Σχολή Θετικών Επιστημών και Τεχνολογίας

Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα (ΠΛΣ)

Διπλωματική Εργασία

Διαχείριση Ραντεβού για Ελεύθερους Επαγγελματίες

Ιωάννης Αυγέρος

Επιβλέπων καθηγητής: Παξιμάδης Κωνσταντίνος

Πάτρα, Μάιος, 2026

Διαχείριση Ραντεβού για Ελεύθερους Επαγγελματίες

Ιωάννης Αυγέρος

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:	Συν-Επιβλέπων Καθηγητής:
Παξιμάδης Κωνσταντίνος	Γκαράνη Γεωργία
Καθηγητής	Καθηγήτρια, τμήμα Ψηφιακών Συστημάτων, Πανεπιστημίου Θεσσαλίας

Πάτρα, Μάιος, 2026

«Ευχαριστίες ή Αφιέρωση»

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο των σπουδών μου στο **Ελληνικό Ανοικτό Πανεπιστήμιο**, στο πρόγραμμα **Μεταπτυχιακής Εξειδίκευσης στα Πληροφοριακά Συστήματα (ΠΛΣ)**.

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα **κ. Παξιμάδη Κωνσταντίνο** για την πολύτιμη καθοδήγηση, την εποικοδομητική ανατροφοδότηση και την εμπιστοσύνη που έδειξε καθ' όλη τη διάρκεια της εργασίας.

Ευχαριστώ επίσης τους διδάσκοντες του προγράμματος για τις γνώσεις και τις δεξιότητες που μου παρείχαν, οι οποίες αποτέλεσαν θεμέλιο για την ολοκλήρωση του έργου. Ιδιαίτερη μνεία οφείλω στην οικογένειά μου και στους φίλους και συναδέλφους μου για τη συνεχή στήριξη, την υπομονή και την ενθάρρυνση.

Η εργασία αυτή πραγματεύεται τον **σχεδιασμό και την υλοποίηση μίας διαδικτυακής εφαρμογής διαχείρισης ραντεβού** για ελεύθερους επαγγελματίες. Επιδίωξή μου ήταν να παραδώσω μια **πρακτική, επεκτάσιμη και εύχρηστη λύση**, αξιοποιώντας **σύγχρονες τεχνολογίες ανάπτυξης λογισμικού**.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως στόχο την ανάπτυξη μιας **διαδικτυακής εφαρμογής διαχείρισης ραντεβού**, απευθυνόμενης σε **ελεύθερους επαγγελματίες** όπως γιατρούς, γυμναστές, αισθητικούς και κομμωτές. Η ανάγκη για τέτοιου είδους λύσεις είναι έντονη, καθώς πολλοί αυτοαπασχολούμενοι και μικρές επιχειρήσεις εξακολουθούν να στηρίζονται σε χειρόγραφες μεθόδους ή γενικά εργαλεία (π.χ. Google Calendar), τα οποία δεν καλύπτουν επαρκώς τις ιδιαιτερότητες του επαγγελματικού τους κύκλου εργασιών. Η προτεινόμενη εφαρμογή **προσφέρει φιλικό περιβάλλον** με ευανάγνωστο ημερολόγιο και απλή ροή εργασίας για διαχείριση ραντεβού. Υλοποιείται **σύστημα πιστοποίησης** (εγγραφή/είσοδος), ώστε κάθε επαγγελματίας να διαθέτει προσωπικό προφίλ και να διαχειρίζεται τα δικά του δεδομένα με ασφάλεια. Οι βασικές λειτουργίες περιλαμβάνουν,**προβολή**, **καταχώρηση**, **επεξεργασία**, **ακύρωση ραντεβού** και διαχείριση πελατολογίου.Τεχνολογικά, το **frontend** υλοποιείται με **React.js** (SPA), το **backend** σε **Python (Flask)** με **RESTful API** και **ORM**, ενώ ως βάση δεδομένων χρησιμοποιείται MySQL. Αξιοποιούνται σύγχρονες πρακτικές ανάπτυξης (**CRUD**, διαχείριση κατάστασης, session/authentication) και προβλέπεται containerization (**Docker**) για ευκολότερη ανάπτυξη. Η συμβολή της εργασίας έγκειται στη δημιουργία μιας πρακτικής, επεκτάσιμης λύσης για ελεύθερους επαγγελματίες και στην τεκμηρίωση μιας ολοκληρωμένης μεθοδολογίας full-stack ανάπτυξης **(React + Flask + MySQL),** ενισχύοντας δεξιότητες σε αρχιτεκτονική συστημάτων, **RESTful APIs, CRUD operations** και **authentication**.

Λέξεις – Κλειδιά

Διαχείριση ραντεβού, ελεύθεροι επαγγελματίες, React.js, Python (Flask), REST API, MySQL, CRUD, Docker, full-stack ανάπτυξη.

Appointment Management for Freelancers

Ioannis Avgeros

Abstract

The present master's thesis aims at developing a web-based appointment management application, targeting freelancers such as doctors, fitness trainers, aestheticians, and hairdressers. The need for such solutions is pressing, as many self-employed professionals and small businesses continue to rely on manual methods or generic tools (e.g., Google Calendar), which often do not adequately address the specificities of their professional workflows.The proposed application offers a user-friendly environment with a clear calendar and a simple workflow for appointment management. An authentication system (registration/login) is implemented, so that each professional has a personal profile and manages their own data securely. Core features include: viewing, creating, editing, canceling appointments, and client management.Technologically, the frontend is implemented with React.js (SPA), the backend in Python (Flask) with RESTful API and ORM, while MySQL is used as the database system. Modern development practices are employed (CRUD, state management, session/authentication) and containerization (Docker) is planned for easier development.The contribution of this work lies in creating a practical, extensible solution for freelancers and in documenting a complete full-stack development methodology (React + Flask + MySQL), enhancing skills in system architecture, RESTful APIs, CRUD operations, and authentication.

Keywords

Appointment management, freelancers, React.js, Python (Flask), REST API, MySQL, CRUD, Docker, full-stack development.

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων

Συντομογραφίες & Ακρωνύμια

ΕΑΠ: Ελληνικό Ανοικτό Πανεπιστήμιο

ΠΛΣ: Πληροφοριακά Συστήματα

API: Application Programming Interface (Διασύνδεση Προγραμματισμού Εφαρμογών)

CRUD: Create, Read, Update, Delete (Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή)

CORS: Cross-Origin Resource Sharing (Κοινή Χρήση Πόρων Διαφορετικών Προελεύσεων)

DB: Database (Βάση Δεδομένων)

GDPR: General Data Protection Regulation (Γενικός Κανονισμός Προστασίας Δεδομένων)

HTTP: HyperText Transfer Protocol (Πρωτόκολλο Μεταφοράς Υπερκειμένου)

JSON: JavaScript Object Notation (Σημειογραφία Αντικειμένων JavaScript)

JWT: JSON Web Token (Διακριτικό Web JSON)

KPI: Key Performance Indicators (Βασικοί Δείκτες Απόδοσης)

MVC: Model-View-Controller (Μοντέλο-Προβολή-Ελεγκτής)

ORM: Object-Relational Mapping (Αντιστοίχιση Αντικειμένων-Σχέσεων)

RBAC: Role-Based Access Control (Έλεγχος Πρόσβασης Βάσει Ρόλων)

REST: Representational State Transfer (Μεταφορά Κατάστασης Αναπαράστασης)

SPA: Single-Page Application (Εφαρμογή Μονής Σελίδας)

SQL: Structured Query Language (Γλώσσα Δομημένων Ερωτημάτων)

UI: User Interface (Διεπαφή Χρήστη)

UX: User Experience (Εμπειρία Χρήστη)

1. Εισαγωγή

1.1 Σκοπός και Κίνητρο

Στόχος της παρούσας διπλωματικής είναι η ανάπτυξη μίας **διαδικτυακής εφαρμογής διαχείρισης ραντεβού**, που απευθύνεται σε **ελεύθερους επαγγελματίες**, όπως γιατρούς, γυμναστές, αισθητικούς και κομμωτές, οι οποίοι βασίζονται σε συστηματικά ραντεβού με πελάτες. Παρά την ευρεία χρήση γενικών εργαλείων (π.χ. ψηφιακά ημερολόγια), παραμένει **κενό για εξειδικευμένες λύσεις** που να προσφέρουν απλή, ασφαλή και αποδοτική ροή εργασίας, ειδικά για επαγγελματικά ραντεβού.

1.2 Ανάγκη και Πρόβλημα

Πολλοί αυτοαπασχολούμενοι ή μικρές επιχειρήσεις καταγράφουν ραντεβού **χειρόγραφα ή με μη εξειδικευμένα εργαλεία**, γεγονός που οδηγεί σε σφάλματα (π.χ. διπλοκρατήσεις), δυσκολία στον προγραμματισμό και ελλιπή εποπτεία του πελατολογίου. Η ανάγκη είναι μία **ελαφριά, εύχρηστη εφαρμογή** με σαφή ροή καταχώρησης, τροποποίησης και ακύρωσης ραντεβού και βασική **διαχείριση πελατών**, προσβάσιμη από οποιαδήποτε συσκευή.

1.3 Στόχοι και Παραδοτέα της Εργασίας

Η εργασία παραδίδει:

- **Διακριτή αρχιτεκτονική** frontend–backend (React SPA + Flask REST API).
- **Ασφαλή αυθεντικοποίηση** με JSON Web Tokens (JWT) και ελέγχους πρόσβασης στα endpoints.
- **Κύριες ροές ραντεβού**: δημιουργία, ενημέρωση, ακύρωση, αποφυγή επικαλύψεων (έλεγχος σύγκρουσης ωρών).
- **Διαχείριση πελατών και υπηρεσιών**, καθώς και βασική υποστήριξη επαγγελματικών ωραρίων.
- **Καθαρή επιχειρησιακή λογική**, με σωστούς μετασχηματισμούς ημερομηνιών και ωρών για αξιόπιστο προγραμματισμό.

1.4 Τεχνολογικό Υπόβαθρο

- **Frontend**: React (single-page application), μοντέρνα UI στοιχεία, dialogs για κρατήσεις, dashboards για επαγγελματίες και πελάτες.
- **Backend**: Flask με **RESTful endpoints**, JWT-based authentication και **SQLAlchemy ORM** για πρόσβαση σε σχεσιακή βάση δεδομένων.
- **Κύρια μοντέλα**: Professionals, Clients, Services, Appointments, Schedules (με σχέσεις και validation).
- **Ποιότητα και Ασφάλεια**: έλεγχοι εισόδου, χειρισμός λαθών, βασικές πολιτικές **CORS** και αποτροπή επικαλύψεων ραντεβού.

1.5 Δομή του Κειμένου

Το κείμενο οργανώνεται ως εξής:

- Στο **Κεφάλαιο 2** παρουσιάζεται η σχετική εργογραφία και το θεωρητικό πλαίσιο.
- Στο **Κεφάλαιο 3** αναλύονται οι απαιτήσεις και προδιαγραφές με χρήστες, ρόλους και σενάρια χρήσης.
- Στο **Κεφάλαιο 4** περιγράφεται η αρχιτεκτονική (ροές δεδομένων, API, μοντέλο δεδομένων).
- Στο **Κεφάλαιο 5** αναπτύσσεται η υλοποίηση (frontend, backend, βάση δεδομένων).
- Στο **Κεφάλαιο 6** παρουσιάζονται η αξιολόγηση και τα αποτελέσματα.
- Τέλος, στο **Κεφάλαιο 7** συνοψίζονται τα συμπεράσματα και προτείνονται μελλοντικές επεκτάσεις.

2. Σχετική Εργογραφία / Θεωρητικό Πλαίσιο

2.1 Αρχιτεκτονικές Web Εφαρμογών (SPA και REST)

Οι σύγχρονες web εφαρμογές υιοθετούν συχνά την αρχιτεκτονική **Single-Page Application (SPA)**, όπου το frontend φορτώνεται μία φορά και στη συνέχεια αλληλεπιδρά με το backend μέσω **RESTful APIs**. Το μοντέλο αυτό μειώνει τους χρόνους απόκρισης, προσφέρει εμπειρία χρήσης παρόμοια με native εφαρμογές και επιτρέπει **σαφή διαχωρισμό ρόλων** μεταξύ πελάτη (client) και διακομιστή (server). Η επικοινωνία πραγματοποιείται συνήθως μέσω **JSON**, με καθαρά ορισμένα **endpoints CRUD** (Create, Read, Update, Delete).

Στην παρούσα εργασία, το frontend βασίζεται στο **React (SPA)**, ενώ το backend υλοποιείται με **Flask** και εκθέτει **REST endpoints** για αυθεντικοποίηση, διαχείριση χρηστών, υπηρεσιών, ωραρίων και ραντεβού. Η προσέγγιση αυτή ενισχύει την **επεκτασιμότητα**, τη **συντηρησιμότητα** και τον **έλεγχο πρόσβασης**, ενώ διευκολύνει τη μελλοντική ενσωμάτωση πρόσθετων λειτουργιών όπως **online πληρωμές** ή **ειδοποιήσεις**.

