="true">·</span>
               <span>{report.citizen_name}</span>
              </div>
             </div>
            </div>
            <div className="cr-report-right">
             <span className={`cr-status-badge ${getStatusClass(report.status)}`}>
              {STATUS_LABELS[report.status] || report.status}
             </span>
             <span className={`cr-priority-badge ${getPriorityClass(report.priority)}`}>
              {PRIORITY_LABELS[report.priority] || report.priority}
             </span>
            </div>
           </div>
           <div className="cr-report-date">
            <i className="fi fi-sr-calendar icon-mr4 icon-gray" aria-hidden="true"></i>
            {new Date(report.created_at).toLocaleString('el-GR', {
```

```
                day: '2-digit', month: '2-digit', year: 'numeric',
                hour: '2-digit', minute: '2-digit'
              })}
              {report.bin_id && (
                <span className="cr-bin-badge">
                  <i className="fi fi-sr-trash icon-mr4" aria-hidden="true"></i>
                  {report.bin_id}
                </span>
              )}
              {report.is_pin_location && (
                <span className="cr-pin-badge">
                  <i className="fi fi-sr-map-marker icon-mr4" aria-hidden="true"></i>
                  Καρφίτσα
                </span>
              )}
            </div>
          </div>
        ))
      )}
    </div>

    {/* Detail Panel */}
    {selectedReport && (
      <div className="cr-detail" aria-label="Λεπτομέρειες αναφοράς">
        <div className="cr-detail-header">
          <h3 className="cr-detail-title">
            <i className={`fi ${TYPE_ICONS[selectedReport.report_type] || 'fi-sr-
comment'} icon-mr8 icon-blue`} aria-hidden="true"></i>
            {selectedReport.title}
          </h3>
          <button
            onClick={() => setSelectedReport(null)}
            className="cr-close-btn"
            aria-label="Κλείσιμο λεπτομερειών"
            title="Κλείσιμο"
          >
            <i className="fi fi-sr-cross" aria-hidden="true"></i>
          </button>
        </div>

        <div className="cr-detail-body">
          <table className="cr-detail-table" aria-label="Στοιχεία αναφοράς">
            <tbody>
              <tr>
                <td className="cr-detail-key">Κωδικός</td>
                <td className="cr-detail-val cr-tracking-
big">{selectedReport.tracking_code}</td>
              </tr>
              <tr>
```

```
              <td className="cr-detail-key">Τύπος</td>
              <td className="cr-detail-
val">{TYPE_LABELS[selectedReport.report_type]}</td>
            </tr>
            <tr>
              <td className="cr-detail-key">Κατάσταση</td>
              <td className="cr-detail-val">
                <span className={`cr-status-badge
${getStatusClass(selectedReport.status)}`}>
                  {STATUS_LABELS[selectedReport.status]}
                </span>
              </td>
            </tr>
            <tr>
              <td className="cr-detail-key">Πολίτης</td>
              <td className="cr-detail-val">{selectedReport.citizen_name}</td>
            </tr>
            {selectedReport.citizen_email && (
              <tr>
                <td className="cr-detail-key">Email</td>
                <td className="cr-detail-val">{selectedReport.citizen_email}</td>
              </tr>
            )}
            {selectedReport.bin_id && (
              <tr>
                <td className="cr-detail-key">Κάδος</td>
                <td className="cr-detail-val">{selectedReport.bin_id}</td>
              </tr>
            )}
            <tr>
              <td className="cr-detail-key">Θέση</td>
              <td className="cr-detail-val">
                {selectedReport.latitude.toFixed(5)}, {selectedReport.longitude.toFixed(5)}
                {selectedReport.is_pin_location && (
                  <span className="cr-pin-badge cr-pin-badge-ml">
                    <i className="fi fi-sr-map-marker icon-mr4" aria-hidden="true"></i>
                    Καρφίτσα
                  </span>
                )}
              </td>
            </tr>
            <tr>
              <td className="cr-detail-key">Υποβολή</td>
              <td className="cr-detail-val">
                {new Date(selectedReport.created_at).toLocaleString('el-GR')}
              </td>
            </tr>
            {selectedReport.resolved_at && (
              <tr>
```

```
                <td className="cr-detail-key">Επίλυση</td>
                <td className="cr-detail-val">
                  {new Date(selectedReport.resolved_at).toLocaleString('el-GR')}
                </td>
              </tr>
            )}
          </tbody>
        </table>

        <div className="cr-description">
          <div className="cr-description-label">
            <i className="fi fi-sr-document icon-mr6 icon-gray" aria-hidden="true"></i>
            Περιγραφή
          </div>
          <p className="cr-description-text">{selectedReport.description}</p>
        </div>

        {/* Map preview */}
        <div className="cr-mini-map-wrap">
          <button
            onClick={() => setMapModal({
              lat: selectedReport.latitude,
              lon: selectedReport.longitude,
              title: selectedReport.title,
            })}
            className="cr-map-link"
            aria-label="Εμφάνιση θέσης αναφοράς στον χάρτη"
            title="Εμφάνιση στον χάρτη"
          >
            <i className="fi fi-sr-map icon-mr6" aria-hidden="true"></i>
            Εμφάνιση στον χάρτη
          </button>
        </div>

        {/* Status update */}
        <div className="cr-status-update" aria-label="Ενημέρωση κατάστασης">
          <div className="cr-status-label">
            <i className="fi fi-sr-settings icon-mr6 icon-gray" aria-hidden="true"></i>
            Αλλαγή Κατάστασης
          </div>
          <div className="cr-status-btns">
            {['open', 'in_progress', 'resolved', 'closed'].map(s => (
              <button
                key={s}
                onClick={() => updateStatus(selectedReport.id, s)}
                disabled={selectedReport.status === s}
                className={`cr-status-btn cr-status-btn-${s} ${selectedReport.status === s
? 'active' : ''}`}
                aria-label={`Ορισμός κατάστασης: ${STATUS_LABELS[s]}`}
```

```
                title={STATUS_LABELS[s]}
              >
                {STATUS_LABELS[s]}
              </button>
            ))}
          </div>
        </div>
      </div>
    </div>
  </div>

    {/* Map Modal */}
    {mapModal && (
     <MapModal
      lat={mapModal.lat}
      lon={mapModal.lon}
      title={mapModal.title}
      onClose={() => setMapModal(null)}
     />
    )}
   </>
 );
}

export default CitizenReportsView;
```

Frontend/src/CitizenReportsView.css

```
/* ===== CitizenReportsView.css ===== */

.icon-mr8  { margin-right: 8px; }
.icon-mr6  { margin-right: 6px; }
.icon-mr4  { margin-right: 4px; }
.icon-blue  { color: #3498db; }
.icon-gray  { color: #7f8c8d; }
.icon-green { color: #27ae60; }

.cr-container {
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
}

/* ===== HEADER ===== */
.cr-header {
  background: white;
  border-radius: 12px;
  padding: 20px;
```

```
  box-shadow: 0 2px 8px rgba(0,0,0,0.08);
  margin-bottom: 16px;
}

.cr-header-top {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 16px;
}

.cr-title {
  font-size: 1.2rem;
  color: #2c3e50;
  margin: 0;
  display: flex;
  align-items: center;
}

.cr-header-actions {
  display: flex;
  align-items: center;
  gap: 10px;
}

.cr-portal-link {
  padding: 8px 16px;
  background: linear-gradient(135deg, #2c3e50, #3498db);
  color: white;
  border-radius: 8px;
  text-decoration: none;
  font-size: 0.85rem;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 6px;
  transition: opacity 0.2s;
}

.cr-portal-link:hover { opacity: 0.85; }

.cr-refresh-btn {
  padding: 8px 12px;
  background: #ecf0f1;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 1rem;
  color: #3498db;
```

```
  font-family: inherit;
}

/* ===== STATS ===== */
.cr-stats {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 10px;
  margin-bottom: 16px;
}

.cr-stat-card {
  text-align: center;
  padding: 12px;
  background: #f8f9fa;
  border-radius: 8px;
}

.cr-stat-val {
  font-size: 1.8rem;
  font-weight: bold;
}

.cr-stat-blue   { color: #3498db; }
.cr-stat-red    { color: #e74c3c; }
.cr-stat-orange { color: #f39c12; }
.cr-stat-green  { color: #27ae60; }

.cr-stat-label {
  font-size: 0.75rem;
  color: #7f8c8d;
  margin-top: 2px;
}

/* ===== FILTERS ===== */
.cr-filters {
  display: flex;
  gap: 16px;
  flex-wrap: wrap;
}

.cr-filter-group {
  display: flex;
  flex-direction: column;
  gap: 4px;
}

.cr-filter-label {
  font-size: 0.8rem;
```

```
  font-weight: bold;
  color: #7f8c8d;
  display: flex;
  align-items: center;
}

.cr-select {
  padding: 7px 10px;
  border: 1px solid #ddd;
  border-radius: 6px;
  font-size: 0.9rem;
  font-family: inherit;
  outline: none;
  min-width: 160px;
}

.cr-select:focus { border-color: #3498db; }

/* ===== MESSAGE ===== */
.cr-message {
  background: #d5f5e3;
  color: #27ae60;
  padding: 10px 16px;
  border-radius: 8px;
  margin-bottom: 12px;
  display: flex;
  align-items: center;
  font-size: 0.9rem;
  border: 1px solid #a9dfbf;
}

/* ===== CONTENT LAYOUT ===== */
.cr-content {
  display: grid;
  grid-template-columns: 1fr;
  gap: 16px;
}

/* When detail is open → 2 columns */
.cr-content:has(.cr-detail) {
  grid-template-columns: 1fr 420px;
}

/* ===== REPORT LIST ===== */
.cr-list {
  display: flex;
  flex-direction: column;
  gap: 10px;
}
```

```
.cr-report-item {
  background: white;
  border-radius: 10px;
  padding: 14px 16px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.07);
  cursor: pointer;
  border: 2px solid transparent;
  transition: all 0.2s;
}

.cr-report-item:hover { border-color: #aed6f1; }
.cr-report-item.selected { border-color: #3498db; background: #f0f8ff; }

.cr-report-header {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  margin-bottom: 6px;
}

.cr-report-left {
  display: flex;
  align-items: flex-start;
  gap: 10px;
}

.cr-type-icon {
  width: 36px;
  height: 36px;
  background: #ebf5fb;
  border-radius: 8px;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 1.1rem;
  color: #3498db;
  flex-shrink: 0;
}

.cr-report-title {
  font-weight: bold;
  font-size: 0.95rem;
  color: #2c3e50;
  margin-bottom: 3px;
}

.cr-report-meta {
  font-size: 0.78rem;
```

```
  color: #7f8c8d;
  display: flex;
  align-items: center;
  gap: 5px;
  flex-wrap: wrap;
}

.cr-tracking {
  font-family: 'Courier New', monospace;
  font-weight: bold;
  color: #3498db;
  background: #ebf5fb;
  padding: 1px 6px;
  border-radius: 4px;
}

.cr-dot { color: #bdc3c7; }

.cr-report-right {
  display: flex;
  flex-direction: column;
  align-items: flex-end;
  gap: 4px;
}

.cr-report-date {
  font-size: 0.75rem;
  color: #95a5a6;
  display: flex;
  align-items: center;
  gap: 8px;
  flex-wrap: wrap;
}

/* ===== BADGES ===== */
.cr-status-badge {
  padding: 3px 10px;
  border-radius: 10px;
  font-size: 0.75rem;
  font-weight: bold;
  display: inline-flex;
  align-items: center;
  white-space: nowrap;
}

.cr-status-open        { background: #fdecea; color: #e74c3c; }
.cr-status-in_progress { background: #fef9e7; color: #f39c12; }
.cr-status-resolved    { background: #d5f5e3; color: #27ae60; }
.cr-status-closed      { background: #f0f3f4; color: #7f8c8d; }
```

```
.cr-priority-badge {
  padding: 2px 7px;
  border-radius: 8px;
  font-size: 0.7rem;
  font-weight: bold;
}

.cr-priority-low      { background: #d5f5e3; color: #27ae60; }
.cr-priority-medium   { background: #ebf5fb; color: #2980b9; }
.cr-priority-high     { background: #fef9e7; color: #f39c12; }
.cr-priority-critical { background: #fdecea; color: #e74c3c; }

.cr-bin-badge {
  background: #f8f9fa;
  color: #2c3e50;
  padding: 1px 7px;
  border-radius: 6px;
  font-size: 0.72rem;
  display: inline-flex;
  align-items: center;
}

.cr-pin-badge {
  background: #f5eef8;
  color: #8e44ad;
  padding: 1px 7px;
  border-radius: 6px;
  font-size: 0.72rem;
  display: inline-flex;
  align-items: center;
}

/* ===== DETAIL PANEL ===== */
.cr-detail {
  background: white;
  border-radius: 12px;
  box-shadow: 0 2px 12px rgba(0,0,0,0.1);
  overflow: hidden;
  align-self: flex-start;
  position: sticky;
  top: 20px;
}

.cr-detail-header {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  padding: 16px 20px;
```

```
  background: linear-gradient(135deg, #2c3e50, #3498db);
  color: white;
}

.cr-detail-title {
  font-size: 0.95rem;
  margin: 0;
  color: white;
  display: flex;
  align-items: center;
}

.cr-close-btn {
  background: rgba(255,255,255,0.2);
  border: none;
  color: white;
  border-radius: 6px;
  padding: 4px 8px;
  cursor: pointer;
  font-size: 0.9rem;
}

.cr-detail-body { padding: 16px 20px; }

.cr-detail-table { width: 100%; border-collapse: collapse; margin-bottom: 16px; }
.cr-detail-table tr { border-bottom: 1px solid #f0f3f4; }

.cr-detail-key {
  padding: 7px 10px 7px 0;
  color: #7f8c8d;
  font-weight: bold;
  font-size: 0.82rem;
  width: 90px;
  vertical-align: top;
}

.cr-detail-val {
  padding: 7px 0;
  color: #2c3e50;
  font-size: 0.87rem;
}

.cr-tracking-big {
  font-family: 'Courier New', monospace;
  font-weight: bold;
  color: #3498db;
  font-size: 1rem;
}
```

```
.cr-description {
  background: #f8f9fa;
  border-radius: 8px;
  padding: 12px;
  margin-bottom: 14px;
}

.cr-description-label {
  font-size: 0.82rem;
  font-weight: bold;
  color: #7f8c8d;
  margin-bottom: 6px;
  display: flex;
  align-items: center;
}

.cr-description-text {
  font-size: 0.87rem;
  color: #2c3e50;
  line-height: 1.5;
  margin: 0;
}

.cr-mini-map-wrap { margin-bottom: 14px; }

.cr-map-link {
  display: inline-flex;
  align-items: center;
  gap: 6px;
  color: #3498db;
  text-decoration: none;
  font-size: 0.85rem;
  font-weight: bold;
  padding: 6px 12px;
  background: #ebf5fb;
  border-radius: 6px;
  border: 1px solid #aed6f1;
}

.cr-map-link:hover { background: #d6eaf8; }

/* ===== STATUS UPDATE ===== */
.cr-status-update {
  border-top: 1px solid #ecf0f1;
  padding-top: 14px;
}

.cr-status-label {
  font-size: 0.82rem;
```

```
  font-weight: bold;
  color: #7f8c8d;
  margin-bottom: 8px;
  display: flex;
  align-items: center;
}

.cr-status-btns {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 6px;
}

.cr-status-btn {
  padding: 7px 10px;
  border: 2px solid transparent;
  border-radius: 7px;
  cursor: pointer;
  font-size: 0.8rem;
  font-weight: bold;
  font-family: inherit;
  transition: all 0.2s;
}

.cr-status-btn:disabled { cursor: default; }

.cr-status-btn-open        { background: #fdecea; color: #e74c3c; border-color: #f5c6cb; }
.cr-status-btn-in_progress { background: #fef9e7; color: #f39c12; border-color: #f9ca74; }
.cr-status-btn-resolved    { background: #d5f5e3; color: #27ae60; border-color: #a9dfbf; }
.cr-status-btn-closed      { background: #f0f3f4; color: #7f8c8d; border-color: #d5d8dc; }

.cr-status-btn.active { opacity: 1; transform: scale(0.97); box-shadow: inset 0 2px 4px
rgba(0,0,0,0.1); }
.cr-status-btn:not(:disabled):hover { opacity: 0.85; }

/* ===== EMPTY / LOADING ===== */
.cr-loading {
  text-align: center; padding: 40px;
  color: #7f8c8d; display: flex;
  flex-direction: column; align-items: center; gap: 10px;
}

.cr-loading i { font-size: 2rem; }

.cr-empty {
  text-align: center; padding: 40px;
  color: #27ae60;
  background: white; border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.07);
```

```
}

.cr-empty i { font-size: 2rem; display: block; margin-bottom: 8px; }

/* ===== RESPONSIVE ===== */
@media (max-width: 900px) {
  .cr-content:has(.cr-detail) {
    grid-template-columns: 1fr;
  }
  .cr-stats { grid-template-columns: repeat(2, 1fr); }
  .cr-detail { position: static; }
}

