



**ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ**

**Σχολή Θετικών Επιστημών και Τεχνολογίας**

**Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα**

**ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

**Διαφορές και Σύγκριση Σχεδιασμού και Ανάπτυξης ηλεκτρονικού καταστήματος για αγορά βιντεοπαιχνιδιών με χρήση Στοιβάς MERN και PERN**

**ΟΝΟΜΑ ΦΟΙΤΗΤΗ**

**ΙΩΑΝΝΟΥ ΙΩΑΝΝΗΣ**

**ΟΝΟΜΑ ΕΠΙΒΛΕΠΟΝΤΑ ΚΑΘΗΓΗΤΗ  
ΜΟΥΤΑΦΗΣ ΠΑΝΑΓΙΩΤΗΣ**

**ΠΑΤΡΑ  
ΜΗΝΑΣ, ΕΤΟΣ 2025**

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του/της φοιτητή/φοιτήτριας («συγγραφέας/δημιουργός») που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο/η 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/της 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/της συγγραφέα/δημιουργού. Ο/Η συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.



Διαφορές και Σύγκριση Σχεδιασμού και Ανάπτυξης ηλεκτρονικού  
καταστήματος για αγορά βιντεοπαιχνιδιών με χρήση Στοίβας MERN  
και PERN

Ιωάννης Ιωάννου

Επιτροπή Κρίσης

Επιβλέπων Καθηγητής:  
Παναγιώτης Μουτάφης  
Μέλος ΣΕΠ ΕΑΠ

Συν-Επιβλέπουσα Καθηγήτρια:  
Χριστοπούλου Ελένη  
Μέλος ΣΕΠ ΕΑΠ

Στο παππού μου Γιάννη



## Περίληψη

Στη παρούσα εργασία παρουσιάζεται η διαδρομή για την κατασκευή ενός ηλεκτρονικού καταστήματος ψηφιακών παιχνιδιών με τους δύο τεχνολογικούς συνδυασμούς, MERN Stack (MongoDB, Express.js, React.js, Node.js) και PERN Stack (PostgreSQL, Express.js, React.js, Node.js) αναλύοντας και σχεδιάζοντας ίδιες λειτουργίες, δεδομένα και στόχους. Επίσης παρουσιάζονται σύγχρονα εργαλεία και μέθοδοι που χρησιμοποιήθηκαν όπως το ODM (Object Data Modeling) Mongoose και ORM (Object Relational Modeling) Sequelize, αυθεντικοποίηση βασισμένη σε κωδικό, εξουσιοδότηση βασισμένη στο ρόλο χρήστη, του Mailtrap για δοκιμές αποστολής email για ανάκτηση κωδικού, του εξωτερικού παρόχου αποθηκευτικού χώρου Cloudinary για την αποθήκευση εικόνων και Stripe για τη πληρωμή των ψηφιακών παιχνιδιών. Το κύριο μέρος της εργασίας ξεκινάει με την γενική περιγραφή του e-shop το οποίο ονομάζεται CheapGames. Οντότητες και ιδιότητες από τη γενική περιγραφή αναλύονται και παρουσιάζονται για να ακολουθήσει ο καθορισμός των λειτουργικών απαιτήσεων και ο διαφορετικός σχεδιασμός των δεδομένων για τις βάσεις MongoDB και PostgreSQL που είναι και η κύρια διαφορά των δύο στίβων. Λεκτικές περιγραφές περιπτώσεων χρήσης συνοδευόμενες από ScreenShots από τις σελίδες δείχνουν το τι σχεδιάστηκε και το αποτέλεσμα στην μεριά του χρήστη. Τα endpoints στα οποία ένας client μπορεί να στέλνει αιτήματα στον server, καθορίστηκαν βάση της αρχιτεκτονικής Rest και παρουσιάζονται σε ένα παράρτημα αναφέροντας την περίπτωση χρήσης στην οποία αντιστοιχούν μέσω του κωδικού της και την εικόνα του screenShot με τη σελίδα του site για να γνωρίζει ο αναγνώστης σε ποιο σημείο της διαδικτυακής εφαρμογής εκτελούνται. Το αποτέλεσμα της ανάλυσης και σχεδίασης οδηγεί στον τρόπο υλοποίησης του backend της διαδικτυακής εφαρμογής CheapGames, συμπεριλαμβάνοντας την δομή των φακέλων και των αρχείων κατά την υλοποίηση με τις δύο στίβες. Ρυθμίσεις περιβάλλοντος, εγκατάσταση και σύνδεση server με τη κάθε βάση και κώδικες αρχείων δείχνουν πως η ανάλυση και η σχεδίαση έπαιξε ρόλο στη κάθε υλοποίηση. Το frontend της διαδικτυακής εφαρμογής CheapGames επίσης παρουσιάζεται μέσω των απαραίτητων ρυθμίσεων, της δομής των φακέλων, ενός παραρτήματος με μία σύντομη περιγραφή για κάθε φάκελο και αρχείο και των βιβλιοθηκών που χρησιμοποιήθηκαν.

Τέλος παρουσιάζονται τα συμπεράσματα για τα κοινά σημεία και τις διαφορές που συναντήθηκαν σε όλη τη διαδρομή μέχρι την υλοποίηση της διαδικτυακής εφαρμογής CheapGames με τις δύο τεχνικές.

## Λέξεις- Κλειδιά

MERN Stack, Pern Stack, MongoDB, Express, ReactJs, PostgreSQL, ODM, Mongoose, ORM, Sequelize, ηλεκτρονικό κατάστημα, e-shop, backend, frontend, διαδικτυακή εφαρμογή, server, περίπτωση χρήσης, endpoints, αρχιτεκτονική Rest

## **Differences and Comparison of Design and Development of an e-store for buying video games using MERN and PERN Stacks**

This paper presents the path for building an online digital games store with the two technological combinations, MERN Stack (MongoDB, Express.js, React.js, Node.js) and PERN Stack (PostgreSQL, Express.js, React.js, Node.js) by analyzing and designing the same functions, data and goals. This paper also talks about some modern tools and methods that were used for the implementation. These include ODM (Object Data Modeling) Mongoose and ORM (Object Relational Modeling) Sequelize, as well as password-based authentication, user role-based authorization, Mailtrap for sending tests emails to find lost passwords, Cloudinary for storing images, and Stripe for purchasing digital games. The main part of the paper begins with the general description of the e-shop called CheapGames. Entities and properties from the general description are analyzed and presented to follow the definition of functional requirements and the different designs of the data for the MongoDB and PostgreSQL databases, which is the main difference between the two stacks. Screenshots from the pages accompany verbal descriptions of use cases, illustrating the design process and the resulting user experience. The endpoints to which a client can send requests to the server were defined based on the Rest architecture and listed in an appendix with their use case code and screenshot image with the site page so the reader knows where they are executed in the web application. The result of the analysis and design leads to the implementation of the backend of the CheapGames web application, including the structure of the folders and files during the implementation with the two stacks. Environment settings, installation and server connection with each database and file codes show how the analysis and design played a role in each implementation. The frontend of the CheapGames web application is also presented through the necessary settings, the folder structure, an appendix with a brief description for each folder and file and the libraries used.

The conclusions are presented on the common points and differences encountered throughout the implementation of the CheapGames web application with the two techniques.

### **Keywords**

MERN Stack, Pern Stack, MongoDB, Express, ReactJs, PostgreSQL, ODM, Mongoose, ORM, Sequelize, online store, e-shop, backend, frontend, web application, server, use case

## ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη.....	1
Differences and Comparison of Design and Development of an e-store for buying video games using MERN and PERN Stacks.....	2
ΠΕΡΙΕΧΟΜΕΝΑ.....	3
Κατάλογος Εικόνων.....	5
Κατάλογος Πινάκων.....	8
Συντομογραφίες και Ακρωνύμια.....	9
1. Εισαγωγή.....	10
1.1 Full Stack Web Development.....	10
1.2 Σκοπός και Στόχοι.....	10
2. Θεωρητικό Υπόβαθρο.....	11
2.1 Βάσεις Δεδομένων.....	11
2.1.1 Σχεσιακές Βάσεις δεδομένων.....	11
2.1.2 Μη Σχεσιακές Βάσεις Δεδομένων.....	12
2.2. MERN STACK και PERN STACK.....	14
2.2.1 ΣΥΝΔΥΑΣΜΟΣ ΤΕΧΝΟΛΟΓΙΩΝ-STACK.....	14
2.2.2 MongoDB.....	15
2.2.3 PostgreSQL.....	18
2.3.4 Express Server.....	19
2.2.5 React.js.....	20
2.2.6 Node.js.....	21
2.3 ODMS ΚΑΙ ORMS.....	22
2.3.1 Object-Data Modeling (ODM):Mongoose.....	22
2.3.2 Object-Relational Mapping (ORM):Sequelize.....	25
2.4 Αρχιτεκτονική Rest.....	29
2.5 Authentication και Authorization.....	31
2.5.1 Authentication Based On Password.....	31
2.5.2 Authorization Based On Role.....	33
3 Εργαλεία Ανάπτυξης και Υποστήριξης.....	35
3.1 Visual Studio Code.....	35
3.2 git και GitHub.....	35
3.3 Postman.....	36
3.4 CLOUDINARY.....	38
3.5 STRIPE.....	39
3.6 MailTrap.....	40
4. Ανάλυση και Σχεδίαση Λειτουργικότητας του Συστήματος Cheap Games.....	42
4.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΟΣ CHEAP GAMES.....	42
4.2 ΛΕΙΤΟΥΡΓΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ.....	43
4.3 Data Modeling.....	48
4.3.1 Εννοιολογικό Επίπεδο (Conceptional level).....	49
4.3.2 Λογικό Επίπεδο (Logical Level).....	50
4.3.3 Μοντέλο Embeeded-Reference.....	56
4.3.4 Φυσικό Επίπεδο.....	59
4.4 Διαγράμματα Περιπτώσεων Χρήσης.....	61
4.5 Λεκτικές Περιγραφές ΠΧ και Στιγμιότυπα Οθόνης από την Υλοποίηση.....	67

4.6 Καταγραφή Endpoints.....	109
5. Υλοποίηση Backend.....	111
5.1 Υλοποίηση Backend με MERN Stack.....	111
5.1.1 Δομή Φακέλων και Αρχείων του Backend (MERN).....	111
5.1.2 Ρύθμιση του EntryPoint Αρχείου.....	113
5.1.3 Σύνδεση Server με τη MongoDB μέσω Mongoose και μηχανισμός GridFs.....	114
5.1.4 Διαχείριση Σφαλμάτων (Error Handling).....	116
5.1.5 Ορισμός Μοντέλων Δεδομένων με Mongoose.....	116
5.1.6 Routes και Middlewares Αυθεντικοποίησης και Εξουσιοδότησης.....	117
5.1.7 Controllers.....	118
5.2 Υλοποίηση Backend με PERN Stack.....	119
6. Υλοποίηση FrontEnd.....	126
6.1 Ρυθμίσεις και Δομή φακέλου Frontend.....	126
7. Συμπεράσματα.....	130
7.1 Ανασκόπηση Σταδίων Ανάπτυξης.....	130
7.2 Συμπεράσματα Ανάλυσης και Σχεδίασης.....	132
7.3 Συμπεράσματα Υλοποίησης Backend και Frontend.....	132
7.4 Ευελιξία και Ταχύτητα.....	133
7.5 Διαχείριση Διαγραφής Δεδομένων.....	134
7.6 Επιλογή Στοιβάς MERN vs PERN.....	134
Βιβλιογραφία.....	136
Παράρτημα Α - Πίνακας Α.1 Καθορισμός EndPoints.....	138
Παράρτημα Β - Πίνακας Β.1 Φάκελοι και Αρχεία Frontend με Περιγραφές.....	144

## Κατάλογος Εικόνων

Εικόνα 2.1 Παράδειγμα Πίνακας Πελάτη.....	12
Εικόνα 2.2 Παράδειγμα JSON.....	16
Εικόνα 2.3 Reference και Embedding subdocument.....	17
Εικόνα 2.4 GridFS – Ο μηχανισμός αποθήκευσης αρχείων στη MongoDB.....	18
Εικόνα 2.5 Components σε προγραμματιστικό Περιβάλλον Visual Studio Code.....	21
Εικόνα 2.6 Λογότυπα MongoDB και Mongoose.....	23
Εικόνα 2.7: Αρχιτεκτονική ενός API με MERN stack και Mongoose.....	23
Εικόνα 2.8 Παράδειγμα Ορισμού Σχήματος και Πεδίων:Τύπων Δεδομένων.....	24
Εικόνα 2.9 Παράδειγμα Ορισμού Σχήματος και εξαγωγή Μοντέλου.....	24
Εικόνα 2.10 Το επίσημο λογότυπο του Sequelize.....	25
Εικόνα 2.11 Αρχιτεκτονική ενός API με PERN stack που χρησιμοποιεί Sequelize ως ORM.....	26
Εικόνα 2.12 Εγκατάσταση του Sequelize μέσω npm.....	27
Εικόνα 2.13 Παράδειγμα σύνδεσης του sequelize με μία ή περισσότερες βάσεις, όπως παρουσιάζεται στην επίσημη τεκμηρίωση του Sequelize.....	27
Εικόνα 2.14: Παράδειγμα Ορισμού Μοντέλου Χρήστη με ιδιότητες username και birthday.....	28
Εικόνα 2.15: Παράδειγμα Ορισμού Μοντέλων WishList και Wish και σχέσεων μεταξύ των δύο μοντέλων.....	28
Εικόνα 2.16 Διάγραμμα κύκλου αιτήματος-απόκρισης HTTP.....	30
Εικόνα 2.17 Διαγραμμα Ροής Δεδομενων σε Mern Stack.....	31
Εικόνα 2.18 Token-Based Authentication: Έκδοση και χρήση JWT για προστατευμένα αιτήματα.....	33
Εικόνα 2.19, Διάγραμμα ροής Password and Role-Based Authorization σε ένα REST API (MERN ή PERN) χρησιμοποιώντας JWT, Middleware.....	34
Εικόνα 3.1 Το λογότυπο του Visual Studio Code(VS Code).....	35
Εικόνα 3.2 Λογότυπα git και GitHub.....	36
Εικόνα 3.3 Λογότυπο Postman.....	36
Εικόνα 3.4 Γραφικό Περιβάλλον Postman.....	37
Εικόνα 3.5 Διαδικασία αποθήκευσης εικόνων σε Cloudinary μέσω REST API σε εφαρμογή MERN/PERN.....	39
Εικόνα 3.6: Διάγραμμα ροής πληρωμών μέσω Stripe Payment Intent API(Ανακτήθηκε από: Stripe Documentation).....	40
Εικόνα 3.7 Διάγραμμα Ακολουθίας για Ανάκτηση Κωδικού(Mailtrap).....	41
Εικόνα 4.1, Επίπεδα Data Modeling.....	49
Εικόνα 4.2, Μοντελο Οντοτήτων Συσχετίσεων στο εννοιολογικό επίπεδο.....	50
Εικόνα 4.3 Στιγμιότυπο από Steam.....	51
Εικόνα 4.4 Στιγμιότυπο με ιδιότητες που αφορούν ένα παιχνίδι.....	51
Εικόνα 4.5 Μοντέλο Οντοτήτων-Συσχετίσεων(ΜΟΣ).....	55
Εικόνα 4.6 Σχεδίαση Μοντέλου Embedded-Reference για MongoDB.....	58
Εικόνα 4.7, ERD από το PgAdmin - Φυσικό Επίπεδο.....	60
Εικόνα 4.8, Αποθήκευση Δεδομένων στη MongoDB- Φυσικό Επίπεδο.....	61
Εικόνα 4.9 Διάγραμμα Ομαδοποιημένων Περιπτώσεων χρήσης για Πλοήγηση,Αναζήτηση Παιχνιδιών και Προσθήκη Παιχνιδιών Στα Αγαπημένα(Για Όλους τους Χρήστες).....	62
Εικόνα 4.10 Διάγραμμα Ομαδοποιημένων Περιπτώσεων χρήσης για Διαχείριση Προφίλ και Στοιχείων Χρήστη και Ανάκτηση Στοιχείων.....	63
Εικόνα 4.11 Διάγραμμα Ομαδοποιημένων Περιπτώσεων χρήσης για Διαχείριση Καλαθιού,	

Παραγγελιών, Αγοράς/Πληρωμή Παραγγελίας, Download Αγορασμένου Παιχνιδιού και Αξιολόγηση Παιχνιδιού .....	64
Εικόνα 4.12 Διάγραμμα Ομαδοποιημένων Περιπτώσεων Χρήσης για Διαχείριση Παιχνιδιών, Χρηστών, Παραγγελιών και Εισόδων και Διαχείριση Slide για Εποχιακές Εκπτώσεις και εφαρμογή Εποχιακών Εκπτώσεων (Για Διαχειριστές).....	66
Εικόνα 4.13 Αρχική σελίδα της εφαρμογής Cheap Games για μη συνδεδεμένο χρήστη.....	67
Εικόνα 4.14 Αρχική σελίδα της εφαρμογής Cheap Games για συνδεδεμένο χρήστη με ρόλο διαχειριστή.....	68
Εικόνα 4.15 Απόσπασμα κεφαλίδας Αρχικής σελίδα της εφαρμογής Cheap Games για συνδεδεμένο χρήστη με ρόλο user.....	68
Εικόνα 4.16 Φόρμα Εισαγωγής Στοιχείων Σύνδεσης.....	69
Εικόνα 4.17, Μήνυμα Ειδοποίησης για εισαγωγή μη εγκυρων στοιχείων σύνδεσης.....	70
Εικόνα 4.18 Κατάλογος αγαπημένων παιχνιδιών χρήστη.....	71
Εικόνα 4.19 Ενημέρωση στον κατάλογο αγαπημένων παιχνιδιών ότι δεν υπάρχουν αγαπημένα παιχνίδια.....	71
Εικόνα 4.20 Κατάλογος αγορασμένων παιχνιδιών χρήστη.....	72
Εικόνα 4.21 Στιγμιότυπο Σελίδας Καταλόγου με εισαγωγή Φίλτρων.....	73
Εικόνα 4.22 Στιγμιότυπο από αναζήτηση στο κατάλογο παιχνιδιών με εισαγωγή τίτλου.....	74
Εικόνα 4.23 Στιγμιότυπο από λεπτομέρειες παιχνιδιού που δεν έχει αγοραστεί από τον χρήστη.....	75
Εικόνα 4.24, Στιγμιότυπο από λεπτομέρειες παιχνιδιού που έχει αγοραστεί.....	76
Εικόνα 4.25, Ενημέρωση χρήστη ότι πρέπει κάνει login για να δει τις λεπτομέρειες ενός παιχνιδιού.....	77
Εικόνα 4.26 Στιγμιότυπο από σελίδα εγγραφής χρήστη.....	78
Εικόνα 4.27 Στιγμιότυπο από σελίδα ανάκτησης στοιχείων.....	79
Εικόνα 4.28 Στιγμιότυπο από εμφάνιση ειδοποίησης αποστολής email με link για ανάκτηση στοιχείων.....	79
Εικόνα 4.29, Στιγμιότυπο από Email Ανάκτησης Κωδικού.....	80
Εικόνα 4.30 Στιγμιότυπο από σελίδα εισαγωγής Νέου Κωδικού.....	81
Εικόνα 4.31 Στιγμιότυπο από εμφάνιση ειδοποίησης για μη καταχωρημένο email στη βάση Δεδομένων.....	81
Εικόνα 4.32 Εμφάνιση Ειδοποίησης μη επιτυχούς εισαγωγής Κωδικού.....	82
Εικόνα 4.33 Μενού επιλογών για συνδεδεμένο χρήστη με ρόλο διαχειριστή(admin).....	83
Εικόνα 4.34 Μενού Επιλογών για συνδεδεμένο χρήστη με ρόλο απλού χρήστη(user).....	83
Εικόνα 4.35 Μενού Επιλογών Τροποποίησης Προφίλ Χρήστη.....	83
Εικόνα 4.36 Σελίδα Τροποποίηση Στοιχείων Χρήστη.....	84
Εικόνα 4.37 Σελίδα Ενημέρωσης Κωδικού.....	85
Εικόνα 4.38 Στιγμιότυπο εμφάνισης ειδοποίησης επιτυχούς πρόσθεσης παιχνιδιού στα αγαπημένα του χρήστη.....	86
Εικόνα 4.39 Πρόσθεση Παιχνιδιού στο καλάθι αγορών.....	87
Εικόνα 4.40 Καλάθι Αγορών.....	88
Εικόνα 4.41 Σελίδα Καλαθιού χωρίς να έχουν προστεθεί παιχνίδια.....	88
Εικόνα 4.42 Σελίδα PersonalInfo-Εισαγωγής Στοιχείων χρήστη κατά την αγορά παιχνιδιών.....	89
Εικόνα 4.43 Επιβεβαίωση Παραγγελίας.....	90
Εικόνα 4.44 Επιλογή Τρόπου Πληρωμής.....	90
Εικόνα 4.45 Περιβάλλον πληρωμών Stripe (test mode).....	91
Εικόνα 4.46 Σελίδα με τις ολοκληρωμένες αγορές του χρήστη.....	91
Εικόνα 4.47 Προβολή Λεπτομερειών Αγοράς Χρήστη.....	92



Εικόνα 4.48 Σελίδα Λήψης Αγορασμένου Παιχνιδιού.....	93
Εικόνα 4.49 Φόρμα εισαγωγής αξιολόγησης.....	94
Εικόνα 4.50 Κονσόλα με μενού επιλογών Διαχειριστή.....	95
Εικόνα 4.51 Φόρμα Εισαγωγής Στοιχείων Καινούριου Παιχνιδιού.....	96
Εικόνα 4.52 Σελίδα με όλα τα καταχωρημένα παιχνίδια(Admin).....	97
Εικόνα 4.53 Φόρμα Εισαγωγής Αρχείου για παιχνίδι.....	98
Εικόνα 4.54 Ανεβασμένο αρχείο και πληροφορίες αρχείου.....	98
Εικόνα 4.55 Ανέβασμα και Διαγραφή Εικόνων Παιχνιδιού.....	99
Εικόνα 4.56 Τροποποίηση Στοιχείων Παιχνιδιού.....	100
Εικόνα 4.57 Διαγραφή Παιχνιδιού.....	101
Εικόνα 4.58 Λίστα Όλων των Χρηστών.....	102
Εικόνα 4.59 Ενημέρωση στοιχείων χρήστη.....	103
Εικόνα 4.60 Λίστα όλων των Αγορών.....	104
Εικόνα 4.61 Πληροφορίες αγοράς.....	105
Εικόνα 4.62 Συνολο Εσόδων και Αγορών με γράφημα.....	106
Εικόνα 4.63 Ρυθμιση Εποχιακής Εκπτώσης και Slide με διαφημίσεις.....	107
Εικόνα 5.1 Δομή Φακέλου backend.....	112
Εικόνα 5.2 Εισαγωγή Βιβλιοθηκών και συναρτήσεων στο αρχείο app.js.....	113
Εικόνα 5.3 Απόσπασμα κώδικα από αρχείο App.js.....	114
Εικόνα 5.4 Αρχείο dbConnect.js - Σύνδεση με τη βάση MongoDB μέσω Mongoose.....	115
Εικόνα 5.5 Αρχείο GridFs.....	116
Εικόνα 5.6 Στιγμιότυπο από αρχείο models/game.js -Validation Rule.....	117
Εικόνα 5.7 Κώδικας για αντιστοίχιση route,controllers, και προστασία controller με middlewares isAuthenticatedUser και authorizeRoles.....	118
Εικόνα 5.8, Δομή Φάκελων και αρχείων Backend (PERN Stack).....	120
Εικόνα 5.9 Σύνδεση με Βάση Δεδομένων μέσω Sequelize.....	121
Εικόνα 5.10 Αποσπάσματα κώδικα Αρχείου User Model-Κάποια πεδία και μέθοδοι για το Password.....	122
Εικόνα 5.11 Απόσπασμα από αρχείο relationships.js.....	123
Εικόνα 5.12 Απόσπασμα από αρχείο orderController.js.....	124
Εικόνα 6.1 Δημιουργία React Προτύπου για Δημιουργία Web Api.....	126
Εικόνα 6.2 Δομή Φακέλων και Αρχείων Frontend.....	127
Εικόνα 6.3 Απόσπασμα Κώδικα από αρχείο package.json.....	128
Εικόνα 7.1 Στάδια από την Ανάλυση ως και την Υλοποίηση.....	131
Εικόνα 6.4 CheapGames logo.....	144
Εικόνα 6.5 Απόσπασμα Από κώδικα αρχείου App.js.....	145

## Κατάλογος Πινάκων

<u>Πίνακας 2.1 Οι τέσσερις κύριες κατηγορίες συστημάτων NoSQL και δείγμα... προϊόντων για κάθε τύπο αποθήκευσης δεδομένων</u>	13
<u>Πίνακας 4.1 Πρόσβαση και Απαγόρευση Χρηστών σε λειτουργίες</u>	46
<u>Πίνακας 4.2 Οντότητες και Ιδιότητες CheapGames</u>	54
<u>Πίνακας 4.3 Ερωτήσεις για απόφαση επιλογής Embedded ή Reference</u>	57
<u>Πίνακας 4.4 Ομαδοποίηση Περιπτώσεων Χρήσης που αφορούν την Διαχείριση Εποχιακής Εκπτώσης για τις ανάγκες τους διαγράμματος</u>	65
<u>Πίνακας 7.1 Συμπεράσματα από υλοποίηση backend και frontend</u>	133
<u>Πίνακας 7.2 Κριτήρια Επιλογής Στοίβας</u>	134
<u>Πίνακας Α.1 Καθορισμός EndPoints</u>	138
<u>Πίνακας Β.1 Φάκελοι και Αρχεία Frontend με Περιγραφές</u>	144



## **Συντομογραφίες και Ακρωνύμια**

PIX	Περίπτωση Χρήσης
ΜΟΣ	Μοντέλο Οντοτήτων-Συσχετίσεων
API	Application Programming Interface
BSON	Binary Json
CRUD	Create, Read, Update, Delete
DBMS	Database Management System
DOM	Document Object Model
FSWD	Full Stack Web Development
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
ID	Identifier(αναγνωριστικό)
JSON	JavaScript Object Notation
JSX	JavaScript XML
ORM	Object-Relational Mapping
ODM	Object-Document Mapping
SQL	Structured Query Language
NoSQL	Not Only SQL
MERN	MongoDB, Express.js, React.js, Node.js
PERN	PostgreSQL, Express.js, React.js, Node.js
JWT	JSON Web Token
REST	Representational State Transfer
SPA	Single Page Application
UI	User Interface

## 1. Εισαγωγή

### 1.1 Full Stack Web Development

Το Full Stack Web Development (FSWD) ως όρος αναφέρεται στην περίπτωση όπου κάποιος web developer κατά την κατασκευή μια Web εφαρμογής, εργάζεται στο κομμάτι του κώδικα που αφορά το backend, δηλαδή σε αυτό που δεν βλέπει ο χρήστης της εφαρμογής και συμβαίνει στη μεριά του server αλλά και στον κώδικα που έχει να κάνει με το frontend, δηλαδή στο κώδικα στον οποίο ο χρήστης έχει πρόσβαση (Singh et al., 2022, p. 3083). Η ανάπτυξη μιας διαδικτυακής εφαρμογής από έναν FSDeveloper περιλαμβάνει τρία επίπεδα, επίπεδο λογικής(backend), επίπεδο παρουσίασης(frontend) και το επίπεδο πληροφορίας (databases) (Singh et al., 2022, p. 3083). Το “stack”(στοίβα) , στον όρο FSWD που περιγράφηκε, αναφέρεται στις γνώσεις συνδυασμού τεχνολογιών που είναι απαραίτητες για την υποστήριξη και κατασκευή μιας Web Εφαρμογής. Στις μέρες μας , λόγω της εξέλιξης του ηλεκτρονικού εμπορίου αλλά και γενικά των ιστοσελίδων, τα εργαλεία και οι τεχνολογίες εξελίσσονται και ανταγωνίζονται ώστε να είναι πιο προσιτές και εύκολες στην εκμάθηση και να χρησιμοποιούνται όλο και από περισσότερους προγραμματιστές. Σε αυτή τη διπλωματική εργασία, θα δούμε τη διαδρομή που ακολουθήθηκε για την κατασκευή ενός ηλεκτρονικού καταστήματος πώλησης ψηφιακών παιχνιδιών με όνομα “CheapGames” με τους συνδυασμούς τεχνολογιών MERN stack και PERN stack χρησιμοποιώντας σύγχρονα εργαλεία ξεκινώντας από την γενική περιγραφή της διαδικτυακής εφαρμογής μέχρι και την υλοποίηση της και καταλήγοντας στα συμπεράσματα για τα κοινά σημεία και διαφορές.