2.2 Frontend Τεχνολογίες και Βέλτιστες Πρακτικές (React)

Το **React** προσφέρει **component-based σχεδίαση**, επαναχρησιμοποίηση κώδικα και προβλέψιμη **διαχείριση κατάστασης**. Σε εφαρμογές διαχείρισης ραντεβού, κρίσιμα στοιχεία του UI είναι τα **ημερολόγια**, τα **dialogs/φόρμες κράτησης** και τα **dashboards**. Η χρήση βιβλιοθηκών διεπαφής (π.χ. **Material UI**) επιταχύνει την ανάπτυξη, εξασφαλίζοντας **προσβασιμότητα** και **συνέπεια οπτικής ταυτότητας**.

Επιπλέον, ο διαχωρισμός σε επιμέρους σελίδες (π.χ. **Login**, **Dashboard Επαγγελματία/Πελάτη**, **Services**, **Clients**, **Schedule**) και η εφαρμογή **προστατευμένων διαδρομών (guarded routes)** με έλεγχο **token** συμβάλλουν στη βελτιστοποίηση της εμπειρίας χρήστη και στην ενίσχυση της ασφάλειας.

2.3 Backend Μικροπλαίσια και RESTful Σχεδιασμός (Flask)

Το **Flask**, ως **μικροπλαίσιο ανάπτυξης web εφαρμογών**, προσφέρει ευελιξία στην οργάνωση των **routes**, **services** και **models**. Η τήρηση των **REST αρχών** — όπως καθαρά resources, κατάλληλες HTTP μέθοδοι (GET, POST, PUT, DELETE) και κωδικοί κατάστασης — συμβάλλει στην **προβλεψιμότητα και επεκτασιμότητα** του API.
Η υλοποίηση της παρούσας εργασίας περιλαμβάνει endpoints για **εγγραφή/είσοδο χρηστών**, **δημιουργία, ενημέρωση και ακύρωση ραντεβού**, καθώς και **ανάγνωση/τροποποίηση πελατών, υπηρεσιών και ωραρίων επαγγελματιών**. Ιδιαίτερη μέριμνα δίνεται στη **διαχείριση ημερομηνιών/ωρών** και στον **έλεγχο συγκρούσεων ραντεβού**, ώστε να εξασφαλίζεται η **ορθότητα του προγραμματισμού**.

2.4 Αυθεντικοποίηση και Εξουσιοδότηση (JWT)

Η χρήση **JSON Web Tokens (JWT)** αποτελεί καθιερωμένη πρακτική για **stateless αυθεντικοποίηση** σε SPAs. Ο πελάτης λαμβάνει **token** μετά από επιτυχή είσοδο και το αποστέλλει σε κάθε αίτημα μέσω του **Authorization header (Bearer)**. Στο backend πραγματοποιείται **επαλήθευση υπογραφής και χρόνου λήξης**, καθώς και **αντιστοίχιση χρήστη**, πριν επιτραπεί η πρόσβαση σε προστατευμένα endpoints.
Στο πλαίσιο της παρούσας εργασίας, ο μηχανισμός **JWT** διασφαλίζει ότι τόσο οι **επαγγελματίες** όσο και οι **πελάτες** έχουν πρόσβαση μόνο στα δεδομένα που τους αφορούν, διατηρώντας έτσι την **ακεραιότητα και εμπιστευτικότητα** των πληροφοριών.

2.5 ORM και Μοντέλο Δεδομένων

Η χρήση μηχανισμών Object-Relational Mapping (ORM), όπως το **SQLAlchemy**, απλοποιεί σημαντικά την πρόσβαση και αλληλεπίδραση με σχεσιακές βάσεις δεδομένων, μετατρέποντας πίνακες σε κλάσεις και εγγραφές σε αντικείμενα.
Στο παρόν σύστημα, οι βασικές οντότητες περιλαμβάνουν τους **Professionals**, **Clients**, **Services**, **Appointments** και **Schedules (ωράρια)**. Οι μεταξύ τους σχέσεις (π.χ. ένα-προς-πολλά μεταξύ επαγγελματία και ραντεβού ή υπηρεσιών) ορίζονται ρητά, ενώ εφαρμόζονται περιορισμοί όπως **μοναδικότητα email** και **έγκυρες χρονικές περίοδοι**, προκειμένου να εξασφαλιστεί η **ακεραιότητα των δεδομένων**.

Η χρήση ORM μειώνει την ανάγκη για γραφή SQL εντολών χαμηλού επιπέδου, επιταχύνοντας την ανάπτυξη και ενισχύοντας τη συντηρησιμότητα του κώδικα.

2.6 Συστήματα Προγραμματισμού Ραντεβού (Scheduling)

Η διεθνής βιβλιογραφία που αφορά τα **συστήματα προγραμματισμού ραντεβού** (scheduling systems) εστιάζει κυρίως στην **αποφυγή επικαλύψεων**, στην **αποτελεσματική κατανομή πόρων** και στη **βελτιστοποίηση χρόνου**.
Σε μικρής κλίμακας επαγγελματικά περιβάλλοντα, οι ανάγκες επικεντρώνονται στην **άμεση ορατότητα των διαθεσίμων**, στη **γρήγορη καταχώρηση ή ακύρωση ραντεβού**, στην **αποτροπή διπλοκρατήσεων** και στη **διαχείριση ωραρίων**.
Η παρούσα λύση ενσωματώνει μηχανισμό ελέγχου συγκρούσεων (π.χ. “time slot already booked”) στο backend, ενώ στο frontend παρέχει φιλική απεικόνιση των διαθέσιμων επιλογών μέσω **διαδραστικών UI στοιχείων** όπως **διαλόγων κράτησης (dialogs)** και **χρονοεπιλογέων (pickers)**.

2.7 Χρηστικότητα και Εμπειρία Χρήστη (UX)

Η **χρηστική σχεδίαση** αποτελεί κρίσιμο παράγοντα για εφαρμογές που απευθύνονται σε μη τεχνικούς χρήστες. Η ύπαρξη **καθαρού πληροφοριακού σχεδιασμού**, **σαφών κουμπιών δράσης** (π.χ. “Νέο Ραντεβού”), **επικύρωσης δεδομένων σε πραγματικό χρόνο** (inline validation) και **οπτικών ανατροφοδοτήσεων** (επιβεβαιώσεις, προειδοποιήσεις, tooltips) βελτιώνει τη συνολική εμπειρία χρήσης.
Η **συνέπεια στη χρήση όρων και ετικετών** (π.χ. “Υπηρεσία”, “Πελάτης”, “Ώρα Έναρξης”) καθώς και η **προσαρμογή για κινητές συσκευές (responsive design)** ενισχύουν τη χρηστικότητα και αυξάνουν την αποδοχή της εφαρμογής από το κοινό-στόχο.

2.8 Ασφάλεια και Προστασία Δεδομένων

Η ασφάλεια αποτελεί θεμέλιο σε κάθε διαδικτυακή εφαρμογή που διαχειρίζεται προσωπικά δεδομένα.

Οι **βέλτιστες πρακτικές** περιλαμβάνουν:

- **Κρυπτογράφηση κωδικών πρόσβασης** με χρήση αλγορίθμων όπως το *bcrypt*.
- **Αποθήκευση tokens** μόνο όπου είναι απαραίτητο.
- **Έλεγχο CORS** για αποτροπή μη εξουσιοδοτημένων αιτήσεων.
- **Επικύρωση εισόδου/δεδομένων** στο backend και στο frontend.
- **Περιορισμό υπερεκτεθειμένων endpoints** και σωστό χειρισμό **ημερομηνιών/ζωνών ώρας**.

Στο πλαίσιο της παρούσας εργασίας, η **επικύρωση token**, οι **έλεγχοι εγκυρότητας πεδίων** και η **αυτόματη λήξη ή ακύρωση tokens** (logout, expiration) αποτελούν τον βασικό μηχανισμό προστασίας έναντι μη εξουσιοδοτημένης πρόσβασης και κακόβουλης χρήσης.

3. Απαιτήσεις και Προδιαγραφές

3.1 Εμπλεκόμενοι Χρήστες και Ρόλοι

Η εφαρμογή διακρίνει τρεις κύριους ρόλους:

- **Επαγγελματίας**:
  Διαχειρίζεται τις παρεχόμενες υπηρεσίες, το ωράριό του, τα ραντεβού και το πελατολόγιο.
  Έχει πρόσβαση σε ημερολόγιο προβολής ραντεβού καθώς και σε λειτουργίες δημιουργίας, τροποποίησης ή ακύρωσης κρατήσεων.

- **Πελάτης**:
  Δημιουργεί λογαριασμό ή συνδέεται, προβάλλει διαθέσιμους επαγγελματίες και υπηρεσίες, προγραμματίζει ραντεβού και μπορεί να τα ακυρώνει όταν χρειάζεται.

- **Σύστημα**:
  Επικυρώνει τις εισόδους χρηστών, αποτρέπει επικαλύψεις ραντεβού, διαχειρίζεται συνεδρίες μέσω **JWT tokens**, και οργανώνει τις βασικές ροές CRUD (Create, Read, Update, Delete).

3.2 Λειτουργικές Απαιτήσεις (Functional Requirements)

Η εφαρμογή πρέπει να υποστηρίζει τις εξής βασικές λειτουργίες:

1. **Εγγραφή και Είσοδος Χρηστών**
   - Δημιουργία λογαριασμού επαγγελματία ή πελάτη.
   - Επικύρωση στοιχείων email και ασφαλής διαχείριση κωδικών (hashing).
   - Έκδοση και επαλήθευση token για επαλήθευση ταυτότητας.
2. **Διαχείριση Υπηρεσιών**
   - Δημιουργία, επεξεργασία και διαγραφή υπηρεσιών (ονομασία, διάρκεια, τιμή).
   - Κάθε υπηρεσία συνδέεται με συγκεκριμένο επαγγελματία.

3. **Διαχείριση Πελατών**
    - Καταχώρηση, προβολή και ενημέρωση στοιχείων πελατών.
    - Σύνδεση ραντεβού με συγκεκριμένο πελάτη.
4. **Διαχείριση Ωραρίου (Schedules)**
    - Καθορισμός ωραρίου ανά ημέρα εβδομάδας από τον επαγγελματία.
    - Δυνατότητα τροποποίησης και προσωρινής απενεργοποίησης ωρών.
5. **Διαχείριση Ραντεβού (Appointments)**
    - Δημιουργία, ανάγνωση, ενημέρωση και ακύρωση ραντεβού.
    - Έλεγχος για συγκρούσεις (double booking) κατά την καταχώρηση ή επεξεργασία.
6. **Προστασία Endpoints & Ρόλων**
    - Πρόσβαση μόνο με έγκυρο token (JWT).
    - Διαχωρισμός ενεργειών και δεδομένων ανά ρόλο χρήστη (επαγγελματίας ή πελάτης).
7. **Πλοήγηση στο Frontend**
    - Υλοποίηση προστατευμένων διαδρομών (guarded routes).
    - Προβολή dashboard μόνο μετά από επιτυχή είσοδο.

3.3 Μη Λειτουργικές Απαιτήσεις (Non-Functional Requirements)

1. **Ασφάλεια**
    - Κρυπτογράφηση κωδικών (bcrypt).
    - Χρήση JWT με ημερομηνία λήξης.
    - Ενεργοποίηση πολιτικών CORS και επικύρωσης δεδομένων εισόδου.

2. **Απόδοση**
    - Γρήγορη απόκριση σε βασικές λειτουργίες (φόρτωση ημερολογίου, λίστες πελατών/υπηρεσιών).
    - Ελαχιστοποίηση περιττών αιτήσεων HTTP.
3. **Χρηστικότητα (UX)**
    - Καθαρό και ευανάγνωστο περιβάλλον διεπαφής.
    - Συνεπής ορολογία και φιλική ροή κράτησης.
    - Οπτική ανατροφοδότηση και ευκολία πλοήγησης.
4. **Επεκτασιμότητα και Συντηρησιμότητα**
    - Διαχωρισμός frontend–backend με καθαρή αρχιτεκτονική.
    - Επαναχρησιμοποίηση components, services και μοντέλων.
    - Ομοιογένεια δομής και τεκμηρίωση κώδικα.
5. **Συμβατότητα και Προσβασιμότητα**
    - Υποστήριξη σύγχρονων browsers (Chrome, Firefox, Edge).
    - Προσαρμοστικότητα σε κινητές και tablet συσκευές (responsive design).

3.4 Σενάρια Χρήσης (Use Cases)

Τα βασικά σενάρια χρήσης της εφαρμογής συνοψίζονται παρακάτω:

- **UC1 – Εγγραφή / Είσοδος Χρήστη**
  Ο χρήστης εισάγει το email και τον κωδικό πρόσβασής του.
  Το σύστημα επικυρώνει τα στοιχεία και, εφόσον είναι έγκυρα, εκδίδει **JWT token** το οποίο χρησιμοποιείται για τις επόμενες αιτήσεις.

- **UC2 – Δημιουργία Υπηρεσίας (Επαγγελματίας)**
  Ο επαγγελματίας ορίζει το όνομα και τη διάρκεια της υπηρεσίας.
  Το σύστημα αποθηκεύει τα δεδομένα και καθιστά τη νέα υπηρεσία διαθέσιμη για κρατήσεις από τους πελάτες.

- **UC3 – Ορισμός Ωραρίου (Επαγγελματίας)**
  Ο επαγγελματίας επιλέγει ημέρες και ώρες διαθεσιμότητας.
  Το σύστημα ελέγχει τη μοναδικότητα ωραρίου ανά ημέρα και αποθηκεύει το πρόγραμμα εργασίας.