/* ===== MAP MODAL ===== */
.cr-modal-overlay {
  position: fixed;
  inset: 0;
  background: rgba(0, 0, 0, 0.65);
  z-index: 9999;
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 20px;
  animation: modalFadeIn 0.2s ease;
}

@keyframes modalFadeIn {
  from { opacity: 0; }
  to   { opacity: 1; }
}

.cr-modal-box {
  background: white;
  border-radius: 14px;
  width: 100%;
  max-width: 800px;
  max-height: 90vh;
  display: flex;
  flex-direction: column;
  box-shadow: 0 20px 60px rgba(0, 0, 0, 0.4);
  animation: modalSlideUp 0.25s ease;
  overflow: hidden;
}

@keyframes modalSlideUp {
  from { transform: translateY(30px); opacity: 0; }
  to   { transform: translateY(0);    opacity: 1; }
}
```

```
.cr-modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 16px 20px;
  background: linear-gradient(135deg, #2c3e50, #3498db);
  flex-shrink: 0;
}

.cr-modal-title {
  color: white;
  font-size: 1rem;
  margin: 0;
  display: flex;
  align-items: center;
}

.cr-modal-close {
  background: rgba(255, 255, 255, 0.2);
  border: none;
  color: white;
  border-radius: 8px;
  padding: 6px 10px;
  cursor: pointer;
  font-size: 0.9rem;
  transition: background 0.2s;
  font-family: inherit;
}

.cr-modal-close:hover { background: rgba(255, 255, 255, 0.35); }

.cr-modal-map-wrap {
  flex: 1;
  min-height: 400px;
}

.cr-modal-map {
  height: 100%;
  width: 100%;
  min-height: 400px;
}

.cr-modal-footer {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 12px 20px;
  background: #f8f9fa;
  border-top: 1px solid #ecf0f1;
```

```
  flex-shrink: 0;
}

.cr-modal-coords {
  font-size: 0.82rem;
  color: #7f8c8d;
  display: flex;
  align-items: center;
  font-family: 'Courier New', monospace;
}

.cr-modal-back-btn {
  padding: 9px 18px;
  background: linear-gradient(135deg, #2c3e50, #3498db);
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 0.9rem;
  font-weight: bold;
  display: flex;
  align-items: center;
  gap: 6px;
  font-family: inherit;
  transition: opacity 0.2s;
}

.cr-modal-back-btn:hover { opacity: 0.85; }

/* Map pin icon */
.cr-map-pin-icon {
  width: 36px;
  height: 36px;
  background: linear-gradient(135deg, #8e44ad, #9b59b6);
  border-radius: 50% 50% 50% 0;
  transform: rotate(-45deg);
  border: 3px solid white;
  box-shadow: 0 3px 10px rgba(142, 68, 173, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  animation: pinPulse 1.5s infinite;
}

.cr-map-pin-icon i {
  transform: rotate(45deg);
  color: white;
  font-size: 1rem;
}
```

```
@keyframes pinPulse {
  0%, 100% { box-shadow: 0 3px 10px rgba(142, 68, 173, 0.4); }
  50%      { box-shadow: 0 3px 20px rgba(142, 68, 173, 0.8); }
}

@media (max-width: 600px) {
  .cr-modal-box { max-height: 100vh; border-radius: 0; }
  .cr-modal-footer { flex-direction: column; gap: 8px; }
  .cr-modal-back-btn { width: 100%; justify-content: center; }
}

/* ===== MAP MODAL POPUP (αντικαθιστά inline styles) ===== */
.cr-modal-popup {
  font-family: inherit;
  min-width: 140px;
}

.cr-modal-popup-title { color: #8e44ad; }
.cr-modal-popup-coords { color: #7f8c8d; }

/* Pin badge με αριστερό περιθώριο (μέσα στον πίνακα λεπτομερειών) */
.cr-pin-badge-ml { margin-left: 8px; }
```

Παράρτημα Γ: «Κώδικας του CollectGarbageTruck του ΠΣ Smart Waste Management System»

Driver-app/index.html

```
<!DOCTYPE html>
<html lang="el">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="apple-mobile-web-app-capable" content="yes">
  <title>CollectionGarbageTruck</title>
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"/>
  <link rel='stylesheet' href='https://cdn-uicons.flaticon.com/2.6.0/uicons-solid-
rounded/css/uicons-solid-rounded.css'>
  <link rel="stylesheet" href="driver-app.css">
</head>
<body>

<!-- LOGIN SCREEN -->
<div id="login-screen">
  <div class="login-card">
    <div class="logo" aria-hidden="true">
      <i class="fi fi-sr-truck-side"></i>
    </div>
    <h2>CollectionGarbageTruck</h2>
    <p>Εφαρμογή Οδηγού Απορριμματοφόρου<br>Δήμος Μαλεβιζίου</p>
    <div id="login-alert" role="alert" aria-live="polite"></div>

    <div class="field">
      <label for="vehicle-select">
        <i class="fi fi-sr-truck-side" aria-hidden="true"></i>
        Όχημα &amp;amp; Οδηγός
      </label>
      <select id="vehicle-select" title="Επιλέξτε το όχημά σας" aria-label="Επιλογή
οχήματος και οδηγού">
        <option value="">-- Επιλέξτε όχημα --</option>
        <optgroup label="Γάζι (GAZ) — 5 οχήματα">
          <option value="VEH-001|GAZ|Γάζι|Μανώλης Σταυρακάκης|18.00">VEH-001 |
HKH-1234 — Μανώλης Σταυρακάκης</option>
          <option value="VEH-007|GAZ|Γάζι|Γιάννης Τζανάκης|18.00">VEH-007 | HKH-
7890 — Γιάννης Τζανάκης</option>
          <option value="VEH-008|GAZ|Γάζι|Αντώνης Σφακιανάκης|15.00">VEH-008 |
HKH-8901 — Αντώνης Σφακιανάκης</option>
          <option value="VEH-P01|GAZ|Γάζι|Θανάσης Κοκολάκης|12.00">VEH-P01 |
HKH-P001 — Θανάσης Κοκολάκης</option>
          <option value="VEH-P04|GAZ|Γάζι|Δημήτρης Κατσαρός|9.00">VEH-P04 | HKH-
P004 — Δημήτρης Κατσαρός</option>
        </optgroup>
```

```
      <optgroup label="Αμμουδάρα (AMM) — 5 οχήματα">
        <option value="VEH-002|AMM|Αμμουδάρα|Γιώργης Παπαδάκης|18.00">VEH-002
| ΗΚΗ-2345 — Γιώργης Παπαδάκης</option>
        <option value="VEH-009|AMM|Αμμουδάρα|Πέτρος Χαλκιαδάκης|18.00">VEH-
009 | ΗΚΗ-9012 — Πέτρος Χαλκιαδάκης</option>
        <option value="VEH-010|AMM|Αμμουδάρα|Βαγγέλης Μαρινάκης|15.00">VEH-
010 | ΗΚΗ-0123 — Βαγγέλης Μαρινάκης</option>
        <option value="VEH-P02|AMM|Αμμουδάρα|Λευτέρης Ορφανάκης|12.00">VEH-
P02 | ΗΚΗ-P002 — Λευτέρης Ορφανάκης</option>
        <option value="VEH-P03|AMM|Αμμουδάρα|Νεκτάριος Αλεξάκης|9.00">VEH-P03
| ΗΚΗ-P003 — Νεκτάριος Αλεξάκης</option>
      </optgroup>
      <optgroup label="Αγία Πελαγία (APE) — 1 όχημα">
        <option value="VEH-003|APE|Αγία Πελαγία|Νίκος Μαρακάκης|15.00">VEH-003 |
ΗΚΗ-3456 — Νίκος Μαρακάκης</option>
      </optgroup>
      <optgroup label="Τύλισος (TYL) — 1 όχημα">
        <option value="VEH-004|TYL|Τύλισος|Κώστας Βασιλάκης|15.00">VEH-004 |
ΗΚΗ-4567 — Κώστας Βασιλάκης</option>
      </optgroup>
      <optgroup label="Κρουσώνας (KRO) — 1 όχημα">
        <option value="VEH-005|KRO|Κρουσώνας|Μιχάλης Δασκαλάκης|18.00">VEH-
005 | ΗΚΗ-5678 — Μιχάλης Δασκαλάκης</option>
      </optgroup>
      <optgroup label="Ροδιά (ROD) — 1 όχημα">
        <option value="VEH-006|ROD|Ροδιά|Στέλιος Καλογεράκης|15.00">VEH-006 |
ΗΚΗ-6789 — Στέλιος Καλογεράκης</option>
      </optgroup>
    </select>
  </div>

  <div class="field">
    <label for="threshold-select">
      <i class="fi fi-sr-settings" aria-hidden="true"></i>
      Όριο Πληρότητας Κάδων
    </label>
    <select id="threshold-select" title="Επιλέξτε όριο πληρότητας" aria-label="Όριο
πληρότητας κάδων">
      <option value="60">60% — Κρίσιμοι + Προσοχή</option>
      <option value="80">80% — Μόνο Κρίσιμοι</option>
      <option value="40">40% — Σχεδόν Όλοι</option>
    </select>
  </div>

  <div class="field">
    <label for="api-url">
      <i class="fi fi-sr-network" aria-hidden="true"></i>
      Διεύθυνση API Server
    </label>
```

```
      <input
        type="text"
        id="api-url"
        value="http://127.0.0.1:8000"
        placeholder="http://IP:8000"
        aria-label="Διεύθυνση API Server"
        title="Εισάγετε τη διεύθυνση του API server"
      />
    </div>

    <button class="btn btn-primary" onclick="doLogin()" aria-label="Σύνδεση στο
σύστημα" title="Σύνδεση">
      <i class="fi fi-sr-sign-in-alt" aria-hidden="true"></i>
      Είσοδος
    </button>
  </div>
</div>

<!-- MAIN APP -->
<div id="app-screen">
  <header class="header">
    <h1>
      <i class="fi fi-sr-truck-side" aria-hidden="true"></i>
      CollectionGarbageTruck
    </h1>
    <div class="header-right">
      <span class="zone-badge" id="header-zone" aria-label="Ζώνη εργασίας">—</span>
      <span class="vehicle-badge" id="header-vehicle" aria-label="Αναγνωριστικό
οχήματος">VEH-001</span>
      <button class="alert-bell" onclick="showTab('notifications')" id="alert-bell-btn"
        aria-label="Ειδοποιήσεις" title="Εμφάνιση ειδοποιήσεων">
        <i class="fi fi-sr-bell" aria-hidden="true"></i>
        <span class="alert-count hidden" id="alert-count" aria-label="Αριθμός
ειδοποιήσεων">0</span>
      </button>
      <button class="header-logout-btn" onclick="doLogout()" aria-label="Έξοδος"
title="Έξοδος">
        <i class="fi fi-sr-sign-out-alt" aria-hidden="true"></i>
      </button>
    </div>
  </header>

  <nav class="tabs" aria-label="Κύρια πλοήγηση">
    <button class="tab-btn active" onclick="showTab('bins')" id="tab-bins"
      aria-label="Κάδοι" title="Κάδοι" aria-current="page">
      <i class="fi fi-sr-trash" aria-hidden="true"></i>
      <span>Κάδοι</span>
    </button>
    <button class="tab-btn" onclick="showTab('map')" id="tab-map"
```

```
    aria-label="Χάρτης" title="Χάρτης">
    <i class="fi fi-sr-map" aria-hidden="true"></i>
    <span>Χάρτης</span>
   </button>
   <button class="tab-btn" onclick="showTab('notifications')" id="tab-notifications"
    aria-label="Ειδοποιήσεις" title="Ειδοποιήσεις">
    <i class="fi fi-sr-bell" aria-hidden="true"></i>
    <span>Ειδοποιήσεις</span>
    <span class="tab-badge hidden" id="notif-tab-badge" aria-hidden="true">0</span>
   </button>
   <button class="tab-btn" onclick="showTab('gps')" id="tab-gps"
    aria-label="GPS Tracking" title="GPS Tracking">
    <i class="fi fi-sr-radar" aria-hidden="true"></i>
    <span>GPS</span>
   </button>
  </nav>

  <!-- BINS TAB -->
  <div class="tab-content active" id="content-bins" role="tabpanel" aria-label="Κάδοι
ζώνης">
   <div id="bins-alert" role="alert" aria-live="polite"></div>

   <div class="zone-info" id="zone-info-bar" aria-label="Πληροφορίες ζώνης">
    <div>
     <div class="zone-label">
      <i class="fi fi-sr-marker" aria-hidden="true"></i> Ζώνη Εργασίας
     </div>
     <div class="zone-name" id="zone-display-name">—</div>
     <div id="driver-display-name"></div>
    </div>
    <div class="zone-stats" id="zone-stats-display" aria-label="Στατιστικά ζώνης">—
</div>
   </div>

   <div id="vehicle-capacity-display" class="card" aria-label="Πληρότητα
οχήματος"></div>

   <div id="zone-vehicles-display" class="card hidden">
    <div class="zone-vehicles-title">
     <i class="fi fi-sr-truck-side" aria-hidden="true"></i>
     Οχήματα Ζώνης
    </div>
    <div id="zone-vehicles-list" aria-label="Λίστα οχημάτων ζώνης"></div>
   </div>

   <div class="stats-row" aria-label="Στατιστικά κάδων">
    <div class="stat-card">
     <div class="val stat-total" id="stat-total">0</div>
     <div class="lbl">Σύνολο</div>
```

```
    </div>
    <div class="stat-card">
      <div class="val stat-pending" id="stat-pending">0</div>
      <div class="lbl">Εκκρεμείς</div>
    </div>
    <div class="stat-card">
      <div class="val stat-collected" id="stat-collected">0</div>
      <div class="lbl">Αδειασμένοι</div>
    </div>
  </div>

  <button class="btn btn-primary btn-load-bins" onclick="loadBins()"
    aria-label="Φόρτωση κάδων ζώνης" title="Φόρτωση κάδων">
    <i class="fi fi-sr-rotate-right" aria-hidden="true"></i>
    Φόρτωση Κάδων Ζώνης
  </button>

  <button class="xyta-btn" onclick="reportXytaTrip()"
    aria-label="Αναφορά μεταφοράς στο ΧΥΤΑ" title="Μεταφορά στο ΧΥΤΑ">
    <i class="fi fi-sr-road" aria-hidden="true"></i>
    Αναφορά Μεταφοράς στο ΧΥΤΑ Πέρα Γαλήνων
  </button>

  <div id="bins-list" class="bins-list" aria-label="Λίστα κάδων" aria-live="polite">
    <div class="empty">
      <i class="fi fi-sr-trash" aria-hidden="true"></i>
      <p>Πατήστε "Φόρτωση Κάδων Ζώνης" για να δείτε τους κάδους σας</p>
    </div>
  </div>
 </div>

 <!-- MAP TAB -->
 <div class="tab-content" id="content-map" role="tabpanel" aria-label="Χάρτης κάδων">
  <div id="map" aria-label="Χάρτης με θέσεις κάδων"></div>
 </div>

 <!-- NOTIFICATIONS TAB -->
 <div class="tab-content" id="content-notifications" role="tabpanel" aria-
label="Ειδοποιήσεις">
  <div class="notifications-header">
    <h3 class="notifications-title">
      <i class="fi fi-sr-bell" aria-hidden="true"></i>
      Ειδοποιήσεις Συστήματος
    </h3>
    <button class="notifications-refresh" onclick="loadAlerts()"
      aria-label="Ανανέωση ειδοποιήσεων" title="Ανανέωση">
      <i class="fi fi-sr-rotate-right" aria-hidden="true"></i>
    </button>
  </div>
```

```
  <div id="notifications-list" aria-live="polite">
    <div class="loading">
      <i class="fi fi-sr-rotate-right" aria-hidden="true"></i>
      <p>Φόρτωση...</p>
    </div>
  </div>
 </div>