### 1.2 Σκοπός και Στόχοι

Σκοπός της έρευνας αυτής της διπλωματικής εργασίας είναι να αναδείξει τα συγκριτικά χαρακτηριστικά της κατασκευής μια διαδικτυακής εφαρμογής για ένα ηλεκτρονικό κατάστημα με τους συνδυασμούς τεχνολογιών MERN και PERN από τα στάδια της ανάλυσης και της σχεδίασης έως και της υλοποίησης. Η διαδικτυακή εφαρμογή αφορά ένα ηλεκτρονικό κατάστημα πώλησης ψηφιακών παιχνιδιών με όνομα CheapGames όπου οι χρήστες θα μπορούν με την αγορά ενός παιχνιδιού να το κάνουν download στη συσκευή τους.

Πρωταρχικός στόχος της εργασίας αυτής, είναι να παρουσιαστούν με σαφήνεια τα βήματα και ο τρόπος για την υλοποίηση του καταστήματος με τις δύο στίβες απαντώντας στο ερώτημα “πώς μπορώ να το φτιάξω με το κάθε συνδυασμό;”. Ένας ακόμα στόχος είναι η ανάδειξη σύγχρονων εργαλείων και μεθόδων ώστε να δοθούν πρακτικές λύσεις και προσεγγίσεις σε ενδεχόμενα προβλήματα που μπορεί κάποιος να συναντά κατά την κατασκευή μιας διαδικτυακής εφαρμογής. Επίσης θα ήθελα η παρούσα εργασία να αποτελέσει έναν αξιόπιστο οδηγό για την επιλογή μεταξύ των δύο τεχνολογικών στοιβών, βοηθώντας προγραμματιστές και επιχειρήσεις να επιλέξουν εκείνη που ταιριάζει καλύτερα στις απαιτήσεις και τις ανάγκες τους ή αυτή που τους προσφέρει έναν πιο εύκολο δρόμο.

## 2. Θεωρητικό Υπόβαθρο

### 2.1 Βάσεις Δεδομένων

Οι βάσεις δεδομένων είναι μια καλά οργανωμένη σύνθεση ηλεκτρονικών και λογικών δομών με σκοπό την συλλογή και αποθήκευση δεδομένων τα οποία σχετίζονται μεταξύ τους για κάποιο σκοπό (Date, 1990, p. 11). Με πιο “απλά” λόγια μία βάση δεδομένων είναι η αποθήκευση πληροφοριών σε μνήμη-ες υπολογιστικών συστημάτων με λογικό τρόπο. Χρησιμοποιούνται στην καθημερινότητα των ανθρώπων σε όλους τους τομείς και η εξέλιξη τους έχουν καθοριστική σημασία σχεδόν σε όλες τις υπηρεσίες που προσφέρονται ή πωλούνται μέσω του διαδικτύου αλλά και χωρίς αυτό. Ένα παράδειγμα χρήσης τους σε ένα eshop είναι η καταχώρηση των στοιχείων των πελατών , των προϊόντων και των παραγγελιών.

Κάποια από τα πλεονεκτήματα της χρήσης τους είναι η εύκολη καταγραφή των δεδομένων αλλά και η γρήγορη ανάκτηση αυτών. Ένα άλλο πλεονέκτημα είναι η πιο εύκολη οργάνωση και ταξινόμηση των δεδομένων. Σημαντικό θα ήταν να αναφέρουμε και τις επιλογές ασφαλείας, συντήρησης, καθώς και την υψηλού βαθμού ακεραιότητα και αξιοπιστία που μπορούν να προσφέρουν.

Μια βάση δεδομένων ελέγχεται συνήθως από ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS). Αυτό το σύστημα, το DBMS, αποτελείται από τα προγράμματα και τις εφαρμογές που δίνουν δυνατότητα στους χρήστες να δημιουργήσουν, διαχειριστούν και να συντηρήσουν μία ή περισσότερες βάσεις δεδομένων. Μαζί, τα δεδομένα και το DBMS, μαζί με τις εφαρμογές που σχετίζονται με αυτά, αναφέρονται ως ένα σύστημα βάσης δεδομένων, το οποίο συχνά συντομεύεται απλά σε βάση δεδομένων (*Oracle, n.d.*).

Αν αναλογιστούμε την ανάπτυξη της επιστήμης της Πληροφορικής, η οποία ασχολείται με το πώς μπορεί να βελτιωθεί η ταχύτητα και η αποδοτικότητα της συλλογής, επεξεργασίας, αποθήκευσης και ανάκτησης πληροφοριών μπορούμε να αντιληφθούμε πόσο σημαντικό ρόλο παίζουν οι βάσεις δεδομένων.

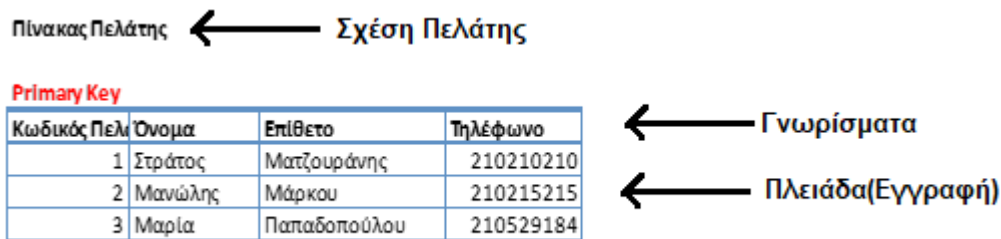
#### 2.1.1 Σχεσιακές Βάσεις δεδομένων

Οι **Σχεσιακές Βάσεις Δεδομένων** κυριάρχησαν τη δεκαετία του 1980 και παραμένουν σήμερα ευρέως χρησιμοποιούμενες, αν όχι οι πιο δημοφιλείς (*Oracle, n.d.*). Στις βάσεις αυτές, τα δεδομένα οργανώνονται ως μια συλλογή από σχέσεις, καθεμία από τις οποίες αντιπροσωπεύει και δίνει πληροφορίες για μια συγκεκριμένη οντότητα ή συμβολική αναπαράσταση της.

**Οντότητα** είναι οτιδήποτε έχει ανεξάρτητη ύπαρξη – μπορεί να είναι κάτι φυσικό, όπως ένας αγοραστής, ή κάτι αφηρημένο, όπως μια παραγγελία. Οποιοδήποτε αντικείμενο ή ιδέα για το οποίο υπάρχει ανάγκη να αποθηκεύουμε δεδομένα, αποτελεί οντότητα. Τα δεδομένα που αποθηκεύουμε για μια οντότητα χωρίζονται σε κατηγορίες ή ιδιότητες, οι οποίες λέγονται γνωρίσματα. Κάθε **γνωρίσμα** είναι μια ξεχωριστή πληροφορία που αφορά την οντότητα.

Από αυτά προκύπτει ότι μια σχέση στη βάση δεδομένων μοιάζει με έναν πίνακα (Εικόνα 2.1) κάθε γραμμή αναπαριστά μια συγκεκριμένη εγγραφή μιας οντότητας η οποία ονομάζεται και

πλειάδα, ενώ κάθε στήλη αναφέρεται σε ένα συγκεκριμένο γνώρισμα. Αυτός είναι και ο λόγος που έχει καθιερωθεί να αναφερόμαστε στις σχέσεις ως πίνακες, και οι Σχεσιακές Βάσεις Δεδομένων συχνά περιγράφονται ως βάσεις που χρησιμοποιούν πίνακες. Κάτι άλλο που είναι σημαντικό να αναφέρουμε είναι ότι κάθε πλειάδα-εγγραφή πρέπει να αναφέρεται σε ένα μοναδικό γεγονός και ταυτόχρονα δεν μπορούν σε μία σχέση δύο ή παραπάνω πλειάδες να αναφέρονται στο ίδιο. Για αυτό και θα πρέπει ένα ή περισσότερα γνώρισμα να επισημαίνονται ως κλειδιά (primary key), δηλώνοντας ότι δεν μπορεί να υπάρχει παραπάνω από μία εγγραφή με την ίδια τιμή στο γνώρισμα αυτό ή γνώρισμα ανάλογα την περίπτωση.



Εικόνα 2.1 Παράδειγμα Πίνακας Πελάτη

## 2.1.2 Μη Σχεσιακές Βάσεις Δεδομένων

Το κύριο χαρακτηριστικό των βάσεων δεδομένων NoSQL (επίσης γνωστών ως Not Only SQL) και των συστημάτων διαχείρισης βάσεων δεδομένων είναι η μη χρήση του μοντέλου RDBMS (Σύστημα Διαχείρισης Βάσεων Δεδομένων Σχέσεων), το οποίο χρησιμοποιείται κυρίως από τα συστήματα στη σημερινή εποχή. Ο όρος NoSQL χρησιμοποιήθηκε για πρώτη φορά το 1998 για μία βάση δεδομένων που δεν χρησιμοποίησε το σχεσιακό μοντέλο και χρησιμοποιήθηκε ξανά το 2009 σε συνέδρια σχετικά με την προώθηση και υποστήριξη μη σχεσιακών βάσεων (Khan et al., 2023). Άρχισαν να εξελίσσονται στα τέλη της δεκαετίας του 2000, κυρίως για να φιλοξενήσουν το διεκταυρνόμενο Διαδίκτυο και τις εφαρμογές που χρειαζόνταν περισσότερη επεξεργαστική ισχύ και μέγεθος.

Οι βάσεις δεδομένων NoSQL χρησιμοποιούν κυρίως μη σχεσιακές μεθόδους οργάνωσης και ανάλυσης δεδομένων αντί πινάκων ή κάποια δομημένη γλώσσα ερωτημάτων (SQL) για διαχείριση δεδομένων. Στην πραγματικότητα είναι κάτι παραπάνω από έναν πίνακα με γραμμές και στήλες. Τα δεδομένα μπορούν να αποθηκεύονται σε διαφορετικούς τύπους βάσεων δεδομένων ανάλογα την περίπτωση και τις ανάγκες και το προϊόν που χρησιμοποιείται. Στον πίνακα 1 βλέπουμε τις τέσσερις κύριες μορφές με τις οποίες μπορούν να αποθηκευτούν τα δεδομένα σε μία NoSQL βάση δεδομένων μαζί με κάποια προϊόντα NoSQL που προσφέρουν αυτές τις υπηρεσίες.

Τύπος Βάσης Δεδομένων	Παραδείγματα τυπικής χρήσης	Προϊόντα Που προσφέρουν αυτούς τύπους NoSql
<b>Key-value stores</b> —Ένα απλό σύστημα αποθήκευσης δεδομένων που χρησιμοποιεί ένα κλειδί για πρόσβαση σε μία τιμή	<ul style="list-style-type: none"> <li>• Εικόνες</li> <li>• Συστήματα αρχείων που βασίζονται σε κλειδιά</li> <li>• Προσωρινή μνήμη αντικειμένων</li> </ul>	<ul style="list-style-type: none"> <li>• Berkeley DB</li> <li>• Memcache</li> <li>• Redis</li> <li>• Riak</li> <li>• DynamoDB</li> </ul>
<b>Column family store</b> —Μπορούν να καταναμηθούν σε πολλούς servers, καθώς κατασκευάστηκαν για να διαχειρίζονται μεγάλους όγκους δεδομένων. Παρόμοια με τα key value stores, ορισμένα κλειδιά σε αυτήν την περίπτωση στοχεύουν πολλά στοιχεία. Εδώ, οι στήλες είναι ταξινομημένες σε οικογένειες στηλών και οι σειρές αναγνωρίζονται από ένα διακριτό κλειδί γραμμής.	<ul style="list-style-type: none"> <li>• Για αποθήκευση αποτελεσμάτων προγράμματος ανίχνευσης ιστού</li> <li>• Για αποθήκευση μεγάλου όγκου δεδομένων</li> </ul>	<ul style="list-style-type: none"> <li>• Apache HBase</li> <li>• Apache Cassandra</li> <li>• Hypertable</li> <li>• Apache Accumulo</li> </ul>
<b>Graph store</b> —Οι κόμβοι, οι συνδέσεις τους και τα χαρακτηριστικά τους αποτελούν τη βάση του. Ακολουθεί ένα ευέλικτο μοντέλο ενός γραφικού (μοντέλο γραφήματος) που μπορεί να χρησιμοποιηθεί και να δημιουργηθεί παράλληλα σε πολυάριθμα μηχανήματα (διακομιστές – κόμβοι) στη θέση πινάκων με στήλες και σειρές.	<ul style="list-style-type: none"> <li>• Κοινωνικά δίκτυα</li> <li>• Ανίχνευση απάτης</li> <li>• Ανάγκη για Υπαρξη Σχέσεων</li> </ul>	<ul style="list-style-type: none"> <li>• Neo4j</li> <li>• AllegroGraph</li> <li>• Bigdata (RDF data store)</li> <li>• InfiniteGraph (Objectivity)</li> </ul>
<b>Document Store</b> —Η δομή με την οποία αποθηκεύονται τα δεδομένα ακολουθούν κάποια ιεραρχία ανάλογα την περίπτωση. Τα δεδομένα αποθηκεύονται σε "συλλογές" δεδομένων με τιμή κλειδιού.	<ul style="list-style-type: none"> <li>• Δεδομένα υψηλής μεταβλητότητας</li> <li>• Αναζήτηση εγγράφων</li> <li>• Διαχείριση περιεχομένου Ιστού</li> <li>• Τα δεδομένα αφορούν δημοσιεύσεις</li> </ul>	<ul style="list-style-type: none"> <li>• MongoDB (10Gen)</li> <li>• CouchDB</li> <li>• Couchbase</li> <li>• MarkLogic</li> <li>• eXist-db</li> <li>• Berkeley DB XML</li> </ul>

Πίνακας 2.1 Οι τέσσερις κύριες κατηγορίες συστημάτων NoSQL και δείγμα προϊόντων για κάθε τύπο αποθήκευσης δεδομένων

Επίσης οι Nosql μπορούν να δουλέψουν σε όλους τους επεξεργαστές και μπορούν να δώσουν υψηλές αποδόσεις ταχύτητας καθώς και μια “συνεπή” αύξηση της απόδοσης και της ταχύτητας όταν αυτοί αυξάνονται.

## **2.2. MERN STACK και PERN STACK**

### **2.2.1 ΣΥΝΔΥΑΣΜΟΣ ΤΕΧΝΟΛΟΓΙΩΝ-STACK**

Κατά την ανάπτυξη διαδικτυακών εφαρμογών μπορούν να χρησιμοποιηθούν διάφορες τεχνολογίες. Αρχικά ο όρος “stack” έγινε γνωστός από την στοίβα LAMP το οποίο αποτελεί συντομογραφία των τεχνολογιών Linux, APACHE, MySQL και PHP και αναφέρεται στον συνδυασμό αυτών των τεχνολογιών (Subramanian, 2017, p. 1).

Η εξέλιξη του Παγκόσμιου Ιστού και η ανάγκη για αλληλεπίδραση μεταξύ χρήστη και ιστοσελίδας οδήγησαν στην ανάγκη για εξέλιξη του τρόπου που “χτίζεται” μια ιστοσελίδα. Ένα κομμάτι αυτής της εξέλιξης είναι τα Single Page Application (SPAs) στα οποία κύριο χαρακτηριστικό είναι η μη ανάγκη αποστολής κάθε φορά όλου του περιεχομένου από τον server στον χρήστη αλλά μόνο οι απαραίτητες πληροφορίες απλουστεύοντας κατά κάποιο τρόπο και διατίθοντας τον όγκο δεδομένων που μπορεί να σταλθούν σε άλλο σκοπό όπως η αλληλεπίδραση μεταξύ χρήστη και ιστοσελίδας (Subramanian, 2017, p. 1). . Με πιο απλά λόγια, σε ένα SPA ο χρήστης λαμβάνει αρχικά τον κώδικα με το περιεχόμενο της ιστοσελίδας, αλλά σε κάθε ενέργεια του χρήστη δεν είναι απαραίτητο να σταλεί ξανά όλο το περιεχόμενο, παρά μόνο αυτά που έχουν οριστεί για αυτή την ενέργεια.

Η ανάγκη ή αλλιώς στόχος για αλληλεπίδραση μεταξύ χρήστη και ιστοσελίδας προϋπέθετε την δημιουργία αρκετών ενεργειών, οι οποίες θα έπρεπε να συμβαίνουν και στην μεριά του χρήστη-πελάτη(client) και όχι αποκλειστικά στην μεριά του server. Αυτό επέφερε και μία εξέλιξη στην δημιουργία των frontend των ιστοσελίδων με χρήση frontend-frameworks, δηλαδή έτοιμα “πλαίσια-τρόποι-δυνατότητες” για να φτιάξει κάποιος την λειτουργία της σελίδας που θα έβλεπε ο τελικός χρήστης στην συσκευή του.

Μαζί με την εξέλιξη όλων των παραπάνω, οι No, Sql βάσεις δεδομένων άρχισαν και αυτές να χρησιμοποιούνται περισσότερο και να αποκτούν δημοφιλία.

Μία στοίβα ή αλλιώς stack η οποία ήταν από τις πρώτες ανοιχτού κώδικα ήταν η στοίβα MEAN όπου και αυτής το όνομα αποτελείται από τα ακρωνύμια των τεχνολογιών που χρησιμοποιούνται. Συγκεκριμένα το ‘M’ αναφέρεται στην βάση δεδομένων που χρησιμοποιείται τη MongoDB, το ‘E’ στον Express Server ο οποίος είναι ένα frame για την κατασκευή του Server σε Node, το ‘A’ στην Angular που είναι ένα frontEnd framework και το ‘N’ στο Node που είναι ένα περιβάλλον για να δουλέψει ο Server.

MERN STACK και PERN STACK είναι λοιπόν δύο συνδυασμοί τεχνολογιών με τις οποίες θα φτιάξουμε ένα site. Στην περίπτωση της MERN STACK συνδυασμός της βάσης δεδομένων που χρησιμοποιείται, MongoDB εξού και το γράμμα ‘M’, του Express Server που αντιστοιχεί στο γράμμα E , του frontEnd framework που χρησιμοποιείται και είναι η ReactJs και αντιστοιχεί στο γράμμα R και του περιβάλλοντος στο οποίο τρέχει ο server που είναι το NodeJs, ενώ στη περίπτωση της PERN Stack το P αναφέρεται στην σχεσιακή βάση δεδομένων PostgreSQL, το E στον Express Server, το R στο frontend frameWork ReactJs και το N στο περιβάλλον Nodejs. Στις επόμενες υποενότητες θα ακολουθήσει αναφορά στις βάσεις δεδομένων MongoDB και PostgreSQL που χρησιμοποιεί ο κάθε συνδυασμός καθώς και στα κοινά σημεία των δύο τεχνολογιών, Express Server, ReactJs και NodeJs.



### **2.2.2 MongoDB**

Η MongoDB, όπως προαναφέρθηκε στον πίνακα 2.1, είναι NoSql βάση δεδομένων η οποία κυκλοφόρησε για πρώτη φορά το 2009 και έδωσε τη δυνατότητα αποθήκευσης διαφορετικών δεδομένων σε κάθε εγγραφή κάτι που διαφέρει σε μεγάλο βαθμό από τις σχεσιακές βάσεις δεδομένων. Ένα βασικό χαρακτηριστικό είναι το ότι χρησιμοποιεί τη μορφή δεδομένων JSON(Java Script Object Notation) για να λάβει ή να εξάγει δεδομένα ενώ τα αποθηκεύει σε μορφή BSON (Binary jSon) (MongoDB, n.d.).