- **UC4 – Καταχώρηση Ραντεβού**
  Ο χρήστης (επαγγελματίας ή πελάτης) επιλέγει υπηρεσία, ημερομηνία και ώρα, καθώς και τον αντίστοιχο πελάτη ή επαγγελματία.
  Το σύστημα ελέγχει για πιθανές επικαλύψεις ωραρίων και αποθηκεύει το νέο ραντεβού.

- **UC5 – Τροποποίηση ή Ακύρωση Ραντεβού**
  Το σύστημα επαληθεύει τα δικαιώματα του χρήστη (επαγγελματίας ή σχετικός πελάτης).
  Εφόσον ο έλεγχος ολοκληρωθεί επιτυχώς, το ραντεβού ενημερώνεται ή ακυρώνεται και οι αλλαγές αποτυπώνονται στο ημερολόγιο.

- **UC6 – Διαχείριση Πελατών (Επαγγελματίας)**
  Ο επαγγελματίας μπορεί να δημιουργεί, ενημερώνει ή διαγράφει πελάτες.
  Κάθε ραντεβού συνδέεται με συγκεκριμένο πελάτη, διασφαλίζοντας συνοχή και εύκολη αναζήτηση ιστορικού.

3.5 Κριτήρια Αποδοχής (Acceptance Criteria)

Η εφαρμογή θεωρείται επιτυχώς ολοκληρωμένη εφόσον πληρούνται τα παρακάτω:

- Το σύστημα **αποτρέπει με αξιοπιστία** επικαλύψεις ραντεβού και λανθασμένες κρατήσεις.
- Η είσοδος και η πρόσβαση σε **προστατευμένες σελίδες** απαιτεί έγκυρο JWT token.
- Οι ενέργειες **δημιουργίας, ενημέρωσης ή ακύρωσης ραντεβού** εμφανίζονται άμεσα στο ημερολόγιο.
- Οι κύριες ροές χρήσης ολοκληρώνονται **χωρίς σφάλματα επικύρωσης** σε τυπικά σενάρια.
- Το περιβάλλον διεπαφής χρήστη (UI) παραμένει **συνεπές**, με ενιαία ορολογία, μηνύματα και λειτουργικότητα σε κοινές αναλύσεις οθόνης.

3.6 Περιορισμοί και Υποθέσεις

Η παρούσα εργασία βασίζεται στις εξής υποθέσεις και περιορισμούς:

- Εφαρμόζεται **βασική πολιτική CORS** και **token-based session management**. Δεν υλοποιούνται σύνθετοι μηχανισμοί RBAC (Role-Based Access Control) πέραν των δύο ρόλων: *επαγγελματίας* και *πελάτης*.
- Η διαχείριση **ζωνών ώρας** είναι απλοποιημένη (τοπική αποθήκευση). Περαιτέρω εξειδίκευση ενδείκνυται για μελλοντική διεθνή χρήση της εφαρμογής.
- Δεν περιλαμβάνεται **υποστήριξη online πληρωμών**, η οποία προβλέπεται ως δυνατότητα επέκτασης σε μελλοντική φάση.
- Η **βάση δεδομένων** θεωρείται αξιόπιστη και προσβάσιμη. Ζητήματα υψηλής διαθεσιμότητας, replication ή distributed υποδομής δεν εξετάζονται στο παρόν έργο.

Αρχιτεκτονική Συστήματος

4.1 Επισκόπηση Αρχιτεκτονικής

Η αρχιτεκτονική της εφαρμογής βασίζεται στη λογική του **διαχωρισμού των επιπέδων παρουσίασης και επιχειρησιακής λογικής** (frontend–backend), εξασφαλίζοντας ευελιξία, επεκτασιμότητα και ευκολία συντήρησης.

- **Frontend (React Single Page Application - SPA)**
  Αποτελεί τη διεπαφή με τον τελικό χρήστη (επαγγελματία ή πελάτη). Περιλαμβάνει τη διαχείριση των ροών χρήσης, των φορμών εισαγωγής δεδομένων και των προστατευμένων διαδρομών (routes). Χρησιμοποιεί τεχνολογίες React, Axios και JWT για την επικοινωνία με το backend.

- **Backend (Flask REST API)**
  Υλοποιεί την επιχειρησιακή λογική, την αυθεντικοποίηση χρηστών, την επικύρωση εισόδων και την πρόσβαση στη βάση δεδομένων μέσω ORM (SQLAlchemy). Παρέχει RESTful endpoints για χρήστες, υπηρεσίες, ωράρια και ραντεβού, με χρήση JSON ως μορφή ανταλλαγής δεδομένων. Η ασφάλεια διασφαλίζεται με χρήση **JWT tokens** και **bcrypt** για κρυπτογράφηση κωδικών.

- **Βάση Δεδομένων (Σχεσιακή – MySQL)**
  Αποθηκεύει τις βασικές οντότητες του συστήματος:
  - Χρήστες (επαγγελματίες/πελάτες)
  - Υπηρεσίες
  - Ραντεβού
  - Ωράρια
    Οι σχέσεις μεταξύ των πινάκων ακολουθούν τη λογική **ένα-προς-πολλά**, ενώ εφαρμόζονται περιορισμοί μοναδικότητας και ακεραιότητας (π.χ. μη επικαλυπτόμενα ραντεβού ανά επαγγελματία και χρονική στιγμή).

Η επιλογή αρχιτεκτονικής **SPA + REST API** επιτρέπει την ανεξάρτητη εξέλιξη της διεπαφής χρήστη και των υπηρεσιών του backend, καθιστώντας το σύστημα πιο **ευέλικτο και δοκιμάσιμο**. Η χρήση **stateless tokens (JWT)** ενισχύει την επεκτασιμότητα και υποστηρίζει μελλοντική συνεργασία με τρίτες εφαρμογές.

4.2 Στόχοι Αρχιτεκτονικής

- **Απλότητα και σαφήνεια** στη ροή κράτησης (δημιουργία, ενημέρωση, ακύρωση).
- **Ασφάλεια** μέσω JWT, bcrypt και πολιτικών CORS.
- **Ακεραιότητα δεδομένων**, με έλεγχο επικαλύψεων και έγκυρων ημερομηνιών.
- **Επεκτασιμότητα**, επιτρέποντας την προσθήκη λειτουργιών όπως πληρωμές, ειδοποιήσεις και πολλαπλά ημερολόγια.
- **Συντηρησιμότητα**, με καθαρή διάκριση μεταξύ routes, services και models, διευκολύνοντας τη μελλοντική ανάπτυξη και τον έλεγχο.

4.3 Μοντέλο Δεδομένων και Σχέσεις

Το μοντέλο δεδομένων της εφαρμογής βασίζεται σε σχεσιακή προσέγγιση, με σαφή ορισμό οντοτήτων και σχέσεων μεταξύ τους. Οι βασικές οντότητες είναι οι εξής:

- **Professional (Επαγγελματίας)**
  Περιλαμβάνει στοιχεία ταυτότητας (όνομα, επώνυμο, email) και hash κωδικού πρόσβασης. Συνδέεται με τις οντότητες **Service**, **Appointment** και **ProfessionalSchedule**.
  Το email είναι μοναδικό ανά εγγραφή. Ένας επαγγελματίας μπορεί να διαθέτει πολλές υπηρεσίες, να εξυπηρετεί πολλούς πελάτες και να έχει πολλαπλά ραντεβού.
- **Client (Πελάτης)**
  Περιέχει βασικά στοιχεία (ονοματεπώνυμο, email ή τηλέφωνο, όπου εφαρμόζεται) και hash κωδικού. Ένας πελάτης μπορεί να έχει πολλά ραντεβού με διαφορετικούς επαγγελματίες.
  Το email μπορεί να είναι προαιρετικό, αλλά όταν παρέχεται πρέπει να είναι μοναδικό για κάθε πελάτη.
- **Service (Υπηρεσία)**
  Ανήκει σε έναν επαγγελματία και περιλαμβάνει πεδία όπως ονομασία και ενδεικτική διάρκεια (π.χ. 30 λεπτά). Χρησιμοποιείται για την κατηγοριοποίηση των διαθέσιμων ραντεβού και τη διευκόλυνση της επιλογής από τον πελάτη.

- **Appointment (Ραντεβού)**
  Συνδέει τον επαγγελματία, τον πελάτη και προαιρετικά την υπηρεσία.
  Περιλαμβάνει χρονικά πεδία (**start_time**, **end_time**) και κατάσταση (**status**: *scheduled*, *completed*, *cancelled*).
  Η ακεραιότητα διασφαλίζεται μέσω ελέγχων μη επικάλυψης για ραντεβού του ίδιου επαγγελματία.

- **ProfessionalSchedule (Ωράριο Επαγγελματία)**
  Ορίζει τη διαθεσιμότητα του επαγγελματία ανά ημέρα της εβδομάδας, με πεδία **day_of_week**, **start_time** και **end_time**. Εφαρμόζεται περιορισμός μοναδικότητας, ώστε κάθε επαγγελματίας να διαθέτει μόνο ένα ωράριο ανά ημέρα.

4.4 Κανόνες Ακεραιότητας και Επιχειρησιακοί Περιορισμοί

- **Μοναδικότητα**
  - Το email κάθε επαγγελματία είναι μοναδικό.
  - Κάθε επαγγελματίας μπορεί να έχει μόνο ένα ωράριο ανά ημέρα.

- **Καταστάσεις Ραντεβού**
  Ο κύκλος ζωής των ραντεβού περιλαμβάνει τις καταστάσεις *scheduled*, *completed* και *cancelled*, επιτρέποντας σαφή διαχείριση και ενημέρωση στο ημερολόγιο.

- **Αποφυγή Διπλοκρατήσεων**
  Πριν την αποθήκευση ραντεβού, πραγματοποιείται έλεγχος επικάλυψης μέσω της συνθήκης: (start_time < end_time_άλλου) AND (end_time > start_time_άλλου) για τον ίδιο επαγγελματία. Έτσι διασφαλίζεται ότι δεν υπάρχουν δύο ταυτόχρονα ενεργά ραντεβού.

- **Χρονικά Πεδία και Μορφοποίηση**
  Όλες οι ημερομηνίες και ώρες υποστηρίζουν το πρότυπο ISO 8601 και αποθηκεύονται σε “naive” μορφή (χωρίς ζώνη ώρας), κατάλληλη για τη στοχευόμενη βάση δεδομένων. Σε μελλοντικές εκδόσεις μπορεί να ενσωματωθεί πλήρης διαχείριση time zones για διεθνή χρήση.

4.5 Σχεδιασμός REST API και Ροές

Τα endpoints οργανώνονται σε ενότητες, κάθε μία με σαφή ευθύνη:

Αυθεντικοποίηση (Auth)

- **Εγγραφή (Registration)** επαγγελματία/πελάτη:
  Έλεγχος μορφής email, μοναδικότητας και επιβεβαίωσης κωδικού.
- **Είσοδος (Login)**:
  Εκδίδεται **JWT token**. Επιστρέφεται επίσης ο τύπος χρήστη (*professional* ή *client*) για ρύθμιση ροής στο frontend.
- **Προφίλ (/me)**:
  Απαιτεί έγκυρο token. Ο server ανακτά τα στοιχεία της οντότητας από το token.

Ραντεβού (Appointments)

- **Δημιουργία Ραντεβού**:
  - Από επαγγελματία: συνδέεται με τον ίδιο ως professional_id.
  - Από πελάτη: απαιτείται professional_id για επιλογή επαγγελματία.
- **Ανάγνωση & Ενημέρωση**: προβολή λίστας ή μεμονωμένου ραντεβού, αλλαγή ώρας ή κατάστασης.
- **Ακύρωση Ραντεβού**: μόνο από τον κάτοχο (client ή professional).
- **Έλεγχος Σύγκρουσης**: ο server απορρίπτει επικαλύψεις με **HTTP 409 Conflict**.

Υπηρεσίες (Services) & Πελάτες (Clients)

- CRUD λειτουργίες με έλεγχο πρόσβασης (μόνο ο επαγγελματίας-κάτοχος).
- Επιστροφή λιστών για επιλογή στο UI (π.χ. picker dialogs).

Ωράρια (Schedules)

- Ορισμός διαθεσιμότητας ανά ημέρα εβδομάδας.
- Περιορισμός μοναδικότητας: ένα ωράριο ανά ημέρα για κάθε επαγγελματία.

4.6 Αυθεντικοποίηση & Έλεγχος Πρόσβασης (JWT)

- Τα **προστατευμένα endpoints** απαιτούν header:
- Authorization: Bearer <token>
- Ο server **αποκωδικοποιεί το JWT**, ελέγχει υπογραφή και λήξη, και ταυτοποιεί τον χρήστη (professional ή client).
- Σε περίπτωση αποτυχίας επαλήθευσης:
    - **401 Unauthorized** → μη έγκυρο token
    - **403 Forbidden** → επαρκή δικαιώματα δεν υπάρχουν

Η χρήση JWT καθιστά την εφαρμογή **stateless**, επιτρέποντας ασφαλή διαχείριση συνεδριών και ανεξάρτητη κλιμάκωση frontend/backend.

4.7 Ασφάλεια και Ποιότητα Υπηρεσίας

- **Hash Κωδικών**: χρήση **bcrypt** για ασφαλή αποθήκευση.
- **Διαχείριση Token**: τα **JWT tokens** έχουν χρονικό ορίζοντα λήξης. Συνιστάται η αποθήκευση μόνο όπου απαιτείται (π.χ. localStorage) με σωστή πρακτική αποσύνδεσης (logout).
- **CORS & Validation**: βασικές πολιτικές CORS και συστηματική επικύρωση εισόδων στα endpoints. Επιστροφή κατάλληλων κωδικών: **400, 401, 403, 404, 409**.
- **Ανθεκτικότητα**: χειρισμός εξαιρέσεων/μη έγκυρων μορφών ημερομηνιών με ασφαλή προεπιλογή και καθαρά μηνύματα σφαλμάτων.