 <!-- GPS TAB -->
 <div class="tab-content" id="content-gps" role="tabpanel" aria-label="GPS Tracking">
  <div class="card">
    <h3 class="card-title">
      <i class="fi fi-sr-radar" aria-hidden="true"></i>
      GPS Tracking
    </h3>
    <div class="gps-status-row">
      <span class="gps-dot inactive" id="gps-dot" aria-hidden="true"></span>
      <span id="gps-status-text" aria-live="polite">GPS Ανενεργό</span>
    </div>
    <div class="gps-info-row">
      <div class="gps-info-label"><i class="fi fi-sr-marker" aria-hidden="true"></i>
Τρέχουσα θέση:</div>
      <div id="gps-coords" class="gps-info-value" aria-live="polite">—</div>
    </div>
    <div class="gps-info-row">
      <div class="gps-info-label"><i class="fi fi-sr-rotate-right" aria-hidden="true"></i>
Τελευταία αποστολή:</div>
      <div id="gps-last-sent" class="gps-info-value" aria-live="polite">—</div>
    </div>
    <div class="gps-info-row">
      <div class="gps-info-label"><i class="fi fi-sr-road" aria-hidden="true"></i>
Αποστολές σήμερα:</div>
      <div id="gps-count" class="gps-count-value" aria-live="polite">0</div>
    </div>
    <button class="btn btn-success" id="gps-start-btn" onclick="startGPS()"
      aria-label="Εκκίνηση GPS Tracking" title="Εκκίνηση GPS">
      <i class="fi fi-sr-radar" aria-hidden="true"></i>
      Εκκίνηση GPS Tracking
    </button>
    <button class="btn btn-danger gps-stop-margin hidden" id="gps-stop-btn"
onclick="stopGPS()"
      aria-label="Διακοπή GPS Tracking" title="Διακοπή GPS">
      <i class="fi fi-sr-stop" aria-hidden="true"></i>
      Διακοπή GPS Tracking
    </button>

    <div class="report-divider">
      <h4 class="report-actions-title">
        <i class="fi fi-sr-triangle-warning" aria-hidden="true"></i>
```

```
        Αναφορά Προβλήματος
       </h4>
       <div class="report-buttons-grid">
        <button class="btn btn-danger btn-sm" onclick="reportBreakdown()"
         aria-label="Αναφορά βλάβης οχήματος" title="Βλάβη">
         <i class="fi fi-sr-tools" aria-hidden="true"></i> Βλάβη
        </button>
        <button class="btn btn-warning btn-sm" onclick="reportMaintenance()"
         aria-label="Αναφορά συντήρησης οχήματος" title="Συντήρηση">
         <i class="fi fi-sr-settings" aria-hidden="true"></i> Συντήρηση
        </button>
        <button class="btn btn-success btn-sm report-btn-full"
onclick="reportXytaArrived()"
         aria-label="Επιβεβαίωση άφιξης στο ΧΥΤΑ" title="Έφτασα στο ΧΥΤΑ">
         <i class="fi fi-sr-check-circle" aria-hidden="true"></i> Έφτασα στο ΧΥΤΑ
        </button>
       </div>
      </div>
     </div>

     <div class="card">
      <h4 class="card-subtitle">
       <i class="fi fi-sr-settings" aria-hidden="true"></i>
       Ρυθμίσεις
      </h4>
      <div class="field">
       <label for="gps-interval">
        <i class="fi fi-sr-time-fast" aria-hidden="true"></i>
        Αποστολή GPS κάθε
       </label>
       <select id="gps-interval" title="Επιλέξτε συχνότητα αποστολής GPS"
        aria-label="Συχνότητα αποστολής GPS">
        <option value="30000">30 δευτερόλεπτα</option>
        <option value="60000">1 λεπτό</option>
        <option value="120000">2 λεπτά</option>
       </select>
      </div>
     </div>
    </div>
   </div>

<!-- Notification Popup -->
<div id="notif-popup" class="notification-popup hidden" role="alert" aria-
live="assertive">
  <i class="fi fi-sr-bell" aria-hidden="true"></i>
  <span id="notif-popup-text"></span>
  <button class="notification-popup-close" onclick="document.getElementById('notif-
popup').classList.add('hidden')"
    aria-label="Κλείσιμο ειδοποίησης" title="Κλείσιμο">
```

```
    <i class="fi fi-sr-cross" aria-hidden="true"></i>
  </button>
</div>

<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
<script>
  // ===== STATE =====
  var API_URL = 'http://127.0.0.1:8000';
  var VEHICLE_ID = '';
  var ZONE_CODE = '';
  var ZONE_NAME = '';
  var DRIVER_NAME = '';
  var THRESHOLD = 60;
  var bins = [];
  var collectedBins = new Set();
  var map = null;
  var mapMarkers = [];
  var gpsInterval = null;
  var alertsInterval = null;
  var gpsCount = 0;
  var mapInitialized = false;
  var knownAlertIds = new Set();
  var lastAlertCount = 0;
  var collectedVolume = 0;
  var vehicleCapacityM3 = 12.0;
  var xytaAlertSent = false;
  var BIN_VOLUME_M3 = 1.1;

  var ZONE_COLORS = {
    GAZ: '#3498db', AMM: '#27ae60', APE: '#e74c3c',
    TYL: '#f39c12', KRO: '#9b59b6', ROD: '#1abc9c'
  };
  var XYTA = { lat: 35.39236790444294, lon: 24.921436396669584 };

  // ===== LOGIN =====
  function doLogin() {
    var vehicleRaw = document.getElementById('vehicle-select').value;
    var threshold = document.getElementById('threshold-select').value;
    var apiUrl = document.getElementById('api-url').value.trim();

    if (!vehicleRaw) { showLoginAlert('error', 'Επιλέξτε όχημα!'); return; }
    if (!apiUrl) { showLoginAlert('error', 'Εισάγετε διεύθυνση API!'); return; }

    var parts = vehicleRaw.split('|');
    VEHICLE_ID = parts[0];
    ZONE_CODE = parts[1];
    ZONE_NAME = parts[2];
    DRIVER_NAME = parts[3];
    vehicleCapacityM3 = parseFloat(parts[4]) || 12.0;
```

```
    THRESHOLD = parseInt(threshold);
    API_URL = apiUrl;

    document.getElementById('header-vehicle').textContent = VEHICLE_ID;
    document.getElementById('header-zone').textContent = ZONE_CODE;
    document.getElementById('header-zone').style.setProperty('--zone-color',
(ZONE_COLORS[ZONE_CODE] || '#3498db') + '80');
    document.getElementById('zone-display-name').textContent = ZONE_NAME;
    document.getElementById('driver-display-name').textContent = DRIVER_NAME;
    document.getElementById('zone-info-bar').style.setProperty('--zone-color',
ZONE_COLORS[ZONE_CODE] || '#2c3e50');

    document.getElementById('login-screen').style.display = 'none';
    document.getElementById('app-screen').style.display = 'block';

    // Επαναφορά αποθηκευμένου state αν υπάρχει (same session)
    var savedState = sessionStorage.getItem('driverState_' + VEHICLE_ID);
    if (savedState) {
      try {
        var state = JSON.parse(savedState);
        var savedTime = new Date(state.savedAt);
        var now = new Date();
        var hoursElapsed = (now - savedTime) / (1000 * 60 * 60);
        // Χρήση state μόνο αν είναι < 8 ώρες παλιό (ίδια βάρδια)
        if (hoursElapsed < 8) {
          collectedBins = new Set(state.collectedBins || []);
          collectedVolume = state.collectedVolume || 0;
          xytaAlertSent = state.xytaAlertSent || false;
        } else {
          sessionStorage.removeItem('driverState_' + VEHICLE_ID);
        }
      } catch(e) { console.error('State restore error:', e); }
    }

    updateCapacityDisplay(collectedBins.size);
    loadBins();
    loadZoneVehicles();
    loadAlerts();
    alertsInterval = setInterval(function() { loadAlerts(); loadZoneVehicles(); }, 30000);
  }

  function doLogout() {
    if (gpsInterval) stopGPS();
    if (alertsInterval) clearInterval(alertsInterval);
    VEHICLE_ID = ''; ZONE_CODE = ''; ZONE_NAME = ''; DRIVER_NAME = '';
    bins = []; collectedBins.clear(); knownAlertIds.clear();
    collectedVolume = 0; xytaAlertSent = false;
    document.getElementById('login-screen').style.display = 'flex';
    document.getElementById('app-screen').style.display = 'none';
```

```
 }

 function showLoginAlert(type, msg) {
  document.getElementById('login-alert').innerHTML =
   '<div class="alert-box alert-' + type + '"><i class="fi fi-sr-' +
   (type === 'error' ? 'cross-circle' : 'check-circle') + '" aria-hidden="true"></i>' + msg +
'</div>';
 }

 // ===== TABS =====
 function showTab(tab) {
  document.querySelectorAll('.tab-content').forEach(function(el) {
el.classList.remove('active'); });
  document.querySelectorAll('.tab-btn').forEach(function(el) {
   el.classList.remove('active');
   el.removeAttribute('aria-current');
  });
  document.getElementById('content-' + tab).classList.add('active');
  var activeBtn = document.getElementById('tab-' + tab);
  activeBtn.classList.add('active');
  activeBtn.setAttribute('aria-current', 'page');
  if (tab === 'map') { setTimeout(function() { initMap(); updateMap(); }, 100); }
  if (tab === 'notifications') { loadAlerts(); }
 }

 // ===== CAPACITY DISPLAY =====
 function updateCapacityDisplay(collectedCount) {
  var collected_vol = collectedCount * BIN_VOLUME_M3;
  var fill_pct = Math.min(100, (collected_vol / vehicleCapacityM3) * 100);
  var remaining = Math.max(0, Math.floor((vehicleCapacityM3 - collected_vol) /
BIN_VOLUME_M3));
  var bucket = fill_pct >= 85 ? 'crit' : fill_pct >= 60 ? 'warn' : 'ok';
  var display = document.getElementById('vehicle-capacity-display');
  if (!display) return;
  display.className = 'card cap-color-' + bucket;
  display.innerHTML =
   '<div class="cap-head">' +
    '<span class="cap-title">Πληρότητα Οχήματος (' + vehicleCapacityM3 +
'm³)</span>' +
    '<span class="cap-pct">' + fill_pct.toFixed(1) + '%</span>' +
   '</div>' +
   '<div class="pwd-strength-bar-bg cap-bar-wrap">' +
    '<div class="cap-bar"></div>' +
   '</div>' +
   '<div class="cap-grid">' +
    '<div class="cap-cell">' +
     '<div class="cap-cell-num blue">' + collectedCount + '</div><div class="cap-cell-
sub">Κάδοι</div>' +
    '</div>' +
```

```
      '<div class="cap-cell">' +
        '<div class="cap-cell-num dyn">' + collected_vol.toFixed(1) + 'm³</div>' +
        '<div class="cap-cell-sub">/' + vehicleCapacityM3 + 'm³</div>' +
      '</div>' +
      '<div class="cap-cell">' +
        '<div class="cap-cell-num green">' + remaining + '</div><div class="cap-cell-
sub">Χωράνε</div>' +
      '</div>' +
    '</div>' +
    (fill_pct >= 85 ?
      '<div class="cap-warning">' +
        '<i class="fi fi-sr-triangle-warning" aria-hidden="true"></i>' +
        '<span>Το όχημα είναι σχεδόν γεμάτο! Μεταβείτε στο ΧΥΤΑ.</span>' +
      '</div>' : '');
   var capBar = display.querySelector('.cap-bar');
   if (capBar) capBar.style.setProperty('--cap-width', fill_pct + '%');

   if (fill_pct >= 85 && !xytaAlertSent && collectedCount > 0) {
     xytaAlertSent = true;
     showPopup('Το όχημα ' + VEHICLE_ID + ' είναι ' + fill_pct.toFixed(0) + '% γεμάτο!
Μεταβείτε στο ΧΥΤΑ.', 'warning');
     reportXytaTrip(true);
   }
 }

 // ===== ZONE VEHICLES =====
 async function loadZoneVehicles() {
   try {
     var res = await fetch(API_URL + '/api/v1/dispatch/zone-vehicles/' + ZONE_CODE);
     var json = await res.json();
     var vehicles = json.vehicles || [];
     var display = document.getElementById('zone-vehicles-display');
     var list = document.getElementById('zone-vehicles-list');
     if (vehicles.length === 0) { display.classList.add('hidden'); return; }
     display.classList.remove('hidden');
     list.innerHTML = vehicles.map(function(v) {
       var isMe = v.vehicle_id === VEHICLE_ID;
       var sCls = v.status === 'active' ? 'status-active' : v.status === 'maintenance' ? 'status-
maintenance' : 'status-inactive';
       var sIcon = v.status === 'active' ? 'fi-sr-check-circle' : v.status === 'maintenance' ? 'fi-
sr-tools' : 'fi-sr-cross-circle';
       var sLabel = v.status === 'active' ? 'Ενεργό' : v.status === 'maintenance' ?
'Συντήρηση' : 'Ανενεργό';
       return '<div class="zone-vehicle-item ' + (isMe ? 'is-me' : 'other') + '">' +
         '<div><div class="zone-vehicle-name">' +
           '<i class="fi fi-sr-truck-side" aria-hidden="true"></i>' + v.vehicle_id +
           (isMe ? '<span class="zone-vehicle-badge-me">ΕΓΩ</span>' : '') +
         '</div>' +
```

```
        '<div class="zone-vehicle-sub">' + v.driver_name + ' | ' + v.capacity_tons + 'τ/' +
v.capacity_m3 + 'm³</div></div>' +
        '<div class="zone-vehicle-status">' +
          '<span class="zone-vehicle-status-badge ' + sCls + '">' +
            '<i class="fi ' + sIcon + '" aria-hidden="true"></i>' + sLabel +
          '</span>' +
          '<div class="zone-vehicle-fuel"><i class="fi fi-sr-gas-pump" aria-
hidden="true"></i> ' + v.fuel_level + '%</div>' +
        '</div></div>';
    }).join('');
  } catch(e) { console.error(e); }
 }

 // ===== SAVE/RESTORE DRIVER STATE =====
 function saveDriverState() {
  if (!VEHICLE_ID) return;
  var state = {
   collectedBins:  Array.from(collectedBins),
   collectedVolume: collectedVolume,
   xytaAlertSent:  xytaAlertSent,
   savedAt:        new Date().toISOString(),
   vehicleId:      VEHICLE_ID,
   zoneCode:       ZONE_CODE,
  };
  sessionStorage.setItem('driverState_' + VEHICLE_ID, JSON.stringify(state));
 }

 // ===== LOAD BINS =====
 async function loadBins() {
  document.getElementById('bins-list').innerHTML =
   '<div class="loading"><i class="fi fi-sr-rotate-right" aria-
hidden="true"></i><p>Φόρτωση κάδων ζώνης ' + ZONE_NAME + '...</p></div>';
  try {
   var res = await fetch(API_URL + '/api/v1/zones/' + ZONE_CODE + '/bins');
   var json = await res.json();
   var allBins = json.bins || [];
   // Κάδοι >= threshold (χρειάζονται αποκομιδή)
   var pendingBins = allBins.filter(function(b) {
    return b.current_fill_level >= THRESHOLD;
   });
   // Κάδοι που αδειάστηκαν σήμερα (fill_level < 10 = μόλις αδειάστηκαν)
   var recentlyEmptied = allBins.filter(function(b) {
    return b.current_fill_level < 10 && collectedBins.has(b.bin_id);
   });
   // Συνδυασμός: pending + αδειασμένοι (χωρίς διπλότυπα)
   var allRelevant = pendingBins.slice();
   recentlyEmptied.forEach(function(b) {
    if (!allRelevant.find(function(x) { return x.bin_id === b.bin_id; })) {
     allRelevant.push(b);
```

```
      }
    });
    bins = allRelevant;

    var critical = allBins.filter(function(b) { return b.current_fill_level >= 80; }).length;
    var warning  = allBins.filter(function(b) { return b.current_fill_level >= 60 &&
b.current_fill_level < 80; }).length;
    document.getElementById('zone-stats-display').innerHTML =
      allBins.length + ' κάδοι<br>' +
      '<span class="zone-stat-critical">' + critical + ' κρίσιμοι</span>' +
      (warning > 0 ? ' | <span class="zone-stat-warning">' + warning + ' προσοχή</span>' :
'');

    if (pendingBins.length === 0 && recentlyEmptied.length === 0) {
      document.getElementById('bins-list').innerHTML =
        '<div class="empty"><i class="fi fi-sr-check-circle" aria-hidden="true"></i>' +
        '<p>Δεν υπάρχουν κάδοι >= ' + THRESHOLD + '% στη ζώνη ' + ZONE_NAME +
'!</p></div>';
    } else {
      renderBins();
      var msg = 'Φορτώθηκαν ' + pendingBins.length + ' κάδοι >= ' + THRESHOLD + '%';
      if (recentlyEmptied.length > 0) msg += ' + ' + recentlyEmptied.length + '
αδειασμένοι';
      showBinsAlert('success', msg);
      setTimeout(function() { document.getElementById('bins-alert').innerHTML = ''; },
3000);
    }
  } catch(e) {
    document.getElementById('bins-list').innerHTML =
      '<div class="empty"><i class="fi fi-sr-cross-circle" aria-
hidden="true"></i><p>Αδυναμία σύνδεσης!</p></div>';
    showBinsAlert('error', 'Αδυναμία σύνδεσης με τον server!');
  }
 }

 function renderBins() {
  updateStats();
  var list = document.getElementById('bins-list');
  list.innerHTML = bins.map(function(bin) {
    var collected = collectedBins.has(bin.bin_id);
    var level = bin.current_fill_level;
    var cls = collected ? 'collected' : level >= 80 ? 'critical' : 'warning';
    return '<div class="bin-item ' + cls + '" id="bin-' + bin.bin_id + '">' +
      '<div class="bin-header">' +
        '<span class="bin-id"><i class="fi fi-sr-trash" aria-hidden="true"></i> ' + bin.bin_id
+
          (bin.active_alerts_count > 0 ? ' <i class="fi fi-sr-triangle-warning bin-alert-icon"
aria-label="Ενεργή ειδοποίηση"></i>' : '') +
        '</span>' +
```