JSON είναι ένας τύπος δεδομένων ο οποίος χρησιμοποιείται για την αποστολή δεδομένων μεταξύ διαδικτυακών εφαρμογών και συστημάτων. Μπορούμε να πούμε πως το JSON είναι μια μορφή ανταλλαγής δεδομένων η οποία βασίζεται σε κείμενο και έχει σύνταξη παρόμοια με τα αντικείμενα JavaScript αλλά ένα αντικείμενο json δεν μπορεί να δεχτεί συναρτήσεις, τύπο ημερομηνίας και απροσδιόριστο τύπο. Στην Εικόνα 2.2 ακολουθεί ένα παράδειγμα δεδομένων που αναφέρονται στα στοιχεία ενός χρήστη (user) και εμφανίζονται σε μορφή JSON (MongoDB, n.d.).

```
json
{
  "user": {
    "id": 12345,
    "username": "gamer123",
    "email": "gamer123@example.com",
    "isPremiumUser": true,
    "age": 25,
    "favorites": [
      {
        "gameId": 101,
        "title": "Game One",
        "isMultiplayer": false
      },
      {
        "gameId": 102,
        "title": "Game Two",
        "isMultiplayer": true
      }
    ]
  },
  "purchasedGames": [
    {
      "gameId": 201,
      "title": "Game Three",
      "purchaseDate": "2023-08-15",
      "price": 29.99
    },
    {
      "gameId": 202,
      "title": "Game Four",
      "purchaseDate": "2023-09-10",
      "price": 49.99
    }
  ]
}
```

#### Χαρακτηριστικά JSON:

- Τα αντικείμενα JSON είναι γραμμένα μέσα σε αγκύλες { }.
- Κάθε στοιχείο είναι ένα ζεύγος κλειδιού και τιμής.
- Τα κλειδιά όπως "id" και οι τιμές του τύπου string γράφονται μέσα σε διπλά εισαγωγικά " " ενώ άλλοι τύποι δεδομένων όπως integer και boolean δεν χρειάζεται να γράφονται σε εισαγωγικά.
- Κάθε στοιχείο διαχωρίζεται από το επόμενο χρησιμοποιώντας κόμμα (.). Δεν υπάρχει κόμμα μετά το τελευταίο στοιχείο.
- Οι πίνακες μέσα σε συμβολοσειρές JSON είναι γραμμένοι εντός αγκυλών [ ] .
- Τα αντικείμενα και οι πίνακες μπορούν να ενσωματωθούν σε ένα αντικείμενο

Εικόνα 2.2 Παράδειγμα JSON

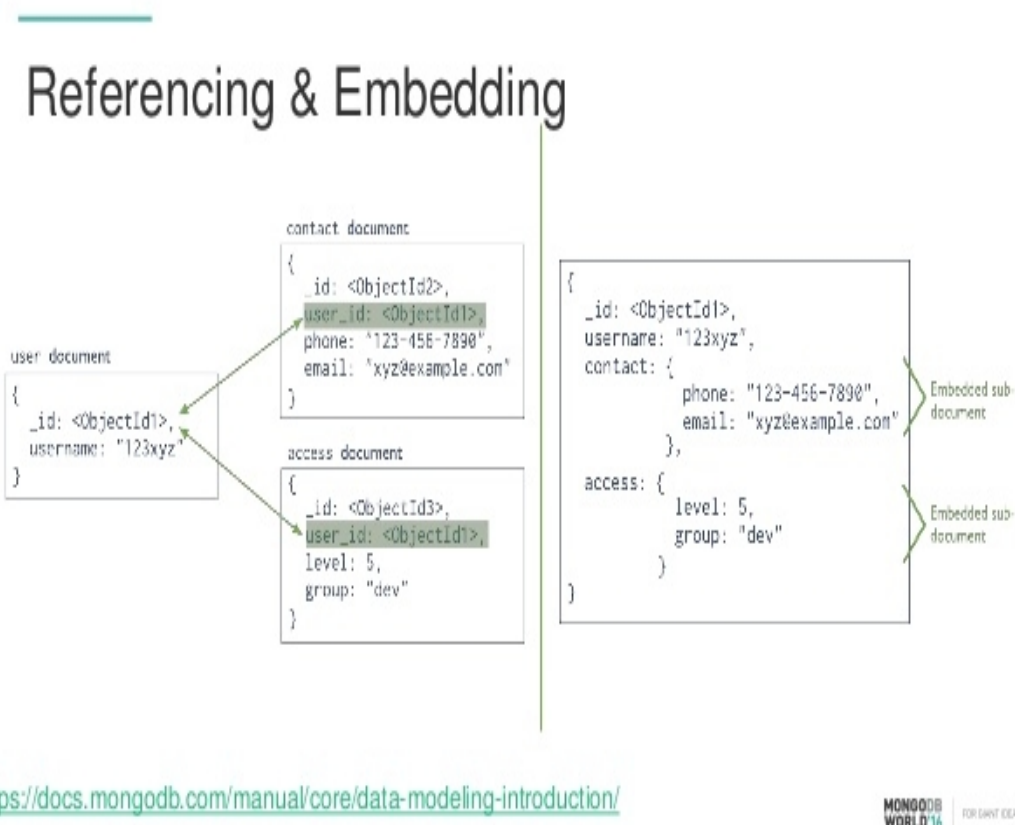
Η μορφή JSON είναι σημαντική να αναφερθεί στη MongoDB καθώς τα δεδομένα μπορούν να σταλούν σε μία βάση Mongo ή να ανακτηθούν σε αυτή την μορφή.

Ωστόσο η MongoDB θα αποθηκεύσει τα δεδομένα σε μορφή BSON (Binary Json) η οποία αποτελεί την δυαδική έκδοση του JSON, ή πιο απλά μια κωδικοποίηση των δεδομένων json. Η αποθήκευση σε BSON δίνει πλεονέκτημα σε ταχύτητα στις αναζητήσεις και στην ανάκτηση των δεδομένων δίνοντας ταυτόχρονα αξιοποιήσιμο χρόνο και για άλλες λειτουργίες-επεξεργασίες των δεδομένων αυτών στους προγραμματιστές καθώς επίσης χρησιμοποιεί και λιγότερο χώρο.

Ένα πολύ σημαντικό χαρακτηριστικό της MongoDB είναι το ότι στη MongoDB δεν



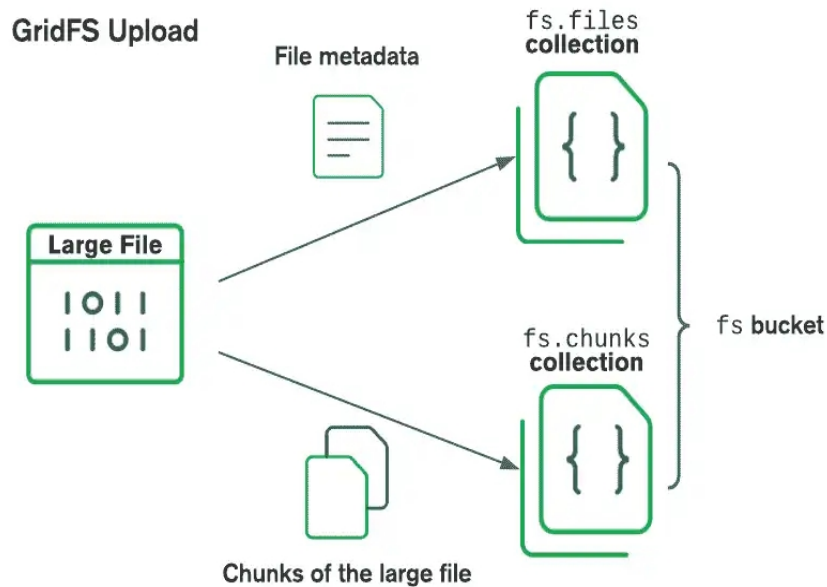
αναφερόμαστε σε έγγραφές και πίνακες όπως στην περίπτωση των σχεσιακών Βάσεων Δεδομένων αλλά σε έγγραφα (documents) και συλλογές (collection) αντίστοιχα. Το παράδειγμα της εικόνας 2.2 θα μπορούσε να αποτελέσει ένα έγγραφο (document) το οποίο αναφέρεται σε έναν χρήστη (user), έχει ένα id (objectId) και όπως προαναφέρθηκε στέλνεται ή ανακτάται από τη βάση σε μορφή Json. Το collection(συλλογή) users θα μπορούσε να είχε όλα τα έγγραφα που αναφέρονται ξεχωριστά σε κάθε χρήστη. Επίσης οι σχέσεις υλοποιούνται μέσω αναφορών(references) ή υποεγγράφων (embedded document). Η αναφορά (reference) υλοποιείται μέσω της σύνδεσης ενός πεδίου ενός εγγράφου με κάποιο άλλο έγγραφο ενός άλλου collection μέσω του objectId του ενώ ένα υποέγγραφο (embedded document) θα είναι μέρος ενός άλλου εγγράφου (Εικόνα 2.3).



**Εικόνα 2.3** Reference και Embedding subdocument(Bernier, A. (n.d.) Document Structure. Available at: <https://hackmd.io/@abernier/S1LLr0uaH> (Accessed: 22-2-2025))

Ένα άλλο χαρακτηριστικό για το οποίο θα πρέπει να αναφερθούν στοιχεία είναι η αποθήκευση αρχείων στη MongoDB. Η MongoDB υποστηρίζει την αποθήκευση αρχείων μέσω του μηχανισμού GridFS. Το GridFs χρησιμοποιείται για την αποθήκευση και ανάκτηση αρχείων που υπερβαίνουν το μέγιστο όριο μεγέθους ενός εγγράφου (16MB). Κατά τη τεχνολογία GridFs, το κάθε αρχείο διασπάται σε μικρότερα πακέτα (chunks), συνήθως 255KB το καθένα (μπορεί και να ρυθμιστεί και διαφορετικό μέγεθος), τα οποία αποθηκεύονται ως ξεχωριστά έγγραφα στη συλλογή chunks. Το κάθε πακέτο έχει μία

αναφορά (reference) προς το objectId ενός έγγραφου το οποίο είναι μοναδικό στη collection files καθώς δεν ένα πακέτο δεν θα μπορούσε να ανήκει σε δύο αρχεία (Εικόνα 2.4). Αυτή η αναφορά λειτουργεί ως δείκτης για να “γνωρίζει” η βάση σε ποιο αρχείο ανήκει το κάθε πακέτο (MongoDB Inc., n.d.; Chodorow, 2013, pp.123-126). Η ομάδα πακέτων που ανήκει σε κάθε αρχείο ονομάζεται bucket (MongoDB Inc.).



**Εικόνα 2.4** GridFS – Ο μηχανισμός αποθήκευσης αρχείων στη MongoDB  
(MongoDB Inc., n.d.) Available at: <https://www.mongodb.com/docs/drivers/node/current/fundamentals/gridfs/>  
(Accessed: 22-2-2025)

### 2.2.3 PostgreSQL

Η PostgreSQL είναι μία βάση δεδομένων η οποία αρχικά ξεκίνησε να δημιουργείται με επικεφαλής τον καθηγητή Michael Stonebraker και χρηματοδότηση από την Υπηρεσία Προηγμένων Ερευνητικών Προγραμμάτων Άμυνας (DARPA), το Γραφείο Ερευνών Στρατού (ARO), την Εθνική Επιστήμη Foundation (NSF) και ESL ενώ ξεκίνησε να χρησιμοποιείται το 1986. Από τότε έχουν γίνει αρκετές βελτιώσεις και ενημερώσεις. (PostgreSQL 16.5 Documentation-The PostgreSQL Global Development Group).

Όπως προαναφέρθηκε και σε προηγούμενο κεφάλαιο η PostgreSQL συγκαταλέγεται στις σχεσιακές βάσεις δεδομένων. Χρησιμοποιείται ως ORDBMS (object relational database management system) για την διαχείριση, αποθήκευση και ανάκτηση των δεδομένων με ασφάλεια ενώ προσφέρει ευελιξία και υψηλή απόδοση για εφαρμογές που χρησιμοποιούνται από μεγάλο αριθμό χρηστών.

Η PostgreSQL υποστηρίζει ερωτοαποκρίσεις (queries) χρησιμοποιώντας την γλώσσα SQL. Επίσης είναι ένα ανοιχτού-κώδικα σύστημα το οποίο επιτρέπει τυχόν αλλαγές και διαμορφώσεις χωρίς περιορισμούς αδειών χρήσης. Η διαχείριση της PostgreSQL μπορεί να πραγματοποιηθεί με δύο βασικούς τρόπους:

- **Μέσω εντολών (commands):** Μπορεί να χρησιμοποιηθεί η γραμμή εντολών του συστήματός ενός υπολογιστή (CMD) ή το περιβάλλον εντολών **psql** το οποίο διατίθεται μαζί με τη PostgreSQL. Μέσω αυτών, είναι δυνατή η εκτέλεση ερωτημάτων SQL και γενικά η διαχείριση της βάσης (δημιουργία πίνακα, διαχείριση χρηστών, διαγραφή και εμφάνιση δεδομένων και άλλα)
- **Μέσω του γραφικού περιβάλλοντος PGAdmin:** Το **PGAdmin** είναι ένα γραφικό περιβάλλον μέσω του οποίου ο χρήστης μπορεί να διαχειριστεί τη βάση δεδομένων χωρίς να χρειάζεται να πληκτρολογεί εντολές. Φυσικά χρειάζεται και εδώ η εξοικείωση του χρήστη με το περιβάλλον ενώ έχει και δυνατότητες προβολής στατιστικών σχετικά με τη χρήση, τη ταχύτητα και άλλες πληροφορίες σχετικά με τη διαχείριση της βάσης. Επίσης διατίθεται sql editor για την γραφή του sql ερωτημάτων.

Ένα άλλο χαρακτηριστικό της PostgreSQL στο οποίο θα πρέπει να αναφερθεί είναι η αποθήκευση αρχείων σε μία βάση δεδομένων PostgreSQL.

Οι περισσότερες βάσεις δεδομένων προσφέρουν ένα τρόπο αποθήκευσης ακατέργαστων δεδομένων δηλαδή δεδομένα που δεν ορίζεται το τι αφορούν (πχ δεν ορίζεται ότι αφορούν έναν ακέραιο αριθμό-Integer). Η PostgreSQL προσφέρει τον τύπο Bytea για αποθήκευση τέτοιων δεδομένων. Ένα πεδίο Bytea στη PostgreSQL μπορεί θεωρητικά να αποθηκεύσει οποιοδήποτε μέγεθος αλλά στη πράξη αυτό δεν μπορεί να είναι μεγαλύτερο από 1 GB. Στην περίπτωση που χρειάζεται μεγαλύτερο μέγεθος από 1 GB μπορεί να χρησιμοποιηθεί ο τύπος LO (Large Objects) (Douglas & Douglas, 2003, pp. 103-104).

Η επιλογή ανάμεσα στους δύο τύπους έχει να κάνει με τις ανάγκες του έργου και παίζει ρόλο και σε περαιτέρω επιλογές όπως η επιλογή ORM το οποίο αναλύεται στο κεφάλαιο 2.3.2.

## 2.3.4 Express Server

Ο Express Server σύμφωνα με αρκετές πηγές στο διαδίκτυο, είναι το πιο δημοφιλές framework για την δημιουργία ενός server στην πλατφόρμα Node.js και επίσης αποτελεί την βάση για άλλα frameworks. Διαφορετικά θα μπορούσαμε να το περιγράψουμε και ως ένα απλό και ευέλικτο πλαίσιο εφαρμογής web node.js, που προσφέρει ένα ισχυρό σύνολο δυνατοτήτων για τη δημιουργία μονόφυλλων και πολυσελίδων site (Brown, 2014, p. 2-3). Ο λόγος που επισημαίνεται ως ευέλικτο είναι το ότι προσφέρει την δυνατότητα να προσθαφαιρούμε λειτουργίες οι οποίες μπορεί να είναι ή να μην είναι απαραίτητες για την δημιουργία του site ανάλογα πάντα την περίπτωση (Brown, 2014, p. 2-3). Αποτελεί μία πολύ καλή βάση και η παραπάνω ευελιξία δίνει και το χαρακτηριστικό της επεκτασιμότητας. Το Express είναι ένας λόγος που ένας συνδυασμός MERN ή PERN θεωρείται εξαιρετική επιλογή για εφαρμογές μιας σελίδας (SPA) , καθώς μπορεί αποτελεσματικά και προσαρμόσιμα να δρομολογεί αιτήματα πελατών στον server χωρίς να χρειάζεται να ανανεώνεται η σελίδα κάθε φορά. Όπως έχει προαναφερθεί στο κεφάλαιο 3.1, οι servers στις εφαρμογές μιας σελίδας SPA, στέλνουν όλο το κώδικα κατά την πρώτη φόρτωση στους clients οι οποίοι συνδέονται σε ένα ιστότοπο και στη συνέχεια στέλνουν τα απαραίτητα

δεδομένα κάθε φορά στον client, μειώνοντας τον όγκο των δεδομένων που στέλνονται αλλά και πιθανό φόρτο στον server αφού δεν χρειάζεται να στέλνουν όλη τη σελίδα κάθε φορά. Όλα αυτά όμως δεν απαγορεύουν την χρήση του express και για πολυσέλιδες διαδικτυακές εφαρμογές, καθώς οι βιβλιοθήκες που παρέχονται παραμένουν σημαντικό πλεονέκτημα.

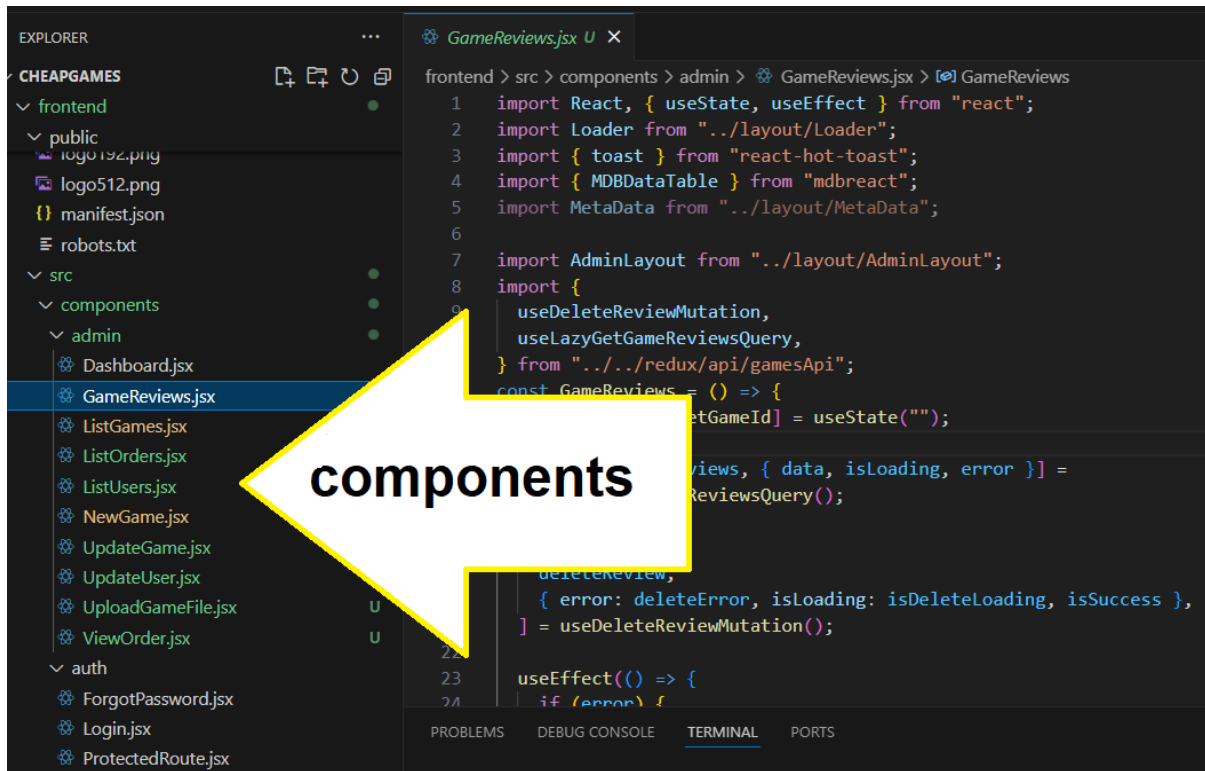
## 2.2.5 React.js

Η React js είναι η τεχνολογία που αφορά το frontEnd και αντιστοιχεί στο τρίτο γράμμα του ακρωνύμιου MERN και του ακρωνύμιου PERN. Δημιουργός της ήταν ο Jordan Walke ένας εργαζόμενος της Meta και αρχικά χρησιμοποιήθηκε στη Διαδικτυακή εφαρμογή Facebook το 2011 ενώ αργότερα ενσωματώθηκε και στο Instagram. Στις μέρες μας χρησιμοποιείται παντού σε μεγάλα και μικρά project διαδικτυακών εφαρμογών.

Στην ουσία, η React.js είναι μία ανοιχτού κώδικα βιβλιοθήκη η οποία βασίζεται στη Javascript και χρησιμοποιείται για να χτιστεί η εμφάνιση μιας διαδικτυακής Εφαρμογής, δηλαδή το frontend. Στα αρχικά της στάδια, η χρήση της React για την κατασκευή του UI δεν ήταν τόσο απλή, καθώς απαιτούσε τη ρύθμιση και διαχείριση διάφορων ανεξάρτητων κομματιών κώδικα. Ωστόσο, σήμερα, η διαδικασία έχει απλοποιηθεί σημαντικά, με εργαλεία όπως το create-react-app. Με μία μόνο εντολή εγκαθίστανται αυτόματα όλα τα απαραίτητα πακέτα Javascript και ρυθμίσεις, προσφέροντας ένα ολοκληρωμένο περιβάλλον εργασίας για την ανάπτυξη εφαρμογών με React και δίνει τη δυνατότητα να ασχοληθούμε με το UI και την εμφάνιση. Επίσης ένα χαρακτηριστικό το οποίο κάνει την React μια δυνατή επιλογή για το frontend, είναι η δυνατότητα να ανανεώνουμε αυτό που φτιάχνουμε αυτόματα όταν κάνουμε τυχόν αλλαγές χωρίς να είναι απαραίτητο να επανεκκινούμε το σύστημα κάθε φορά από την αρχή (Subramanian, 2017, p.5). Σε συνδυασμό με την δομημένη και εύκολη διαχείριση και μετακίνηση μεταξύ των αρχείων του κωδικα, θα λέγαμε ότι κατά τη διάρκεια της γραφής του κώδικα, μπορούμε να διαμορφώνουμε αυτό που φτιάχνουμε και να δοκιμάζουμε σχετικά εύκολα διαφορετικές ιδέες ενώ ένα σύστημα είναι σε λειτουργία βλέποντας το αποτέλεσμα με απλές ανανεώσεις και ενημερώσεις του κώδικα .

Ο λόγος για την παραπάνω ταχύτητα και ευελιξία κατά το στάδιο της γραφής του κώδικα, οφείλεται στο ότι η React μπορεί να χρησιμοποιεί ένα εικονικό DOM (Αντικείμενο που περιέχει όλο το αποτέλεσμα), το οποίο ανανεώνει το πραγματικό HTML DOM μόνο με τα στοιχεία που έχουν αλλάξει (Subramanian, 2017, pp. 6-7).

Επίσης για να συμπεράνουμε όλα τα παραπάνω, σημαντικό θα ήταν να εστιάσουμε και στο τρόπο με τον οποίο ο κώδικας χωρίζεται σε κομμάτια-components αρχεία, με το κάθε component να αφορά ένα συγκεκριμένο στοιχείο στη σελίδα (Εικόνα 2.5)



Εικόνα 2.5 Components σε προγραμματιστικό Περιβάλλον Visual Studio Code

Ο κάθε component είναι ένα αρχείο JSX (Javascript XML) το οποίο περιέχει κώδικα Javascript σε συνδυασμό με HTML. Φυσικά υπάρχουν και άλλοι τύποι αρχείων που μπορούν ή πρέπει να χρησιμοποιηθούν αλλά ως επί των πλείστων, ο περισσότερος κώδικας είναι JSX και μπορεί να αντιστοιχεί σε ένα στοιχείο της σελίδας του site, για παράδειγμα τη κεφαλίδα ή το μενού που ενδεχομένως να υπάρχουν σε μία σελίδα.

## 2.2.6 Node.js

Το Node.js είναι ένα δωρεάν περιβάλλον JavaScript runtime, ανοιχτού κώδικα, που επιτρέπει στους προγραμματιστές να δημιουργούν διακομιστές, εφαρμογές ιστού, εργαλεία γραμμής εντολών και σενάρια (Node.js, n.d). Αναφέροντας ότι το Nodejs είναι ένα runtime javascript εργαλείο εννοούμε πως δίνεται η δυνατότητα ασύγχρονα ή συγχρονισμένα να εκτελούνται προγράμματα στην μεριά του server.

Έχει πολλά κοινά με άλλες πλατφόρμες που χρησιμοποιούνται για την δημιουργία server όπως οι Microsoft's Internet Information Services (IIS) ή η Apache αλλά ωστόσο υπάρχουν διαφορές. Θα μπορούσαμε να πούμε πως το δυνατό χαρτί του Node είναι η χρήση του Express Server και η απλή χρήση του συνδυασμού αυτών των δύο χωρίς να στερείται σε τίποτα σε υψηλή απόδοση, λειτουργίες και δυνατότητες. Ένα άλλο χαρακτηριστικό στο οποίο το Nodejs διαφέρει από άλλες πλατφόρμες είναι το ότι είναι single-threaded, δηλαδή έχει μια κύρια "γραμμή εκτέλεσης" ενώ οι άλλες πιο παραδοσιακές πλατφόρμες είναι multi-threaded (πολυνηματικοί), πράγμα που σημαίνει ότι μπορούν να εκτελούν πολλαπλές εργασίες ταυτόχρονα σε πολλές γραμμές εκτέλεσης (Subramanian, 2017, p. 8). Αυτό ακούγεται ως κάτι αρνητικό αλλά υπάρχουν περιπτώσεις που μειώνουν την πολυπλοκότητα



καθώς το multi-threaded μπορεί να απαιτεί τη λύση πολύπλοκων προβλημάτων όσον αφορά την ταυτόχρονη εκτέλεση εργασιών από τον server. Πχ στο περιβάλλον Node μπορούμε να χρησιμοποιούμε promises και callbacks για να προγραμματίζουμε ασύγχρονα, δηλαδή να έχουμε μέρη προγράμματος τα οποία θα εκτελούνται ταυτόχρονα με όλες τις άλλες λειτουργίες σε μία γραμμή λειτουργίας όλων. Δεν χρειάζεται να προγραμματίζουμε πολλές γραμμές λειτουργίας. Είναι σημαντικό όμως να αναφερθεί ότι σε περίπτωση που η αξιοποίηση του υλικού παίζει σημαντικό ρόλο, οι πιο παραδοσιακές πλατφόρμες προσφέρουν καλύτερες αλλά και πιο πολύπλοκες όπως προαναφέρθηκε λύσεις. Δηλαδή ανάλογα την περίπτωση και τις απαιτήσεις αξιοποίησης των πόρων του συστήματος σε καταμερισμό εργασιών, έχουμε να επιλέξουμε ή σε μια πάρα πολύ έξυπνη, απλή και “ευφύεστατη” πλατφόρμα όπως το Node η οποία στην περίπτωση των προγραμματιστών θα προσφέρει λιγότερο πολύπλοκα προβλήματα ή στις παραδοσιακές πλατφόρμες οι οποίες όμως σε τέτοιες περιπτώσεις απαιτούν σε υψηλότερο βαθμό εξειδίκευση, χρόνο και εργασία.