4.8 Διαχείριση Ημερομηνιών/Ωρών

- Αποδοχή **ISO 8601** μορφών (π.χ. με "Z") και μετασχηματισμός για αποθήκευση χωρίς ζώνη ώρας (**naive**) σύμφωνα με τον τύπο πεδίου της DB.
- Παρέχεται fallback αναλυτής για κοινές μορφές (YYYY-MM-DD HH:MM:SS), μειώνοντας σφάλματα εισαγωγής.
- Έλεγχος επικαλύψεων μέσω ανισοτήτων στα **start_time / end_time** για τον ίδιο επαγγελματία, αποτρέποντας διπλοκρατήσεις.

4.9 Frontend Ροές & UX

- **Προστατευμένες Διαδρομές (Protected Routes)**: οι σελίδες ημερολογίου, ρυθμίσεων, πελατών και υπηρεσιών απαιτούν έγκυρο token. Σε απουσία του, γίνεται ανακατεύθυνση στο login.
- **Διαλόγοι Κράτησης (Modals)**: δημιουργία ραντεβού με επιλογή ημερομηνίας/ώρας/υπηρεσίας. Άμεση ανάδραση για διαθέσιμα slots και μηνύματα σφάλματος.
- **Dashboards**: ξεχωριστές προβολές για επαγγελματίες και πελάτες, με λίστες ραντεβού, φίλτρα και ενέργειες (create/edit/cancel).
- **Συνέπεια UI**: σαφής ορολογία (**Υπηρεσία, Πελάτης, Ημερομηνία, Ώρα**), αναδυόμενα μηνύματα επιβεβαίωσης, καθαρά κουμπιά δράσης για βελτιωμένη χρηστικότητα.

5. Υλοποίηση

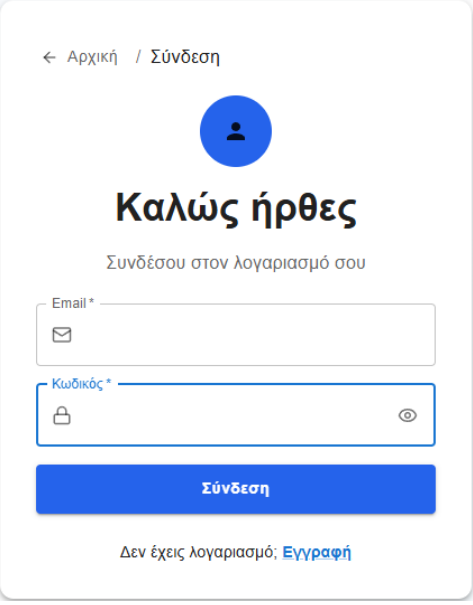
5.1 Επισκόπηση Υλοποίησης

Η υλοποίηση ακολουθεί σαφή διαχωρισμό καθηκόντων: το **frontend (React SPA)** αναλαμβάνει την παρουσίαση, την πλοήγηση και τις ροές αλληλεπίδρασης του χρήστη, ενώ το **backend (Flask REST API)** ενσωματώνει την επιχειρησιακή λογική, την **αυθεντικοποίηση (JWT)** και την πρόσβαση στη **σχεσιακή βάση δεδομένων** μέσω **SQLAlchemy ORM**. Οι δύο πλευρές επικοινωνούν με **JSON** μέσω προστατευμένων endpoints. Η εφαρμογή οργανώνεται σε διακριτές ενότητες/σελίδες (π.χ. **Login**, **Professional Dashboard**, **Client Dashboard**, **Services**, **Clients**, **Schedule**, **Settings**), με **προστατευμένες διαδρομές** ώστε το περιεχόμενο να καθίσταται προσβάσιμο μόνο μετά από επιτυχή είσοδο. Ιδιαίτερη έμφαση δίνεται στη ροή δημιουργίας/ακύρωσης ραντεβού και στον **έλεγχο σύγκρουσης χρονικών διαστημάτων**.

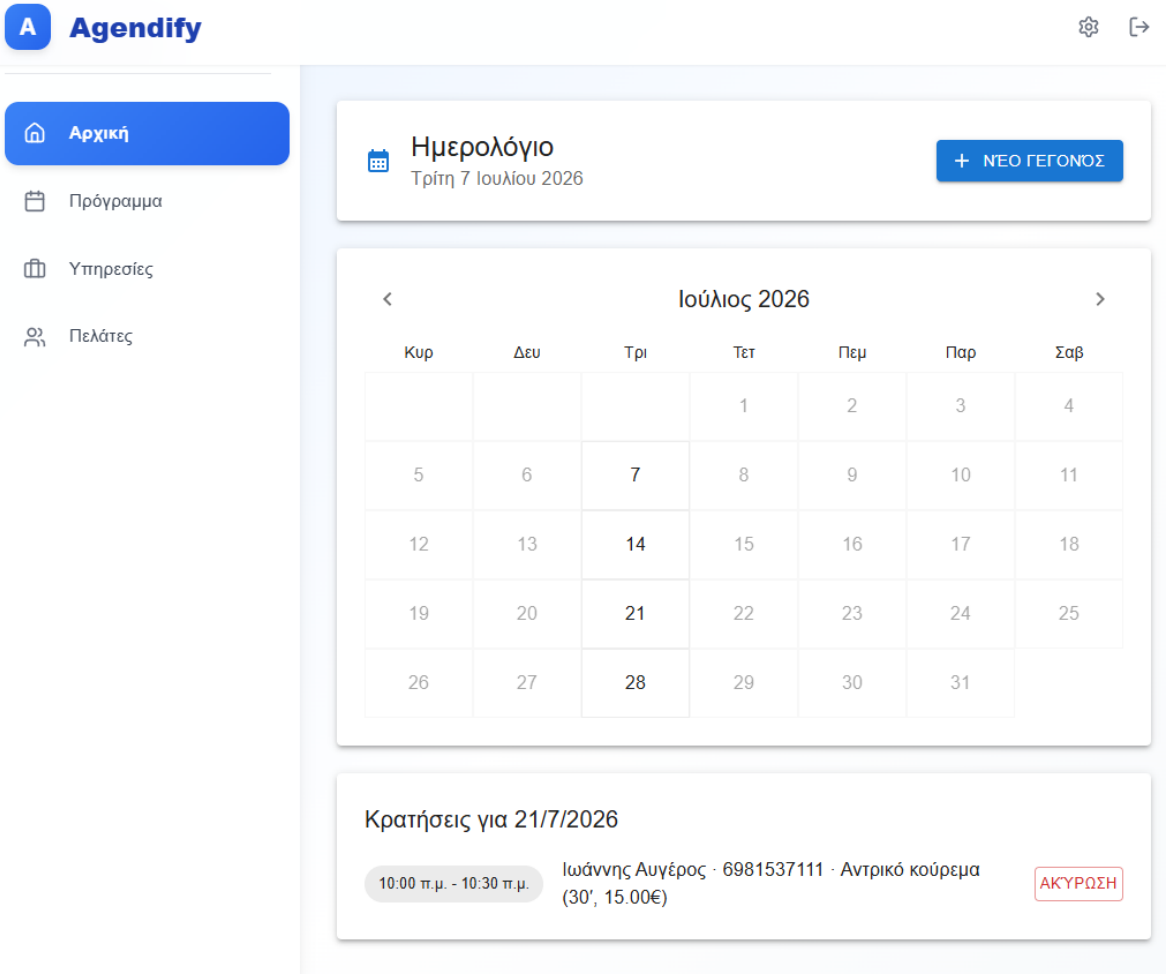
5.2 Frontend (React) — Δομή και Πλοήγηση

- **Routing & Προστασία**: Η πλοήγηση υλοποιείται με router· οι διαδρομές του **ημερολογίου**, των **ρυθμίσεων**, των **υπηρεσιών** και των **πελατών** προστατεύονται με έλεγχο ύπαρξης **JWT token** (ανακατεύθυνση σε login αν απουσιάζει).
- **Σελίδες/Components**:
    - **Επαγγελματίας**: **Professional Dashboard** (ημερολόγιο, λίστα/ενέργειες ραντεβού), **Services**, **Clients**, **Schedule**, **Settings**.
    - **Πελάτης**: **Client Dashboard** με αναζήτηση/φίλτρα επαγγελματιών/κατηγοριών και ροή αίτησης κράτησης.
    - Κοινά components κρατήσεων: **BookingModal** (επαγγελματίας), **ClientBookingModal** (πελάτης) με επιλογή **υπηρεσίας**, **ημερομηνίας** και **ωραρίου**.
- **UI Βιβλιοθήκες**: Χρήση **Material UI (MUI)** για τυποποιημένα στοιχεία, και **date pickers** με ελληνικό locale για ορθή εγχώρια εμπειρία.
- **Διαχείριση Κατάστασης & Κλήσεις API**: Οι κλήσεις προς τα endpoints συγκεντρώνονται σε **service layer** (π.χ. authService) με συνεπή χειρισμό headers (Authorization: Bearer <token>), επιτυχίας και σφαλμάτων.

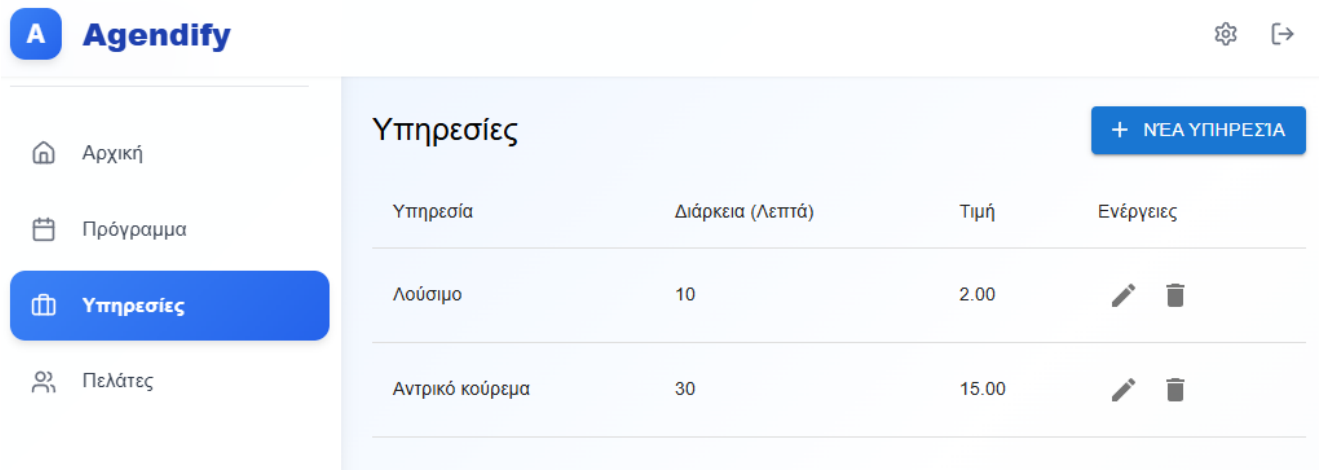
Εικόνα 5.2.1 Οθόνη Login (σύνδεση)



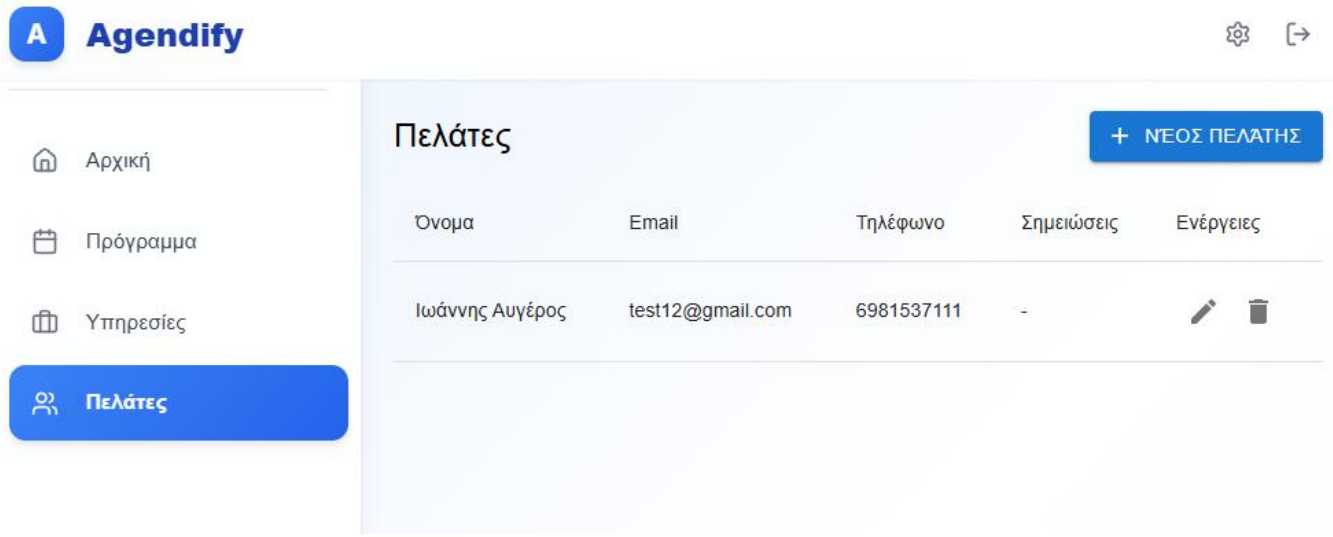
Εικόνα 5.2.2 Professional Dashboard (ημερολόγιο ραντεβού)