```
        '<span class="bin-pct">' +
          (collected ? '<i class="fi fi-sr-check-circle" aria-label="Αδειάστηκε"></i>' : level +
'%') +
        '</span>' +
       '</div>' +
       '<div class="bin-location"><i class="fi fi-sr-marker" aria-hidden="true"></i> ' +
bin.location_name + '</div>' +
       '<div class="bin-address">' + (bin.address || '') + '</div>' +
       '<div class="fill-bar" role="progressbar" aria-valuenow="' + Math.round(level) + '"
aria-valuemin="0" aria-valuemax="100" aria-label="Πληρότητα κάδου ' + bin.bin_id + ': '
+ level + '%">' +
        '<div class="fill-progress" data-w="' + level + '"></div>' +
       '</div>' +
       '<div class="bin-actions">' +
        (!collected ?
          '<button class="btn btn-success btn-sm" onclick="markCollected(\'' + bin.bin_id +
'\')" aria-label="Καταγραφή αποκομιδής κάδου ' + bin.bin_id + '" title="Αδειάστηκε">' +
            '<i class="fi fi-sr-check-circle" aria-hidden="true"></i> Αδειάστηκε' +
          '</button>' +
          '<button class="btn btn-secondary btn-sm" onclick="navigateToBin(' + bin.latitude
+ ',' + bin.longitude + ')" aria-label="Εμφάνιση στον χάρτη" title="Χάρτης">' +
            '<i class="fi fi-sr-map" aria-hidden="true"></i> Χάρτης' +
          '</button>' :
          '<span class="bin-collected-tag" aria-label="Αδειάστηκε">' +
            '<i class="fi fi-sr-check-circle" aria-hidden="true"></i> Αδειάστηκε' +
          '</span>') +
       '</div>' +
     '</div>';
   }).join('');
   // Πλάτος μπάρας πληρότητας ως CSS variable (χωρίς inline style)
   list.querySelectorAll('.fill-progress[data-w]').forEach(function(el) {
    el.style.setProperty('--w', el.getAttribute('data-w') + '%');
   });
  }

  function updateStats() {
   var pending = bins.filter(function(b) { return !collectedBins.has(b.bin_id); }).length;
   document.getElementById('stat-total').textContent = bins.length;
   document.getElementById('stat-pending').textContent = pending;
   document.getElementById('stat-collected').textContent = collectedBins.size;
  }

  function showBinsAlert(type, msg) {
   document.getElementById('bins-alert').innerHTML =
    '<div class="alert-box alert-' + type + '">' +
     '<i class="fi fi-sr-' + (type==='success'?'check-circle':'cross-circle') + '" aria-
hidden="true"></i>' + msg +
    '</div>';
  }
```

```
 async function markCollected(binId) {
  try {
   var res = await fetch(API_URL + '/api/v1/bins/' + binId + '/collect', {
    method: 'POST', headers: { 'Content-Type': 'application/json' }
   });
   var json = await res.json();
   if (res.ok) {
    collectedBins.add(binId);
    collectedVolume += BIN_VOLUME_M3;
    saveDriverState();
    updateCapacityDisplay(collectedBins.size);
    renderBins(); updateMap();
    showBinsAlert('success', 'Ο κάδος ' + binId + ' καταγράφηκε στο σύστημα!');
    setTimeout(function() { document.getElementById('bins-alert').innerHTML = ''; },
3000);
    sendDriverAction('bin_collected', { bin_id: binId, collected_bins_count:
collectedBins.size, fill_level_pct: (collectedBins.size * BIN_VOLUME_M3 /
vehicleCapacityM3) * 100 });
   } else {
    showBinsAlert('error', 'Σφάλμα: ' + (json.detail || 'Άγνωστο'));
   }
  } catch(e) {
   collectedBins.add(binId);
   collectedVolume += BIN_VOLUME_M3;
   updateCapacityDisplay(collectedBins.size);
   renderBins(); updateMap();
   showBinsAlert('error', 'Offline — ενημερώθηκε μόνο τοπικά.');
  }
 }

 async function reportXytaTrip(auto) {
  if (!VEHICLE_ID) return;
  var fill_pct = Math.min(100, (collectedBins.size * BIN_VOLUME_M3 /
vehicleCapacityM3) * 100);
  try {
   var res = await fetch(API_URL + '/api/v1/dispatch/xyta-trip', {
    method: 'POST', headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ vehicle_id: VEHICLE_ID, reason: 'full',
collected_bins_count: collectedBins.size, fill_level_pct: fill_pct })
   });
   var json = await res.json();
   if (res.ok) {
    showPopup((auto ? 'Αυτόματη: ' : '') + 'XYTA καταγράφηκε! ' +
json.xyta.distance_km + ' km (~' + json.xyta.estimated_minutes + ' λεπτά)', 'success');
    sendDriverAction('vehicle_full', { fill_level_pct: fill_pct });
    setTimeout(function() {
     collectedBins.clear(); collectedVolume = 0;
     xytaAlertSent = false;
```

```
        saveDriverState();
        updateCapacityDisplay(0); renderBins();
      }, 2000);
    }
  } catch(e) { showPopup('Αδυναμία σύνδεσης.', 'error'); }
}

function navigateToBin(lat, lon) {
  showTab('map');
  setTimeout(function() { initMap(); map.setView([lat, lon], 17); }, 150);
}

// ===== DRIVER ACTIONS =====
async function sendDriverAction(actionType, extraData) {
  try {
    await fetch(API_URL + '/api/v1/dispatch/driver-action', {
      method: 'POST', headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(Object.assign({ vehicle_id: VEHICLE_ID, action_type:
actionType, driver_name: DRIVER_NAME, zone_code: ZONE_CODE,
collected_bins_count: collectedBins.size, fill_level_pct: (collectedBins.size *
BIN_VOLUME_M3 / vehicleCapacityM3) * 100 }, extraData || {}))
    });
  } catch(e) { console.error(e); }
}

async function reportBreakdown() {
  var notes = prompt('Περιγράψτε τη βλάβη:') || '';
  await sendDriverAction('breakdown', { notes: notes });
  showPopup('Αναφορά βλάβης στάλθηκε!', 'warning');
  loadAlerts();
}

async function reportMaintenance() {
  var notes = prompt('Σημειώσεις συντήρησης:') || '';
  await sendDriverAction('maintenance', { notes: notes });
  showPopup('Αναφορά συντήρησης στάλθηκε!', 'warning');
  loadAlerts();
}

async function reportXytaArrived() {
  await sendDriverAction('xyta_arrived', {});
  showPopup('Επιβεβαιώθηκε η άφιξη στο XYTA!', 'success');
  collectedBins.clear(); collectedVolume = 0;
  xytaAlertSent = false;
  saveDriverState();
  updateCapacityDisplay(0); renderBins();
}

// ===== MAP =====
```

```
 function initMap() {
  if (mapInitialized) return;
  map = L.map('map').setView([35.35, 25.06], 12);
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', { attribution: '© OpenStreetMap' }).addTo(map);
  mapInitialized = true;
  L.marker([XYTA.lat, XYTA.lon], {
   icon: L.divIcon({
    html: '<div class="driver-xyta-marker"><i class="fi fi-sr-road" aria-hidden="true"></i> XYTA Πέρα Γαλήνων</div>',
    className: '', iconSize: [160, 28], iconAnchor: [80, 14]
   })
  }).bindPopup('<strong>XYTA Πέρα Γαλήνων</strong><br>Χώρος Υγειονομικής Ταφής').addTo(map);
 }

 function updateMap() {
  if (!map) return;
  mapMarkers.forEach(function(m) { m.remove(); });
  mapMarkers = [];
  bins.forEach(function(bin) {
   if (!bin.latitude || !bin.longitude) return;
   var collected = collectedBins.has(bin.bin_id);
   var cls = collected ? 'collected' : bin.current_fill_level >= 80 ? 'critical' : 'warning';
   var pcls = collected ? 'ok' : bin.current_fill_level >= 80 ? 'crit' : 'warn';
   var markerIcon = collected ? 'fi-sr-check-circle' : 'fi-sr-trash';
   var icon = L.divIcon({
    html: '<div class="driver-bin-marker ' + cls + '" role="img" aria-label="' + bin.bin_id + '"><i class="fi ' + markerIcon + '" aria-hidden="true"></i></div>',
    className: '', iconSize: [30, 30], iconAnchor: [15, 15]
   });
   var marker = L.marker([bin.latitude, bin.longitude], { icon }).addTo(map);
   marker.bindPopup(
    '<strong>' + bin.bin_id + '</strong><br>' + bin.location_name + '<br>' +
    '<strong class="driver-popup-pct ' + pcls + '">Πληρότητα: ' + bin.current_fill_level + '%</strong><br>' +
    (collected ? '<em><i class="fi fi-sr-check-circle" aria-hidden="true"></i> Αδειάστηκε</em>' :
     '<button onclick="markCollected(\'' + bin.bin_id + '\')" class="driver-popup-btn" aria-label="Αποκομιδή κάδου"><i class="fi fi-sr-check-circle" aria-hidden="true"></i> Αδειάστηκε</button>')
   );
   mapMarkers.push(marker);
  });
 }

 // ===== ALERTS =====
 async function loadAlerts() {
  try {
```

```
    // Φόρτωση ειδοποιήσεων για το όχημα ΚΑΙ τη ζώνη
    var params = new URLSearchParams({ vehicle_id: VEHICLE_ID });
    if (ZONE_CODE) params.append('zone_code', ZONE_CODE);
    var res = await fetch(API_URL + '/api/v1/dispatch/alerts?' + params.toString());
    var json = await res.json();
    var myAlerts = json.alerts || [];
    var activeCount = myAlerts.filter(function(a) { return a.status === 'active'; }).length;
    if (activeCount > 0) {
      document.getElementById('alert-count').textContent = activeCount;
      document.getElementById('alert-count').classList.remove('hidden');
      document.getElementById('notif-tab-badge').textContent = activeCount;
      document.getElementById('notif-tab-badge').classList.remove('hidden');
    } else {
      document.getElementById('alert-count').classList.add('hidden');
      document.getElementById('notif-tab-badge').classList.add('hidden');
    }
    myAlerts.forEach(function(alert) {
      if (!knownAlertIds.has(alert.id) && alert.status === 'active') {
        knownAlertIds.add(alert.id);
        if (lastAlertCount > 0) {
          showPopup(alert.message.substring(0, 120), alert.alert_type === 'breakdown' ?
'error' : 'info');
        }
      }
    });
    lastAlertCount = myAlerts.length;
    renderAlerts(myAlerts);
  } catch(e) {
    document.getElementById('notifications-list').innerHTML =
      '<div class="empty"><i class="fi fi-sr-cross-circle" aria-
hidden="true"></i><p>Αδυναμία σύνδεσης</p></div>';
  }
 }

 function renderAlerts(alerts) {
  var list = document.getElementById('notifications-list');
  if (alerts.length === 0) {
    list.innerHTML = '<div class="empty"><i class="fi fi-sr-check-circle" aria-
hidden="true"></i><p>Δεν υπάρχουν ειδοποιήσεις</p></div>';
    return;
  }
  var labels = { bin_collected:'Αποκομιδή', vehicle_full:'Γεμάτο', breakdown:'Βλάβη',
maintenance:'Συντήρηση', xyta_arrived:'ΧΥΤΑ', xyta_trip:'Προς ΧΥΤΑ',
zone_change:'Ζώνη' };
  list.innerHTML = alerts.map(function(a) {
    var isForMe = a.vehicle_id === VEHICLE_ID;
    var isNearby = a.nearest_vehicle_id === VEHICLE_ID;
    var isDriver = a.initiated_by === 'driver';
    var variant = isForMe ? 'for-me' : isNearby ? 'nearby' : 'other';
```

```
    var label = labels[a.alert_type] || a.alert_type;
    return '<div class="alert-item ' + variant + '" role="article" aria-label="Ειδοποίηση: ' +
label + '">' +
      '<div class="alert-row-top">' +
       '<div class="alert-tags">' +
        '<span class="alert-type-label">' + label + '</span>' +
        (isForMe ? '<span class="alert-tag alert-tag-me">ΤΟ ΟΧΗΜΑ ΣΑΣ</span>' : '') +
        (isNearby ? '<span class="alert-tag alert-tag-called">ΚΛΗΘΗΚΑΤΕ</span>' : '') +
        '<span class="alert-tag ' + (isDriver ? 'alert-tag-driver' : 'alert-tag-system') + '">' +
         (isDriver ? 'Οδηγός' : 'Σύστημα') +
        '</span>' +
       '</div>' +
       '<span class="alert-status-badge ' + (a.status === 'active' ? 'active' : 'ok') + '">' +
        (a.status === 'active' ? 'Ενεργή' : 'OK') +
       '</span>' +
      '</div>' +
      '<p class="alert-message">' + a.message + '</p>' +
      (isNearby && a.status === 'active' ?
       '<div class="alert-called-box">' +
        '<p class="alert-called-text">Καλείστε να καλύψετε τη ζώνη ' +
a.suggested_zone_code + '!</p>' +
       '</div>' : '') +
      '<div class="alert-row-bottom">' +
       '<span class="alert-time">' + new Date(a.created_at).toLocaleString('el-GR') +
'</span>' +
       (a.status === 'active' ?
        '<button onclick="acknowledgeAlert(' + a.id + ')" class="alert-ack-btn" aria-
label="Επιβεβαίωση ειδοποίησης" title="Επιβεβαίωση">' +
         '<i class="fi fi-sr-check-circle" aria-hidden="true"></i> Επιβεβαίωση</button>' :
'') +
      '</div>' +
     '</div>';
   }).join('');
  }

  async function acknowledgeAlert(alertId) {
   try {
    await fetch(API_URL + '/api/v1/dispatch/alerts/' + alertId + '/acknowledge', { method:
'PUT' });
    loadAlerts();
   } catch(e) { console.error(e); }
  }

  // ===== POPUP =====
  function showPopup(msg, type) {
   type = type || 'error';
   var popup = document.getElementById('notif-popup');
   popup.className = 'notification-popup ' + (type === 'error' ? '' : type);
   document.getElementById('notif-popup-text').textContent = msg;
```

```
  popup.classList.remove('hidden');
  setTimeout(function() { popup.classList.add('hidden'); }, 6000);
 }

 // ===== GPS =====
 function startGPS() {
  if (!navigator.geolocation) { alert('GPS δεν υποστηρίζεται!'); return; }
  var intervalMs = parseInt(document.getElementById('gps-interval').value);
  document.getElementById('gps-dot').className = 'gps-dot active';
  document.getElementById('gps-status-text').textContent = 'GPS Ενεργό';
  document.getElementById('gps-start-btn').classList.add('hidden');
  document.getElementById('gps-stop-btn').classList.remove('hidden');
  sendGPS();
  gpsInterval = setInterval(sendGPS, intervalMs);
 }

 function stopGPS() {
  if (gpsInterval) { clearInterval(gpsInterval); gpsInterval = null; }
  document.getElementById('gps-dot').className = 'gps-dot inactive';
  document.getElementById('gps-status-text').textContent = 'GPS Ανενεργό';
  document.getElementById('gps-start-btn').classList.remove('hidden');
  document.getElementById('gps-stop-btn').classList.add('hidden');
 }

 function sendGPS() {
  navigator.geolocation.getCurrentPosition(
   async function(position) {
    var lat = position.coords.latitude;
    var lon = position.coords.longitude;
    var acc = position.coords.accuracy;
    document.getElementById('gps-coords').textContent = lat.toFixed(6) + ', ' +
lon.toFixed(6) + ' (±' + Math.round(acc) + 'm)';
    document.getElementById('gps-last-sent').textContent = new
Date().toLocaleTimeString('el-GR');
    gpsCount++;
    document.getElementById('gps-count').textContent = gpsCount;
    try {
     await fetch(API_URL + '/api/v1/vehicles/' + VEHICLE_ID, {
      method: 'PUT', headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ latitude: lat, longitude: lon })
     });
    } catch(e) { console.error(e); }
   },
   function(error) { document.getElementById('gps-status-text').textContent = 'Σφάλμα
GPS: ' + error.message; },
   { enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }
  );
 }
</script>
```

```
</body>
</html>
```

Driver-app/driver-app.css

```
/* ===== CollectionGarbageTruck PWA - driver-app.css ===== */

* { box-sizing: border-box; margin: 0; padding: 0; font-family: 'Consolas', monospace; }
body { background: #f5f6fa; min-height: 100vh; }