## 2.3 ODMS ΚΑΙ ORMS

### 2.3.1 Object-Data Modeling (ODM):Mongoose

Τα ODMs (Object Data Modeling) είναι βιβλιοθήκες που επιτρέπουν στον προγραμματιστή να δημιουργεί και να διαχειρίζεται μια μη σχεσιακή βάση δεδομένων, χρησιμοποιώντας τη γλώσσα προγραμματισμού στην οποία αναπτύσσει την εφαρμογή του. Ένα από τα βασικά πλεονεκτήματα των ODMs είναι η ευελιξία που προσφέρουν, καθώς επιτρέπουν στους προγραμματιστές να κάνουν αλλαγές στα μοντέλα δεδομένων και τα σχήματα, χωρίς να απαιτείται άμεση παρέμβαση στη βάση δεδομένων .

Επίσης, τα ODMs δίνουν την δυνατότητα προσθήκης κανόνων επικύρωσης δεδομένων (*validation rules*), όπως:

- Περιορισμός του μήκους των δεδομένων πριν από την αποθήκευσή τους (π.χ., ένα πεδίο να μην ξεπερνά τους 20 χαρακτήρες).
- Ορισμός προκαθορισμένων τιμών (*default values*), μειώνοντας την πολυπλοκότητα και αυξάνοντας την ακεραιότητα των δεδομένων.

Παρόλο που η χρήση ενός ODM δεν είναι απόλυτα απαραίτητη για την ανάπτυξη εφαρμογών με μη σχεσιακές βάσεις δεδομένων, η δημοτικότητά τους είναι ιδιαίτερα αυξημένη.

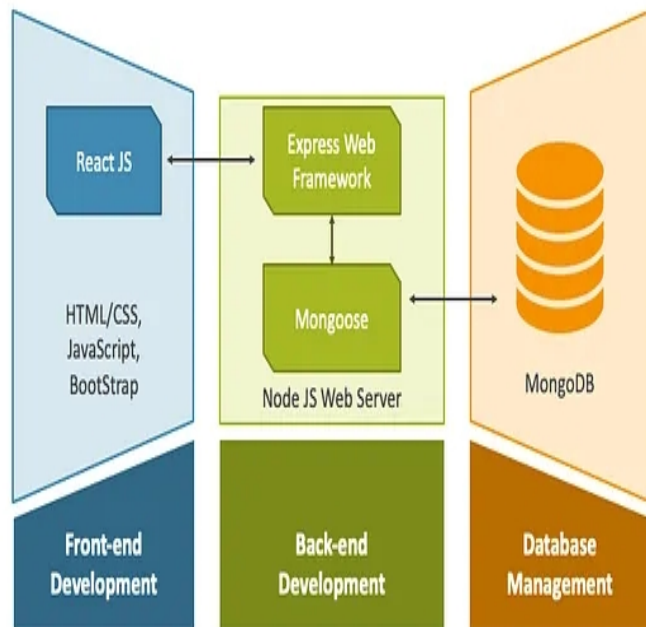
Το Mongoose είναι ένα ODM το οποίο μπορούμε να χρησιμοποιήσουμε με τη MongoDB (Εικόνα 2.6). Σύμφωνα με το επίσημο site της MongoDB, αρκετοί προγραμματιστές επιλέγουν να χρησιμοποιήσουν το Mongoose, καθώς παρέχει δυνατότητες όπως μοντελοποίηση δεδομένων, έλεγχο σχημάτων, επικύρωση μοντέλων και διαχείριση δεδομένων (Hall, 2022). Η αρχιτεκτονική ενός Mern Stack που χρησιμοποιεί mongoose μπορεί να φανεί στην εικόνα 2.7. Μέσω της βιβλιοθήκης γίνεται η σύνδεση του express server με τη βάση δεδομένων και γίνεται η διαχείριση των δεδομένων.



**Εικόνα 2.6** Λογότυπα MongoDB και Mongoose  
(Πηγή: Hall, 2022, <https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/#connecting-to-mongodb>)

## MERN STACK

MERN Stack Development



**Εικόνα 2.7:** Αρχιτεκτονική ενός API με MERN stack και Mongoose(Πηγή: Abhisri, 2023, <https://medium.com/@ashish-k-mishra/getting-started-with-mern-stack-building-your-first-web-application-0bbb6799d251>)

Για να χρησιμοποιήσουμε το ODM Mongoose σε ένα PERN API , θα πρέπει πρώτα να το εγκαταστήσουμε ως Dependency, ανοίγοντας κάποιο τερματικό και μεταφέροντας την “τοποθεσία” στο φάκελο του backend. Στη συνέχεια μέσω του npm μπορούμε να το εγκαταστήσουμε μέσω της εντολής “npm install mongoose” (Mongoose, n.d.). Όπως αναφέρει ο Hall (2022), ένας προγραμματιστής που θα χρησιμοποιήσει τη Mongoose θα πρέπει να ορίσει τη σύνδεση του Server με τη βάση δεδομένων μέσω του Mongoose και να ορίσει σχήματα (schemas) και μοντέλα δεδομένων (models)(Hall, 2022). Σύμφωνα με τον Hall, σχήμα(schema) είναι ο καθορισμός της δομής μιας συλλογής στη βάση δεδομένων και με τα σχήματα ορίζουμε των τύπο των ιδιοτήτων-πεδίων(Hall, 2022). Στην εικόνα 2.8 βλέπουμε ένα παράδειγμα με τον ορισμό ενός σχήματος και τα πεδία με τους τύπους των δεδομένων που το αφορούν.

```
1 const blog = new Schema({
2   title: String,
3   slug: String,
4   published: Boolean,
5   author: String,
6   content: String,
7   tags: [String],
8   createdAt: Date,
9   updatedAt: Date,
10  comments: [{
11    user: String,
12    content: String,
13    votes: Number
14  }]
15 });
```

Εικόνα 2.8 Παράδειγμα Ορισμού Σχήματος και Πεδίων:Τύπων Δεδομένων  
(Πηγή: [What is A schema](#))

Μοντέλο είναι η χρήση ενός σχήματος ως μοντέλο δεδομένων για λειτουργίες CRUD ή άλλες. Στην εικόνα 2.8 για να μπορεί να γίνει χρήση του μοντέλου των δεδομένων, θα πρέπει να προσθέσουμε την γραμμή κώδικα

```
const Blog = mongoose.model('Blog', blog);
```

Στην εικόνα 2.9 βλέπουμε πως θα είναι ο κώδικας για τον ορισμό του σχήματος του παραδείγματος της εικόνας 2.8 αλλά και της εξαγωγής του μοντέλου.

```
1 import mongoose from 'mongoose';
2 const { Schema, model } = mongoose;
3
4 const blogSchema = new Schema({
5   title: String,
6   slug: String,
7   published: Boolean,
8   author: String,
9   content: String,
10  tags: [String],
11  createdAt: Date,
12  updatedAt: Date,
13  comments: [{
14    user: String,
15    content: String,
16    votes: Number
17  }]
18 });
19
20 const Blog = model('Blog', blogSchema);
21 export default Blog;
```

Εικόνα 2.9 Παράδειγμα Ορισμού Σχήματος και εξαγωγή Μοντέλου  
(Πηγή: [Create Schema and Model](#) )

Στη συνέχεια μπορεί να χρησιμοποιήσει το κάθε μοντέλο για να εκτελεί λειτουργίες CRUD



και άλλες μέσω των διαθέσιμων μεθόδων από το Mongoose.

### 2.3.2 Object-Relational Mapping (ORM):Sequelize

Τα **Object-Relational Mapping (ORM)** είναι τεχνικές που διευκολύνουν τον προγραμματισμό διαδικτυακών και τοπικών εφαρμογών. Συγκεκριμένα, δίνουν τη δυνατότητα στους προγραμματιστές να χρησιμοποιούν τη γλώσσα προγραμματισμού με την οποία κατασκευάζουν μια εφαρμογή τόσο για τον ορισμό της δομής όσο και για την επικοινωνία της εφαρμογής τους με μία ή περισσότερες σχεσιακές βάσεις δεδομένων.

Σύμφωνα με τους Ireland et al. (2009), τα ORMs συμβάλλουν στη γεφύρωση του χάσματος μεταξύ του αντικειμενοστραφούς προγραμματισμού και των σχεσιακών βάσεων δεδομένων, εξαλείφοντας την ανάγκη για τη χρήση SQL ερωτημάτων από τους προγραμματιστές κάτι που θα συναντήσουμε και στην περίπτωση της υλοποίησης με PERN stack.

Η βασική λειτουργία των ORMs είναι να παρέχουν έναν μηχανισμό αντιστοίχισης μεταξύ των κλάσεων(οντότητες-αντικείμενα) του κώδικα και των πινάκων της βάσης δεδομένων. Αυτή η αντιστοίχιση μειώνει την πολυπλοκότητα που προκύπτει από το **Object-Relational Impedance Mismatch**—ένα πρόβλημα που ανακύπτει λόγω των διαφορών στον τρόπο αποθήκευσης, επεξεργασίας και αναπαράστασης δεδομένων μεταξύ αντικειμενοστραφών γλωσσών προγραμματισμού και σχεσιακών βάσεων δεδομένων(Ireland et al., 2009).

Ο Barnes (2007), στο πλαίσιο του Honors Project του στο **Macalester College**, επισημαίνει ότι τα ORMs προσφέρουν έναν αυτοματοποιημένο μηχανισμό για τη διαχείριση της αντιστοίχισης αντικειμένων και δεδομένων, μειώνοντας την ανάγκη για επαναλαμβανόμενο και χειροκίνητο κώδικα. Επιπλέον, αναφέρει ότι τα ORMs βοηθούν στην εξάλειψη της διπλής δουλειάς που απαιτείται όταν οι κλάσεις και οι πίνακες χρειάζονται ξεχωριστή συντήρηση.

Εδώ και αρκετά χρόνια τα ORMs εξελίσσονται και σε κάποιες περιπτώσεις διευκολύνουν σε ικανοποιητικό βαθμό την εργασία του προγραμματιστή για αυτό και αρκετοί προγραμματιστές τα χρησιμοποιούν.

Στην υλοποίηση με PERN STACK επιλέγεται να χρησιμοποιηθεί το ORM Sequelize (Εικόνα 2.10).

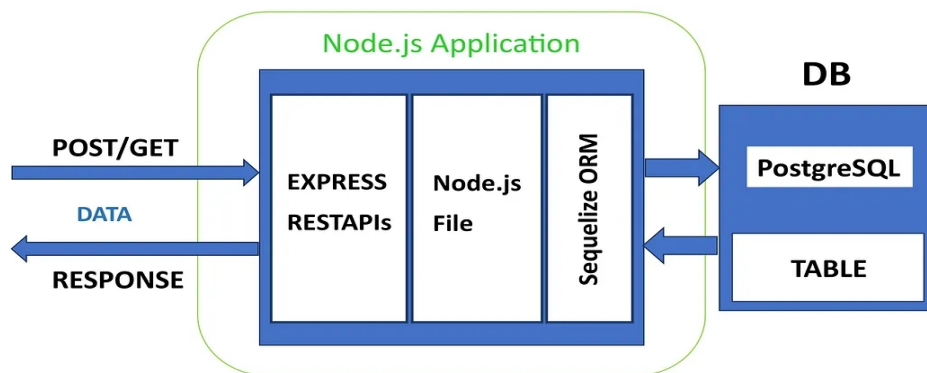


Εικόνα 2.10 Το επίσημο λογότυπο του Sequelize. (Πηγή: <https://sequelize.org> )

Το sequelize είναι ένα διαδεδомένο εργαλείο για σχεσιακές βάσεις το οποίο μας επιτρέπει να διαχειριστούμε τη βάση δεδομένων χρησιμοποιώντας την γλώσσα προγραμματισμού

JavaScript με την οποία γράφουμε τον κώδικα της διαδικτυακής μας εφαρμογής αντι για SQL και μπορεί να εκτελεστεί στο περιβάλλον Node. Σύμφωνα με την τεκμηρίωση του Sequelize (n.d.), τα **queries** (ερωτήματα προς τη βάση) απλοποιούνται μέσω μεθόδων όπως `.findOne()`, `.findAll()`, `.create()`, `.update()`, και `.destroy()`. Επίσης μία βασική δυνατότητα σύμφωνα με την τεκμηρίωση του Sequelize (n.d.) την οποία προσφέρει το ORM Sequelize είναι η επικοινωνία μιας διαδικτυακής εφαρμογής με μία ή και περισσότερες διαφορετικές σχεσιακές βάσεις δεδομένων και συγκεκριμένα με τις Postgres, MySQL, MariaDB, SQLite, Microsoft SQL Server, Oracle Database, Amazon Redshift and Snowflake's Data Cloud, ένα πλεονέκτημα το οποίο στην περίπτωση του Site μας δεν αναδεικνύεται μιας και θα επικοινωνεί με ένα τύπο βάσης Postgres.

Το σχήμα της εικόνα 2.11 είναι αντιπροσωπευτικό του πως αλληλεπιδρά το ORM Sequelize με τα υπόλοιπα μέρη του PERN API. Ο server του οποίου τα περιθώρια καθορίζονται με πράσινη γραμμή, διαχειρίζεται τα αιτήματα Post/Get από τους clients και μέσω του εργαλείου Sequelize επικοινωνεί με την βάση δεδομένων.



Εικόνα 2.11 Αρχιτεκτονική ενός API με PERN stack που χρησιμοποιεί Sequelize ως ORM.  
(Πηγή: Abhisri, 2024, [Medium](#) )

Σύμφωνα με το επίσημο site του Sequelize μπορούμε να διακρίνουμε τα τρία βασικά βήματα, για να το χρησιμοποιήσουμε. Θα πρέπει:

- αρχικά να το εγκαταστήσουμε ως dependency αναφέροντας και τον τύπο της βάσης την οποία χρησιμοποιούμε, αφού συνδεθούμε στο φάκελο με το κώδικα που αφορά το Backend μέσω ενός τερματικού (terminal) χρησιμοποιώντας το npm(node package manager) (Εικόνα 2.12).

## Getting Started

In this tutorial, you will learn to make a simple setup of Sequelize.

### Installing

Sequelize is available via [npm](#) (or [yarn](#)).

```
npm install --save sequelize
```

You'll also have to manually install the driver for your database of choice:

```
# One of the following:  
$ npm install --save pg pg-hstore # Postgres  
$ npm install --save mysql2  
$ npm install --save mariadb  
$ npm install --save sqlite3  
$ npm install --save tedious # Microsoft SQL Server  
$ npm install --save oracledb # Oracle Database
```

**Εικόνα 2.12** Εγκατάσταση του Sequelize μέσω npm  
(Πηγή: [Sequelize Official Documentation](#) )

- Αφού εγκατασταθεί το Sequelize ως dependency για να χρησιμοποιηθεί θα πρέπει να καθοριστεί η σύνδεση του sequelize με τη βάση δεδομένων. Στην εικόνα 2.13 βλέπουμε ένα παράδειγμα των τρόπων σύνδεσης είτε περνώντας κάποιο σύνδεσμο (uri), είτε περνώντας τις παραμέτρους ξεχωριστά.

```
const { Sequelize } = require('sequelize');  
  
// Option 1: Passing a connection URI  
const sequelize = new Sequelize('sqlite::memory:') // Example for sqlite  
const sequelize = new Sequelize('postgres://user:pass@example.com:5432/dbname') // Example for postgres  
  
// Option 2: Passing parameters separately (sqlite)  
const sequelize = new Sequelize({  
  dialect: 'sqlite',  
  storage: 'path/to/database.sqlite'  
});  
  
// Option 3: Passing parameters separately (other dialects)  
const sequelize = new Sequelize('database', 'username', 'password', {  
  host: 'localhost',  
  dialect: /* one of 'mysql' | 'postgres' | 'sqlite' | 'mariadb' | 'mssql' | 'db2' | 'snowflake' | 'oracle' */  
});
```

**Εικόνα 2.13** Παράδειγμα σύνδεσης του sequelize με μία ή περισσότερες βάσεις, όπως παρουσιάζεται στην επίσημη τεκμηρίωση του Sequelize.  
(Πηγή: [Sequelize Official Documentation](#) )

- Να ορίσουμε τα μοντέλα δεδομένων δηλ. να ορίσουμε μία οντότητα με τις ιδιότητες

της η οποία θα πρέπει να αποθηκεύεται με τις ιδιότητες της. Στην εικόνα 2.14 βλέπουμε ένα σύντομο παράδειγμα της οντότητας χρήστη που υπάρχει και στην επίσημη τεκμηρίωση του Sequelize.

## 2. DEFINE MODELS

```
import { Sequelize, DataTypes } from 'sequelize';

const sequelize = new Sequelize('sqlite::memory:');
const User = sequelize.define('User', {
  username: DataTypes.STRING,
  birthday: DataTypes.DATE,
});
```

Defining Models →

**Εικόνα 2.14:** Παράδειγμα Ορισμού Μοντέλου Χρήστη με ιδιότητες username και birthday  
(Πηγή: <https://sequelize.org/>)

- Να ορίσουμε τις σχέσεις μεταξύ των μοντέλων των δεδομένων. Κάθε μοντέλο αντιστοιχεί σε ένα πίνακα. Στην εικόνα 2.15 βλέπουμε ένα σύντομο παράδειγμα καθορισμού δύο μοντέλων και των σχέσεων όπως θα μπορούσαν να οριστούν.

Data Modeling	Associations
Define your models with ease and make optional use of automatic database synchronization.	Define associations between models and let Sequelize handle the heavy lifting.
<pre>const Wishlist = sequelize.define("Wishlist", {   title: DataTypes.STRING, }); const Wish = sequelize.define("Wish", {   title: DataTypes.STRING,   quantity: DataTypes.NUMBER, });  // Automatically create all tables await sequelize.sync();</pre>	<pre>Wish.belongsTo(Wishlist); Wishlist.hasMany(Wish);  const wishlist = await Wishlist.findOne(); const wishes = await wishlist.getWishes(); const wish = await wishlist.createWish({   title: 'Toys', quantity: 3, });  await wishlist.removeWish(wish);</pre>

**Εικόνα 2.15:** Παράδειγμα Ορισμού Μοντέλων WishList και Wish και σχέσεων μεταξύ των δύο μοντέλων  
(Πηγή: <https://sequelize.org/>)

Όταν χρησιμοποιούμε το sequelize ως ORM έχουμε τη δυνατότητα να δημιουργούμε αυτόματα πίνακες κατά την εκκίνηση της διαδικτυακής εφαρμογής και της σύνδεσης του ORM με τη βάση ή να χρησιμοποιούμε ήδη υπάρχοντες πίνακες με ήδη καταχωρημένα δεδομένα (Πηγή: Sequelize, n.d., [New databases versus existing databases](#)). Στην εικόνα 2.15 στο κομμάτι του κώδικα του παραδείγματος για το Data Modeling, βλέπουμε στο τέλος την εντολή “await sequelize.sync()”, μια εντολή που δημιουργεί αυτόματα τους πίνακες στην βάση δεδομένων στην περίπτωση που δεν υπάρχουν.

Ένα άλλο χαρακτηριστικό του Sequelize στο οποίο θα πρέπει να αναφερθούμε είναι ο τύπος δεδομένων που χρησιμοποιείται για τα αρχεία. Το sequelize προσφέρει τον τύπο BLOB για τον ορισμό ενός πεδίου το οποίο αφορά αυθαίρετο μέγεθος (Sequelize Documentation, n.d.)

Κατά τη χρήση του Sequelize με PostgreSQL, όλοι οι τύποι BLOB μετατρέπονται αυτόματα σε BYTEA, καθώς η PostgreSQL δεν υποστηρίζει διαφορετικά μεγέθη BLOB όπως στη MySQL (π.χ., TINYBLOB, LONGBLOB). Αυτό σημαίνει ότι η αποθήκευση αρχείων στη βάση δεδομένων μέσω Sequelize και PostgreSQL πραγματοποιείται αποκλειστικά με τον τύπο BYTEA, ο οποίος επιτρέπει την αποθήκευση δυαδικών δεδομένων ως byte arrays (Supercharging Node.js Applications with Sequelize, 2022, p. 48).

## 2.4 Αρχιτεκτονική Rest

Αρχιτεκτονική Rest (Representational State Transfer) είναι μία αρχιτεκτονική άποψη για την δημιουργία APIS η οποία τα τελευταία χρόνια έχει γίνει αρκετά δημοφιλής. Υπάρχουν και άλλες επιλογές ως προς την αρχιτεκτονική όπως η Soap (Vasan Subramanian, 2017, p.69).

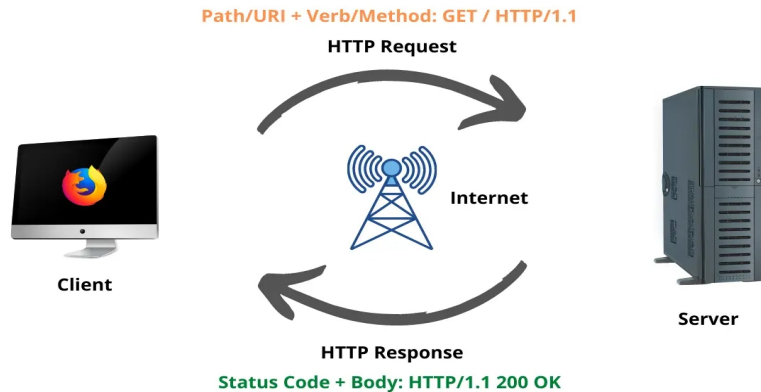
Για να καταλάβουμε καλύτερα το τι είναι η αρχιτεκτονική Rest, θα πρέπει να κάνουμε μια σύντομη αναφορά στο πως λειτουργεί το διαδίκτυο το οποίο χρησιμοποιεί το μοντέλο Πελάτη-Εξυπηρετητή και το πρωτόκολλο HTTP. Όταν ένας Web Browser(client) κάνει ένα HTTP αίτημα(request) για να λάβει δεδομένα, ο server απαντάει με ένα έγγραφο (συνήθως σε μορφή HTML, η με κάποιο Binary Image ή κάτι διαφορετικό) (Richardson & Amundsen, 2013, p. 3).

Κατά την αρχιτεκτονική Rest, κάθε αίτημα από τον client προς τον server αποστέλλεται σε ένα συγκεκριμένο route (διαδρομή). Ο προγραμματιστής ορίζει αυτές τις διευθύνσεις (routes) στον server, όπου "ακούει" για εισερχόμενα αιτήματα και διαχειρίζεται την επικοινωνία. Έτσι, ο client μπορεί να στείλει δεδομένα ή να ζητήσει πληροφορίες, ενώ ο server εκτελεί την αντίστοιχη λειτουργία σύμφωνα με την επιχειρηματική λογική της εφαρμογής.

Κάθε HTTP αίτημα περιλαμβάνει, εκτός από το πρωτόκολλο (π.χ. "http://") και τη βασική διεύθυνση του site (π.χ. www.cheapGames.gr), και μια διαδρομή (route) που, μαζί με τις παραμέτρους του, σχηματίζουν το Uniform Resource Identifier (URI) – συχνά αναφερόμενο και ως "endpoint". Επίσης, το κάθε αίτημα περιέχει και τη μέθοδο καθώς και το κύριο σώμα (body), όπου ενδέχεται να ενσωματώνονται δεδομένα σε κάποια μορφή(πχ. json) προς τον server. Οι 5 από τις μεθόδους τις οποίες θα χρειαστούμε στην συνέχεια και θα μπορούσαμε να χαρακτηρίσουμε πιο σημαντικές σε πρώτο επίπεδο είναι οι :

- **GET:** Η μέθοδος GET χρησιμοποιείται όταν θέλουμε να δώσουμε την δυνατότητα στον client να ανακτήσει δεδομένα από τον server.
- **POST:** Η μέθοδος POST χρησιμοποιείται σε HTTP μηνύματα αίτησης για να δώσουμε τη δυνατότητα στον client να στείλει δεδομένα στον server.
- **PUT:** Η μέθοδος put δίνει τη δυνατότητα στον client να αντικαταστήσει δεδομένα σε αυτά που είναι διαθέσιμα μέσω του link
- **DELETE:** Διαγράφει όλες τις τρέχουσες αναπαραστάσεις των δεδομένων που έχουν δοθεί από ένα URL.
- **PATCH:** Η μέθοδος PATCH χρησιμοποιείται σε HTTP αιτήματα όταν θέλουμε ο client να έχει τη δυνατότητα να ενημερώσει μερικώς ή όλα κάποια δεδομένα.

Η απάντηση(response) του Server θα περιέχει εκτός των άλλων έναν τριψήφιο κωδικό( status Code ο οποίος ανάλογα τα ψηφία από τα οποία αποτελείται, δηλώνει την επιτυχία ή αποτυχία ή άλλες περιπτώσεις και τα δεδομένα(Body) που πρέπει να στείλει ανάλογα την περίπτωση (Εικόνα 2.16).