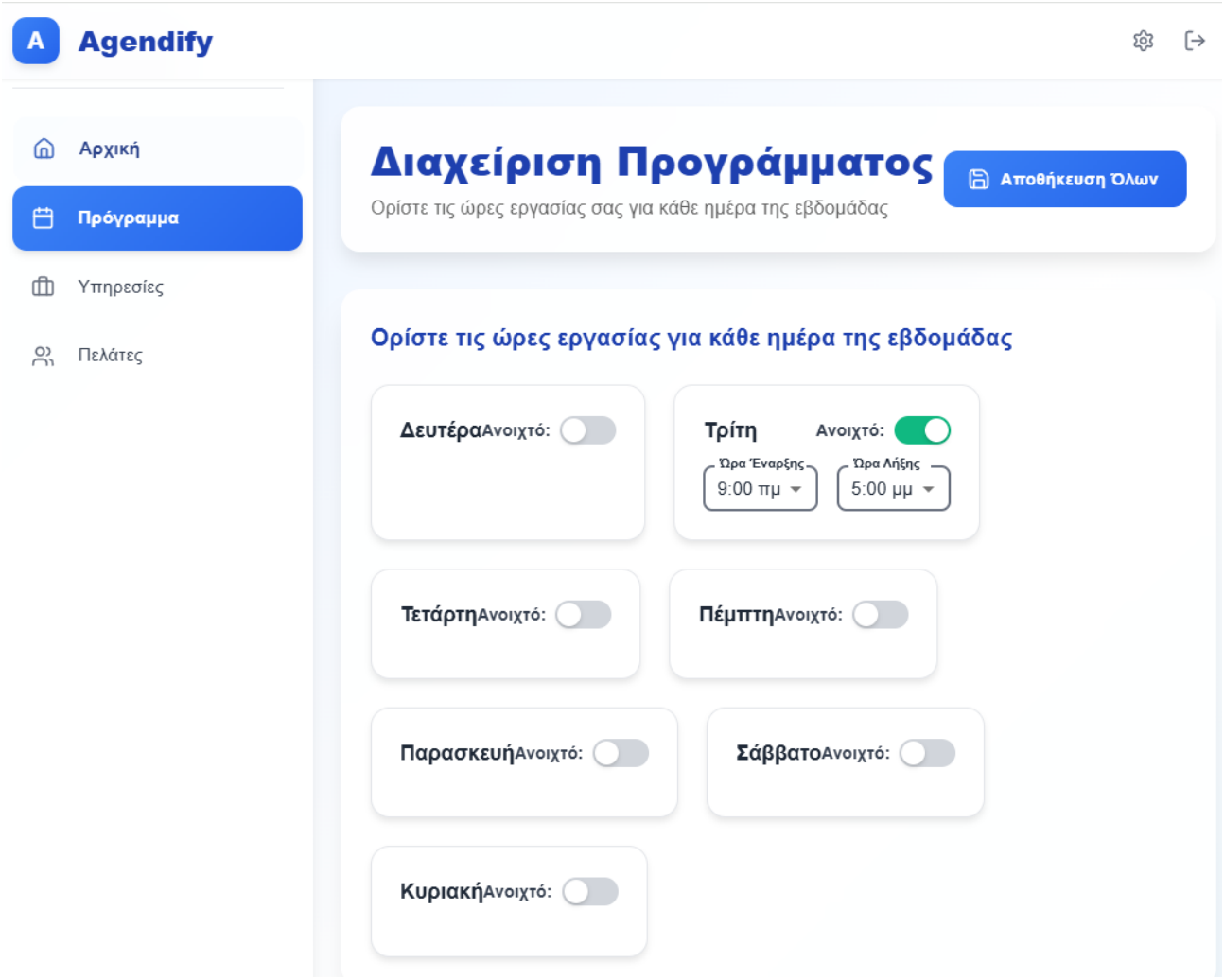
Εικόνα 5.2.3 Σελίδα Υπηρεσίες (Services)



Εικόνα 5.2.4 Σελίδα Πελάτες (Clients)



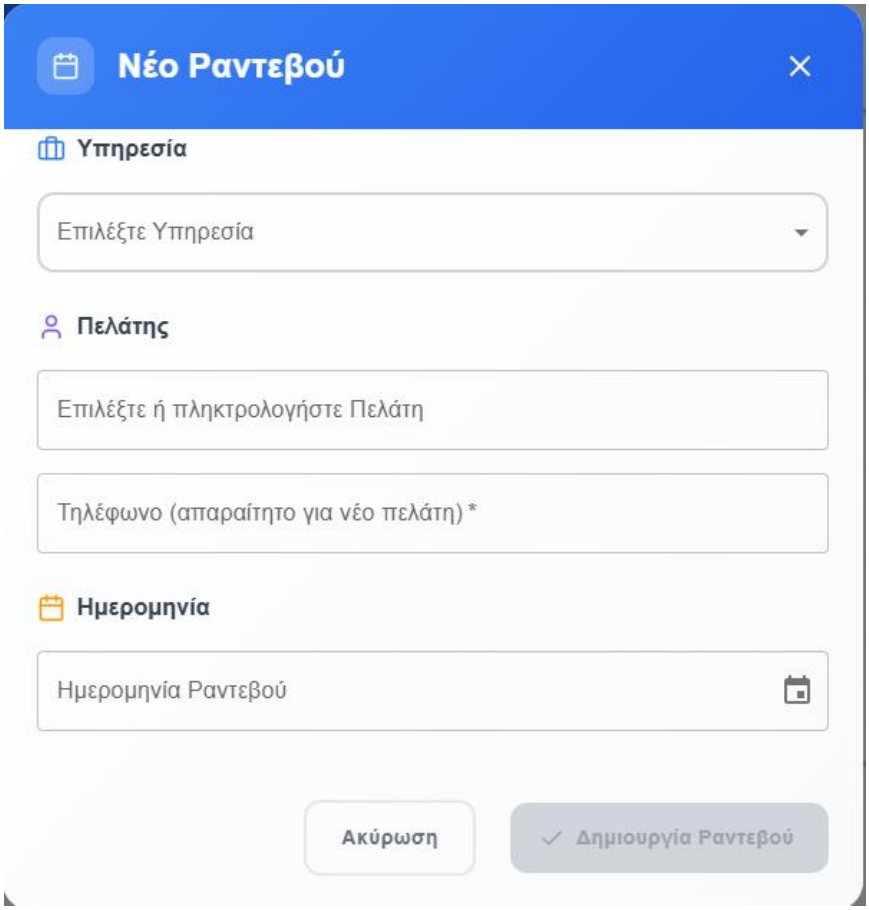
Εικόνα 5.2.5 Σελίδα Ωράριο (Schedule)



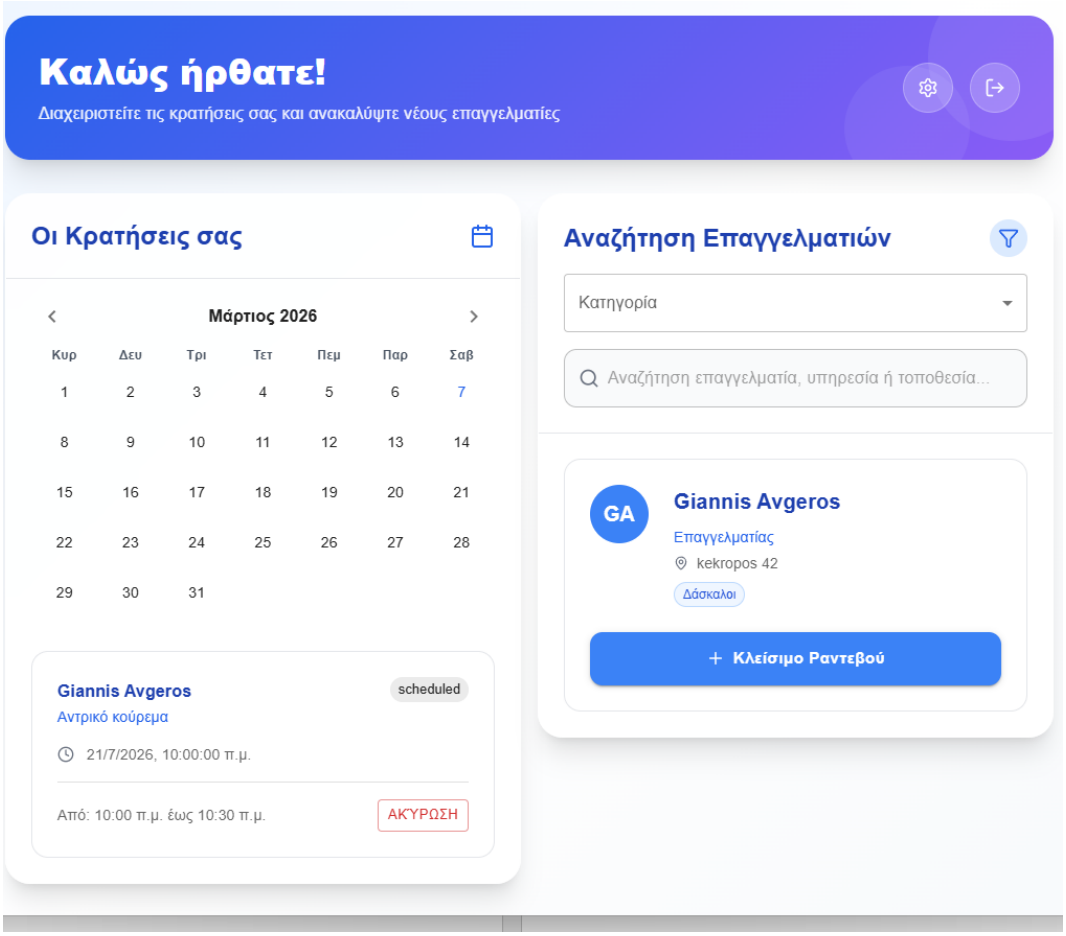
5.3 Frontend — Ροές Κράτησης

Η ροή κράτησης περιλαμβάνει: επιλογή **υπηρεσίας (service)**, επιλογή **ημερομηνίας** και **διαθέσιμης χρονικής ζώνης (slot)** βάσει ωραρίου/υφιστάμενων ραντεβού, **επιβεβαίωση** και αποστολή στο **backend**. Σε αποτυχημένη αποθήκευση λόγω σύγκρουσης, προβάλλεται **μήνυμα σφάλματος** και προτείνεται εναλλακτικό slot. Μετά από επιτυχή καταχώρηση, ανανεώνεται η **λίστα ραντεβού** και εμφανίζεται **μήνυμα επιβεβαίωσης**.

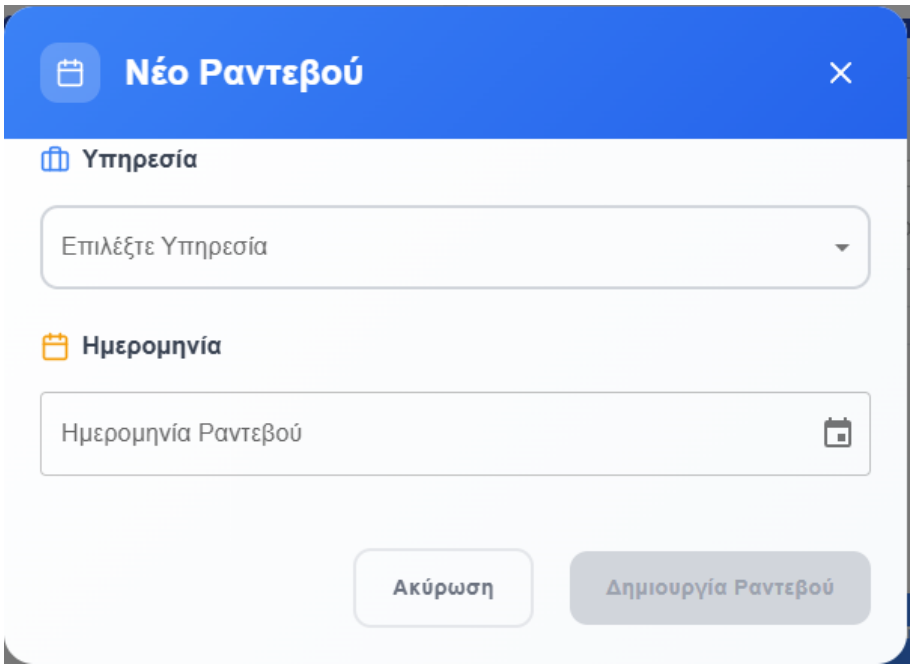
Εικόνα 5.3.1 Διάλογος κράτησης (BookingModal) – επαγγελματίας



Εικόνα 5.3.2 Client Dashboard (προβολή επαγγελματιών/κράτηση)



Εικόνα 5.3.3 Διάλογος κράτησης πελάτη (ClientBookingModal)



5.4 Backend (Flask) — Δομή και Υλοποίηση

- **Blueprints & Οργάνωση**: Το backend οργανώνεται σε blueprints ανά λειτουργικό κομμάτι: auth, appointments, services, clients, schedules. Αυτό επιτρέπει σαφή διαχωρισμό ευθυνών και ευκολία συντήρησης.
- **Endpoints**:
    - **Auth**: POST /register, POST /login, GET /me.
    - **Appointments**: GET /appointments, POST /appointments, PUT /appointments/<id>, DELETE /appointments/<id> με έλεγχο δικαιωμάτων και validation.
    - **Services & Clients**: CRUD endpoints με περιορισμένη πρόσβαση στον ιδιοκτήτη επαγγελματία.
    - **Schedules**: GET /schedules, POST /schedules, PUT /schedules/<id>, DELETE /schedules/<id>, με έλεγχο μοναδικότητας ανά ημέρα.
- **Αυθεντικοποίηση & Έλεγχος Πρόσβασης**: Χρήση **JWT** για stateless sessions. Προστατευμένα endpoints απαιτούν έγκυρο token (Authorization: Bearer <token>), με επιστροφή **401 Unauthorized** ή **403 Forbidden** για μη έγκυρη πρόσβαση.
- **Validation & Error Handling**: Όλες οι εισόδοι επικυρώνονται για τύπους/εύρος τιμών, ημερομηνίες/ώρες και μοναδικότητα. Σφάλματα επιστρέφονται με κατάλληλο HTTP status (400, 409, κλπ).