/* Header */
.header {
  background: linear-gradient(135deg, #2c3e50, #3498db);
  color: white; padding: 12px 15px;
  display: flex; align-items: center; justify-content: space-between;
  position: sticky; top: 0; z-index: 1000;
  box-shadow: 0 2px 8px rgba(0,0,0,0.3);
}
.header h1 { font-size: 1rem; display: flex; align-items: center; gap: 6px; }
.header-right { display: flex; align-items: center; gap: 8px; }
.vehicle-badge { background: rgba(255,255,255,0.2); padding: 4px 10px; border-radius:
12px; font-size: 0.8rem; }
.zone-badge { padding: 3px 8px; border-radius: 10px; font-size: 0.75rem; font-weight:
bold; }
.alert-bell { position: relative; cursor: pointer; background: none; border: none; color:
white; font-size: 1.1rem; padding: 4px; }
.alert-count { position: absolute; top: -4px; right: -4px; background: #e74c3c; color: white;
border-radius: 50%; width: 16px; height: 16px; font-size: 0.65rem; display: flex; align-
items: center; justify-content: center; font-weight: bold; }
.header-logout-btn { background: none; border: none; color: white; cursor: pointer; font-
size: 1rem; }

/* Login Screen */
#login-screen { display: flex; align-items: center; justify-content: center; min-height:
100vh; background: linear-gradient(135deg, #2c3e50, #3498db); }
.login-card { background: white; border-radius: 16px; padding: 30px; width: 90%; max-
width: 380px; box-shadow: 0 20px 60px rgba(0,0,0,0.3); }
.login-card .logo { text-align: center; margin-bottom: 15px; }
.login-card .logo i { font-size: 2.5rem; color: #3498db; }
.login-card h2 { text-align: center; color: #2c3e50; margin-bottom: 4px; font-size: 1.2rem;
}
.login-card p { text-align: center; color: #7f8c8d; font-size: 0.8rem; margin-bottom: 20px;
}

/* Form fields */
.field { margin-bottom: 14px; }
.field label { display: flex; align-items: center; gap: 6px; margin-bottom: 5px; font-weight:
bold; font-size: 0.85rem; color: #2c3e50; }
.field input, .field select { width: 100%; padding: 10px; border: 2px solid #ecf0f1; border-
radius: 8px; font-size: 0.9rem; outline: none; font-family: 'Consolas', monospace; }
```

```
.field input:focus, .field select:focus { border-color: #3498db; }

/* Buttons */
.btn { width: 100%; padding: 12px; border: none; border-radius: 8px; font-size: 0.95rem;
font-weight: bold; cursor: pointer; display: flex; align-items: center; justify-content:
center; gap: 8px; font-family: 'Consolas', monospace; }
.btn-primary { background: #3498db; color: white; }
.btn-success { background: #27ae60; color: white; }
.btn-danger  { background: #e74c3c; color: white; }
.btn-warning { background: #f39c12; color: white; }
.btn-secondary { background: #ecf0f1; color: #2c3e50; }
.btn-sm { padding: 7px 12px; font-size: 0.8rem; width: auto; border-radius: 6px; }

/* App screen */
#app-screen { display: none; }

/* Tabs */
.tabs { display: flex; background: white; border-bottom: 2px solid #ecf0f1; position:
sticky; top: 54px; z-index: 999; }
.tab-btn { flex: 1; padding: 10px 6px; border: none; background: none; cursor: pointer;
font-size: 0.75rem; color: #7f8c8d; display: flex; flex-direction: column; align-items:
center; gap: 3px; border-bottom: 3px solid transparent; transition: all 0.2s; font-family:
'Consolas', monospace; position: relative; }
.tab-btn.active { color: #3498db; border-bottom-color: #3498db; }
.tab-btn i { font-size: 1.1rem; }
.tab-badge { position: absolute; top: 6px; right: 20%; background: #e74c3c; color: white;
border-radius: 50%; width: 14px; height: 14px; font-size: 0.6rem; display: flex; align-
items: center; justify-content: center; }

/* Tab content */
.tab-content { display: none; padding: 12px; }
.tab-content.active { display: block; }

/* Cards */
.card { background: white; border-radius: 10px; padding: 14px; box-shadow: 0 2px 8px
rgba(0,0,0,0.08); margin-bottom: 12px; }

/* Stats row */
.stats-row { display: grid; grid-template-columns: repeat(3, 1fr); gap: 8px; margin-bottom:
12px; }
.stat-card { background: white; border-radius: 10px; padding: 10px; text-align: center;
box-shadow: 0 2px 8px rgba(0,0,0,0.08); }
.stat-card .val { font-size: 1.5rem; font-weight: bold; }
.stat-card .lbl { font-size: 0.65rem; color: #7f8c8d; margin-top: 2px; }

/* Zone info bar */
.zone-info { background: linear-gradient(135deg, #2c3e50, #3498db); color: white;
border-radius: 10px; padding: 12px; margin-bottom: 12px; display: flex; justify-content:
space-between; align-items: center; }
```

```
.zone-info .zone-label { font-size: 0.8rem; opacity: 0.8; }
.zone-info .zone-name { font-size: 1.1rem; font-weight: bold; }
.zone-info .zone-stats { text-align: right; font-size: 0.8rem; }

/* Capacity display */
#vehicle-capacity-display { margin-bottom: 12px; padding: 12px; }

/* Bin items */
.bin-item { background: white; border-radius: 10px; padding: 12px; margin-bottom: 10px;
box-shadow: 0 2px 8px rgba(0,0,0,0.08); border-left: 4px solid #ecf0f1; }
.bin-item.critical { border-left-color: #e74c3c; }
.bin-item.warning  { border-left-color: #f39c12; }
.bin-item.collected { border-left-color: #27ae60; opacity: 0.7; }
.bin-header { display: flex; justify-content: space-between; align-items: center; margin-
bottom: 6px; }
.bin-id { font-weight: bold; color: #3498db; font-size: 0.8rem; }
.bin-pct { font-weight: bold; font-size: 1rem; }
.fill-bar { background: #ecf0f1; border-radius: 10px; height: 7px; margin: 5px 0; overflow:
hidden; }
.fill-progress { height: 100%; border-radius: 10px; transition: width 0.3s; }
.bin-location { font-size: 0.82rem; color: #2c3e50; font-weight: bold; margin-bottom: 2px;
display: flex; align-items: center; gap: 4px; }
.bin-address { font-size: 0.72rem; color: #7f8c8d; margin-bottom: 7px; }
.bin-actions { display: flex; gap: 7px; }

/* Map */
#map { height: calc(100vh - 160px); width: 100%; }

/* GPS dot */
.gps-dot { width: 10px; height: 10px; border-radius: 50%; margin-right: 6px; display:
inline-block; }
.gps-dot.active { background: #27ae60; animation: pulse 1.5s infinite; }
.gps-dot.inactive { background: #e74c3c; }

@keyframes pulse { 0%, 100% { opacity: 1; } 50% { opacity: 0.4; } }

/* Alert items */
.alert-item { border-radius: 8px; padding: 12px; margin-bottom: 10px; border-left: 4px
solid #e74c3c; background: white; box-shadow: 0 2px 8px rgba(0,0,0,0.08); }
.alert-item.acknowledged { border-left-color: #27ae60; opacity: 0.7; }

/* Notification popup */
.notification-popup { position: fixed; top: 70px; left: 50%; transform: translateX(-50%);
background: #e74c3c; color: white; padding: 12px 20px; border-radius: 10px; z-index:
9999; box-shadow: 0 4px 20px rgba(0,0,0,0.3); max-width: 90%; font-size: 0.85rem;
display: flex; align-items: center; gap: 10px; animation: slideDown 0.3s ease; }
.notification-popup.info    { background: #3498db; }
.notification-popup.success { background: #27ae60; }
.notification-popup.warning { background: #f39c12; }
```

```
.notification-popup-close { background: none; border: none; color: white; cursor: pointer; margin-left: auto; }

@keyframes slideDown { from { top: 50px; opacity: 0; } to { top: 70px; opacity: 1; } }

/* Alert boxes */
.alert-box { padding: 10px 14px; border-radius: 8px; margin-bottom: 10px; font-size: 0.85rem; display: flex; align-items: center; gap: 8px; }
.alert-success { background: #d5f5e3; color: #27ae60; }
.alert-error   { background: #fdecea; color: #e74c3c; }
.alert-info    { background: #ebf5fb; color: #2980b9; }
.alert-warning { background: #fef9e7; color: #f39c12; }

/* Loading / Empty */
.loading { text-align: center; padding: 30px; color: #7f8c8d; }
.loading i { font-size: 1.8rem; display: block; margin-bottom: 8px; color: #3498db; }
.empty { text-align: center; padding: 30px 15px; color: #7f8c8d; }
.empty i { font-size: 2rem; display: block; margin-bottom: 8px; }

/* XYTA button */
.xyta-btn { background: linear-gradient(135deg, #8e44ad, #9b59b6); color: white; border: none; border-radius: 8px; padding: 10px 15px; width: 100%; font-weight: bold; cursor: pointer; display: flex; align-items: center; justify-content: center; gap: 8px; margin-top: 10px; font-family: 'Consolas', monospace; font-size: 0.9rem; }

/* Notifications header */
.notifications-header { display: flex; justify-content: space-between; align-items: center; margin-bottom: 12px; }
.notifications-title { margin: 0; color: #2c3e50; display: flex; align-items: center; gap: 6px; }
.notifications-refresh { background: none; border: none; cursor: pointer; color: #3498db; font-size: 1rem; }

/* GPS section */
.gps-status-row { display: flex; align-items: center; gap: 8px; margin-bottom: 12px; }
.gps-info-row { margin-bottom: 12px; }
.gps-info-label { font-size: 0.8rem; color: #7f8c8d; margin-bottom: 3px; display: flex; align-items: center; gap: 4px; }
.gps-info-value { font-weight: bold; color: #2c3e50; font-size: 0.9rem; }
.gps-count-value { font-weight: bold; color: #3498db; }

/* Report buttons grid */
.report-actions-title { margin: 0 0 10px; color: #2c3e50; font-size: 0.9rem; display: flex; align-items: center; gap: 6px; }
.report-buttons-grid { display: grid; grid-template-columns: 1fr 1fr; gap: 8px; }
.report-btn-full { grid-column: 1 / -1; }
.report-divider { margin-top: 15px; border-top: 1px solid #ecf0f1; padding-top: 15px; }

/* Zone vehicles */
```

```
.zone-vehicles-title { font-weight: bold; font-size: 0.85rem; color: #2c3e50; margin-bottom: 8px; display: flex; align-items: center; gap: 6px; }
.zone-vehicle-item { display: flex; justify-content: space-between; align-items: center; padding: 7px 8px; border-radius: 7px; margin-bottom: 5px; }
.zone-vehicle-item.is-me { background: #ebf5fb; border: 1px solid #3498db; }
.zone-vehicle-item.other { background: #f8f9fa; border: 1px solid #ecf0f1; }
.zone-vehicle-name { font-weight: bold; font-size: 0.8rem; color: #2c3e50; display: flex; align-items: center; gap: 4px; }
.zone-vehicle-badge-me { background: #3498db; color: white; padding: 1px 5px; border-radius: 6px; font-size: 0.65rem; }
.zone-vehicle-sub { font-size: 0.72rem; color: #7f8c8d; }
.zone-vehicle-status { text-align: right; }
.zone-vehicle-status-badge { display: inline-flex; align-items: center; gap: 3px; padding: 2px 7px; border-radius: 8px; font-size: 0.72rem; font-weight: bold; }
.zone-vehicle-fuel { font-size: 0.7rem; color: #7f8c8d; margin-top: 2px; }

/* Status colors */
.status-active     { background: rgba(39,174,96,0.15);  color: #27ae60; }
.status-maintenance{ background: rgba(243,156,18,0.15); color: #f39c12; }
.status-inactive   { background: rgba(231,76,60,0.15);  color: #e74c3c; }

/* PWD strength */
.pwd-strength-bar-bg { background: #ecf0f1; border-radius: 10px; height: 5px; }
.pwd-strength-bar { height: 100%; border-radius: 10px; transition: width 0.3s; }

/* Utility: hidden (αντί για display:none inline) */
.hidden { display: none !important; }

/* Stat card colors */
.stat-total    { color: #3498db; }
.stat-pending  { color: #e74c3c; }
.stat-collected{ color: #27ae60; }

/* Bins list */
.bins-list { margin-top: 12px; }

/* Load bins button */
.btn-load-bins { margin-bottom: 10px; }

/* GPS stop button (το κρύψιμο γίνεται με την .hidden· εδώ μόνο το margin) */
.gps-stop-margin { margin-top: 8px; }

/* Card titles */
.card-title {
  margin-bottom: 14px;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
```

```
}
.card-subtitle {
  margin-bottom: 10px;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 6px;
}

/*
==========================================================================
===========
   ΚΛΑΣΕΙΣ ΠΟΥ ΑΝΤΙΚΑΘΙΣΤΟΥΝ INLINE STYLES ΑΠΟ ΤΑ JS TEMPLATE
STRINGS
   Τα συνεχή πλάτη μπαρών (--cap-width / --w) ορίζονται από JS μέσω setProperty.
   ==========================================================================
============ */

/* ---- Πληρότητα οχήματος (updateCapacityDisplay) ---- */
.cap-color-ok   { --cap-color: #27ae60; }
.cap-color-warn { --cap-color: #f39c12; }
.cap-color-crit { --cap-color: #e74c3c; }

.cap-head {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 4px;
}
.cap-title { font-size: 0.8rem; font-weight: bold; color: #2c3e50; }
.cap-pct { font-weight: bold; font-size: 1rem; color: var(--cap-color, #27ae60); }

.cap-bar-wrap { margin-bottom: 8px; }
.cap-bar {
  height: 100%;
  border-radius: 10px;
  transition: width 0.3s;
  width: var(--cap-width, 0%);
  background: var(--cap-color, #27ae60);
}

.cap-grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 6px;
  font-size: 0.72rem;
  text-align: center;
}
.cap-cell { background: #f8f9fa; border-radius: 6px; padding: 5px; }
```

```
.cap-cell-num { font-weight: bold; }
.cap-cell-num.blue { color: #3498db; }
.cap-cell-num.green { color: #27ae60; }
.cap-cell-num.dyn { color: var(--cap-color, #27ae60); }
.cap-cell-sub { color: #7f8c8d; }

.cap-warning {
  background: #fdecea;
  border-radius: 6px;
  padding: 8px;
  margin-top: 8px;
  display: flex;
  align-items: center;
  gap: 6px;
}
.cap-warning i { color: #e74c3c; }
.cap-warning span { font-size: 0.8rem; color: #e74c3c; font-weight: bold; }

/* ---- Στατιστικά ζώνης (μέσα στη σκούρα μπάρα) ---- */
.zone-stat-critical { color: #ffcccc; }
.zone-stat-warning  { color: #ffe8a1; }

/* ---- Λίστα κάδων (renderBins): χρώμα ανά κατάσταση μέσω parent class ---- */
.fill-progress { width: var(--w, 0%); }
.bin-item.critical  .fill-progress { background: #e74c3c; }
.bin-item.warning   .fill-progress { background: #f39c12; }
.bin-item.collected .fill-progress { background: #27ae60; }

.bin-item.critical  .bin-pct { color: #e74c3c; }
.bin-item.warning   .bin-pct { color: #f39c12; }
.bin-item.collected .bin-pct { color: #27ae60; }

.bin-alert-icon { color: #e74c3c; }

.bin-collected-tag {
  color: #27ae60;
  font-size: 0.8rem;
  display: flex;
  align-items: center;
  gap: 4px;
}

/* ---- Markers χάρτη (Leaflet divIcon) ---- */
.driver-bin-marker {
  width: 30px;
  height: 30px;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
```

```
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 13px;
  color: white;
}
.driver-bin-marker.collected { background: #27ae60; }
.driver-bin-marker.critical  { background: #e74c3c; }
.driver-bin-marker.warning   { background: #f39c12; }

.driver-xyta-marker {
  background: #8e44ad;
  color: white;
  padding: 5px 8px;
  border-radius: 6px;
  font-size: 0.7rem;
  font-weight: bold;
  white-space: nowrap;
  display: flex;
  align-items: center;
  gap: 4px;
}

/* ---- Popup χάρτη ---- */
.driver-popup-pct { font-weight: bold; }
.driver-popup-pct.ok   { color: #27ae60; }
.driver-popup-pct.warn { color: #f39c12; }
.driver-popup-pct.crit { color: #e74c3c; }

.driver-popup-btn {
  margin-top: 6px;
  padding: 5px 10px;
  background: #27ae60;
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  width: 100%;
  font-family: 'Consolas', monospace;
}