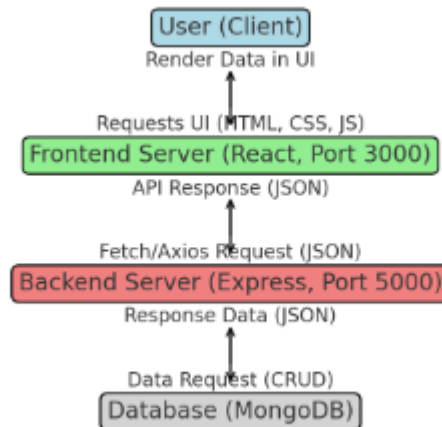


Εικόνα 2.16 Διάγραμμα κύκλου αιτήματος-απόκρισης HTTP (Πηγή: Gitonga, 2022, <https://davisgitonga.dev/blog/request-response-cycle> )

Στην αρχιτεκτονική REST, η πιο διαδεδομένη μορφή ανταλλαγής δεδομένων μεταξύ client και server είναι το JSON (JavaScript Object Notation) το οποίο προανέφερα στο κεφάλαιο 2.2.2 και αυτή θα είναι και η μορφή που θα ανταλλάσσονται δεδομένα όπως πχ. τα στοιχεία ενός χρήστη. Σε ένα MERN STACK ή PERN stack υπάρχουν διαφορετικές πρακτικές ή αλλιώς δρόμοι ως προς τον προγραμματιστικό κώδικα και την δομή αυτού για την εφαρμογή μιας αρχιτεκτονικής REST. Αυτό όμως που είναι σίγουρο είναι το ότι ο Server θα έχει συγκεκριμένα routes για συγκεκριμένες αποστολές και λήψεις δεδομένων.

Για την κατασκευή ενός RestFul Api είτε με συνδυασμό MERN είτε με συνδυασμό PERN, μία κοινή πρακτική είναι ο διαχωρισμός του κώδικα σε δύο φακέλους όπου ο ένας αφορά το backend και ο άλλος το frontend. Συνηθίζονται ονόματα όπως backend και frontend ή server και client. Ο φάκελος backend-server έχει όλο τον κώδικα που αφορά το παρασκήνιο και αποτελείται από άλλους υποφακέλους διαχωρίζοντας σε κατηγορίες ή ομάδες τα αρχεία με τον κώδικα. Συνήθως στο backend για την οργάνωση του, ένας υποφάκελος του backend θα αποτελείται από αρχεία που περιέχουν τα routes που προανέφερα. Σε ένα άλλο υποφάκελο του backend θα περιέχονται τα αρχεία με τα controllers όπου controllers είναι κομμάτια κώδικα με το τι ακριβώς θέλουμε να εκτελείται όταν γίνεται ένα συγκεκριμένο HTTP αίτημα προς κάποιο route πχ. η ανάκτηση δεδομένων από την βάση και η αποστολή προς τον client ή η δημιουργία μιας εγγραφής στη βάση και η αποστολή επιτυχούς επιβεβαίωσης. Αντίστοιχα ο frontend-client έχει και αυτός υποφακέλους με αρχεία-components τα οποία έχουν να κάνουν με αυτό που βλέπει ο χρήστης και το πως αυτό λειτουργεί στη πλευρά του client. Όταν ο χρήστης μπαίνει σε μια σελίδα του site, τότε εκείνη τη στιγμή κάνει ένα αίτημα στον server για να λάβει τον Html κώδικα που καθορίζεται από τον φάκελο frontend-client και ακούει σε συγκεκριμένη θύρα του server. Αυτός ο HTML κώδικας δίνει την δυνατότητα στον χρήστη να κάνει αιτήματα HTTP μέσω της βιβλιοθήκης fetch ή axios προς τον server σε καθορισμένα routes και να λαμβάνει δεδομένα σε μορφή JSON χωρίς κάθε φορά να χρειάζεται να στέλνεται ολόκληρη η σελίδα (Εικόνα 2.17).





Εικόνα 2.17 Διάγραμμα Ροής Δεδομένων σε Mern Stack

## 2.5 Authentication και Authorization

### 2.5.1 Authentication Based On Password

Authentication είναι η διαδικασία επαλήθευσης της ταυτότητας ενός χρήστη ο οποίος κάνει ένα αίτημα request προς τον server. Η απαίτηση αυτή συνήθως εκτελείται απαιτώντας από τον χρήστη να παρέχει τον σωστό κωδικό πρόσβασης του όταν συνδέεται και χρησιμοποιώντας cookies για να θυμούνται τη ταυτότητα του και να τη συσχετίζουν με μεταγενέστερα αιτήματα-requests (Chapman & Chapman, 2019, σελ. 45). Ένας κοινός τρόπος authentication σε μια διαδικτυακή εφαρμογή είναι η χρήση κωδικού (Password Based) και Token Based Authentication.

Κάθε χρήστης κατά την εγγραφή του σε μια διαδικτυακή εφαρμογή, του ζητείται να καταχωρήσει κάποια στοιχεία σε κάποιο φόρμα όπως email, name και μαζί με αυτά και έναν κωδικό τα οποία στέλνονται μέσω ενός αιτήματος Request στο server και αυτός είναι προγραμματισμένος να τα αποθηκεύσει στη βάση δεδομένων.

Η διατήρηση των κωδικών πρόσβασης είναι υψηλή προτεραιότητα, καθώς αποτελούν την κύρια άμυνα κατά της μη εξουσιοδοτημένης πρόσβασης στα περισσότερα συστήματα. Ο κωδικός πρόσβασης είναι μια πληροφορία, η οποία πρέπει να αποθηκεύεται κάπου για να μπορεί να ελέγχεται κάθε φορά που ένας χρήστης προσπαθεί να συνδεθεί. Για να ελαχιστοποιήσουμε τις συνέπειες της μη εξουσιοδοτημένης πρόσβασης σε κωδικούς πρόσβασης, οι κωδικοί πρόσβασης των χρηστών θα πρέπει να αποθηκεύονται σε κρυπτογραφημένη μορφή χρησιμοποιώντας κάποια κρυπτογραφική διαδικασία hash (Chapman & Chapman, 2019, σελ. 45).

Μια από τις πιο γνωστές βιβλιοθήκες που χρησιμοποιούνται για αυτόν τον σκοπό σε περιβάλλον Node.js είναι η bcrypt.js. Η βιβλιοθήκη bcrypt.js χρησιμοποιεί τον αλγόριθμο bcrypt, ο οποίος εφαρμόζει hashing στον κωδικό πρόσβασης του χρήστη, καθιστώντας πολύ δύσκολη την αναστροφή του σε αρχική μορφή ακόμα και σε περίπτωση παραβίασης της

βάσης δεδομένων. Όταν ένας χρήστης εισάγει τον κωδικό του κατά την εγγραφή, ο server εκτελεί έναν αλγόριθμο hashing μέσω της bcrypt.js και αποθηκεύει το αποτέλεσμα στη βάση δεδομένων. Στη συνέχεια, κατά τη διαδικασία σύνδεσης του χρήστη, η βιβλιοθήκη χρησιμοποιείται ξανά για να συγκρίνει τον κατακερματισμένο κωδικό με τον εισαχθέντα από τον χρήστη, προκειμένου να επιβεβαιώσει την ταυτότητά του (Panta et al., 2023).

Η βιβλιοθήκη bcrypt είναι διαθέσιμη στο node μέσω του NPM (NPM, n.d.).

Το Token Based Authentication είναι μια άλλη τεχνική όπου αφού οι χρήστες εισάγουν το κωδικό τους μία φορά και συνδεθούν, λαμβάνουν ως απάντηση ένα μοναδικό, κρυπτογραφημένο διακριτικό (token), το οποίο αποτελείται από τυχαίους χαρακτήρες. Αντί να εισάγουν εκ νέου τα διαπιστευτήριά τους, οι χρήστες μπορούν να χρησιμοποιούν το token για την πρόσβαση σε προστατευμένα συστήματα. Το ψηφιακό αυτό διακριτικό επιβεβαιώνει ότι ο χρήστης έχει ήδη εξουσιοδοτηθεί.(Panta et al., 2023).

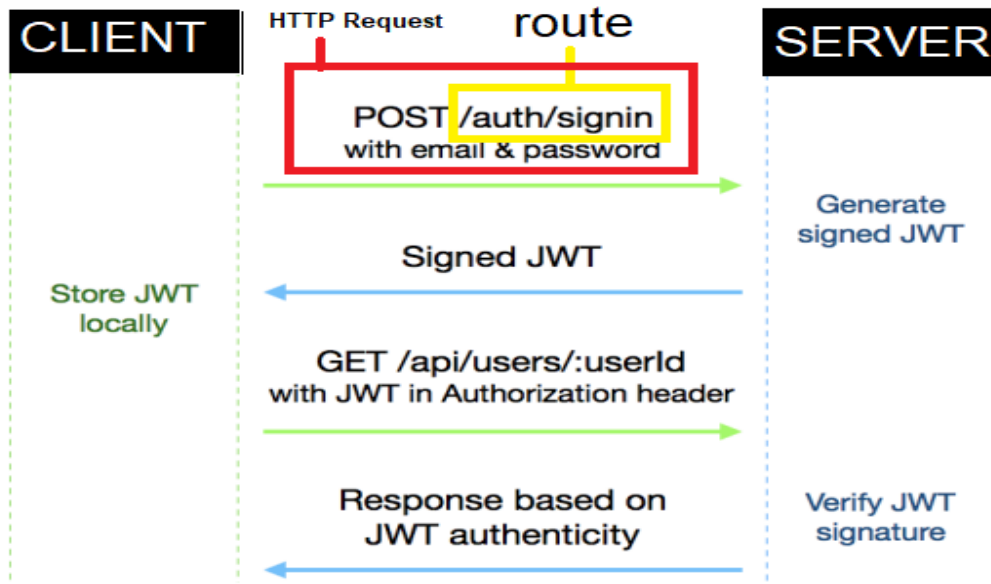
Κατά την υλοποίηση της μεθόδου Token-Based Authentication σε ένα MERN ή PERN API, οι προγραμματιστές μπορούν να εγκαταστήσουν τη βιβλιοθήκη jsonwebtoken (διαθέσιμη μέσω NPM) για τη δημιουργία και διαχείριση JWT (JSON Web Tokens).

Όταν ένας χρήστης εισάγει επιτυχώς τα στοιχεία σύνδεσής του, ο server δημιουργεί ένα JWT, το οποίο περιέχει κωδικοποιημένες πληροφορίες σχετικά με την ταυτότητα του χρήστη και το αποθηκεύει στη Βάση Δεδομένων. Το token επιστρέφεται ως απόκριση και μπορεί να αποθηκευτεί είτε ως HTTP-only cookie είτε στο localStorage/ sessionStorage του browser (Shama, 2018, “How JWT works”).

Κάθε φορά που ο χρήστης πραγματοποιεί ένα αίτημα προς τον server, το JWT που έχει ληφθεί επιτυχώς ως respond προηγουμένως κατά την επιτυχή σύνδεση του, αποστέλλεται μαζί με το request, συνήθως μέσω της Authorization Header ή ως cookie. Ο server, από την πλευρά του, αποκωδικοποιεί και επαληθεύει το token, ελέγχοντας την εγκυρότητά του και επιτρέποντας ή απορρίπτοντας την πρόσβαση σε περιορισμένες λειτουργίες(Shama, 2018, “How JWT works”).

Το JWT έχει περιορισμένη διάρκεια ισχύος, η οποία καθορίζεται από τον προγραμματιστή κατά τη δημιουργία του token. Μετά τη λήξη του, απαιτείται είτε ανανέωση μέσω ενός μηχανισμού refresh tokens, είτε εκ νέου είσοδος του χρήστη. Επιπλέον, ο χρήστης μπορεί να αποσυνδεθεί και να διαγράψει το token από τον browser (cookie ή storage). Ένα παράδειγμα όλων των παραπάνω φαίνεται στο σχήμα της εικόνας 2.18.





Εικόνα 2.18 Token-Based Authentication: Έκδοση και χρήση JWT για προστατευμένα αιτήματα

## 2.5.2 Authorization Based On Role

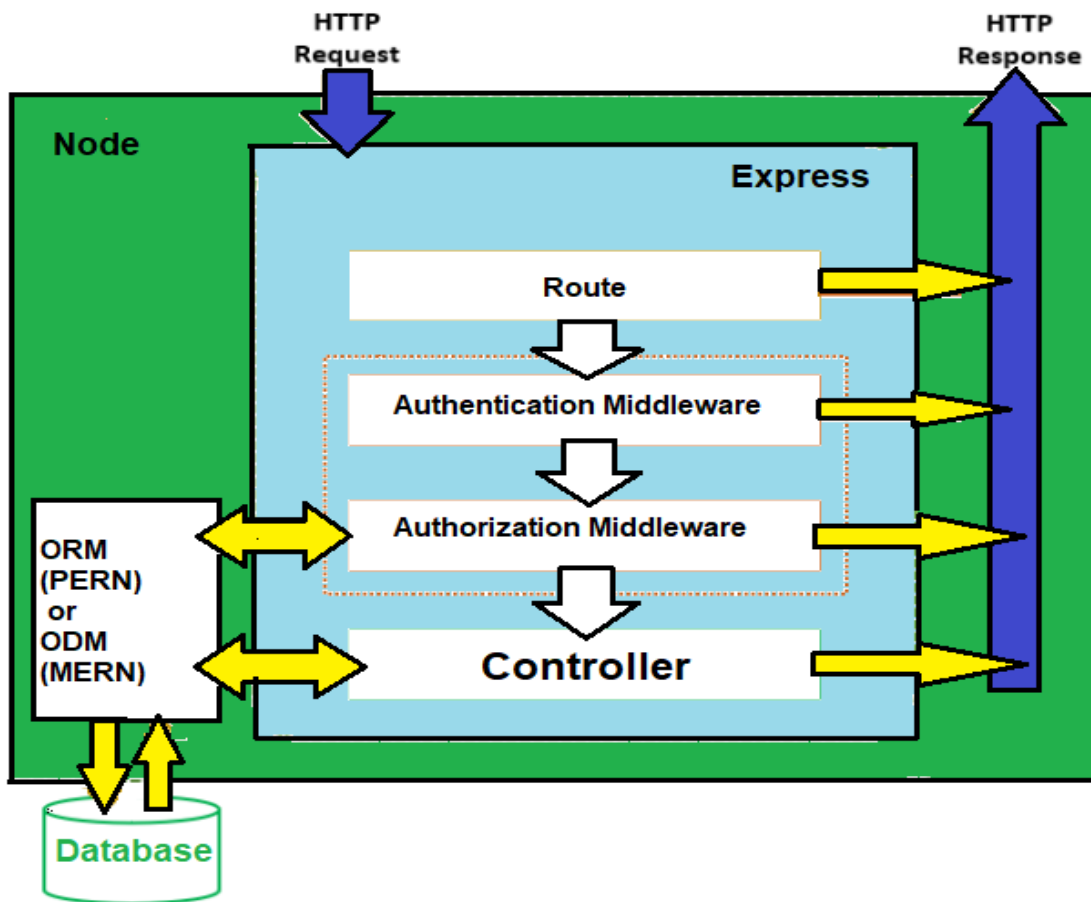
Το authorization Based On Role είναι ένας άλλος τρόπος να προστατεύσουμε τις λειτουργίες και τα δεδομένα μιας διαδικτυακής εφαρμογής από μη εξουσιοδοτημένη πρόσβαση.

Είναι δεδομένο πως στις διαδικτυακές εφαρμογές μπορεί να υπάρχει η ανάγκη για την ύπαρξη μιας κατηγορίας χρηστών οι οποίοι είναι διαχειριστές και έχουν πρόσβαση σε διαφορετικές λειτουργίες από αυτές των απλών χρηστών ή ανάγκη να έχουμε διαφορετικές κατηγορίες χρηστών με διαφορετικές επίπεδα πρόσβασης όπως πχ. premium χρήστες και απλούς χρήστες (Chapman & Chapman, 2012, Role-Based Authorization).

Για να υπάρξει Authorization Based On Role αρχικά θα πρέπει ο χρήστης να έχει κάνει login και να έχει αυθεντικοποιηθεί η πρόσβαση του με κάποιον τρόπο όπως το Authentication based Password. Όπως προανέφερα σε προηγούμενο κεφάλαιο, είναι γεγονός πως σε ένα Rest Api ο χρήστης αλληλεπιδρά με τον Server μέσω HTTP request τα οποία στέλνονται προς συγκεκριμένα routes-διαδρομές του Server ανάλογα την περίπτωση και του τι θέλει να εκτελεστεί. Το Authorization Based On Role είναι στην ουσία η προστασία των λειτουργιών αυτών των routes ώστε να μη μπορέσει ένα χρήστης που ο ρόλος του δεν είναι εξουσιοδοτημένος να μπορέσει να εκτελέσει τις λειτουργίες που έχουν καθοριστεί σε ένα controller.

Πως μπορούμε να πετύχουμε μια τέτοια προστασία; Τα δεδομένα κάθε χρήστη αποθηκεύονται στη βάση δεδομένων, και κάποιο πεδίο αφορά τον ρόλο του χρήστη πχ admin ή user. Κάθε φορά που ο χρήστης κάνει ένα HTTP αίτημα request σε ένα προστατευμένο Route του server μαζί με ότι στοιχεία πρόκειται να στείλει στέλνει και το JWT (Json Web

Token). Πριν εκτελεστούν οι λειτουργίες που καθορίζονται στο controller που αφορούν το συγκεκριμένο route, θα πρέπει να υπάρχει ένα ενδιάμεσο κομμάτι κώδικα το οποίο ελέγχει τα στοιχεία του χρήστη, τον αναζητάνε στη βάση δεδομένων, ελέγχουν την τιμή στο πεδίο που αφορά τον ρόλο του και αν είναι εξουσιοδοτημένος του επιτρέπεται η πρόσβαση στις λειτουργίες του controller. Αυτό το κομμάτι κώδικα έχει επικρατήσει να ονομάζεται middleware και χρησιμοποιείται και σε άλλες περιπτώσεις αναγκών ελέγχου όπως το Authentication Based On Password. Όλα τα παραπάνω αναπαριστώνται στο σχήμα της εικόνας 2.19.



**Εικόνα 2.19,** Διάγραμμα ροής Password and Role-Based Authorization σε ένα REST API (MERN ή PERN) χρησιμοποιώντας JWT, Middleware

## 3 Εργαλεία Ανάπτυξης και Υποστήριξης

### 3.1 Visual Studio Code

Το Visual Studio Code (VS Code) είναι ένα δωρεάν προγραμματιστικό περιβάλλον, το οποίο είναι ελαφρύ, μπορεί να εγκατασταθεί και να λειτουργήσει στα λειτουργικά συστήματα macOS, Linux και Windows. Είναι αρκετά εύχρηστο καθώς δίνει την δυνατότητα καθορισμού του τρόπου εμφάνισης του UI με διαφορετικά θέματα, debuggers και επεκτάσεις ώστε να χρησιμοποιηθούν διάφορες γλώσσες προγραμματισμού ανάλογα τις ανάγκες του προγραμματιστή. Υπάρχει μεγάλη γκάμα πακέτων-βιβλιοθηκών και πρόσθετων λειτουργιών για κάθε ανάγκη και είναι διαθέσιμες μέσω του npm package manager (Microsoft, n.d.).



**Εικόνα 3.1** Το λογότυπο του Visual Studio Code(VS Code)

### 3.2 git και GitHub

Το git είναι ένα κατακευματισμένο σύστημα ελέγχου εκδόσεων (Distributed Version Control System - DVCS) το οποίο δίνει την δυνατότητα στους προγραμματιστές να πραγματοποιούν αλλαγές στον κώδικα δημιουργώντας εκδόσεις αυτού. Με το εργαλείο αυτό, οι developers μπορούν να αποθηκεύουν το κώδικα που αναπτύσσουν τοπικά στην συσκευή τους ή σε άλλα απομακρυσμένα αποθετήρια, να συνεργάζονται μεταξύ τους και να σχολιάζουν(commit) κώδικα, να συγχωνεύουν τροποποιήσεις και εκδόσεις κώδικα εύκολα και να δοκιμάζουν εκδόσεις πριν την παραγωγή. (Git, n.d.).

Στο προγραμματιστικό περιβάλλον VS Code που ανέφερα στο προηγούμενο υποκεφάλαιο, μπορούμε να εγκαταστήσουμε το Git και να το χρησιμοποιήσουμε με γραφικά στοιχεία(Microsoft, n.d.).

Το GitHub, είναι μια διαδικτυακή πλατφόρμα αποθετηρίων κώδικα στην οποία οι developers μπορούν να συνεργάζονται, να αποθηκεύουν και να ανακτούν κώδικα και εκδόσεις των project τους.(GitHub, n.d.) Ανάλογα τις ανάγκες πάντα, μπορεί να παρέχει ένα ή περισσότερα κεντρικά σημεία για τη συνεργασία των προγραμματιστών.

Ο συνδυασμός των VS Code, Git και GitHub αποτελεί μία καλή πρακτική και ως backup ενός project καθώς αποθήκευση του σε παραπάνω από ένα σημεία (πχ. αποθήκευση project τοπικά και στο GitHub) εξασφαλίζει τη μη απώλεια του λόγω απρόοπτων γεγονότων όπως πχ η καταστροφή του σκληρού δίσκου που είναι αποθηκευμένο.



**Εικόνα 3.2** Λογότυπα git και GitHub

### 3.3 Postman

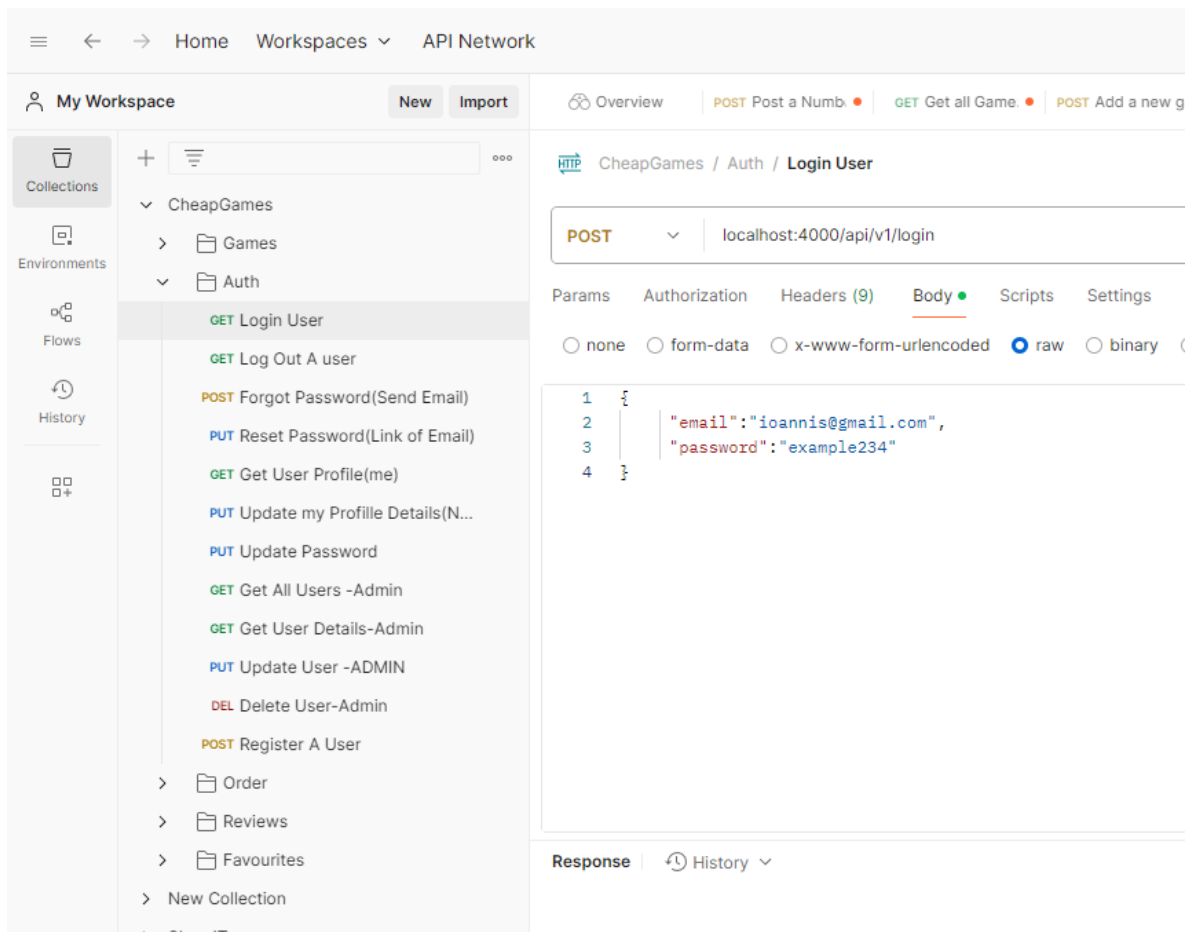
Το Postman αποτελεί ένα δημοφιλές εργαλείο λογισμικού καθώς το εμπιστεύονται και το χρησιμοποιούν πάνω από 500.000 εταιρείες και 35 εκατομμύρια προγραμματιστές το οποίο δίνει την δυνατότητα στους developers να στέλνουν αιτήματα προς έναν server, να ελέγχουν και να αναλύουν την λειτουργία και απαντήσεις του (Postman, n.d.).