5.5 Βάση Δεδομένων και ORM (SQLAlchemy)

- **Βασικά Μοντέλα**: Professional, Client, Service, Appointment, ProfessionalSchedule.
- **Σχέσεις & Περιορισμοί**:
    - Professional → Services (1:N)
    - Professional → Appointments (1:N)
    - Professional → Schedules (1:N, μοναδικότητα ανά ημέρα)
    - Client → Appointments (1:N)
    - Appointment συνδέει Professional, Client και προαιρετικά Service.
- **Έλεγχος Διπλοκρατήσεων**: Πριν την αποθήκευση ενός ραντεβού, γίνεται έλεγχος αν κάποιο υπάρχον ραντεβού του επαγγελματία επικαλύπτεται χρονικά (start_time < end_time_άλλου AND end_time > start_time_άλλου).

5.6 Ροές Backend — Συνεργασία με Frontend

1. Ο χρήστης υποβάλλει αίτημα μέσω UI (π.χ. κράτηση ραντεβού).
2. Το React SPA στέλνει **HTTP POST** με token στο Flask endpoint.
3. Το backend:
    - Αποκωδικοποιεί το JWT, επαληθεύει τα δικαιώματα.
    - Ελέγχει επικαλύψεις και εγκυρότητα δεδομένων.
    - Αποθηκεύει στην DB ή επιστρέφει σφάλμα (409 Conflict).
4. Το frontend λαμβάνει απάντηση:
    - **Επιτυχία**: ενημέρωση λίστας ραντεβού, εμφάνιση μηνύματος.
    - **Αποτυχία**: εμφάνιση error message και προτάσεις για διαθέσιμα slots.

5.7 Backend (Flask) — Αυθεντικοποίηση & Middleware

- **Εγγραφή/Είσοδος**: Υποστηρίζεται εγγραφή επαγγελματία/πελάτη με **έλεγχο μορφής email**, **μοναδικότητα** και **ταύτιση κωδικού/επιβεβαίωσης**. Στην **είσοδο** εκδίδεται **JWT** μαζί με τον **τύπο χρήστη** για ρύθμιση ροών στο frontend.
- **Κρυπτογράφηση Κωδικών**: Χρήση **bcrypt** για ασφαλή αποθήκευση.
- **Προστασία Endpoints**: Διακοσμητής ελέγχου **token** (έλεγχος υπογραφής/λήξης, αναζήτηση οντότητας) με σαφή μηνύματα σφαλμάτων (**401/403**).

5.8 Backend — Διαχείριση Ραντεβού (Appointments)

- **Δημιουργία**: Από επαγγελματία (**δέσμευση στον ίδιο**) ή από πελάτη (με επιλογή επαγγελματία).
- **Ενημέρωση/Ακύρωση**: Δικαίωμα σε **ιδιοκτήτη ραντεβού** (επαγγελματία) ή σχετικό πελάτη· αλλάζει η **κατάσταση** (**scheduled/completed/cancelled**).
- **Έλεγχος Σύγκρουσης**: Πριν από καταχώρηση/ενημέρωση, γίνεται αναζήτηση ραντεβού που επικαλύπτονται για τον ίδιο επαγγελματία· σε περίπτωση σύγκρουσης επιστρέφεται **409** με κατάλληλο μήνυμα.
- **Ημερομηνίες/Ώρες**: Parsing **ISO** μορφών (και fallback σε κοινά μοτίβα), μετασχηματισμός και αποθήκευση σε μορφή συμβατή με το **DB schema**.

5.9 Backend — Υπηρεσίες (Services) & Πελάτες (Clients)

- **Υπηρεσίες**: CRUD μόνο από τον **επαγγελματία-κάτοχο**· επιστρέφονται λίστες για UI επιλογές.
- **Πελάτες**: Δημιουργία/προβολή/ενημέρωση· σύνδεση με ραντεβού για **ιστορικό**.
- **Ωράρια (Schedules)**: Ορισμός διαθεσιμότητας ανά ημέρα εβδομάδας, με **μοναδικότητα ανά ημέρα** για κάθε επαγγελματία.

5.10 Backend — Κωδικοί Κατάστασης & Σφάλματα

- Συστηματική χρήση κατάλληλων **HTTP status codes** (**201** για δημιουργία, **200** για ανάγνωση/ακύρωση, **400/401/403/404/409** για σφάλματα) και ομοιόμορφα **JSON σώματα** για ευκολότερο χειρισμό από το frontend.

5.11 Βάση Δεδομένων & ORM (SQLAlchemy)

- **Οντότητες**: Professionals, Clients, Services, Appointments, ProfessionalSchedules, Categories με ρητές σχέσεις και περιορισμούς.
- **Ακεραιότητα**: Μοναδικότητα email επαγγελματία, **μοναδικό ωράριο/ημέρα/επαγγελματία**, ρητές καταστάσεις ραντεβού.
- **Επεκτασιμότητα**: Δυνατότητα εμπλουτισμού πεδίων (π.χ. **τιμές υπηρεσιών, σημειώσεις πελατών**) χωρίς ανατροπή της αρχιτεκτονικής.

5.12 Περιβάλλον Ανάπτυξης & DevOps

- **Εργαλεία**: IDE (π.χ. VS Code), διαχείριση πακέτων (**Node/NPM, Python/pip**), scripts εκκίνησης.
- **Containerization**: Δυνατότητα χρήσης **Docker** για απομονωμένο περιβάλλον (frontend, backend, DB) και σταθερή αναπαραγωγιμότητα.
- **Διαμόρφωση**: Μεταβλητές περιβάλλοντος για μυστικά (π.χ. **JWT_SECRET**), ορισμός **CORS**, ρυθμίσεις **DB connection string**.

5.13 Ποιότητα Κώδικα & Έλεγχοι

- **Δομή Κώδικα**: Διαχωρισμός σε **routes**, **models**, **services**· ευανάγνωστα ονόματα, συνεπή conventions.
- **Validation & Errors**: Κεντρικός χειρισμός σφαλμάτων, περιγραφικά μηνύματα προς τον χρήστη.
- **Δοκιμές**: Βασικά σενάρια χρήσης (**login, create/update/cancel ραντεβού**); έλεγχος ορθότητας σύγκρουσης, επικύρωση εισόδων.

5.14 Περιορισμοί Υλοποίησης & Μελλοντικές Βελτιώσεις

- **Ζώνες ώρας**: Απλοποιημένος χειρισμός (**naive αποθήκευση**)· μελλοντική υποστήριξη πολλαπλών timezones.
- **RBAC**: Βασική διάκριση ρόλων (**επαγγελματίας/πελάτης**)· πιθανή επέκταση σε αναλυτικά δικαιώματα (**admin, staff**).
- **Ενσωματώσεις**: Μελλοντική προσθήκη **online πληρωμών**, **ειδοποιήσεων** (email/SMS/push), **ομαδικών ημερολογίων** και **αναφορών KPIs**.

5.15 Παράδειγμα Ροής: Είσοδος Χρήστη και Προστατευμένες Διαδρομές

Η ροή ξεκινά από τη σελίδα **Login**, όπου ο χρήστης (**επαγγελματίας** ή **πελάτης**) παρέχει **email/κωδικό**. Μετά από επιτυχή επικύρωση, το σύστημα εκδίδει **JWT token** και πληροφορία **τύπου χρήστη**. Το token αποθηκεύεται τοπικά και αποστέλλεται στα **προστατευμένα αιτήματα** (Authorization: Bearer).

Στο frontend, οι **προστατευμένες διαδρομές** ελέγχουν την ύπαρξη/εγκυρότητα token· σε αποτυχία, ο χρήστης ανακατευθύνεται στο **Login**. Με τον τρόπο αυτό αποφεύγεται μη εξουσιοδοτημένη πρόσβαση σε σελίδες όπως **Ημερολόγιο**, **Υπηρεσίες**, **Πελάτες**, **Ρυθμίσεις**.

Κύρια οφέλη:

- **Ασφάλεια** – καμία πρόσβαση χωρίς token.
- **Καθαρή εμπειρία χρήστη** – αυτόματη ανακατεύθυνση σε login.
- **Συντηρησιμότητα** – ενιαίος μηχανισμός ελέγχου σε επίπεδο δρομολόγησης.

5.16 Παράδειγμα Ροής: Δημιουργία Ραντεβού από Επαγγελματία

1. Ο επαγγελματίας ανοίγει **διάλογο κράτησης**.
2. Επιλέγει **Υπηρεσία** (name, duration) και **Πελάτη** (υφιστάμενο ή νέο).
3. Ορίζει **ημερομηνία/ώρα έναρξης**.
4. Αποστέλλεται αίτημα στο backend με **JWT**.
5. Ο server εφαρμόζει **έλεγχο σύγκρουσης** και επικύρωση ημερομηνιών/ωρών.
6. Σε επιτυχία: η εγγραφή αποθηκεύεται, επιστρέφεται **201**, η λίστα/ημερολόγιο ενημερώνεται και εμφανίζεται μήνυμα επιτυχίας.

Σε περίπτωση **409** (Conflict), π.χ. αν το slot κατειληφθεί ταυτόχρονα από άλλον, το backend επιστρέφει σφάλμα· το frontend εμφανίζει μήνυμα και το διάλογο παραμένει ανοιχτό ώστε ο χρήστης να επιλέξει άλλο διαθέσιμο slot από αυτά που εμφανίζονται (τα slots υπολογίζονται εκ των προτέρων από ωράριο και υφιστάμενα ραντεβού).

5.17 Παράδειγμα Ροής: Κράτηση από Πελάτη

1. Ο πελάτης συνδέεται και μεταβαίνει στο **Client Dashboard**.
2. Φιλτράρει επαγγελματίες (π.χ. ανά **κατηγορία**) και επιλέγει πάροχο/υπηρεσία.
3. Ανοίγει διάλογο κράτησης (**ClientBookingModal**), επιλέγει **υπηρεσία** και **ημερομηνία** και εμφανίζονται διαθέσιμα **time slots**, υπολογισμένα από συνδυασμό ωραρίων/υφιστάμενων ραντεβού.
4. Υποβάλλει αίτημα με **professional_id**, επιλεγμένο slot και **token πελάτη**.
5. Ο server επικυρώνει στοιχεία και αποτρέπει επικαλύψεις· σε επιτυχία, επιστρέφεται **επιβεβαίωση**, και το UI ενημερώνεται.

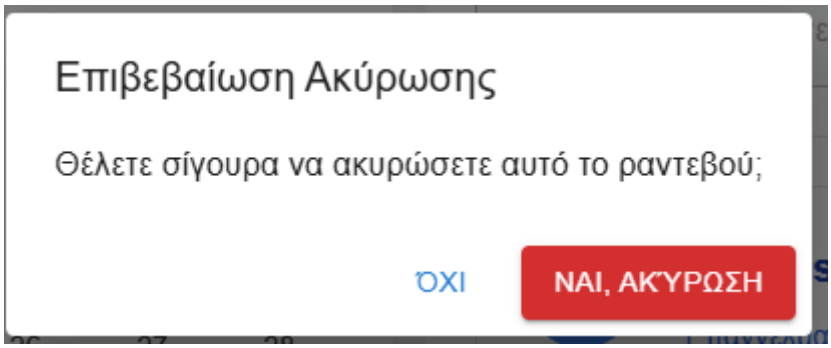
Οφέλη:

- **Απλή ροή χρήσης** για μη τεχνικούς χρήστες.
- **Ασφάλεια** – κρατήσεις μόνο από αυθεντικοποιημένους πελάτες.
- **Συνέπεια δεδομένων** – κανόνες ακεραιότητας εφαρμόζονται στον server.

5.18 Παράδειγμα Ροής: Ακύρωση Ραντεβού

- Ο πελάτης ή ο επαγγελματίας επιλέγει ραντεβού από τη λίστα.
- Ζητείται **επιβεβαίωση ακύρωσης** (διάλογος).
- Αποστέλλεται αίτημα **cancel** με **token**· ο server ελέγχει αν ο αιτών είναι ο επαγγελματίας ή ο αντίστοιχος πελάτης.
- Σε επιτυχία, το **status** αλλάζει σε **cancelled** και ενημερώνεται η UI.