/* ---- Ειδοποιήσεις (renderAlerts) ---- */
.alert-item.for-me { border-left-color: #e74c3c; background: #fdecea; }
.alert-item.nearby { border-left-color: #27ae60; background: #d5f5e3; }
.alert-item.other  { border-left-color: #3498db; background: #ebf5fb; }

.alert-row-top {
  display: flex;
  justify-content: space-between;
```

```
  align-items: flex-start;
  margin-bottom: 6px;
}
.alert-tags { display: flex; gap: 5px; flex-wrap: wrap; align-items: center; }
.alert-type-label { font-size: 0.75rem; font-weight: bold; }

.alert-tag { padding: 1px 6px; border-radius: 8px; font-size: 0.65rem; }
.alert-tag-me     { background: #e74c3c; color: white; }
.alert-tag-called { background: #27ae60; color: white; }
.alert-tag-driver { background: #ebf5fb; color: #2980b9; }
.alert-tag-system { background: #fef9e7; color: #f39c12; }

.alert-status-badge {
  font-size: 0.7rem;
  padding: 2px 6px;
  border-radius: 8px;
  color: white;
}
.alert-status-badge.active { background: #e74c3c; }
.alert-status-badge.ok     { background: #27ae60; }

.alert-message { margin: 0 0 6px; font-size: 0.82rem; color: #2c3e50; }

.alert-called-box {
  background: #d5f5e3;
  border-radius: 6px;
  padding: 7px;
  margin-top: 6px;
}
.alert-called-text { margin: 0; font-size: 0.8rem; color: #27ae60; font-weight: bold; }

.alert-row-bottom {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 6px;
}
.alert-time { font-size: 0.7rem; color: #7f8c8d; }
.alert-ack-btn {
  padding: 4px 10px;
  background: #27ae60;
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-size: 0.75rem;
  font-family: 'Consolas', monospace;
}
```

```
/* ---- Δυναμικό χρώμα ζώνης (header badge + info bar) ---- */
.zone-badge { background: var(--zone-color, rgba(255,255,255,0.2)); }
.zone-info  { background: linear-gradient(135deg, var(--zone-color, #2c3e50), #2c3e50); }
```

Παράρτημα Δ: «Κώδικας της Πύλης Πολιτών του ΠΣ Smart Waste Management System»

Citizen-portal/citizen-portal.html

```
<!DOCTYPE html>
<html lang="el">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Πύλη Πολιτών — Δήμος Μαλεβιζίου</title>
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"/>
      <link rel='stylesheet' href='https://cdn-uicons.flaticon.com/2.6.0/uicons-solid-rounded/css/uicons-solid-rounded.css'>
  <link rel="stylesheet" href="citizen-portal.css">
</head>
<body>


<header class="cp-header">
  <div class="cp-header-inner">
    <div class="cp-logo">
      <i class="fi fi-sr-trash" aria-hidden="true"></i>
      <div>
        <div class="cp-logo-title">Δήμος Μαλεβιζίου</div>
        <div class="cp-logo-sub">Πύλη Αναφορών Πολιτών</div>
      </div>
    </div>
```

```
  <nav class="cp-nav" aria-label="Κύρια πλοήγηση">
    <button class="cp-nav-btn active" onclick="showSection('report')" id="nav-report"
      aria-current="page" title="Νέα Αναφορά">
      <i class="fi fi-sr-add" aria-hidden="true"></i> Νέα Αναφορά
    </button>
    <button class="cp-nav-btn" onclick="showSection('track')" id="nav-track"
      title="Παρακολούθηση">
      <i class="fi fi-sr-search" aria-hidden="true"></i> Παρακολούθηση
    </button>
  </nav>
 </div>
</header>

<main class="cp-main">

 <section id="section-report" class="cp-section active" aria-label="Νέα αναφορά">
  <div class="cp-card">
   <h2 class="cp-card-title">
    <i class="fi fi-sr-add icon-mr8 icon-blue" aria-hidden="true"></i>
    Νέα Αναφορά
   </h2>

   <div id="report-alert" role="alert" aria-live="polite"></div>

   <div class="cp-steps" aria-label="Βήματα υποβολής">
```

```
  <div class="cp-step active" id="step-1" aria-label="Βήμα 1: Τύπος">
    <div class="cp-step-num">1</div>
    <div class="cp-step-label">Τύπος</div>
  </div>
  <div class="cp-step-line"></div>
  <div class="cp-step" id="step-2" aria-label="Βήμα 2: Θέση">
    <div class="cp-step-num">2</div>
    <div class="cp-step-label">Θέση</div>
  </div>
  <div class="cp-step-line"></div>
  <div class="cp-step" id="step-3" aria-label="Βήμα 3: Στοιχεία">
    <div class="cp-step-num">3</div>
    <div class="cp-step-label">Στοιχεία</div>
  </div>
  <div class="cp-step-line"></div>
  <div class="cp-step" id="step-4" aria-label="Βήμα 4: Υποβολή">
    <div class="cp-step-num">4</div>
    <div class="cp-step-label">Υποβολή</div>
  </div>
</div>

<form id="report-form" novalidate>

  <div class="cp-form-step active" id="form-step-1">
    <h3 class="cp-step-title">
```

```
  <i class="fi fi-sr-list icon-mr6 icon-blue" aria-hidden="true"></i>
  Επιλέξτε τύπο αναφοράς
 </h3>
 <div class="cp-type-grid" role="group" aria-label="Τύπος αναφοράς">
  <label class="cp-type-card" for="type-overflow">
   <input type="radio" id="type-overflow" name="report_type"
    value="overflow" aria-label="Υπερχείλιση κάδου"/>
   <i class="fi fi-sr-water icon-type" aria-hidden="true"></i>
   <span class="cp-type-label">Υπερχείλιση</span>
   <span class="cp-type-desc">Ο κάδος είναι γεμάτος/χυμένος</span>
  </label>
  <label class="cp-type-card" for="type-missed">
   <input type="radio" id="type-missed" name="report_type"
    value="missed" aria-label="Χαμένη αποκομιδή"/>
   <i class="fi fi-sr-truck-side icon-type" aria-hidden="true"></i>
   <span class="cp-type-label">Χαμένη Αποκομιδή</span>
   <span class="cp-type-desc">Δεν έγινε αποκομιδή</span>
  </label>
  <label class="cp-type-card" for="type-damage">
   <input type="radio" id="type-damage" name="report_type"
    value="damage" aria-label="Ζημιά κάδου"/>
   <i class="fi fi-sr-tools icon-type" aria-hidden="true"></i>
   <span class="cp-type-label">Ζημιά Κάδου</span>
   <span class="cp-type-desc">Φθορά ή σπασμένος κάδος</span>
  </label>
```

```
    <label class="cp-type-card" for="type-new-bin">
      <input type="radio" id="type-new-bin" name="report_type"
        value="new_bin" aria-label="Αίτημα νέου κάδου"/>
      <i class="fi fi-sr-add icon-type" aria-hidden="true"></i>
      <span class="cp-type-label">Νέος Κάδος</span>
      <span class="cp-type-desc">Αίτημα τοποθέτησης κάδου</span>
    </label>
    <label class="cp-type-card" for="type-other">
      <input type="radio" id="type-other" name="report_type"
        value="other" aria-label="Άλλο πρόβλημα"/>
      <i class="fi fi-sr-comment icon-type" aria-hidden="true"></i>
      <span class="cp-type-label">Άλλο</span>
      <span class="cp-type-desc">Άλλο πρόβλημα</span>
    </label>
  </div>
  <button type="button" class="cp-btn cp-btn-primary" onclick="nextStep(2)"
    aria-label="Επόμενο βήμα">
    <i class="fi fi-sr-arrow-right" aria-hidden="true"></i> Επόμενο
  </button>
</div>

<div class="cp-form-step" id="form-step-2">
  <h3 class="cp-step-title">
    <i class="fi fi-sr-marker icon-mr6 icon-blue" aria-hidden="true"></i>
    Εντοπίστε τον κάδο στον χάρτη
```

```
</h3>
<p class="cp-hint">
  <i class="fi fi-sr-info icon-mr4 icon-blue2" aria-hidden="true"></i>
  Κάντε κλικ σε έναν κάδο ή καρφιτσώστε τη θέση αν δεν τον βρίσκετε.
</p>

<div class="cp-map-controls">
  <button type="button" class="cp-map-btn active" id="btn-select-bin"
    onclick="setMapMode('select')"
    aria-label="Επιλογή υπάρχοντος κάδου" title="Επιλογή κάδου">
    <i class="fi fi-sr-trash" aria-hidden="true"></i> Επιλογή Κάδου
  </button>
  <button type="button" class="cp-map-btn" id="btn-pin-location"
    onclick="setMapMode('pin')"
    aria-label="Καρφίτσωμα θέσης στον χάρτη" title="Καρφίτσωμα θέσης">
    <i class="fi fi-sr-map-marker" aria-hidden="true"></i>
    Καρφίτσωμα Θέσης
  </button>
</div>

<div id="citizen-map" aria-label="Χάρτης κάδων Δήμου Μαλεβιζίου"></div>

<div id="selected-location" class="cp-selected-location hidden"
  role="status" aria-live="polite">
  <i class="fi fi-sr-check-circle icon-mr6 icon-green" aria-hidden="true"></i>
```

```
  <span id="selected-location-text">—</span>
</div>

<div class="cp-form-field">
  <label for="address-hint" class="cp-label">
    <i class="fi fi-sr-marker icon-mr4 icon-gray" aria-hidden="true"></i>
    Διεύθυνση / Περισσότερες πληροφορίες θέσης
  </label>
  <input type="text" id="address-hint" name="address_hint"
    placeholder="π.χ. Δίπλα στο σχολείο, γωνία οδού..."
    class="cp-input"
    aria-label="Διεύθυνση ή περιγραφή θέσης"
    title="Εισάγετε διεύθυνση ή περιγραφή θέσης"/>
</div>

<div class="cp-form-actions">
  <button type="button" class="cp-btn cp-btn-secondary" onclick="prevStep(1)"
    aria-label="Προηγούμενο βήμα">
    <i class="fi fi-sr-arrow-left" aria-hidden="true"></i> Πίσω
  </button>
  <button type="button" class="cp-btn cp-btn-primary" onclick="nextStep(3)"
    aria-label="Επόμενο βήμα">
    <i class="fi fi-sr-arrow-right" aria-hidden="true"></i> Επόμενο
  </button>
</div>
```

```
</div>

<div class="cp-form-step" id="form-step-3">
 <h3 class="cp-step-title">
  <i class="fi fi-sr-document icon-mr6 icon-blue" aria-hidden="true"></i>
  Συμπληρώστε τα στοιχεία
 </h3>

 <div class="cp-form-field">
  <label for="report-title" class="cp-label">
   <i class="fi fi-sr-text icon-mr4 icon-gray" aria-hidden="true"></i>
   Τίτλος αναφοράς <span class="cp-required">*</span>
  </label>
  <input type="text" id="report-title" name="title"
   placeholder="Σύντομος τίτλος..." class="cp-input"
   aria-required="true" aria-label="Τίτλος αναφοράς"
   title="Εισάγετε τίτλο αναφοράς" required/>
 </div>

 <div class="cp-form-field">
  <label for="report-description" class="cp-label">
   <i class="fi fi-sr-document icon-mr4 icon-gray" aria-hidden="true"></i>
   Περιγραφή <span class="cp-required">*</span>
  </label>
  <textarea id="report-description" name="description"
```

```
    placeholder="Περιγράψτε το πρόβλημα..." class="cp-textarea"
    rows="4" aria-required="true" aria-label="Περιγραφή προβλήματος"
    title="Περιγράψτε το πρόβλημα" required></textarea>
</div>

<div class="cp-form-field">
  <label for="report-photo" class="cp-label">
    <i class="fi fi-sr-camera icon-mr4 icon-gray" aria-hidden="true"></i>
    Φωτογραφία (προαιρετικά)
  </label>
  <div class="cp-photo-upload">
    <input type="file" id="report-photo" name="photo"
      accept="image/*" class="cp-file-input"
      aria-label="Ανέβασμα φωτογραφίας"
      title="Επιλέξτε φωτογραφία (προαιρετικά)"
      onchange="previewPhoto(this)"/>
    <label for="report-photo" class="cp-photo-label">
      <i class="fi fi-sr-upload icon-mr6" aria-hidden="true"></i>
      Επιλογή Φωτογραφίας
    </label>
  </div>
  <div id="photo-preview" class="cp-photo-preview hidden"
    aria-live="polite"></div>
</div>
```

```
<div class="cp-form-field">
  <label for="citizen-name" class="cp-label">
    <i class="fi fi-sr-user icon-mr4 icon-gray" aria-hidden="true"></i>
    Όνομα <span class="cp-required">*</span>
  </label>
  <input type="text" id="citizen-name" name="citizen_name"
    placeholder="Το όνομά σας..." class="cp-input"
    aria-required="true" aria-label="Όνομα πολίτη"
    title="Εισάγετε το όνομά σας" required/>
</div>

<div class="cp-form-field">
  <label for="citizen-email" class="cp-label">
    <i class="fi fi-sr-envelope icon-mr4 icon-gray" aria-hidden="true"></i>
    Email (για επιβεβαίωση — προαιρετικά)
  </label>
  <input type="email" id="citizen-email" name="citizen_email"
    placeholder="email@example.com" class="cp-input"
    aria-label="Email για επιβεβαίωση"
    title="Εισάγετε email για λήψη επιβεβαίωσης"/>
</div>

<div class="cp-form-actions">
  <button type="button" class="cp-btn cp-btn-secondary" onclick="prevStep(2)"
    aria-label="Προηγούμενο βήμα">
```

```
      <i class="fi fi-sr-arrow-left" aria-hidden="true"></i> Πίσω
    </button>
    <button type="button" class="cp-btn cp-btn-primary" onclick="nextStep(4)"
      aria-label="Επόμενο βήμα">
      <i class="fi fi-sr-arrow-right" aria-hidden="true"></i> Επόμενο
    </button>
  </div>
</div>

<div class="cp-form-step" id="form-step-4">
  <h3 class="cp-step-title">
    <i class="fi fi-sr-check-circle icon-mr6 icon-green" aria-hidden="true"></i>
    Επισκόπηση &amp;amp; Υποβολή
  </h3>

  <div id="review-content" class="cp-review" aria-label="Επισκόπηση αναφοράς">
  </div>

  <div class="cp-form-actions">
    <button type="button" class="cp-btn cp-btn-secondary" onclick="prevStep(3)"
      aria-label="Προηγούμενο βήμα">
      <i class="fi fi-sr-arrow-left" aria-hidden="true"></i> Πίσω
    </button>
    <button type="submit" class="cp-btn cp-btn-submit" id="submit-btn"
      aria-label="Υποβολή αναφοράς" title="Υποβολή αναφοράς">
```

```
          <i class="fi fi-sr-paper-plane" aria-hidden="true"></i> Υποβολή Αναφοράς
        </button>
       </div>
      </div>


    </form>
   </div>


    <div id="success-card" class="cp-card cp-success-card hidden" role="status" aria-live="polite">
     <div class="cp-success-icon" aria-hidden="true">
      <i class="fi fi-sr-check-circle"></i>
     </div>
     <h2 class="cp-success-title">Η αναφορά υποβλήθηκε!</h2>
     <p class="cp-success-sub">Κωδικός παρακολούθησης:</p>
     <div class="cp-tracking-code" id="success-tracking-code" aria-live="polite">—</div>
     <p class="cp-success-hint">
      Αποθηκεύστε αυτόν τον κωδικό για να παρακολουθείτε
      την πρόοδο της αναφοράς σας.
     </p>
     <div id="success-email-note" class="hidden cp-email-note">
      <i class="fi fi-sr-envelope icon-mr6 icon-blue" aria-hidden="true"></i>
      Στάλθηκε email επιβεβαίωσης.
     </div>
     <button class="cp-btn cp-btn-primary" onclick="resetForm()"
      aria-label="Υποβολή νέας αναφοράς">
```

```
    <i class="fi fi-sr-add" aria-hidden="true"></i> Νέα Αναφορά
   </button>
   <button class="cp-btn cp-btn-secondary" onclick="goTrack()"
    aria-label="Παρακολούθηση αναφοράς">
    <i class="fi fi-sr-search" aria-hidden="true"></i> Παρακολούθηση
   </button>
  </div>
</section>

<section id="section-track" class="cp-section" aria-label="Παρακολούθηση αναφοράς">
 <div class="cp-card">
  <h2 class="cp-card-title">
   <i class="fi fi-sr-search icon-mr8 icon-blue" aria-hidden="true"></i>
   Παρακολούθηση Αναφοράς
  </h2>

  <div class="cp-form-field">
   <label for="track-code" class="cp-label">
    <i class="fi fi-sr-key icon-mr4 icon-gray" aria-hidden="true"></i>
    Κωδικός Παρακολούθησης
   </label>
   <div class="cp-track-row">
    <input type="text" id="track-code" placeholder="π.χ. CR1-0001"
     class="cp-input cp-track-input"
     aria-label="Κωδικός παρακολούθησης αναφοράς"
```