**Εικόνα 3.3** Λογότυπο Postman

Για να το χρησιμοποιήσουμε αρχικά πρέπει να το εγκαταστήσουμε σε κάποιον υπολογιστή και να δημιουργήσουμε κάποιο λογαριασμό (Postman, n.d.).

Η εκτέλεση των αιτημάτων γίνεται μέσω του γραφικού περιβάλλοντος (Εικόνα 3.4) και δεν χρειάζονται συγκεκριμένες εντολές, ωστόσο θα πρέπει κανείς να έχει κατανοήσει τις αρχές του πρωτοκόλλου HTTP και να ακολουθήσει τις οδηγίες από το επίσημο site και να εξοικειωθεί με το πρόγραμμα (Postman, n.d.).



Εικόνα 3.4 Γραφικό Περιβάλλον Postman

Τα requests μπορούν να οργανωθούν και να αποθηκευτούν σε φακέλους και collections διευκολύνοντας τις εκτελέσεις και δοκιμές χωρίς να χρειάζεται κάθε φορά να δημιουργούνται από την αρχή. Τα requests ελέγχονται ως προς τις απαντήσεις τους δηλαδή αν επιστρέφονται τα σωστά δεδομένα και status code ενώ βοηθάει και στον έλεγχο και των τεχνικών ασφαλείας μιας διαδικτυακής εφαρμογής (πχ Authentication Based On Password και Authorization Based On Role) (Postman, n.d.).

Κάποιες από τις υπηρεσίες και χαρακτηριστικά του εργαλείου είναι τα εξής:

- Παρέχεται η δυνατότητα εξαγωγής δεδομένων σε διάφορες μορφές (JSON, CSV) (Postman, n.d.).
- Μέσω της Postman API και των workspaces συνεργασίας, οι ομάδες μπορούν να μοιράζονται collections, requests και αποτελέσματα δοκιμών, βελτιώνοντας τη συνεργασία και τη διαχείριση των API tests (Postman, n.d.).
- Το Postman περιλαμβάνει επίσης εργαλεία για ανάλυση και απεικόνιση των API απαντήσεων, καθώς και για διαχείριση cookies, επιτρέποντας τη βελτίωση της ασφάλειας και της διαχείρισης των αιτήσεων (Postman, n.d.).
- Επιπλέον, διαθέτει δυνατότητες για καταγραφή και επιθεώρηση της κυκλοφορίας των API(Postman, n.d.).
- Επιπλέον, το Postman δεν περιορίζεται μόνο σε HTTP requests αλλά υποστηρίζει και

διαφορετικά πρωτόκολλα επικοινωνίας, όπως GraphQL, gRPC, WebSocket, MQTT και SOAP (Postman, n.d.).

### 3.4 CLOUDINARY

Το Cloudinary είναι μια πλατφόρμα διαχείρισης πολυμέσων η οποία παρέχει υπηρεσίες (SaaS) αποθήκευσης, μετατροπής και διανομής εικόνων και βίντεο μέσω του διαδικτύου. (Cloudinary, n.d.).

Με το Cloudinary έχουμε τη δυνατότητα, αρχεία εικόνων και βίντεο που πρέπει να αποθηκευτούν κατά τη λειτουργία και χρήση ενός διαδικτυακού τόπου, να αποθηκεύονται σε έναν εξωτερικό πάροχο χωρίς να επιβαρύνεται η βάση ή βάσεις δεδομένων που χρησιμοποιούνται για άλλους σκοπούς.

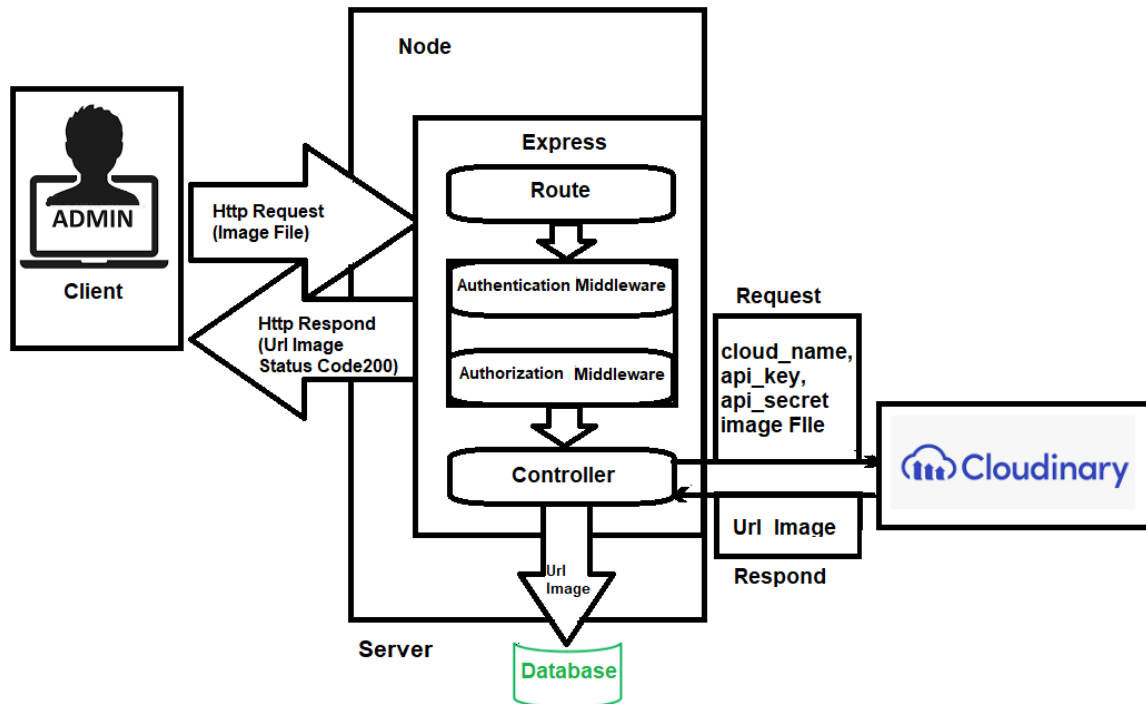
Το Cloudinary επίσης προσφέρει αρκετά οφέλη και στους προγραμματιστές. Ένα από τα πιο σημαντικά είναι η ύπαρξη έτοιμων βιβλιοθηκών (SDKs), οι οποίες επιτρέπουν την εύκολη ενσωμάτωση της υπηρεσίας σε εφαρμογές MERN και PERN μέσω του npm installer (Cloudinary, n.d.). Με αυτό τρόπο επιτυγχάνεται η ανάπτυξη ενός Project μειώνοντας τον χρόνο της κατασκευής καθώς δεν ξεκινάμε από το 0.

Στην περίπτωση της υλοποίησης του διαδικτυακού καταστήματος CheapGames θα υπάρξει η ανάγκη για αποθήκευση εικόνων οι οποίες θα αφορούν τα προϊόντα-παιχνίδια.

Για να χρησιμοποιήσουμε το Cloudinary, θα πρέπει να έχουμε φτιάξει αρχικά ένα λογαριασμό στην επίσημη πλατφόρμα. Ύστερα η πλατφόρμα μας παρέχει στοιχεία (Cloud Name, Api\_Key και Secret Key) τα οποία απαιτούνται για τη ρύθμιση της επικοινωνίας μεταξύ του server και του Cloudinary.

Αυτά τα στοιχεία ενσωματώνονται στο backend της εφαρμογής και χρησιμοποιούνται για την ταυτοποίηση της υπηρεσίας όταν στέλνουμε αιτήματα αποθήκευσης εικόνων. Συνήθως, τα credentials αποθηκεύονται με ασφαλή τρόπο μέσω μεταβλητών περιβάλλοντος (environment variables) και δεν περιλαμβάνονται απευθείας στον κώδικα, για λόγους ασφαλείας.

Στην εικόνα 3.5, απεικονίζεται η διαδικασία κατά την οποία ένας διαχειριστής ανεβάζει μία εικόνα σχετική με ένα παιχνίδι. Αρχικά, το αρχείο αποστέλλεται από τον client μέσω HTTP Request προς τον server. Ο server, αφού περάσει τα δεδομένα από τα authentication και authorization middleware, προωθεί την εικόνα στο Cloudinary μέσω API Request. Μετά την επιτυχή αποθήκευση, το Cloudinary επιστρέφει το URL της εικόνας, το οποίο στη συνέχεια αποθηκεύεται στη βάση δεδομένων και αποστέλλεται ως απόκριση στον client.



Εικόνα 3.5 Διαδικασία αποθήκευσης εικόνων σε Cloudinary μέσω REST API σε εφαρμογή MERN/PERN

### 3.5 STRIPE

Το Stripe είναι μια διαδεδωμένη και πιστοποιημένη ως προς την ασφάλεια πλατφόρμα ηλεκτρονικών πληρωμών μέσω διαδικτύου, βασίζεται στην αρχιτεκτονική Rest και μπορεί να ενσωματωθεί εύκολα μέσω διαθέσιμων Βιβλιοθηκών (SDKs) για κάθε γλώσσα και περιβάλλον προγραμματισμού (Stripe, n.d.).

Στο ηλεκτρονικό κατάστημα CheapGames θα πρέπει να υπάρχει η δυνατότητα αγοράς παιχνιδιών μέσω πιστωτικής κάρτας. Αρα θα πρέπει να ενσωματώσουμε ένα εργαλείο κατάλληλο για αυτό το σκοπό σαν το Stripe. Κατά την υλοποίηση της διαδικτυακής εφαρμογής μπορούμε να εκτελέσουμε πληρωμές σε test\_mode , δηλαδή δοκιμαστικά ελέγχοντας την ορθή λειτουργία χωρίς να χρεώνουμε πραγματικές κάρτες.

Για να ενσωματώσουμε το Stripe σε ένα MERN ή PERN stack, θα πρέπει αρχικά να δημιουργήσουμε έναν λογαριασμό στην πλατφόρμα του Stripe και να εγκαταστήσουμε το Stripe SDK για Node.js. Αφού δημιουργηθεί ο λογαριασμός, θα αποκτήσουμε τα απαραίτητα API keys (Publishable Key & Secret Key) τα οποία χρησιμοποιούνται για την επικοινωνία του server με το Stripe.

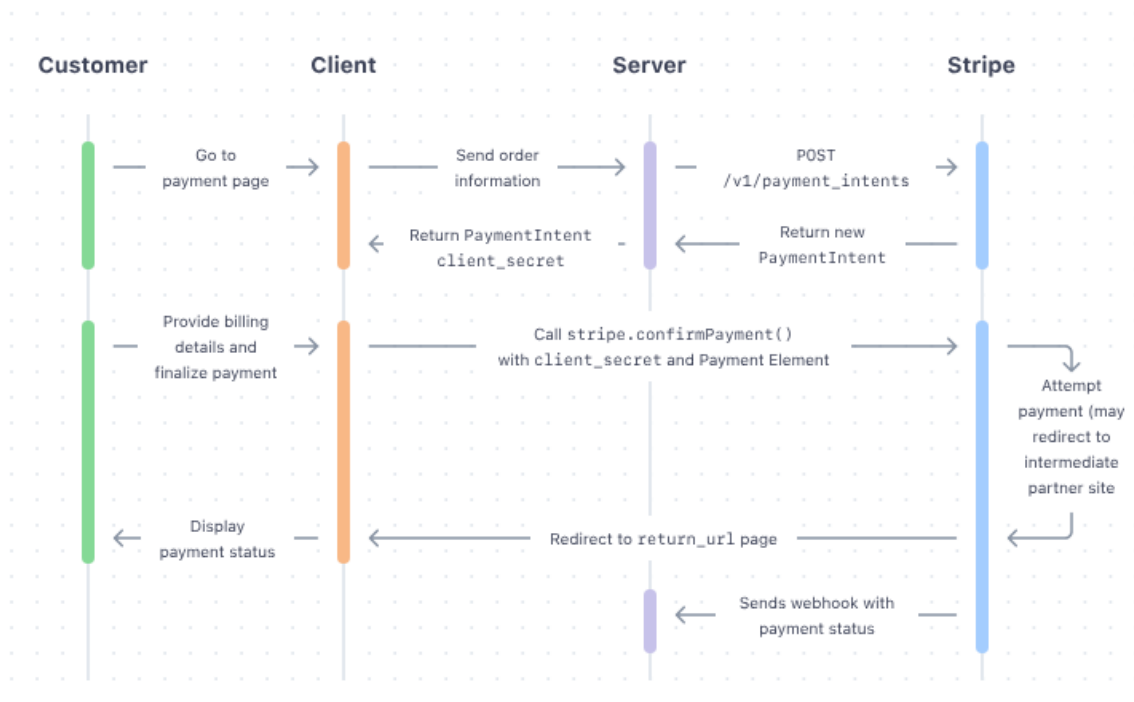
Στην Εικόνα 3.6, απεικονίζεται σε μια γενική άποψη η διαδικασία ολοκλήρωσης μιας πληρωμής μέσω πιστωτικής κάρτας, χρησιμοποιώντας τον πάροχο Stripe και τη μέθοδο Payment Intent. Ο πελάτης (client) προσθέτει προϊόντα στο καλάθι και προχωρά στην πληρωμή (checkout process). Ο server λαμβάνει τα στοιχεία της παραγγελίας και, μέσω της βιβλιοθήκης Stripe και των API Keys (τα οποία είναι αποθηκευμένα σε env variables),



στέλνει αίτημα στον Stripe Server.

Ο Stripe Server ανταποκρίνεται δημιουργώντας ένα αντικείμενο Payment Intent, το οποίο περιλαμβάνει ένα Client Secret URL. Αυτό το URL στέλνεται στον client για να ανακατευθυνθεί σε ασφαλές περιβάλλον του Stripe, όπου ο πελάτης εισάγει τα στοιχεία της πιστωτικής κάρτας του. Στη συνέχεια, η Stripe επεξεργάζεται τη συναλλαγή, επικοινωνεί με τις τράπεζες και τα δίκτυα πληρωμών, και επιστρέφει Status Code στους client και server της εφαρμογής για το αν η πληρωμή ολοκληρώθηκε ή όχι.

Αν η πληρωμή είναι επιτυχής, ο server μπορεί να δημιουργήσει μία εγγραφή στη βάση δεδομένων σχετικά με την ολοκλήρωση της παραγγελίας και ο client ανακατευθύνεται πίσω στο ηλεκτρονικό κατάστημα, όπου εμφανίζεται το σχετικό μήνυμα επιβεβαίωσης (display status code).



**Εικόνα 3.6:** Διάγραμμα ροής πληρωμών μέσω Stripe Payment Intent API(Ανακτήθηκε από: Stripe Documentation, <https://docs.stripe.com/payments/link/add-link-elements-integration> )

### 3.6 MailTrap

Το Mailtrap είναι μια πλατφόρμα αυτοματοποιημένης αποστολής email η οποία προσφέρει testing αποστολής email σε ασφαλές περιβάλλον κατά την ανάπτυξη μιας διαδικτυακής εφαρμογής ώστε αυτά να μην στέλνονται σε πραγματικούς χρήστες αλλά να γίνεται ο έλεγχος της ορθής λειτουργίας (Mailtrap, n.d.).

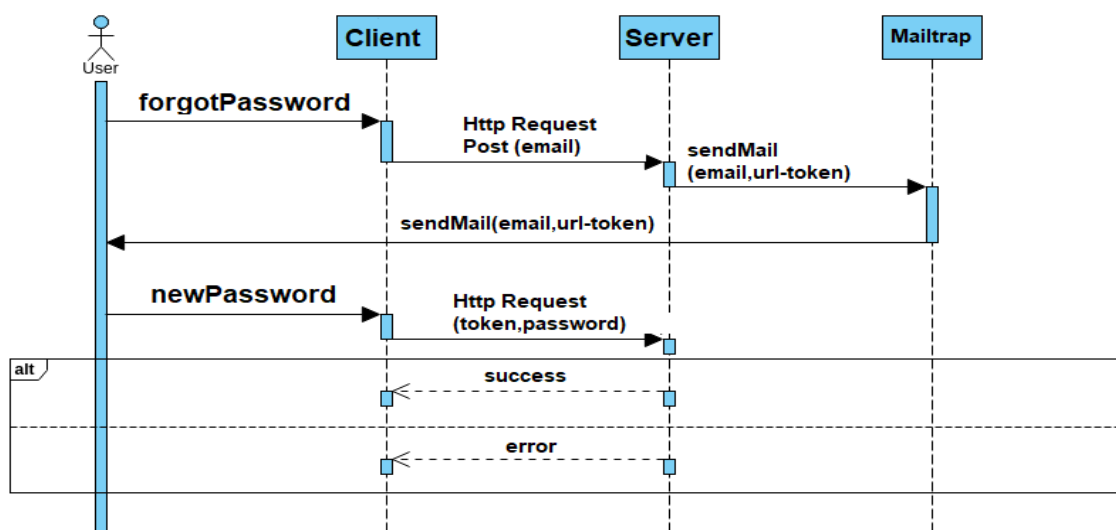
Στο ηλεκτρονικό κατάστημα CheapGames, θα υπάρξει η ανάγκη για ανάκτηση στοιχείων

πρόσβασης σε περίπτωση που κάποιος χρήστης ξεχάσει τα στοιχεία του. Μία τέτοια ανάγκη θα πρέπει να εξυπηρετείται από μία αυτοματοποιημένη διαδικασία. Ένας κοινός τρόπος ο οποίος μπορεί να παρατηρηθεί σε αρκετές ιστοσελίδες, είναι ο χρήστης να επιλέγει σε κάποιο σημείο της σελίδας την επιλογή “ανάκτηση στοιχείων” ή “forgot password” ή κάποια άλλη αντίστοιχη επιλογή. Η σελίδα εμφανίζει μία φόρμα εισαγωγής στοιχείων όπου ο χρήστης εισάγει τη διεύθυνση email του, το σύστημα επιβεβαιώνει ότι ο χρήστης έχει κάνει εγγραφή με τα στοιχεία αυτά και στέλνει ένα email με έναν υπερσύνδεσμο στη διεύθυνση email. Ο χρήστης λαμβάνει το email και ο υπερσύνδεσμος τον ανακατευθύνει σε σελίδα όπου εισάγει καινούριο κωδικό, επιλέγει επιβεβαίωση και αν όλα έχουν γίνει σωστά μπορεί να συνδεθεί με τον κωδικό αυτό.

Το Mailtrap μπορεί να χρησιμοποιηθεί για την αποστολή email από το σύστημα σε κάποιο χρήστη εξυπηρετώντας την ανάκτηση στοιχείων. Αντί να στέλνεται το email ανάκτησης απευθείας σε πραγματικούς χρήστες, το Mailtrap δημιουργεί ένα εικονικό περιβάλλον (sandbox) για την αποστολή και τον έλεγχο των email, εξασφαλίζοντας ότι η διαδικασία αποστολής λειτουργεί σωστά.

Στο σχήμα της εικόνας 3.7 βλέπουμε την αλληλεπίδραση μεταξύ client,server και MailTrap για την ανάκτηση στοιχείων.Ο Client (Frontend) στέλνει ένα HTTP POST Request στον Server με το email του χρήστη. Ο Server δημιουργεί έναν μοναδικό κωδικό ανάκτησης (token) και τον συμπεριλαμβάνει σε ένα URL(Μέσω αυτού του Url ο χρήστης μπορεί να δημιουργήσει ένα καινούριο κωδικό).Το url-token στέλνεται μέσω του mailTrap στο email του χρήστη. Το link οδηγεί τον χρήστη σε κατάλληλη σελίδα της διαδικτυακή εφαρμογής όπου εισάγει το καινούριο κωδικό.Ο Client στέλνει με ένα HTTP request τον καινούριο password στο Server.Αν ο κωδικός έχει ενημερωθεί σωστά, επιστρέφεται επιτυχής απάντηση στον Client.

Σημαντικό να αναφέρουμε ότι το token που δημιουργείται από τον server, ρυθμίζεται ώστε να είναι ενεργό για συγκεκριμένο χρονικό διάστημα. Το token σε Mern Stack ή Pern μπορεί να δημιουργηθεί μέσω του JWT(Json Web Token).



Εικόνα 3.7 Διάγραμμα Ακολουθίας για Ανάκτηση Κωδικού(Mailtrap)

## **4. Ανάλυση και Σχεδίαση Λειτουργικότητας του Συστήματος Cheap Games**

### **4.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΟΣ CHEAP GAMES.**

Το δημοφιλές ηλεκτρονικό κατάστημα πώλησης βιντεοπαιχνιδιών “CheapGames” διαθέτει ψηφιακά βιντεοπαιχνίδια τα οποία οι εγγεγραμμένοι χρήστες μπορούν με την αγορά τους να αξιολογούν και να κάνουν download στον υπολογιστή τους αν τους το επιτρέπει η ηλικία τους.

Στην ιστοσελίδα θα μπορεί να συνδέεται καταρχάς ο διαχειριστής, ο οποίος θα μπορεί να προσθαφαιρεί/ενημερώνει/διαγράφει στο σύστημα τα προϊόντα-παιχνίδια, καταχωρώντας για το κάθε παιχνίδι δεδομένα όπως αρχείο παιχνιδιού, κατηγορία, έκπτωση, τιμή, όριο ηλικίας, εικόνες, δημιουργό και εκδότη, ελάχιστες και προτεινόμενες απαιτήσεις συστήματος. Ο διαχειριστής επίσης θα έχει τη δυνατότητα να ρυθμίζει την εμφάνιση ενός διαφημιστικού Slide στους χρήστες, το οποίο θα αφορά εποχιακή έκπτωση για συγκεκριμένο χρονικό διάστημα σε επιλεγμένες κατηγορίες με ειδικό προωθητικό μήνυμα και εικόνες για το κάθε παιχνίδι. Στα πλαίσια των ρυθμίσεων αυτών, ο διαχειριστής θα έχει τη δυνατότητα να εφαρμόζει την εποχιακή έκπτωση στις επιλεγμένες κατηγορίες παιχνιδιών καθώς και να επαναφέρει τις τιμές των εκπτώσεων στην αρχική τους μορφή κατά την αρχή και λήξη της περιόδου της εποχιακής έκπτωσης. Κάποιες άλλες δυνατότητες των διαχειριστών είναι να βλέπουν στοιχεία για τα έσοδα τους επιλέγοντας συγκεκριμένο ημερολογιακό διάστημα, να διαγράφουν αξιολογήσεις παιχνιδιών, να βλέπουν/ενημερώνουν/διαγράφουν στοιχεία των εγγεγραμμένων χρηστών και όλων τις αγορών.

Ο κάθε χρήστης που συνδέεται στη σελίδα θα μπορεί να βλέπει έναν κατάλογο των προς πώληση παιχνιδιών με τις πιο σημαντικές πληροφορίες για το κάθε παιχνίδι όπως τίτλο, τιμή, όριο ηλικίας και σκορ αξιολογήσεων αλλά για να δει περισσότερες λεπτομέρειες για το καθένα, θα πρέπει να έχει κάνει εγγραφή όπου θα δηλώνει και την ημερομηνία γέννησης. Ο εγγεγραμμένος χρήστης για να αγοράσει ή να δει περισσότερες λεπτομέρειες για ένα παιχνίδι όπως περιγραφή, αξιολογήσεις, θα πρέπει η ηλικία του να είναι μεγαλύτερη από το καταχωρημένο όριο ηλικίας του παιχνιδιού, διαφορετικά θα εμφανίζεται κατάλληλο μήνυμα. Ο εγγεγραμμένος χρήστης θα μπορεί να προσθέτει παιχνίδια σε καλάθι αγορών και να προχωράει στην αγορά τους μέσω κάρτας, να βλέπει τα αγορασμένα του παιχνίδια καθώς και προσωπική λίστα αγαπημένων παιχνιδιών άσχετα από αν είναι αγορασμένα ή όχι από αυτόν. Επίσης ο εγγεγραμμένος χρήστης μπορεί να αξιολογεί μόνο τα παιχνίδια που αγοράζει με βαθμολογία και ένα σύντομο σχόλιο και να βλέπει τις αγορές του χωρίς να έχει δυνατότητα διαγραφής κάποιας. Τέλος ο εγγεγραμμένος χρήστης θα μπορεί να προσθέτει εθελοντικά τον χρόνο που έπαιξε το αγορασμένο παιχνίδι και αυτές να καταγράφονται από το σύστημα.

## 4.2 ΛΕΙΤΟΥΡΓΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ

Σε αυτό το κεφάλαιο θα καθορίσουμε τις λειτουργικές απαιτήσεις του συστήματος και θα τις χωρίσουμε σε κατηγορίες ανάλογα του τι αφορούν π.χ χρήστες, διαχειριστές, βιντεοπαιχνίδια-προϊόντα για να αντικατοπτρίσουμε στο δυνατότερο σημείο το λειτουργικό κομμάτι του Api.

Αρχικά στο έγγραφο των απαιτήσεων θα τοποθετήσουμε τους **διαχειριστές**.