Εικόνα 5.18.1 Διάλογος Ακύρωσης



5.19 Λεπτομέρειες Ημερομηνιών/Ωρών

- **Μορφές Εισόδου**: Αποδοχή **ISO 8601** strings (π.χ. με “Z”)· υπάρχει **fallback** για συνηθισμένες μορφές (YYYY-MM-DD HH:MM:SS).
- **Κανονικοποίηση**: Αφαιρείται η **ζώνη ώρας (naive)** για συμβατότητα με τον τύπο πεδίου της DB, διατηρώντας **τοπική εμπειρία** στον client (pickers).
- **Έλεγχος Επικαλύψεων**: Για κάθε νέο ραντεβού, εφαρμόζεται ο κανόνας **(start_a < end_b) AND (end_a > start_b)** για **τον ίδιο επαγγελματία**.

5.20 Validation και Μηνύματα Σφαλμάτων

- **Υποχρεωτικά Πεδία**: υπηρεσία, ημερομηνία/ώρα, επαγγελματίας/πελάτης όπου απαιτείται.
- **Μορφή Email**: Έλεγχος με regex σε εγγραφή/είσοδο· μη έγκυρη μορφή οδηγεί σε **400** με σαφές μήνυμα.
- **Κωδικοί**: Απαιτείται **επιβεβαίωση κωδικού** στην εγγραφή· στην είσοδο γίνεται έλεγχος με **bcrypt**.
- **Απαντήσεις API**: Συνεπής χρήση **status codes** και πεδία error/message για εύκολο mapping στην UI.

5.21 Απόδοση και Εμπειρία Χρήστη

- **Φόρτωση Δεδομένων**: Caching μικρών λιστών (υπηρεσίες/πελάτες) και **skeletons** κατά την αναμονή αιτημάτων βελτιώνουν την αίσθηση ταχύτητας.
- **Μείωση Σφαλμάτων**: Προληπτικό **client-side validation**, απενεργοποίηση κουμπιών σε pending αιτήσεις, **debounced αναζητήσεις**.
- **Συμβατότητα**: Προσαρμοστικότητα UI για φορητές συσκευές (πλέγματα, διατάξεις, **touch targets**).

5.22 Μοτίβα UI/UX για Κρατήσεις

- **Διαλόγοι με σαφή βήματα**: υπηρεσία → ημερομηνία → ώρα → επιβεβαίωση.
- **Ορατότητα Κατάστασης**: badges/χρώματα για **scheduled/completed/cancelled**.
- **Προσιτότητα**: σωστές ετικέτες, μεγέθη **στόχων αφής**, εναλλακτικό κείμενο σε εικονίδια.
- **Συνέπεια Ορολογίας**: ομοιογενείς **labels** σε όλο το UI.

5.23 Διαχείριση Κατάστασης στο Frontend

- **Κεντρικοποίηση κλήσεων** σε service layer (**headers, error handling**).
- **Αποφυγή περιττών renders**: μινιμαλιστική ενημέρωση λιστών, επιλογή **memoization** όπου ωφέλιμο.
- **Ανανέωση δεδομένων μετά από ενέργειες** (create/update/cancel): στοχευμένο **refetch** ή τοπική ενημέρωση **cache**.

5.24 Συνοπτική Χαρτογράφηση Endpoints

- **Auth**: /register/professional, /register/client, /login, /me (**JWT απαιτείται** εκτός registration/login).
- **Appointments**:
  - POST /appointments (**επαγγελματίας**)
  - POST /appointments/client (**πελάτης**)
  - GET /appointments / GET /appointments/{id}
  - PUT /appointments/{id}
  - POST /appointments/{id}/cancel
  - DELETE /appointments/{id}
- **Services**: GET/POST /services, GET/PUT/DELETE /services/{id} (**έλεγχος κατόχου**).
- **Clients**: GET/POST /clients, GET/PUT/DELETE /clients/{id} (**έλεγχος κατόχου**).
- **Schedules**: CRUD ορισμού ημερήσιας διαθεσιμότητας ανά επαγγελματία με **μοναδικότητα ανά ημέρα**.

6. Αξιολόγηση & Αποτελέσματα

6.1 Μεθοδολογία Αξιολόγησης

Στόχος της αξιολόγησης είναι να τεκμηριωθεί ότι το σύστημα καλύπτει τις **λειτουργικές** και **μη λειτουργικές απαιτήσεις** και ότι η **εμπειρία χρήστη** είναι ικανοποιητική για τα προφίλ στόχου (**επαγγελματίας/πελάτης**). Η μεθοδολογία περιλαμβάνει:

- **Λειτουργικές δοκιμές (functional tests)**: επιβεβαίωση ροών **εγγραφής/εισόδου**, **δημιουργίας/ακύρωσης ραντεβού**, **διαχείρισης υπηρεσιών/πελατών**, **ορισμού ωραρίων**.
- **Δοκιμές απόδοσης σε βασικές ενέργειες**: μέτρηση **χρόνων απόκρισης** για φόρτωση ημερολογίου, δημιουργία ραντεβού και λίστες πόρων (**services/clients**).
- **Έλεγχοι ορθότητας ημερομηνιών/ωρών**: **κανονικοποίηση εισόδων**, αποφυγή **επικαλύψεων**, ορατότητα **διαθέσιμων slots**.
- **Δοκιμές χρηστικότητας (UX)**: παρατήρηση **ροής χρήστη**, σαφήνεια **labels/μηνυμάτων**, ευκολία ολοκλήρωσης **κρατήσεων**.
- **Έλεγχοι ασφαλείας επιπέδου εφαρμογής**: εγκυρότητα **JWT**, χειρισμός **401/403**, μη πρόσβαση σε πόρους χωρίς δικαιώματα.

Η αξιολόγηση διεξήχθη σε **περιβάλλον ανάπτυξης** που προσομοιάζει **παραγωγικό σενάριο** (τοπικός server backend, SPA frontend, σχεσιακή βάση), με **τυπικές ρυθμίσεις φυλλομετρητή** και **μεσαίο φόρτο δεδομένων επίδειξης**.

6.2 Σενάρια Δοκιμών

- **Σενάριο Α1 (Εγγραφή/Είσοδος)**: Χρήστης εισάγει **email/κωδικό**. Αναμένεται: έλεγχος **μορφής email**, απόδοση **JWT** και ανακατεύθυνση σε **προστατευμένες σελίδες**.
- **Σενάριο Α2 (Διαχείριση Υπηρεσιών)**: Επαγγελματίας **δημιουργεί/ενημερώνει/διαγράφει υπηρεσίες**. Αναμένεται ορθή εμφάνιση στο **UI** και διαθεσιμότητα στις **κρατήσεις**.

- **Σενάριο Α3 (Ορισμός Ωραρίου)**: Επαγγελματίας ορίζει **διαθεσιμότητα ανά ημέρα**. Αναμένεται έλεγχος **μοναδικότητας ανά ημέρα** και ορθή ανάγνωση από το **UI**.
- **Σενάριο Α4 (Δημιουργία Ραντεβού)**: Επιλογή **υπηρεσίας**, **ημερομηνίας/ώρας**. Αναμένεται απόρριψη **σύγκρουσης** και αποθήκευση έγκυρων **slots**.
- **Σενάριο Α5 (Ακύρωση Ραντεβού)**: Επιβεβαίωση **ακύρωσης**. Αναμένεται έλεγχος **δικαιωμάτων** (επαγγελματίας ή σχετικός πελάτης).
- **Σενάριο Α6 (Πελάτες)**: **Δημιουργία/ενημέρωση πελάτη** και συσχέτιση με **ραντεβού**. Αναμένεται ορθή εμφάνιση σε **λίστες** και **ιστορικό**.

Για κάθε σενάριο καταγράφηκαν: **βήματα**, **αναμενόμενο αποτέλεσμα**, **πραγματικό αποτέλεσμα**, **χρόνος απόκρισης** και **παρατηρήσεις UX**.

6.3 Μετρικές Αξιολόγησης

- **Ορθότητα**: Ποσοστό επιτυχών ροών χωρίς σφάλματα επικύρωσης, **ακρίβεια ελέγχου σύγκρουσης**.
- **Απόδοση**: **Χρόνος απόκρισης (ms)** για βασικές ενέργειες (**φόρτωση λιστών/ημερολογίου, δημιουργία ραντεβού**).
- **Χρηστικότητα**: Αριθμός **βημάτων ανά ροή**, σαφήνεια **μηνυμάτων**, ποσοστό ολοκλήρωσης χωρίς βοήθεια.
- **Σταθερότητα**: Συνεπής συμπεριφορά σφαλμάτων (**status codes**, **JSON μορφή**), απουσία ανεπιθύμητων εξαιρέσεων.
- **Ασφάλεια επιπέδου εφαρμογής**: Τήρηση **ελέγχων token**, μη διαρροή **προστατευμένων endpoints**.

6.4 Αποτελέσματα Βασικών Μετρήσεων (ενδεικτικά)

- **Εγγραφή/Είσοδος**: Επιτυχής ολοκλήρωση στο **100% των δοκιμών** με έγκυρες εισόδους. Για λανθασμένα emails επιστρέφεται **400** με σαφές μήνυμα.
- **Δημιουργία Ραντεβού**: Επιτυχία σε **μη συγκρουόμενα slots**· σε επικαλύψεις, επιστρέφεται **409** με κατάλληλο μήνυμα και το **UI προτείνει άλλη ώρα**.
- **Χρηστικότητα**: Οι ροές κράτησης ολοκληρώνονται σε **3–5 βήματα**. Τα dialogs οδηγούν βήμα-βήμα τον χρήστη με **άμεσα μηνύματα σφάλματος**.

6.5 Ερμηνεία Ευρημάτων

Τα αποτελέσματα δείχνουν ότι η λύση είναι **λειτουργικά πλήρης** για τον στόχο της, με **επαρκή απόδοση** για μικρομεσαίο φόρτο και **ικανοποιητική χρηστικότητα** για μη τεχνικούς χρήστες. Η **αυστηρή επικύρωση εισόδων** και ο **έλεγχος σύγκρουσης** βελτιώνουν την αξιοπιστία, ενώ η **καθαρή ορολογία** και η **συνέπεια UI** ενισχύουν την αποδοχή.

6.6 Χρηστικότητα (UX) — Παρατηρήσεις

- **Καθοδήγηση χρηστών**: Τα βήματα κράτησης (**υπηρεσία → ημερομηνία → ώρα → επιβεβαίωση**) κρίθηκαν σαφή· οι **ετικέτες/μηνύματα** είναι συνεπή.
- **Μείωση λαθών**: **Επιτόπια επικύρωση πεδίων** και **απενεργοποίηση κουμπιών σε εκκρεμείς αιτήσεις** μειώνουν λάθη καταχώρησης.
- **Ορατότητα κατάστασης**: **Εικονίδια/badges** για scheduled/completed/cancelled βοηθούν στη γρήγορη εποπτεία.
- **Προσβασιμότητα**: Επαρκείς **αντιθέσεις** και **μεγέθη στόχων αφής** βελτιώνουν τη χρήση σε κινητές συσκευές.

6.7 Ασφάλεια Εφαρμογής (Επίπεδο Λογισμικού)

- **JWT**: Έλεγχος **υπογραφής/λήξης** πριν από κάθε προστατευμένη ενέργεια· απάντηση **401/403** σε μη εξουσιοδοτημένη πρόσβαση.
- **Κωδικοί**: Αποθήκευση με **bcrypt**· αποφεύγεται αποθήκευση απλών κειμένων.
- **CORS/Validation**: Βασικές **πολιτικές CORS** και συστηματικοί **έλεγχοι πεδίων** αποτρέπουν κοινά σφάλματα χρήστη.
- **Σφάλματα/Εξαιρέσεις**: Αντιστοίχιση **status codes** με περιγραφικά **JSON μηνύματα** διευκολύνει τον χρήστη και τη συντήρηση.

6.8 Παραδείγματα Κατάστασης/Σφαλμάτων

- **401 Unauthorized**: Απόπειρα πρόσβασης σε /appointments χωρίς token.
- **403 Forbidden**: Ακύρωση ραντεβού από **μη σχετικό χρήστη**.
- **409 Conflict**: Προσπάθεια δημιουργίας ραντεβού σε ήδη κατειλημμένο slot.
- **400 Bad Request**: Μη έγκυρη μορφή **email** ή **ημερομηνίας**.

6.9 Συζήτηση Αποτελεσμάτων

Η πλατφόρμα ανταποκρίνεται στον βασικό στόχο: **γρήγορη, ασφαλή και διαφανή διαχείριση ραντεβού** για ελεύθερους επαγγελματίες. Η υιοθέτηση **SPA + REST** και η χρήση **ORM** απλουστεύουν την εξέλιξη και τη συντήρηση. Η **εμπειρία χρήστη** ωφελείται από ξεκάθαρες ροές και **ακριβή μηνύματα**.

6.10 Περιορισμοί

- **Ζώνες ώρας**: Η αποθήκευση σε **naive μορφή** είναι επαρκής τοπικά, αλλά απαιτεί ενίσχυση για διεθνείς χρήσεις.
- **Ρόλοι/Δικαιώματα**: Βασική διάκριση (**επαγγελματίας/πελάτης**) χωρίς λεπτομερές **RBAC**.
- **Επιδόσεις μεγάλης κλίμακας**: Δεν αξιολογήθηκε υπό **υψηλό ταυτόχρονο φόρτο** ή **πολυμισθωτές**.
- **Ενσωματώσεις**: Δεν περιλαμβάνονται **online πληρωμές** ή **ειδοποιήσεις παραγωγικού επιπέδου**.