```
          title="Εισάγετε τον κωδικό παρακολούθησης"/>
        <button type="button" class="cp-btn cp-btn-primary" onclick="trackReport()"
          aria-label="Αναζήτηση αναφοράς" title="Αναζήτηση">
          <i class="fi fi-sr-search" aria-hidden="true"></i> Αναζήτηση
        </button>
      </div>
    </div>

    <div id="track-result" aria-live="polite"></div>
  </div>
 </section>

</main>

<footer class="cp-footer">
  <p>Smart Waste Management — Δήμος Μαλεβιζίου © 2026</p>
  <p>
    <a href="http://localhost:3000" class="cp-footer-link" title="Κεντρική εφαρμογή">
      <i class="fi fi-sr-home" aria-hidden="true"></i> Κεντρική Εφαρμογή
    </a>
  </p>
</footer>

<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
<script>
```

```
var API_URL = 'http://localhost:8000';
var currentStep = 1;
var selectedBinId = null;
var selectedLat = null;
var selectedLon = null;
var selectedLocationName = '';
var isPinLocation = false;
var mapMode = 'select';
var citizenMap = null;
var pinMarker = null;
var binMarkers = [];
var lastTrackingCode = '';
var selectedMarker = null;

var TYPE_LABELS = {
  overflow: 'Υπερχείλιση Κάδου',
  missed:   'Χαμένη Αποκομιδή',
  damage:   'Ζημιά Κάδου',
  new_bin:  'Αίτημα Νέου Κάδου',
  other:    'Άλλο',
};

var STATUS_LABELS = {
  open:        'Ανοιχτή',
  in_progress: 'Σε Εξέλιξη',
```

```
  resolved:   'Επιλύθηκε',
  closed:     'Κλειστή',
};

function showSection(section) {
  document.querySelectorAll('.cp-section').forEach(function(s) {
    s.classList.remove('active');
  });
  document.querySelectorAll('.cp-nav-btn').forEach(function(b) {
    b.classList.remove('active');
    b.removeAttribute('aria-current');
  });
  document.getElementById('section-' + section).classList.add('active');
  var btn = document.getElementById('nav-' + section);
  btn.classList.add('active');
  btn.setAttribute('aria-current', 'page');

  if (section === 'report') {
    if (!citizenMap) {
      setTimeout(function() { initMap(); loadBins(); }, 200);
    } else {
      setTimeout(function() { citizenMap.invalidateSize(); }, 100);
    }
  }
}
```

```
function nextStep(step) {
  if (step === 2) {
    var type = document.querySelector('input[name="report_type"]:checked');
    if (!type) { showAlert('Επιλέξτε τύπο αναφοράς.', 'error'); return; }
    // The map needs its container visible before it can size itself.
    setTimeout(function() {
      if (!citizenMap) {
        initMap();
        loadBins();
      } else {
        citizenMap.invalidateSize();
      }
    }, 150);
  }
  if (step === 3) {
    if (!selectedLat || !selectedLon) {
      showAlert('Επιλέξτε θέση στον χάρτη ή καρφιτσώστε τη θέση σας.', 'error');
      return;
    }
  }
  if (step === 4) {
    var title = document.getElementById('report-title').value.trim();
    var desc  = document.getElementById('report-description').value.trim();
    var name  = document.getElementById('citizen-name').value.trim();
```

```
    if (!title || !desc || !name) {
      showAlert('Συμπληρώστε Τίτλο, Περιγραφή και Όνομα.', 'error');
      return;
    }
    buildReview();
  }
  setStep(step);
}

function prevStep(step) { setStep(step); }

function setStep(step) {
  currentStep = step;
  document.querySelectorAll('.cp-form-step').forEach(function(s) {
    s.classList.remove('active');
  });
  document.getElementById('form-step-' + step).classList.add('active');
  document.querySelectorAll('.cp-step').forEach(function(s, i) {
    s.classList.remove('active', 'done');
    if (i + 1 < step) s.classList.add('done');
    if (i + 1 === step) s.classList.add('active');
  });
  document.getElementById('report-alert').innerHTML = '';
}
```

```
function initMap() {
  if (citizenMap) return;
  citizenMap = L.map('citizen-map', {
    center: [35.33, 25.06],
    zoom: 13,
    doubleClickZoom: false,
  });
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '© OpenStreetMap'
  }).addTo(citizenMap);

  // Tell a click apart from the end of a drag by timing the mouse press.
  var mouseDownTime = 0;
  citizenMap.on('mousedown', function() {
    mouseDownTime = Date.now();
  });
  citizenMap.on('click', function(e) {
    var elapsed = Date.now() - mouseDownTime;
    if (mapMode !== 'pin') return;
    if (elapsed > 200) return;
    setPinLocation(e.latlng.lat, e.latlng.lng);
  });

  loadBins();
}
```

```
async function loadBins() {
  try {
    var res = await fetch(API_URL + '/api/v1/citizen/bins');
    var json = await res.json();
    var bins = json.bins || [];

    binMarkers.forEach(function(m) { m.remove(); });
    binMarkers = [];

    bins.forEach(function(bin) {
      var statusCls = bin.fill_status === 'critical' ? 'critical' :
                      bin.fill_status === 'warning'  ? 'warning' : 'ok';
      var icon = L.divIcon({
        html: '<div class="cp-bin-marker ' + statusCls + '" role="img" aria-label="' +
bin.bin_id + '">' +
          '<i class="fi fi-sr-trash" aria-hidden="true"></i></div>',
        className: '', iconSize: [26,26], iconAnchor: [13,13]
      });
      var marker = L.marker([bin.latitude, bin.longitude], { icon: icon });
      marker.bindPopup(
        '<div class="cp-bin-popup">' +
          '<strong>' + bin.location_name + '</strong><br>' +
          '<small>' + bin.bin_id + '</small><br>' +
          '<small>' + (bin.address || '') + '</small><br>' +
          '<button onclick="selectBin(\'' + bin.bin_id + '\',\'' +
```

```
          bin.location_name.replace(/'/g,"\'") + '\',' +
          bin.latitude + ',' + bin.longitude + '); return false;" ' +
          'class="cp-bin-popup-btn" ' +
          'aria-label="Επιλογή κάδου ' + bin.bin_id + '">' +
          '<i class=\"fi fi-sr-check-circle icon-mr4\" aria-hidden=\"true\"></i>' +
          'Επιλογή κάδου</button>' +
        '</div>'
      );
      marker.addTo(citizenMap);
      binMarkers.push(marker);
    });
  } catch(e) { console.error(e); }
}

function selectBin(binId, name, lat, lon) {
  selectedBinId = binId;
  selectedLat = parseFloat(lat);
  selectedLon = parseFloat(lon);
  selectedLocationName = name;
  isPinLocation = false;
  if (pinMarker) { pinMarker.remove(); pinMarker = null; }

  // Dim every marker, then highlight the one the citizen picked.
  binMarkers.forEach(function(m) {
    var el = m.getElement();
```

```
    if (el) el.style.opacity = '0.5';
  });
  selectedMarker = binMarkers.find(function(m) {
    return m.getLatLng().lat.toFixed(5) === parseFloat(lat).toFixed(5);
  });
  if (selectedMarker) {
    var el = selectedMarker.getElement();
    if (el) {
      el.style.opacity = '1';
      el.style.transform = 'scale(1.4)';
      el.style.zIndex = '1000';
    }
  }

  showSelectedLocation('Επιλεγμένος κάδος: ' + name + ' (' + binId + ')');
  citizenMap.closePopup();
  citizenMap.setView([lat, lon], 17);
}

function setPinLocation(lat, lon) {
  selectedLat = parseFloat(lat.toFixed(8));
  selectedLon = parseFloat(lon.toFixed(8));
  selectedBinId = null;
  isPinLocation = true;
  selectedLocationName = lat.toFixed(5) + ', ' + lon.toFixed(5);
```

```
  if (pinMarker) pinMarker.remove();
  // Re-enable dragging now that the pin is placed.
  setMapMode('select');
  pinMarker = L.marker([lat, lon], {
    icon: L.divIcon({
      html: '<div class="cp-pin-marker" role="img" aria-label="Καρφίτσα θέσης">' +
            '<i class="fi fi-sr-map-marker" aria-hidden="true"></i></div>',
      className: '', iconSize: [32,32], iconAnchor: [16,16]
    })
  }).addTo(citizenMap);
  showSelectedLocation('Καρφιτσωμένη θέση: ' + selectedLocationName);
}

function setMapMode(mode) {
  mapMode = mode;
  document.getElementById('btn-select-bin').classList.toggle('active', mode === 'select');
  document.getElementById('btn-pin-location').classList.toggle('active', mode === 'pin');
  if (citizenMap) {
    var mapEl = document.getElementById('citizen-map');
    if (mode === 'pin') {
      citizenMap.getContainer().style.cursor = 'crosshair';
      citizenMap.dragging.disable();
      citizenMap.touchZoom.disable();
      if (mapEl) mapEl.classList.add('pin-mode');
    } else {
```

```
      citizenMap.getContainer().style.cursor = '';
      citizenMap.dragging.enable();
      citizenMap.touchZoom.enable();
      if (mapEl) mapEl.classList.remove('pin-mode');
    }
  }
}

function showSelectedLocation(text) {
  var el = document.getElementById('selected-location');
  el.classList.remove('hidden');
  document.getElementById('selected-location-text').textContent = text;
}

function previewPhoto(input) {
  var preview = document.getElementById('photo-preview');
  if (input.files && input.files[0]) {
    var reader = new FileReader();
    reader.onload = function(e) {
      preview.innerHTML =
        '<img src="' + e.target.result + '" alt="Προεπισκόπηση φωτογραφίας" ' +
        'class="cp-photo-img"/>' +
        '<button type="button" onclick="removePhoto()" class="cp-photo-remove" ' +
        'aria-label="Αφαίρεση φωτογραφίας" title="Αφαίρεση">' +
        '<i class="fi fi-sr-cross-circle" aria-hidden="true"></i></button>';
```

```
    preview.classList.remove('hidden');
   };
   reader.readAsDataURL(input.files[0]);
  }
}

function removePhoto() {
  document.getElementById('report-photo').value = '';
  var preview = document.getElementById('photo-preview');
  preview.innerHTML = '';
  preview.classList.add('hidden');
}

function buildReview() {
  var type = document.querySelector('input[name="report_type"]:checked');
  var title = document.getElementById('report-title').value;
  var desc  = document.getElementById('report-description').value;
  var name  = document.getElementById('citizen-name').value;
  var email = document.getElementById('citizen-email').value;
  var hint  = document.getElementById('address-hint').value;

  var rows = [
   ['Τύπος', TYPE_LABELS[type ? type.value : ''] || '—'],
   ['Θέση', selectedLocationName || '—'],
   ['Κάδος', selectedBinId || (isPinLocation ? 'Καρφιτσωμένη θέση' : '—')],
```

```
    ['Διεύθυνση', hint || '—'],
    ['Τίτλος', title],
    ['Περιγραφή', desc],
    ['Όνομα', name],
    ['Email', email || 'Δεν δόθηκε'],
  ];

  var html = '<table class="cp-review-table" aria-label="Επισκόπηση αναφοράς">';
  rows.forEach(function(row) {
    html += '<tr><td class="cp-review-key">' + row[0] +
            '</td><td class="cp-review-val">' + row[1] + '</td></tr>';
  });
  html += '</table>';
  document.getElementById('review-content').innerHTML = html;
}

document.getElementById('report-form').addEventListener('submit', async function(e) {
  e.preventDefault();
  var btn = document.getElementById('submit-btn');
  btn.disabled = true;
  btn.innerHTML = '<i class="fi fi-sr-rotate-right" aria-hidden="true"></i> Υποβολή...';

  try {
    var type   = document.querySelector('input[name="report_type"]:checked');
    var title  = document.getElementById('report-title').value.trim();
```

```
var desc   = document.getElementById('report-description').value.trim();
var name   = document.getElementById('citizen-name').value.trim();
var email  = document.getElementById('citizen-email').value.trim();
var hint   = document.getElementById('address-hint').value.trim();
var photo  = document.getElementById('report-photo').files[0];

var formData = new FormData();
formData.append('report_type',    type ? type.value : 'other');
formData.append('title',          title);
formData.append('description',    desc);
formData.append('latitude',       parseFloat(selectedLat).toString());
formData.append('longitude',      parseFloat(selectedLon).toString());
formData.append('citizen_name',   name);
if (email)       formData.append('citizen_email',  email);
if (selectedBinId) formData.append('bin_id',       selectedBinId);
if (hint)        formData.append('address_hint',   hint);
formData.append('is_pin_location', isPinLocation ? 'true' : 'false');
if (photo)       formData.append('photo',          photo);

var res = await fetch(API_URL + '/api/v1/citizen/reports', {
  method: 'POST', body: formData
});
var json = await res.json();

if (res.ok) {
```

```
      lastTrackingCode = json.tracking_code;
      document.getElementById('success-tracking-code').textContent = json.tracking_code;
      document.getElementById('report-form').classList.add('hidden');
      document.getElementById('success-card').classList.remove('hidden');
      if (json.email_sent) {
        document.getElementById('success-email-note').classList.remove('hidden');
      }
    } else {
      showAlert(json.detail || 'Σφάλμα υποβολής.', 'error');
      btn.disabled = false;
       btn.innerHTML = '<i class="fi fi-sr-paper-plane" aria-hidden="true"></i> Υποβολή Αναφοράς';
    }
  } catch(err) {
    showAlert('Αδυναμία σύνδεσης με τον server.', 'error');
    btn.disabled = false;
     btn.innerHTML = '<i class="fi fi-sr-paper-plane" aria-hidden="true"></i> Υποβολή Αναφοράς';
  }
});

async function trackReport() {
  var code = document.getElementById('track-code').value.trim().toUpperCase();
  if (!code) { return; }
  var result = document.getElementById('track-result');
```

```
    result.innerHTML = '<div class="cp-loading"><i class="fi fi-sr-rotate-right" aria-hidden="true"></i> Αναζήτηση...</div>';

  try {
    var res = await fetch(API_URL + '/api/v1/citizen/reports/' + code);
    var json = await res.json();

    if (!res.ok) {
      result.innerHTML =
        '<div class="cp-alert cp-alert-error" role="alert">' +
        '<i class="fi fi-sr-cross-circle" aria-hidden="true"></i> ' +
        (json.detail || 'Δεν βρέθηκε αναφορά.') + '</div>';
      return;
    }

    var r = json.report;

    result.innerHTML =
      '<div class="cp-track-result" role="region" aria-label="Αποτέλεσμα αναζήτησης">' +
        '<div class="cp-track-header">' +
          '<span class="cp-track-code">' + r.tracking_code + '</span>' +
          '<span class="cp-status-badge ' + r.status + '">' +
            r.status_label + '</span>' +
        '</div>' +
        '<table class="cp-review-table" aria-label="Στοιχεία αναφοράς">' +
```

```
            '<tr><td class="cp-review-key">Τύπος</td><td class="cp-review-val">' +
r.type_label + '</td></tr>' +
        '<tr><td class="cp-review-key">Τίτλος</td><td class="cp-review-val">' + r.title +
'</td></tr>' +
          '<tr><td class="cp-review-key">Περιγραφή</td><td class="cp-review-val">' +
r.description + '</td></tr>' +
        (r.bin_id ? '<tr><td class="cp-review-key">Κάδος</td><td class="cp-review-val">'
+ r.bin_id + '</td></tr>' : '') +
         '<tr><td class="cp-review-key">Υποβολή</td><td class="cp-review-val">' + new
Date(r.created_at).toLocaleString('el-GR') + '</td></tr>' +
           (r.resolved_at ? '<tr><td class="cp-review-key">Επίλυση</td><td class="cp-
review-val">' + new Date(r.resolved_at).toLocaleString('el-GR') + '</td></tr>' : '') +
       '</table>' +
       '<div class="cp-status-timeline">' +
        ['open','in_progress','resolved'].map(function(s) {
          var done = ['open','in_progress','resolved'].indexOf(r.status) >=
                ['open','in_progress','resolved'].indexOf(s);
          return '<div class="cp-timeline-step ' + (done ? 'done' : '') + '">' +
                '<div class="cp-timeline-dot"></div>' +
                '<div class="cp-timeline-label">' + (STATUS_LABELS[s] || s) + '</div>' +
              '</div>';
        }).join('<div class="cp-timeline-line"></div>') +
       '</div>' +
     '</div>';
  } catch(err) {
    result.innerHTML =
     '<div class="cp-alert cp-alert-error" role="alert">' +
```