### Λειτουργικές Απαιτήσεις για διαχειριστές

Οι διαχειριστές είναι υπεύθυνοι για την διαχείριση των προϊόντων των παιχνιδιών και την ομαλή λειτουργία του περιεχομένου του eshop καθώς και την έγκαιρη αντιμετώπιση τυχόν προβλημάτων. Μπορούν να εκτελέσουν όλες τις λειτουργίες που τους προσφέρονται από το σύστημα αφού πρώτα συνδεθούν εισάγοντας email και κωδικό πρόσβασης. Αυτό σημαίνει ότι θα έχουν τις ίδιες δυνατότητες με τους εγγεγραμμένους χρήστες και συμπληρωματικά κάποιες που δεν θα μπορούν οι υπόλοιποι χρήστες. Πιο συγκεκριμένα οι απαιτήσεις που αφορούν αποκλειστικά μόνο τους διαχειριστές:

- **Προσθήκη τίτλων και αρχείων παιχνιδιών που αφορούν τους τίτλους στη βάση Δεδομένων:**
  - Προσθήκη νέων παιχνιδιών με πληροφορίες όπως τίτλος, περιγραφή, εικόνες, βίντεο, τιμή, περιορισμός ηλικίας, επιλογή κατηγορίας, απαιτήσεις συστήματος, τιμής, έκπτωσης και ανέβασμα αρχείων που αφορούν το κάθε παιχνίδι στη βάση Δεδομένων
- **Ενημέρωση/Update υπαρχόντων παιχνιδιών**
  - Τροποποίηση στοιχείων που αφορούν το κάθε παιχνίδι και ενημέρωση αρχείων παιχνιδιών
- **Διαγραφή παιχνιδιών**
  - Διαγραφή παιχνιδιών που δεν είναι πλέον διαθέσιμα
- **Διαχείριση Χρηστών**
  - Τροποποίηση στοιχείων Χρηστών Name, Email, dateOfBirth και Role από χρήστη σε διαχειριστή και το αντίστροφο
  - Διαγραφή Χρηστών
- **Διαγραφή αξιολογήσεων για το κάθε παιχνίδι**
  - Διαγραφή Αξιολογήσεων
- **Διαχείριση Παραγγελιών**
  - Ελεγχος Παραγγελιών
  - Διαγραφή Παραγγελιών
- **Προβολή Εισόδων ανα συγκεκριμένο ημερολογιακό διάστημα**
- **Καθορισμός Εκπτώσεων**
- **Διαχείριση διαφημιστικού slide εποχιακής έκπτωσης για επιλεγμένες κατηγορίες και εφαρμογή εποχιακής έκπτωσης στα αντίστοιχα προϊόντα**
  - Δημιουργία δεδομένων εποχιακής έκπτωσης
  - Επιλογή ενεργοποίησης /απενεργοποίησης του διαφημιστικού Slide στους χρήστες
  - Εφαρμογή της εποχιακής έκπτωσης στις κατηγορίες παιχνιδιών
  - Επαναφορά των εκπτώσεων των κατηγοριών στην αρχική τους τιμή

Στη συνέχεια θα δούμε τις λειτουργικές απαιτήσεις για του Χρήστες.

Οι χρήστες χωρίζονται στους χρήστες που είναι εγγεγραμμένοι και στους μη εγγεγραμμένους.

Αρχικά θα δούμε τις Λειτουργικές απαιτήσεις για τους μη εγγεγραμμένους χρήστες

**Λειτουργικές Απαιτήσεις για μη εγγεγραμμένους χρήστες**

- Περιήγηση στον κατάλογο των παιχνιδιών εμφανίζοντας για το κάθε παιχνίδι το τίτλο, το όριο ηλικίας, την τιμή, τυχόν έκπτωση, Σκορ Αξιολογήσεων
- Αναζήτηση Τίτλων παιχνιδιών βάση Τίτλου, Κατηγορίας, και φίλτρων Τιμής Min και Max και min Σκορ Αξιολογήσεων
- Εγγραφή στο σύστημα  
-Υποχρεωτική Εισαγωγή στοιχείων Name, Email, dateOfBirth, Password
- Εμφάνιση κατάλληλου μηνύματος σε περίπτωση Εισαγωγής λάθος στοιχείων Εγγραφής όπως πχ. email που χρησιμοποιεί άλλος χρήστης

Στην επόμενη φάση θα δούμε τις λειτουργικές απαιτήσεις για τους εγγεγραμμένους Χρήστες  
Οι χρήστες αφού επιλέξουν Login και συνδεθούν με τα στοιχεία τους Email και Password αποκτούν δυνατότητες σε περισσότερες λειτουργίες της σελίδας σε σχέση με τους μη εγγεγραμμένους.

**Λειτουργικές Απαιτήσεις για εγγεγραμμένους χρήστες**

- Επεξεργασία Προφίλ  
-Τροποποίηση στοιχείων Name, Email, dateOfBirth
- Ανάκτηση Στοιχείων Πρόσβασης  
-Δυνατότητα Εισαγωγής Email με περιεχόμενο Link-σύνδεσμο που αφορά μόνο τον χρήστη με σκοπό την ανάκτηση στοιχείων πρόσβασης.
- Περιήγηση στον κατάλογο των παιχνιδιών εμφανίζοντας για το κάθε παιχνίδι το τίτλο, το όριο ηλικίας, την τιμή, τυχόν έκπτωση, Σκορ και Πλήθους Αξιολογήσεων και εμφάνισης επιλογής προβολής περισσότερων λεπτομερειών
- Περιήγηση στον κατάλογο των αγαπημένων παιχνιδιών εμφανίζοντας για το κάθε παιχνίδι το τίτλο, το όριο ηλικίας, την τιμή, τυχόν έκπτωση, Σκορ και πλήθους Αξιολογήσεων
- Περιήγηση στον κατάλογο των αγορασμένων παιχνιδιών εμφανίζοντας για το κάθε παιχνίδι το τίτλο, το όριο ηλικίας, την τιμή, τυχόν έκπτωση, Σκορ και πλήθους Αξιολογήσεων  
-Εμφάνιση Επιλογής για το κάθε παιχνίδι προβολής Λεπτομερειών  
-Εμφάνιση Επιλογής Εναρξης για το κάθε παιχνίδι όπου ο χρήστης μπορεί να κατεβάσει το παιχνίδι
- Αναζήτηση Τίτλων παιχνιδιών βάση Τίτλου, Κατηγορίας, Τιμής Min και Max και Αξιολογήσεων
- Προβολή λεπτομερειών παιχνιδιών  
-Εικόνες, Περιγραφή, στιγμιότυπα, τιμή, πιθανή έκπτωση, σκορ αξιολογήσεων, minimum τεχνικές απαιτήσεις συστήματος, συνιστώμενες τεχνικές προδιαγραφές, Αξιολογήσεις, αν Είναι αγαπημένο ή όχι  
-Αν το παιχνίδι δεν είναι εντός του ορίου ηλικίας του χρήστη δεν θα επιτρέπεται η πρόσβαση.
- Προσθήκη Παιχνιδιού στα αγαπημένα  
-Στις λεπτομέρειες για το κάθε παιχνίδι θα διατίθεται η δυνατότητα προσθήκης παιχνιδιού στη λίστα αγαπημένων του χρήστη
- Προσθήκη παιχνιδιού στο καλάθι αγορών  
-Στις λεπτομέρειες για το κάθε παιχνίδι θα διατίθεται η δυνατότητα προσθήκης παιχνιδιού στο καλάθι αγοράς
- Προβολή καλάθιού αγοράς

- Προβολή παιχνιδιών που έχουν προστεθεί στο καλάθι αγορών μαζί με τιμή του κάθε παιχνιδιού
- Προβολή Συνολικής τιμής προϊόντων και Πλήθους Παιχνιδιών που είναι στο καλάθι αγοράς
- Εμφάνιση Επιλογής για προώθηση στο επόμενο βήμα Εισαγωγής Στοιχείων Πελάτη(Διεύθυνσης, Πόλης, Τηλεφώνου, Ταχ. Κώδικα, και Χώρας).
- Δυνατότητα επιλογής αφαίρεσης παιχνιδιού από το καλάθι
- **Προβολή Όλων των αγοράς και λεπτομέρειες Αγοράς του χρήστη**
- **Download και Εθελοντική Καταχώρηση χρόνου παιξίματος αγορασμένου παιχνιδιού**
- **Αξιολόγηση παιχνιδιού μόνο αν έχει αγοραστεί από το χρήστη**
  - Εισαγωγή κειμένου που αφορά το παιχνίδι
  - Εισαγωγή Σκορ

Θα μπορούσαμε να συντάξουμε και έναν πίνακα με όλες τις δυνατές λειτουργίες για την κάθε κατηγορία χρήστη (Διαχειριστής, μη Εγγεγραμμένος Χρήστης , Εγγεγραμμένος χρήστης) βοηθώντας στον σχεδιασμό αλλά και στον αναγνώστη να καταλάβει τα επόμενα μέρη της εργασίας. Στη κάθε Λειτουργία θα δώσουμε έναν αριθμό τον οποίο θα χρησιμοποιήσουμε στο επόμενο κεφάλαιο με τις περιπτώσεις χρήσης και τα Διαγράμματα Περιπτώσεων Χρήσης.

Πίνακας 4.1 Πρόσβαση και Απαγόρευση Χρηστών σε λειτουργίες

Αριθμός	Λειτουργία	Διαχειριστής	Εγγεγραμμένος χρήστης	Μη Εγγεγραμμένος χρήστης
1	Περιήγηση στον κατάλογο παιχνιδιών (τίτλος, όριο ηλικίας, τιμή, έκπτωση, σκορ αξιολογήσεων)	✓	✓	✓
2	Σύνδεση Χρήστη	✓	✓	✗
3	Περιήγηση στα αγαπημένα παιχνίδια (τίτλος, όριο ηλικίας, τιμή, έκπτωση, σκορ, πλήθος αξιολογήσεων)	✓	✓	✗
4	Περιήγηση στα αγορασμένα παιχνίδια	✓	✓	✗
5	Αναζήτηση παιχνιδιών (τίτλος, κατηγορία, φίλτρα τιμής Min/Max, σκορ αξιολογήσεων)	✓	✓	✓



6	Προβολή λεπτομερειών παιχνιδιού (περιγραφή, εικόνες, στιγμιότυπα, απαιτήσεις συστήματος, σκορ αξιολογήσεων, αν είναι αγαπημένο, αξιολογήσεις)	✓	✓	✗
7	Εγγραφή στο σύστημα (name, email, dateOfBirth, password)	✗	✗	✓
8	Ανάκτηση στοιχείων πρόσβασης (αποστολή email με σύνδεσμο)	✓	✓	✓
9	Επεξεργασία προφίλ (τροποποίηση name, email, dateOfBirth, προσθήκη/αλλαγή φωτογραφίας)	✓	✓	✗
10	Τροποποίηση/ Ενημέρωση Κωδικού	✓	✓	✗
11	Προσθήκη/Αφαίρεση παιχνιδιού από τα αγαπημένα παιχνίδια	✓	✓	✗
12	Προσθήκη παιχνιδιού στο καλάθι αγορών	✓	✓	✗
13	Προβολή καλαθιού αγορών (τίτλοι παιχνιδιών, τιμή, συνολική τιμή, πλήθος προϊόντων)	✓	✓	✗
14	Αφαίρεση παιχνιδιού από το καλάθι	✓	✓	✗
15	Εισαγωγή στοιχείων πελάτη (διεύθυνση, τηλέφωνο, χώρα κ.λπ.) και Στοιχείων Κάρτας-Πληρωμή Παραγγελίας	✓	✓	✗
16	Προβολή Λίστας Αγορών και	✓	✓	✗



	Λεπτομερειών Αγοράς του χρήστη			
17	Download και Εθελοντική καταχώρηση Αγορασμένου παιχνιδιού	✓	✓	✗
18	Αξιολόγηση παιχνιδιού (μόνο αν έχει αγοραστεί από το χρήστη)	✓	✓	✗
19	Προσθήκη παιχνιδιών (τίτλος, περιγραφή, εικόνες, βίντεο, τιμή, έκπτωση, απαιτήσεις συστήματος, αρχεία παιχνιδιών)	✓	✗	✗
20	Ενημέρωση/ τροποποίηση Στοιχείων παιχνιδιού	✓	✗	✗
21	Διαγραφή παιχνιδιού	✓	✗	✗
22	Τροποποίηση στοιχείων άλλων χρηστών, π.χ αλλαγή ρόλου από χρήστη σε διαχειριστή, διαγραφή)	✓	✗	✗
23	Διαγραφή χρήστη	✓	✗	✗
24	Διαχείριση παραγγελιών (έλεγχος, διαγραφή)	✓	✗	✗
25	Διαχείριση Εισόδων ανα συγκεκριμένο ημερολογιακό διάστημα	✓	✗	✗
26	Δημιουργία/Ενημέρωση Εποχιακής Εκπτώσης	✓	✗	✗
27	Εφαρμογή εποχιακών εκπτώσεων σε συγκεκριμένες	✓	✗	✗

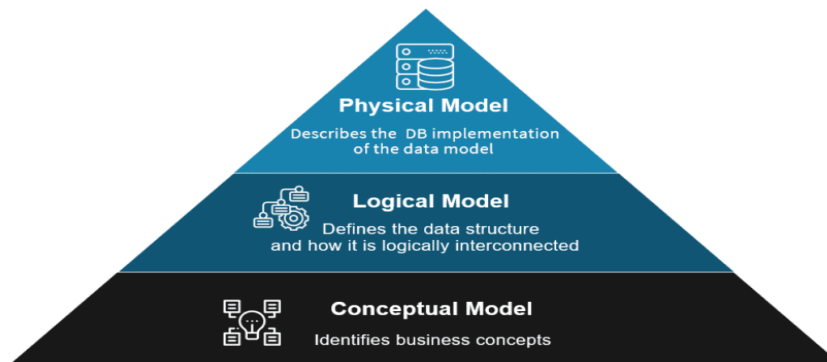
	κατηγορίες παιχνιδιών			
28	Επαναφορά εκπτώσεων παιχνιδιών με εφαρμοσμένη την εποχιακή έκπτωση στις αρχικές-προηγούμενες τους τιμές	✓	✗	✗
29	Απενεργοποίηση Εμφάνισης Διαφημιστικού Slide	✓	✗	✗

Πίνακας 4.1 Πρόσβαση και Απαγόρευση Χρηστών σε λειτουργίες

### 4.3 Data Modeling

Η μοντελοποίηση δεδομένων (data modeling) είναι η διαδικασία που προσδιορίζει τα σχετικά δεδομένα, τον τρόπο συλλογής και επεξεργασίας αυτών των δεδομένων και, τελικά, πώς τα δεδομένα μπορούν να οπτικοποιηθούν ως διάγραμμα. Αυτή η οπτική αναπαράσταση δεν βοηθά μόνο στον εντοπισμό όλων των στοιχείων δεδομένων, αλλά βοηθά επίσης στον προσδιορισμό των σχέσεων μεταξύ των στοιχείων δεδομένων. (MongoDB, 2022). Υπάρχουν τρία βασικά μέρη κατά την μοντελοποίηση (Εικόνα 4.1).

- **Εννοιολογικό (Conceptual):** Χρησιμοποιείται κυρίως για τον προσδιορισμό των δεδομένων που είναι χρήσιμα και απαραίτητα για τον επιχειρηματικό σκοπό και λειτουργία της διαδικτυακής εφαρμογής. Το εννοιολογικό μοντέλο είναι ανεξάρτητο από τη τεχνολογία ή το τύπο της βάσης που θα χρησιμοποιηθεί (Simsion & Witt, 2005, p. 17).
- **Λογικό (Logical):** Περιγράφει πώς θα οργανωθούν τα δεδομένα σε μια βάση, καθορίζοντας τους τύπους δεδομένων και τις μεταξύ τους σχέσεις, ανεξάρτητα από το συγκεκριμένο Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS) (MongoDB, 2022).
- **Φυσικό (Physical):** Αποτυπώνει την πραγματική υλοποίηση της βάσης δεδομένων, δηλαδή πώς τα δεδομένα θα αποθηκευτούν στο επιλεγμένο DBMS, συμπεριλαμβανομένων των πρωτεύοντων και ξένων κλειδιών (primary & foreign keys) στη PostgreSQL ή των αναφορών (references) και των ενσωματωμένων εγγράφων (embedded documents) στη MongoDB (MongoDB, 2022).



Εικόνα 4.1, Επίπεδα Data Modeling

πηγή: <https://bigdataanalyticsnews.com/difference-between-conceptual-logical-data-models/>

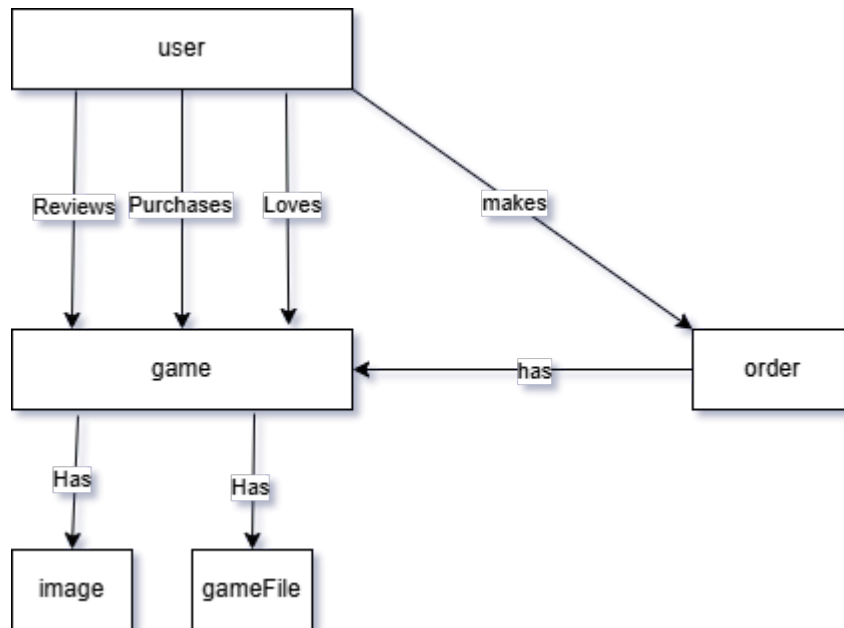
Ωστόσο αυτά τα επίπεδα δεν υπαγορεύουν ρητά τον τρόπο που θα γίνει η μοντελοποίηση. Διαφορετικές μεθοδολογίες, συνδυασμοί και συγχωνεύσεις αυτών μπορούν να χρησιμοποιηθούν (Simsion & Witt, 2005, p. 17 ). Για παράδειγμα κάποιος αναλυτής μπορεί να παραλείψει το εννοιολογικό επίπεδο και να προχωρήσει στο λογικό και το φυσικό, κάποιος αναλυτής να χρησιμοποιήσει διαγράμματα και σχήματα για την μοντελοποίηση και κάποιος άλλος πίνακες.

### 4.3.1 Εννοιολογικό Επίπεδο (Conceptional level)

Στο εννοιολογικό επίπεδο καθορίζονται τα δεδομένα που θα αποθηκεύονται στην βάση δεδομένων και ο τρόπος που σχετίζονται. Για αυτό το λόγο θα πρέπει να καθορίσουμε τις βασικές οντότητες και τις μεταξύ του σχέσεις. Οι βασικές οντότητες βάση της γενική περιγραφής και των λειτουργικών απαιτήσεων όπως καθορίστηκαν στο κεφάλαιο 4.2 είναι οι εξής : game, user, order, image και file.

Το Μοντέλο Οντοτήτων-Συσχετίσεων (Entity-Relational Diagram) είναι ένα δημοφιλές μοντέλο το οποίο χρησιμοποιείται συχνά και παραδοσιακά για τον εννοιολογικό σχεδιασμό μια βάσης δεδομένων και πολλά εργαλεία και περιβάλλοντα σχεδιασμού το έχουν ενσωματώσει στις διαθέσιμες επιλογές τους (Elmasri & Navathe, 2016, p.59).

Για τις ανάγκες του εννοιολογικού σχεδιασμού σχεδιάστηκε ένα ERD σε μία απλή μορφή (Εικόνα 4.2) όπου απεικονίζονται οι σχέσεις μεταξύ των βασικών οντοτήτων του συστήματος.



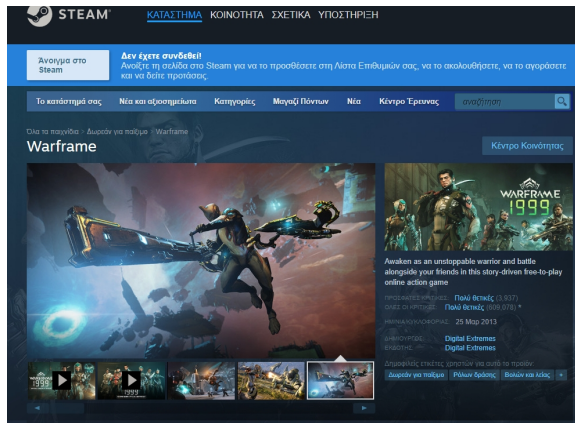
Εικόνα 4.2, Μοντελο Οντοτήτων Συσχετίσεων στο εννοιολογικό επίπεδο

### 4.3.2 Λογικό Επίπεδο (Logical Level)

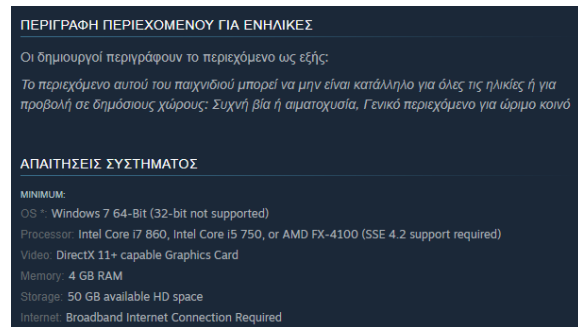
Στη συνέχεια προχωρώντας στο λογικό επίπεδο θα πρέπει αρχικά να καθορίσουμε όλες τις οντότητες, τις ιδιότητες (attributes) τους και τα πρωτεύον κλειδιά.

Ο καθορισμός των ιδιοτήτων εκτός από την αναζήτηση των οντοτήτων που ξεκίνησε στο εννοιολογικό επίπεδο, είναι ένας συνδυασμός του εντοπισμού τους εντός του κειμένου της γενικής περιγραφής αλλά και της παρατήρησης άλλων ηλεκτρονικών καταστημάτων Ψηφιακών παιχνιδιών όπως το Steam (Εικόνα 4.3 και 4.4). Φυσικά λαμβάνονται υπόψη και οι λειτουργικές απαιτήσεις όπως καθορίστηκαν στο κεφάλαιο 4.2.

ΤΙΤΛΟΣ: Warframe  
ΕΙΔΟΣ: Δράση, RPG, Δωρεάν για παιχνίδι  
ΔΗΜΙΟΥΡΓΟΣ: Digital Extremes  
ΕΚΔΟΤΗΣ: Digital Extremes  
ΗΜΕΡΙΑ ΚΥΚΛΟΦΟΡΙΑΣ: 25 Μαρ 2013



**Εικόνα 4.3** Στιγμιότυπο από Steam,  
Πηγή: <https://store.steampowered.com/>



**Εικόνα 4.4** Στιγμιότυπο με ιδιότητες που αφορούν  
ένα παιχνίδι  
Πηγή: <https://store.steampowered.com/>

Στον πίνακα 4.2 παρουσιάζονται όλες οι οντότητες με μια σύντομη περιγραφή, τις ιδιότητες τους και τους χαρακτηρισμούς PK για πρωτεύον κλειδί και FK για ξένο κλειδί.