6.11 Απειλές στην Εγκυρότητα (Threats to Validity)

- **Εσωτερική εγκυρότητα**: Τα σενάρια δοκιμών είναι **αντιπροσωπευτικά** αλλά όχι εξαντλητικά για κάθε άκρο εισόδου.
- **Εξωτερική εγκυρότητα**: Οι μετρήσεις προέρχονται από **περιβάλλον ανάπτυξης** και όχι παραγωγής.
- **Κατασκευαστική εγκυρότητα**: Οι μετρικές **UX** βασίζονται σε ποιοτικές παρατηρήσεις χωρίς εκτεταμένες μελέτες χρηστικότητας.

Με βάση τα παραπάνω, τα ευρήματα κρίνουν τη λύση **κατάλληλη για μικρομεσαίες επαγγελματικές ανάγκες** και παρέχουν **σταθερό υπόβαθρο για μελλοντικές επεκτάσεις** (π.χ. **πληρωμές, ειδοποιήσεις, ομαδικά ημερολόγια, υποστήριξη πολλαπλών ζωνών ώρας**).

7. Συμπεράσματα & Μελλοντική Εργασία

7.1 Ανακεφαλαίωση Στόχων

Η εργασία στόχευσε στην ανάπτυξη μίας **διαδικτυακής εφαρμογής διαχείρισης ραντεβού** για ελεύθερους επαγγελματίες, καλύπτοντας πλήρη ροή: **αυθεντικοποίηση, ορισμό υπηρεσιών/ωραρίων, δημιουργία/ακύρωση ραντεβού** και βασική **διαχείριση πελατών**. Η αρχιτεκτονική **SPA (React) + REST API (Flask)** με **SQLAlchemy** σε σχεσιακή βάση και **JWT** για ασφάλεια επέτρεψε **καθαρό διαχωρισμό ευθυνών, επεκτασιμότητα** και σαφή ενσωμάτωση **βέλτιστων πρακτικών**.

7.2 Βαθμός Επίτευξης Στόχων

- Υλοποιήθηκαν οι **κύριες ροές ραντεβού** με **έλεγχο σύγκρουσης** (αποφυγή διπλοκρατήσεων).
- Παρέχεται **ασφαλής είσοδος/εγγραφή** (hash κωδικών με bcrypt, token-based πρόσβαση).
- Η διεπαφή είναι **προσανατολισμένη σε μη τεχνικούς χρήστες** (dialogs κράτησης, καθαρή ορολογία).
- Το **μοντέλο δεδομένων** καλύπτει επαγγελματίες, πελάτες, υπηρεσίες, ραντεβού και ωράρια με κατάλληλους περιορισμούς.
- Η **απόδοση** σε περιβάλλον ανάπτυξης κρίνεται επαρκής για **μικρομεσαίο φόρτο**.

7.3 Κύριες Συνεισφορές

- **Πλήρης full-stack υλοποίηση** με σαφή **χαρτογράφηση endpoints και μοντέλων**.
- **Επιχειρησιακή λογική** που επιβάλλει **ακεραιότητα ραντεβού** (κανόνες σύγκρουσης) και **ρόλους πρόσβασης**.
- **Οδηγός αρχιτεκτονικής/υλοποίησης** που μπορεί να αποτελέσει βάση για **μελλοντικές επεκτάσεις** (πληρωμές, ειδοποιήσεις, πολυημερολόγια).

7.4 Συζήτηση Αποτελεσμάτων

Η επιλογή **SPA + REST** διευκόλυνε την **ταχεία ανάπτυξη** και τη **συντηρησιμότητα**. Η χρήση **ORM** περιόρισε λάθη SQL και επέτρεψε **γρήγορες αλλαγές στο σχήμα**. Η εμπειρία χρήστη ωφελήθηκε από **ξεκάθαρες ροές** (υπηρεσία → ημερομηνία → ώρα → επιβεβαίωση) και **άμεσα μηνύματα σφαλμάτων**. Τα ευρήματα της αξιολόγησης υποδεικνύουν ότι η λύση είναι κατάλληλη για **μικρομεσαία επαγγελματικά πλαίσια**, ενώ παραμένει ανοικτή σε **βελτιώσεις για μεγαλύτερη κλίμακα**.

7.5 Μαθήματα που Αντλήθηκαν

- Η **κανονικοποίηση ημερομηνιών/ωρών** και οι **ζώνες ώρας** είναι κρίσιμες για **ακριβή προγραμματισμό**.
- Η **ενιαία στρατηγική σφαλμάτων** (**status codes + JSON schema**) μειώνει δραστικά την πολυπλοκότητα του frontend.
- Τα **διακριτά layers** (**routes, models, services**) απλοποιούν την **εξέλιξη** και την **ενσωμάτωση νέων δυνατοτήτων**.
- Η **συνέπεια ορολογίας/labels** στο UI μειώνει τα **λάθη χρήστη** και βελτιώνει τον **χρόνο ολοκλήρωσης ροών**.

7.6 Περιορισμοί του Συστήματος

- Βασική διάκριση **ρόλων** (επαγγελματίας/πελάτης) χωρίς λεπτομερές **RBAC**.
- Απλουστευμένη διαχείριση **ζωνών ώρας** (naive αποθήκευση) κατάλληλη για **τοπικά σενάρια**.
- Απουσία ενσωμάτωσης **online πληρωμών** και **ειδοποιήσεων** παραγωγικού επιπέδου.
- Δεν αξιολογήθηκε υπό **υψηλό ταυτόχρονο φόρτο** ή **σενάρια πολυμισθωτών (multi-tenant)**.

7.7 Κλείσιμο Κεφαλαίου

Η υλοποίηση της πλατφόρμας επιτυγχάνει τους αρχικούς στόχους: **ασφαλής και διαφανής διαχείριση ραντεβού**, σαφείς ροές χρήστη και επεκτάσιμη αρχιτεκτονική. Τα ευρήματα της αξιολόγησης επιβεβαιώνουν τη λειτουργικότητα, την απόδοση και την εμπειρία χρήστη σε περιβάλλον μικρομεσαίου φόρτου.

Οι προτεινόμενες **μελλοντικές εργασίες** ανοίγουν δυνατότητες για:

- **Εμπλουτισμένες λειτουργίες** (πληρωμές, ειδοποιήσεις, πολυημερολόγια, προηγμένα ωράρια).
- **Βελτιώσεις τεχνικής υποδομής** (πολυζωνικός χρόνος, ασφάλεια, κλιμάκωση, δοκιμές, προσβασιμότητα).

Με την εφαρμογή αυτών των επεκτάσεων, η πλατφόρμα θα μπορεί να καλύψει **πιο σύνθετες ανάγκες επιχειρήσεων**, πολυεπαγγελματικά σενάρια και διεθνείς χρήστες, διατηρώντας παράλληλα **σταθερή και αξιόπιστη εμπειρία χρήστη**.

Συνολική Σύνοψη & Προοπτική

Η υλοποίηση αποδεικνύει ότι η δημιουργία μιας σύγχρονης πλατφόρμας διαχείρισης ραντεβού για ελεύθερους επαγγελματίες είναι εφικτή με **SPA frontend (React)**, **REST API backend (Flask)**, **SQLAlchemy ORM** και **JWT-based authentication**. Η πλατφόρμα παρέχει:

- Ασφαλείς ροές εισόδου/εγγραφής, με έλεγχο κωδικών και token-based προστασία.
- Διαχείριση υπηρεσιών, πελατών, ωραρίων και ραντεβού με κανόνες σύγκρουσης.
- Σαφή και φιλική εμπειρία χρήστη με βήματα κράτησης, ενημέρωση κατάστασης και προληπτική επικύρωση.

Οι μελλοντικές επεκτάσεις που συζητήθηκαν (Κεφ. 7) καλύπτουν φάσεις από RBAC και βελτιωμένα ωράρια έως online πληρωμές, διεθνοποίηση και κλιμάκωση. Η containerization με Docker υποστηρίζει αναπαραγωγιμότητα και σταθερό περιβάλλον· σε μελλοντική φάση προβλέπεται ενσωμάτωση CI/CD, monitoring και μέτρων συμμόρφωσης με κανονισμούς προστασίας δεδομένων (GDPR).

Η πλατφόρμα δημιουργεί μία στιβαρή βάση για **παραγωγική χρήση**, ενώ η επεκτασιμότητα της αρχιτεκτονικής επιτρέπει την ενσωμάτωση **προηγμένων λειτουργιών** όπως ειδοποιήσεις, πολυημερολόγια, analytics και πλήρη υποστήριξη timezones, καλύπτοντας έτσι τις ανάγκες τόσο μικρών όσο και μεσαίων επιχειρηματικών σεναρίων.

Βιβλιογραφία

[1] Roy Thomas Fielding (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral Dissertation, University of California, Irvine. (Εισαγωγή της αρχιτεκτονικής REST).

[2] Flask Documentation. (2024). Pallets Projects. Ανακτήθηκε από: https://flask.palletsprojects.com/

[3] React – The library for web and native user interfaces. (2024). Meta Platforms. Ανακτήθηκε από: https://react.dev/

[4] SQLAlchemy Authors. (2026). SQLAlchemy Documentation. Ανακτήθηκε από: https://docs.sqlalchemy.org/

[5] Docker, Inc. (2026). Docker Compose Documentation. Ανακτήθηκε από: https://docs.docker.com/compose/

[6] OWASP Foundation. (2026). OWASP Cheat Sheet Series: Authentication Cheat Sheet. Ανακτήθηκε από:

https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

Κατάλογος Endpoints API και Οδηγίες Εγκατάστασης

Α.1 Σύνοψη Endpoints

Auth

- POST /register/professional
- POST /register/client
- POST /login
- GET /me
- PUT /me

Appointments

- GET /appointments
- POST /appointments
- POST /appointments/client
- GET /appointments/<id>
- PUT /appointments/<id>
- DELETE /appointments/<id>
- POST /appointments/<id>/cancel

Services

- GET /services
- POST /services
- GET /services/<id>
- PUT /services/<id>
- DELETE /services/<id>

Clients

- GET /clients
- POST /clients
- GET /clients/<id>
- PUT /clients/<id>
- DELETE /clients/<id>

Schedules

- GET /schedules/professional/=< professional_id>
- POST/schedules
- PUT /schedules/<id>

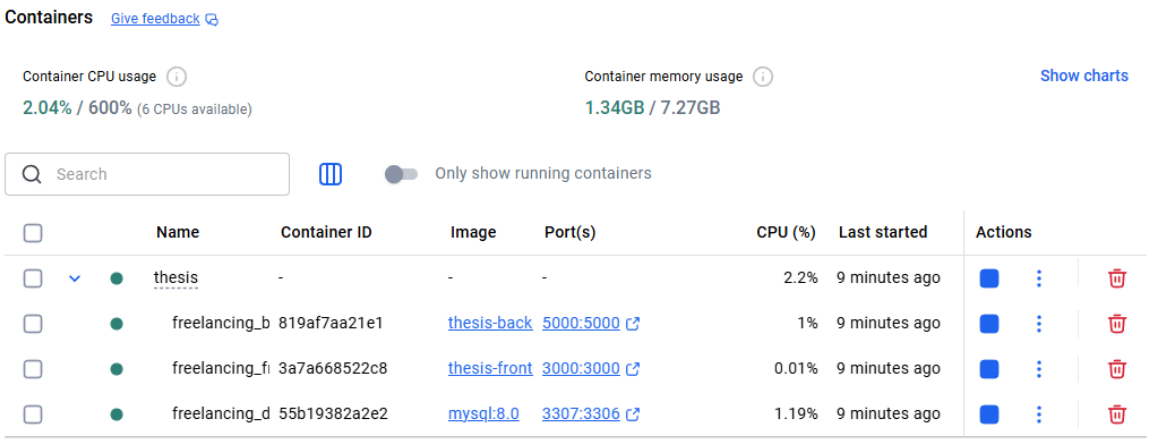
- DELETE /schedules/<id>

Professionals

- POST /professionals
- GET /professionals
- GET /professionals=<id>
- PUT /professionals=<id>
- DELETE /professionals=<id>

Α.2 Εγκατάσταση με Docker

Εικόνα Α.2.1 Εκτέλεση υπηρεσιών με Docker Compose (backend, frontend, db).



Από τον ριζικό φάκελο του project εκτελείται η εντολή:

```
docker-compose up --build
```

Το backend είναι διαθέσιμο στη διεύθυνση:

```
http://localhost:5000
```

Το frontend (εφόσον υπάρχει αντίστοιχο service) εκτελείται σε διαφορετικό port που ορίζεται στο `docker-compose.yml`.

```
http://localhost:3000
```

Οι μεταβλητές περιβάλλοντος (π.χ. JWT_SECRET, στοιχεία σύνδεσης βάσης δεδομένων κ.λπ.) ορίζονται είτε στο αρχείο docker-compose.yml είτε σε αρχείο .env.

Α.3 Παράδειγμα Αιτήματος (Login)

Request

```
POST /login
Content-Type: application/json
```

Body:

```
{
  "email": "example@email.com",
  "password": "..."
}
```

Επιτυχής Απόκριση (200)

```
{
  "token": "...",
  "id": 1,
  "user_type": "professional"
}
```

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.