```
    '<i class="fi fi-sr-cross-circle" aria-hidden="true"></i> Αδυναμία σύνδεσης.</div>';
  }
}

function showAlert(msg, type) {
  document.getElementById('report-alert').innerHTML =
    '<div class="cp-alert cp-alert-' + type + '" role="alert">' +
      '<i class="fi fi-sr-' + (type==='error'?'cross-circle':'check-circle') +
      '" aria-hidden="true"></i> ' + msg + '</div>';
}

function resetForm() {
  currentStep = 1;
  selectedBinId = null; selectedLat = null; selectedLon = null;
  selectedLocationName = ''; isPinLocation = false;
  selectedMarker = null;
  binMarkers.forEach(function(m) {
    var el = m.getElement();
    if (el) { el.style.opacity = '1'; el.style.transform = ''; }
  });
  document.getElementById('report-form').reset();
  document.getElementById('report-form').classList.remove('hidden');
  document.getElementById('success-card').classList.add('hidden');
  document.getElementById('photo-preview').innerHTML = '';
  document.getElementById('photo-preview').classList.add('hidden');
```

```
  document.getElementById('selected-location').classList.add('hidden');
  setStep(1);
 }

 function goTrack() {
  showSection('track');
  document.getElementById('track-code').value = lastTrackingCode;
  if (lastTrackingCode) trackReport();
 }

 showSection('report');
</script>
</body>
</html>
```

Citizen-portal/citizen-portal.css

```
/* ===== citizen-portal.css ===== */

* { box-sizing: border-box; margin: 0; padding: 0; font-family: 'Segoe UI', Arial, sans-serif;
}
body { background: #f0f4f8; min-height: 100vh; }

/* Icon utilities */
.icon-mr8  { margin-right: 8px; }
.icon-mr6  { margin-right: 6px; }
.icon-mr4  { margin-right: 4px; }
.icon-blue  { color: #3498db; }
```

```
.icon-blue2 { color: #2980b9; }
.icon-green { color: #27ae60; }
.icon-gray  { color: #7f8c8d; }
.hidden { display: none !important; }

/* ===== HEADER ===== */
.cp-header {
  background: linear-gradient(135deg, #1a252f, #2c3e50);
  color: white;
  padding: 0 20px;
  box-shadow: 0 2px 10px rgba(0,0,0,0.3);
  position: sticky; top: 0; z-index: 1000;
}

.cp-header-inner {
  max-width: 1000px; margin: 0 auto;
  display: flex; align-items: center;
  justify-content: space-between;
  padding: 12px 0;
}

.cp-logo {
  display: flex; align-items: center; gap: 12px;
}
```

```
.cp-logo i { font-size: 2rem; color: #3498db; }

.cp-logo-title { font-size: 1.1rem; font-weight: bold; }

.cp-logo-sub { font-size: 0.8rem; opacity: 0.8; }


.cp-nav { display: flex; gap: 8px; }


.cp-nav-btn {

  padding: 8px 16px; background: rgba(255,255,255,0.1);

  color: white; border: 1px solid rgba(255,255,255,0.3);

  border-radius: 20px; cursor: pointer; font-size: 0.9rem;

  display: flex; align-items: center; gap: 6px;

  transition: all 0.2s; font-family: inherit;

}


.cp-nav-btn:hover { background: rgba(255,255,255,0.2); }

.cp-nav-btn.active { background: white; color: #2c3e50; border-color: white; }


/* ===== MAIN ===== */

.cp-main { max-width: 1000px; margin: 0 auto; padding: 24px 16px; }


.cp-section { display: none; }

.cp-section.active { display: block; }


/* ===== CARD ===== */

.cp-card {
```

```
  background: white; border-radius: 12px;
  padding: 28px; box-shadow: 0 2px 12px rgba(0,0,0,0.08);
  margin-bottom: 20px;
}

.cp-card-title {
  font-size: 1.3rem; color: #2c3e50;
  margin-bottom: 20px;
  display: flex; align-items: center;
  padding-bottom: 12px; border-bottom: 2px solid #ecf0f1;
}

/* ===== STEPS ===== */
.cp-steps {
  display: flex; align-items: center;
  margin-bottom: 28px;
}

.cp-step {
  display: flex; flex-direction: column; align-items: center; gap: 4px;
  flex-shrink: 0;
}

.cp-step-num {
  width: 34px; height: 34px; border-radius: 50%;
```

```
  background: #ecf0f1; color: #7f8c8d;
  display: flex; align-items: center; justify-content: center;
  font-weight: bold; font-size: 0.9rem;
  transition: all 0.3s;
}

.cp-step.active .cp-step-num { background: #3498db; color: white; }
.cp-step.done   .cp-step-num { background: #27ae60; color: white; }

.cp-step-label { font-size: 0.75rem; color: #7f8c8d; }
.cp-step.active .cp-step-label { color: #3498db; font-weight: bold; }
.cp-step.done   .cp-step-label { color: #27ae60; }

.cp-step-line { flex: 1; height: 2px; background: #ecf0f1; margin: 0 6px; margin-bottom: 18px; }

/* ===== FORM STEPS ===== */
.cp-form-step { display: none; }
.cp-form-step.active { display: block; }

.cp-step-title {
  font-size: 1rem; color: #2c3e50; margin-bottom: 16px;
  display: flex; align-items: center;
}

/* ===== TYPE GRID ===== */
```

```
.cp-type-grid {
  display: grid; grid-template-columns: repeat(auto-fill, minmax(160px, 1fr));
  gap: 12px; margin-bottom: 20px;
}

.cp-type-card {
  border: 2px solid #ecf0f1; border-radius: 10px;
  padding: 16px 12px; cursor: pointer; text-align: center;
  transition: all 0.2s; display: flex; flex-direction: column;
  align-items: center; gap: 6px;
}

.cp-type-card:hover { border-color: #aed6f1; background: #f8f9fa; }
.cp-type-card input[type="radio"] { position: absolute; opacity: 0; width: 0; height: 0; }

.cp-type-card:has(input:checked) {
  border-color: #3498db; background: #ebf5fb;
}

.icon-type { font-size: 1.8rem; color: #3498db; }
.cp-type-label { font-weight: bold; font-size: 0.9rem; color: #2c3e50; }
.cp-type-desc { font-size: 0.75rem; color: #7f8c8d; }

/* ===== MAP ===== */
.cp-hint {
```

```
  font-size: 0.85rem; color: #5d6d7e;
  margin-bottom: 12px; display: flex; align-items: center; gap: 6px;
}

.cp-map-controls {
  display: flex; gap: 8px; margin-bottom: 10px;
}

.cp-map-btn {
  padding: 7px 14px; border: 2px solid #ecf0f1;
  border-radius: 8px; cursor: pointer; background: white;
  font-size: 0.85rem; color: #7f8c8d;
  display: flex; align-items: center; gap: 6px;
  transition: all 0.2s; font-family: inherit;
}

.cp-map-btn:hover { border-color: #3498db; color: #3498db; }
.cp-map-btn.active { border-color: #3498db; background: #ebf5fb; color: #3498db; font-weight: bold; }
.cp-map-btn.active[id="btn-pin-location"] {
  border-color: #8e44ad; background: #f5eef8; color: #8e44ad;
}

#citizen-map {
  height: 420px; border-radius: 10px; overflow: hidden;
  border: 2px solid #3498db; margin-bottom: 12px;
```

```
  box-shadow: 0 2px 12px rgba(52,152,219,0.15);
  transition: border-color 0.2s;
}

#citizen-map.pin-mode {
  border-color: #8e44ad;
  box-shadow: 0 2px 12px rgba(142,68,173,0.3);
  cursor: crosshair !important;
}

#citizen-map.pin-mode * {
  cursor: crosshair !important;
}

.cp-selected-location {
  background: #d5f5e3; border-radius: 8px; padding: 12px 16px;
  font-size: 0.9rem; color: #27ae60; margin-bottom: 12px;
  display: flex; align-items: center; gap: 8px;
  border: 2px solid #27ae60; font-weight: bold;
  animation: fadeIn 0.3s ease;
}
@keyframes fadeIn { from { opacity: 0; transform: translateY(-4px); } to { opacity: 1; transform: none; } }

/* ===== FORM FIELDS ===== */
.cp-form-field { margin-bottom: 16px; }
```

```
.cp-label {
  display: flex; align-items: center; gap: 5px;
  margin-bottom: 6px; font-weight: bold;
  font-size: 0.87rem; color: #2c3e50;
}

.cp-required { color: #e74c3c; }

.cp-input, .cp-textarea {
  width: 100%; padding: 10px 14px;
  border: 2px solid #ecf0f1; border-radius: 8px;
  font-size: 0.95rem; outline: none;
  transition: border-color 0.2s; font-family: inherit;
  box-sizing: border-box;
}

.cp-input:focus, .cp-textarea:focus { border-color: #3498db; }
.cp-textarea { resize: vertical; min-height: 100px; }

/* ===== PHOTO UPLOAD ===== */
.cp-photo-upload { position: relative; }

.cp-file-input {
  position: absolute; opacity: 0; width: 0; height: 0;
```

```
}

.cp-photo-label {
  display: inline-flex; align-items: center; gap: 8px;
  padding: 9px 18px; background: #ecf0f1; color: #2c3e50;
  border-radius: 8px; cursor: pointer; font-size: 0.9rem;
  font-weight: bold; border: 2px solid #ecf0f1;
  transition: all 0.2s;
}

.cp-photo-label:hover { border-color: #3498db; color: #3498db; background: #ebf5fb; }

.cp-photo-preview {
  margin-top: 10px; position: relative; display: inline-block;
}

.cp-photo-img {
  max-width: 200px; max-height: 150px;
  border-radius: 8px; border: 2px solid #ecf0f1;
  display: block;
}

.cp-photo-remove {
  position: absolute; top: -8px; right: -8px;
  background: #e74c3c; color: white; border: none;
```

```
  border-radius: 50%; width: 22px; height: 22px;
  cursor: pointer; font-size: 0.75rem;
  display: flex; align-items: center; justify-content: center;
}

/* ===== BUTTONS ===== */
.cp-btn {
  padding: 10px 20px; border: none; border-radius: 8px;
  font-size: 0.95rem; font-weight: bold; cursor: pointer;
  display: inline-flex; align-items: center; gap: 8px;
  transition: all 0.2s; font-family: inherit;
}

.cp-btn-primary { background: #3498db; color: white; }
.cp-btn-primary:hover { background: #2980b9; }
.cp-btn-secondary { background: #ecf0f1; color: #2c3e50; }
.cp-btn-secondary:hover { background: #d5d8dc; }
.cp-btn-submit {
  background: linear-gradient(135deg, #27ae60, #2ecc71);
  color: white; font-size: 1rem; padding: 12px 24px;
}
.cp-btn-submit:hover { background: linear-gradient(135deg, #229954, #27ae60); }
.cp-btn:disabled { opacity: 0.6; cursor: not-allowed; }

.cp-form-actions { display: flex; gap: 10px; justify-content: flex-end; margin-top: 20px; }
```

```
/* ===== REVIEW ===== */
.cp-review { margin-bottom: 20px; }

.cp-review-table { width: 100%; border-collapse: collapse; }
.cp-review-table tr { border-bottom: 1px solid #ecf0f1; }
.cp-review-key {
  padding: 8px 12px; color: #7f8c8d; font-weight: bold;
  font-size: 0.85rem; width: 140px; background: #f8f9fa;
}
.cp-review-val { padding: 8px 12px; color: #2c3e50; font-size: 0.9rem; }

/* ===== SUCCESS ===== */
.cp-success-card { text-align: center; padding: 40px; }
.cp-success-icon { font-size: 4rem; color: #27ae60; margin-bottom: 16px; }
.cp-success-title { font-size: 1.5rem; color: #27ae60; margin-bottom: 8px; }
.cp-success-sub { color: #7f8c8d; margin-bottom: 8px; }
.cp-tracking-code {
  font-size: 2rem; font-weight: bold; color: #3498db;
  background: #ebf5fb; padding: 12px 24px; border-radius: 10px;
  display: inline-block; margin: 12px 0; letter-spacing: 2px;
  font-family: 'Courier New', monospace;
}
.cp-success-hint { color: #7f8c8d; font-size: 0.9rem; margin-bottom: 20px; }
.cp-email-note {
```

```
  background: #ebf5fb; color: #2980b9; padding: 8px 16px;
  border-radius: 8px; margin-bottom: 20px; font-size: 0.85rem;
  display: inline-flex; align-items: center; gap: 6px;
}
.cp-success-card .cp-btn { margin: 6px; }

/* ===== TRACKING ===== */
.cp-track-row { display: flex; gap: 10px; }
.cp-track-input { flex: 1; text-transform: uppercase; letter-spacing: 1px; }

.cp-track-result {
  background: #f8f9fa; border-radius: 10px;
  padding: 20px; margin-top: 16px;
  border: 1px solid #ecf0f1;
}

.cp-track-header {
  display: flex; justify-content: space-between; align-items: center;
  margin-bottom: 16px;
}

.cp-track-code {
  font-size: 1.1rem; font-weight: bold; color: #3498db;
  font-family: 'Courier New', monospace; letter-spacing: 1px;
}
```

```
.cp-status-badge {
  padding: 4px 12px; border-radius: 12px;
  font-weight: bold; font-size: 0.85rem;
}

/* ===== TIMELINE ===== */
.cp-status-timeline {
  display: flex; align-items: center;
  margin-top: 20px; padding-top: 16px;
  border-top: 1px solid #ecf0f1;
}

.cp-timeline-step {
  display: flex; flex-direction: column; align-items: center; gap: 4px;
}

.cp-timeline-dot {
  width: 16px; height: 16px; border-radius: 50%;
  background: #ecf0f1; border: 2px solid #bdc3c7;
}

.cp-timeline-step.done .cp-timeline-dot {
  background: #27ae60; border-color: #27ae60;
}
```

```
.cp-timeline-label { font-size: 0.75rem; color: #7f8c8d; }
.cp-timeline-step.done .cp-timeline-label { color: #27ae60; font-weight: bold; }

.cp-timeline-line { flex: 1; height: 2px; background: #ecf0f1; margin-bottom: 18px; }

/* ===== ALERTS ===== */
.cp-alert {
  padding: 10px 14px; border-radius: 8px;
  margin-bottom: 12px; font-size: 0.87rem;
  display: flex; align-items: center; gap: 8px;
}
.cp-alert-error   { background: #fdecea; color: #e74c3c; border: 1px solid #f5c6cb; }
.cp-alert-success { background: #d5f5e3; color: #27ae60; border: 1px solid #a9dfbf; }

.cp-loading {
  text-align: center; padding: 20px; color: #7f8c8d;
  display: flex; align-items: center; justify-content: center; gap: 8px;
}

/* ===== FOOTER ===== */
.cp-footer {
  text-align: center; padding: 20px;
  color: #7f8c8d; font-size: 0.85rem; margin-top: 20px;
}
```

```
.cp-footer-link {
  color: #3498db; text-decoration: none;
  display: inline-flex; align-items: center; gap: 5px;
}

.cp-footer-link:hover { text-decoration: underline; }

/* ===== RESPONSIVE ===== */
@media (max-width: 600px) {
  .cp-header-inner { flex-direction: column; gap: 10px; padding: 12px 0; }
  .cp-type-grid { grid-template-columns: 1fr 1fr; }
  .cp-form-actions { flex-direction: column; }
  .cp-btn { width: 100%; justify-content: center; }
  .cp-track-row { flex-direction: column; }
}

/*
================================================================================
   ΚΛΑΣΕΙΣ ΠΟΥ ΑΝΤΙΚΑΘΙΣΤΟΥΝ INLINE STYLES ΑΠΟ ΤΑ JS TEMPLATE STRINGS
   ================================================================================ */

/* Marker κάδου στον χάρτη (loadBins) */
```

```
.cp-bin-marker {
  width: 26px;
  height: 26px;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 12px;
  color: white;
}
.cp-bin-marker.critical { background: #e74c3c; }
.cp-bin-marker.warning  { background: #f39c12; }
.cp-bin-marker.ok       { background: #27ae60; }

/* Popup κάδου */
.cp-bin-popup {
  font-family: 'Consolas', monospace;
  min-width: 160px;
}
.cp-bin-popup-btn {
  margin-top: 6px;
  padding: 5px 10px;
  background: #3498db;
```

```
  color: white;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  width: 100%;
  font-family: 'Consolas', monospace;
  font-size: 0.85rem;
}

/* Marker καρφίτσας (setPinLocation) */
.cp-pin-marker {
  background: #8e44ad;
  color: white;
  padding: 6px;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
  font-size: 16px;
  display: flex;
  align-items: center;
  justify-content: center;
}

/* Status badge στην παρακολούθηση (tint ανά κατάσταση) */
.cp-status-badge.open        { background: #fdecea; color: #e74c3c; }
```

```
.cp-status-badge.in_progress { background: #fef9e7; color: #f39c12; }
.cp-status-badge.resolved    { background: #d5f5e3; color: #27ae60; }
.cp-status-badge.closed      { background: #f0f3f4; color: #7f8c8d; }
```

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.