Οντότητα	Ιδιότητες
<p>user (χρήστης)</p> <p>Περιγραφή: Στη κεντρική βάση δεδομένων θα πρέπει να αποθηκεύονται τα προσωπικά στοιχεία και στοιχεία Σύνδεση του κάθε χρήστη καθώς και ο ρόλος του (user ή admin) βάση του οποίου μπορεί να δίνεται ή να μη δίνεται η πρόσβαση στις επιλογές διαχείρισης.</p>	<ul style="list-style-type: none"> <li>● IDuser (αναγνωριστικό χρήστη) <b>-PK</b></li> <li>● name (όνομα)</li> <li>● dateOfBirth (ημερομηνία γέννησης Χρήστη)</li> <li>● email</li> <li>● role (ρόλος)</li> <li>● password (κωδικός πρόσβασης)</li> </ul>
<p>game (παιχνίδι)</p> <p>Περιγραφή: Στη κεντρική βάση δεδομένων θα πρέπει να αποθηκεύονται τα στοιχεία του κάθε παιχνιδιού. Στα πεδία έχουν συμπεριληφθεί τα πεδία previousDiscount και το πεδίο isSeasonalDiscountActive τα οποία χρησιμοποιούν για την εποχιακή έκπτωση. Στο previousDiscount αποθηκεύεται η αρχική έκπτωση πριν την εφαρμογή</p>	<ul style="list-style-type: none"> <li>● idgame (Αναγνωριστικό παιχνιδιού) <b>-PK</b></li> <li>● title (τίτλος),</li> <li>● description (περιγραφή),</li> <li>● category (κατηγορία),</li> <li>● previousDiscount (ποσοστό έκπτωσης πριν την εφαρμογή της Εποχιακής Εκπτώσης)</li> <li>● isSeasonalDiscountActive (Εφαρμοσμένες ή μη εφαρμοσμένες οι εποχιακές εκπτώσεις)</li> <li>● discount (έκπτωση που ισχύει και εφαρμόζεται κατά την αγορά του παιχνιδιού)</li> <li>● price (τιμή),</li> <li>● ageLimit (περιορισμός ηλικίας)</li> <li>● ratings (Σκορ Αξιολόγησης)</li> <li>● numOfReviews (Πλήθος Αξιολογήσεων)</li> </ul>

της εποχιακής έκπτωσης ενώ στο πεδίο isSeasonalDiscountActive το αν είναι ενεργή η εποχιακή έκπτωση.	<ul style="list-style-type: none"> <li>• minRequirements (Ελάχιστες Απαιτήσεις)</li> <li>• reqRequirements (Συνιστώμενες δυνατότητες)</li> <li>• Developer (Δημιουργός παιχνιδιού)</li> <li>• Publisher (Εκδότης)</li> <li>• ReleaseDate (Ημ/μηνία Κυκλοφορίας)</li> </ul>
<p>favouriteGame (Αγαπημένα Παιχνίδια)</p> <p>Περιγραφή: Στη κεντρική βασική δεδομένων θα πρέπει να αποθηκεύονται τα παιχνίδια που είναι αγαπημένα και ο χρήστης του οποίου είναι αγαπημένα.</p>	<ul style="list-style-type: none"> <li>• UserId (Αναγνωριστικό Χρήστη)-<b>PK</b></li> <li>• gameId (Αναγνωριστικό Παιχνιδιού)-<b>PK</b></li> </ul>
<p>gameFile (αρχείο παιχνιδιού)</p> <p>Το κάθε αρχείο θα πρέπει να έχει id και gameId για να ξέρουμε σε ποιο παιχνίδι αντιστοιχεί το κάθε αρχείο καθώς επίσης και όνομα αρχείου, τύπο αρχείου και τα δεδομένα του αρχείου</p>	<ul style="list-style-type: none"> <li>• Id (αναγνωριστικό αρχείου) - <b>PK</b></li> <li>• fileName (όνομα αρχείου)</li> <li>• fileType (τύπος αρχείου)</li> <li>• fileData (αρχείο παιχνιδιού)</li> <li>• gameId (αναγνωριστικό παιχνιδιού) – <b>FK (game)</b></li> </ul>
<p>review (αξιολόγηση)</p> <p>Περιγραφή: Θα πρέπει να υπάρχουν δεδομένα με όλα τα review (βαθμούς και Σχόλια) . Επίσης θα πρέπει να υπάρχουν το Id του παιχνιδιού που αφορά το review και ο χρήστης που το εισήγαγε</p>	<ul style="list-style-type: none"> <li>• IDReview (αναγνωριστικό αξιολόγησης)-<b>PK</b></li> <li>• IdGame (αναγνωριστικό παιχνιδιού) – <b>FK (game)</b></li> <li>• IdUser (Αναγνωριστικό Χρήστη) -<b>FK (user)</b></li> <li>• rating (βαθμός Αξιολόγησης)</li> <li>• comment (Κείμενο Αξιολόγησης)</li> </ul>
<p>image (εικόνα)</p> <p>Περιγραφή: Στη κεντρική βάση δεδομένων θα πρέπει να αποθηκεύονται τα Url των εικόνων καθώς οι εικόνες θα αποθηκεύονται σε εξωτερικό πάροχο (cloudinary).Επίσης θα αποθηκεύουμε και το δημόσιο Id στο Cloudinary και το id του παιχνιδιού που αφορά η εικόνα.</p>	<ul style="list-style-type: none"> <li>• Id (αναγνωριστικό εικόνας) - <b>PK</b></li> <li>• url (διεύθυνση εικόνας εξωτερικού παρόχου cloudinary)</li> <li>• publicId (Δημόσιο Id Εικόνας στον εξωτερικό πάροχο)</li> <li>• gameId (αναγνωριστικό παιχνιδιού) - <b>FK (game)</b></li> </ul>
<p>order (παραγγελία)</p> <p>Περιγραφή: Στη κεντρική βάση δεδομένων θα πρέπει να</p>	<ul style="list-style-type: none"> <li>• idorder (αναγνωριστικό παραγγελίας) - <b>PK</b></li> <li>• userId (αναγνωριστικό χρήστη) – <b>FK (user)</b></li> <li>• itemsPrice (Καθαρό Κόστος παιχνιδιών),</li> </ul>

αποθηκεύονται τα στοιχεία κάθε παραγγελίας καθώς και ο χρήστης που τον αφορά	<ul style="list-style-type: none"> <li>• taxAmount (Φορολογικό Κόστος),</li> <li>• totalAmount (Συνολικό Κόστος),</li> <li>• discount (έκπτωση)</li> <li>• orderStatus Paied/No Paied (Κατάσταση Παραγγελίας Πληρωμένη/Μη πληρωμένη)</li> </ul>
<p>orderGame (παιχνίδι παραγγελίας)</p> <p>Περιγραφή: Στη κεντρική βάση δεδομένων θα πρέπει να αποθηκεύεται το κάθε παιχνίδι που αγοράζεται, σε ποια παραγγελία και από ποιον καθώς και ο τίτλος του η τιμή και η έκπτωση που εφαρμόστηκε</p>	<ul style="list-style-type: none"> <li>• orderId (αναγνωριστικό παραγγελίας), <b>PK</b></li> <li>• gameId (αναγνωριστικό παιχνιδιού), <b>PK - FK</b> (game)</li> <li>• title (τίτλος παιχνιδιού)</li> <li>• price (τιμή παιχνιδιού)</li> <li>• discount (ποσό έκπτωσης)</li> </ul>
<p>purchasedGame (αγορασμένο παιχνίδι)</p> <p>Περιγραφή: Θα πρέπει να αποθηκεύονται για κάθε χρήστη τα αγορασμένα παιχνίδια και ο χρόνος που τα έπαιξε. Θα μπορούσαν να συμπτυχθούν σε μία οντότητα μαζί με τη orderGame αλλά επιλέγεται να υπάρξει για προαιρετικούς λόγους όπως πχ. στη περίπτωση διαγραφής μιας παραγγελίας να μην διαγράφεται το αγορασμένο παιχνίδι του χρήστη ή να γνωρίζουμε ότι αγοράστηκε το τάδε παιχνίδι από το χρήστη σε οποιαδήποτε περίπτωση</p>	<ul style="list-style-type: none"> <li>• ID (αναγνωριστικό εγγραφής)</li> <li>• purchasedDate (ημερομηνία αγοράς)</li> <li>• Idgame (αναγνωριστικό παιχνιδιού) - <b>PK</b></li> <li>• IDUser (αναγνωριστικό χρήστη) - <b>PK</b></li> <li>• orderId (Αναγνωριστικό παραγγελίας) - <b>FK</b> (order)</li> <li>• TimePlayed (χρόνος που έχει παιχθεί)</li> </ul>
<p>seasonalDiscount (εποχιακή έκπτωση)</p> <p>Στη βάση θα πρέπει να αποθηκεύονται οι ρυθμίσεις για την εποχιακή έκπτωση και το διαφημιστικό Slide. Αποτελεί μία κατάσταση η οποία μπορεί να δημιουργηθεί μία φορά και να ενημερώνεται κάθε φορά. Ανάλογα αν η</p>	<ul style="list-style-type: none"> <li>• ID (αναγνωριστικό εποχιακής έκπτωσης)</li> <li>• category (Κατηγορία)</li> <li>• seasonalDiscountMessage (Μήνυμα εμφάνισης εποχιακής έκπτωσης)</li> <li>• discount (έκπτωση)</li> <li>• Applied (Εφαρμοσμένη/Μη Εκπτωση στα παιχνίδια)</li> <li>• isActive (Ενεργή/Μη Ενεργή)</li> </ul>

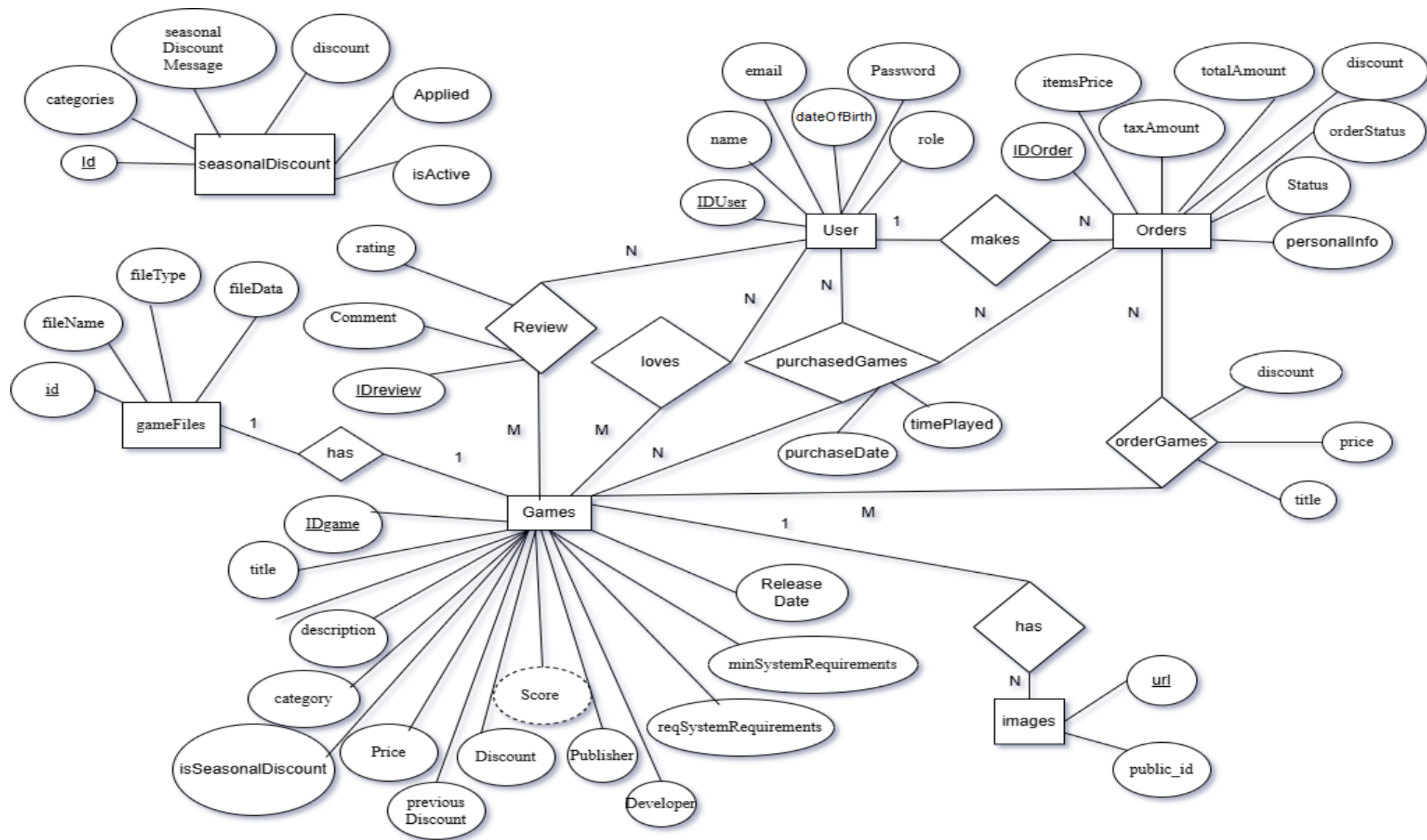


τιμή του πεδίου isActive είναι true(ενεργή) ή false το διαφημιστικό slide εμφανίζεται. Ανάλογα την τιμή του πεδίου Applied φαίνεται αν η εποχιακή έκπτωση έχει εφαρμοστεί στα παιχνίδια.	
--	--

Πίνακας 4.2 Οντότητες και Ιδιότητες CheapGames

### Μοντέλο Οντοτήτων Συσχετίσεων

Στην εικόνα 4.5 μπορούμε να δούμε το μοντέλο ΟΣ.



Εικόνα 4.5 Μοντέλο Οντοτήτων-Συσχετίσεων(MOΣ)

Σε αυτό το σημείο θα κάνουμε κάποιες παραδοχές και σχόλια:

- Η ιδιότητα Score της Οντότητας Games είναι ένα πεδίο το οποίο θα υπολογίζεται ως μέσος όρος των σκορ που θα δίνει ο κάθε χρήστης
- Η οντότητα seasonalDiscount θα είναι μία οντότητα η οποία δεν θα συνδέεται με καμία άλλη οντότητα. Η οντότητα αυτή θα αποτελέσει μία ρύθμιση του συστήματος της οποίας τα δεδομένα θα μπορούσαν να αποθηκεύονται και στον server και όχι στη βάση. Από τη στιγμή που επιλέγεται να αποθηκευτεί στην βάση αποτελεί όμως οντότητα για αυτό και απεικονίζεται στο μοντέλο. Οι ιδιότητες που την αφορούν θα ρυθμίζουν την εμφάνιση ή όχι ενός διαφημιστικού slide για εποχιακές εκπτώσεις σε συγκεκριμένες κατηγορίες που επιλέγει ο διαχειριστής.
- Η ύπαρξη και του purchasedGames και του orderGames θα μπορούσε να συγχωνευθεί σε μία οντότητα αλλά στη περίπτωση που θέλουμε η αγορά να παραμένει ως οικονομικό στοιχείο στη βάση ανεξάρτητα από τη διαγραφή του χρήστη προτιμάται η χρήση και ξεχωριστής οντότητας. Είναι ένα σημείο το οποίο θα μπορούσε να αναθεωρηθεί.
- Η ύπαρξη των πεδίων discount(έκπτωση) και Price στην οντότητα orderGame δεν αποτελεί επανάληψη καθώς θα πρέπει να αποθηκεύεται η τιμή που αγοράστηκε το παιχνίδι και η έκπτωση και όχι να ενημερώνεται ταυτόχρονα σε κάθε αλλαγή τιμής από το διαχειριστή.
- Το σχήμα τηρεί στο βασικό του μέρος (user, game, order, favouriteGame, purchasedGame, review και image) τις αρχές της κανονικοποίησης. Οι μόνες περιπτώσεις που δεν τηρούνται είναι στην περίπτωση της εισαγωγής json δεδομένων σε ένα πεδίο, όπως στο personalInfo στο orders ή στο minRequirements και reqRequirements όπου μπορεί να υπάρξει επανάληψη δεδομένων με τα ίδια δεδομένα. Ο λόγος που δεν εφαρμόστηκε κανονικοποίηση και σε αυτές τις περιπτώσεις είναι η μείωση του βαθμού πολυπλοκότητας του διαγράμματος ΜΟΣ στην εικόνα 4.5.
- Βλέπουμε ότι όλες οι σχέσεις είναι δυαδικού βαθμού εκτός από τη purchasedGames που είναι τριαδική.  
Ο βαθμός μιας σχέσης καθορίζεται από τις οντότητες που σχετίζονται μέσω αυτής. Μια τριαδική σχέση αποτελείται από τρεις οντότητες (Elmasri & Navathe, 2016, p. 73). Σε κάποιες περιπτώσεις μία τριαδική σχέση μπορεί να υλοποιηθεί από τρεις δυαδικές σχέσεις μεταξύ των τριών οντοτήτων και σε κάποιες αποτελεί απόλυτη προσέγγιση (Elmasri & Navathe, 2016, p.88,90).

### 4.3.3 Μοντέλο Embeeded-Reference

Η σχεδίαση του σχήματος (δομή Δεδομένων) σε μια NoSQL βάση δεδομένων όπως η MongoDB, δεν ακολουθεί αυστηρούς κανόνες σχέσεων και οικονομίας αποθηκευτικού χώρου μέσω κανονικοποίησης, όπως συμβαίνει στις σχεσιακές βάσεις, αλλά βασίζεται στη δομή των συλλογών (Collection) και στον τρόπο που η εφαρμογή λειτουργεί και διαχειρίζεται τα δεδομένα (Karlsson, 2022). Σύμφωνα με τις βέλτιστες πρακτικές σχεδίασης της MongoDB, υπάρχουν δύο βασικές προσεγγίσεις:

- Embedded (Ενσωμάτωση εγγράφων): Χρησιμοποιείται όταν τα δεδομένα συχνά ανακτώνται μαζί, καθώς επιτρέπει την ανάκτηση όλων των σχετικών δεδομένων με

ένα μόνο ερώτημα.

- Reference: Εφαρμόζεται όταν η αποθήκευση των δεδομένων μέσα στο ίδιο έγγραφο θα προκαλούσε υψηλή πιθανότητα αύξησης του όγκου των δεδομένων, καθώς τα έγγραφα στη MongoDB έχουν όριο 16MB (Best Practices Guide for MongoDB, 2022).

Για να αποφασίσουμε τι θα ενσωματωθεί ως embedded και τι ως reference μπορούμε να κάνουμε μια σειρά από ερωτήσεις με βάση το ΜΟΣ της εικόνας 4.5. Οι ερωτήσεις αυτές είναι στον πίνακα 4.3.

Ερωτήσεις για Embedded	Ερωτήσεις για Reference
<b>Απλότητα:</b> Η διατήρηση των πληροφοριών μαζί απλοποιεί το μοντέλο δεδομένων και τον κώδικα;	<b>Πληθυσμικότητα :</b> Υπάρχει μεγάλη πληθυσμικότητα στη σχέση (π.χ. ένας χρήστης έχει χιλιάδες παραγγελίες);
<b>Συχνότητα ανάκτησης:</b> Τα δεδομένα αυτά ανακτώνται πάντα μαζί;	<b>Διαχείριση δεδομένων:</b> Θα προκαλούσε η αντιγραφή δεδομένων περιττή πολυπλοκότητα;
<b>Ενημερώσεις:</b> Οι πληροφορίες ενημερώνονται ταυτόχρονα;	<b>Μέγεθος εγγράφου:</b> Τα δεδομένα θα γίνονταν πολύ μεγάλα αν αποθηκευτούν μαζί;
<b>Αρχειοθέτηση:</b> Πρέπει τα δεδομένα να αρχειοθετούνται μαζί;	<b>Ανεξαρτησία:</b> Τα δεδομένα μπορούν να υπάρχουν και χωρίς τον γονέα τους;

Πίνακας 4.3 Ερωτήσεις για απόφαση επιλογής Embedded ή Reference

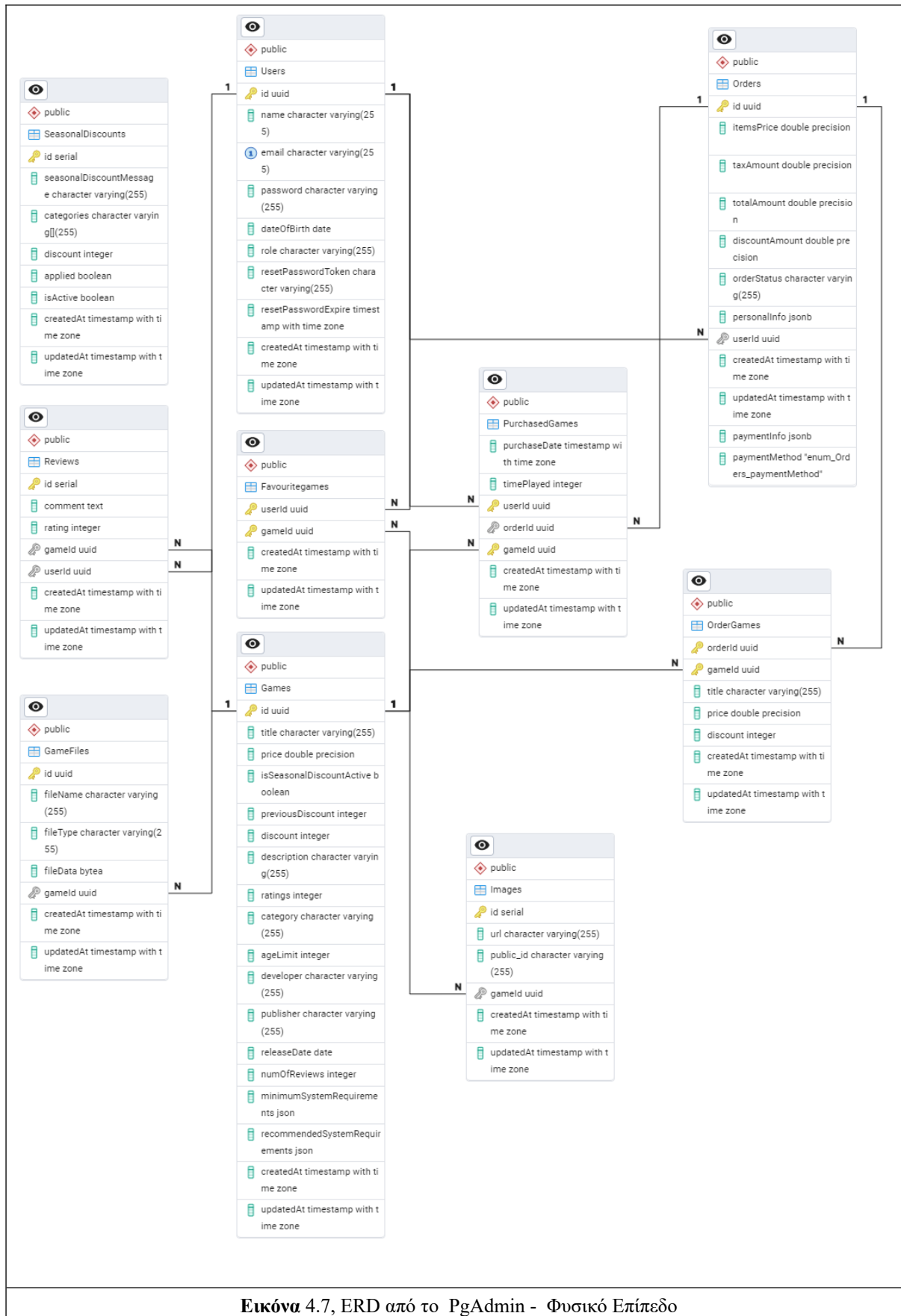
Απαντώντας τις παραπάνω ερωτήσεις (Πίνακας 4.3) και χρησιμοποιώντας το ΜΟΣ της εικόνας 4.5 οδηγούμαστε στον καθορισμό των επιλογών Embedded ή Reference ο οποίος αποτυπώνεται σε ένα Διάγραμμα Embedded-Reference (Εικόνα 4.6).



#### **4.3.4 Φυσικό Επίπεδο**

Στο φυσικό επίπεδο θα δούμε την δομή των δεδομένων, μαζί με τους τύπους αυτών όπως αποθηκεύονται στη κάθε βάση δεδομένων:

- Μέσω του εργαλείου PGADMIN το οποίο προσφέρεται μαζί με την PostgreSQL και χρησιμοποιείται για την διαχείριση της βάσης, έγινε εξαγωγή ενός λεπτομερούς μοντέλου ER ώστε να παρουσιασθεί ο τρόπος αποθήκευσης των δεδομένων στη βάση μαζί με τον τύπο κάθε πεδίου (Εικόνα 4.7).
- Για το φυσικό επίπεδο των δεδομένων δημιουργήθηκε το σχήμα της εικόνας 4.8 όπου μπορούμε να δούμε την δομή των δεδομένων και το τύπο δεδομένων όπως αποθηκεύονται στη MongoDB. Κάθε έγγραφο μέσα σε κάθε collection χαρακτηρίζεται από ένα μοναδικό ID με το οποίο μπορεί γίνεται η αναφορά σε αυτό.



Εικόνα 4.7, ERD από το PgAdmin - Φυσικό Επίπεδο



game	user	order	seasonalDiscount
<pre>{   "_id": "ObjectId",   "title": "string",   "price": "number",   "isSeasonalDiscountActive": "boolean",   "previousDiscount": "number",   "discount": "number",   "description": "string",   "ratings": "number",   "images": [     {       "public_id": "string",       "url": "string"     }   ],   "category": "string",   "ageLimit": "number",   "Developer": "string",   "Publisher": "string",   "ReleaseDate": "string",   "numOfReviews": "number",   "reviews": [     {       "user": "ObjectId",       "rating": "number",       "comment": "string"     }   ],   "user": "ObjectId",   "gameFile": "ObjectId",   "minimumSystemRequirements": {     "OS": "string",     "processor": "string",     "memory": "string",     "graphics": "string",     "storage": "string"   },   "recommendedSystemRequirements": {     "OS": "string",     "processor": "string",     "memory": "string",     "graphics": "string",     "storage": "string"   },   "createdAt": "string",   "updatedAt": "string" }</pre>	<pre>{   "_id": "ObjectId",   "name": "string",   "email": "string",   "password": "string",   "dateOfBirth": "string",   "role": "string",   "resetPasswordToken": "string",   "resetPasswordExpire": "string",   "purchasedGames": [     {       "game": {         "type": "ObjectId",         "ref": "Game"       },       "purchaseDate": "string",       "timePlayed": "number"     }   ],   "favoriteGames": [     {       "type": "ObjectId",       "ref": "Game"     }   ],   "createdAt": "string",   "updatedAt": "string" }</pre>	<pre>{   "id": "ObjectId",   "personalInfo": {     "address": "string",     "city": "string",     "phoneNo": "string",     "zipCode": "string",     "country": "string"   },   "user": {     "type": "ObjectId",     "ref": "User"   },   "orderItems": [     {       "title": "string",       "price": "string",       "discount": "number",       "discountCategory": "number",       "game": {         "type": "ObjectId",         "ref": "Game"       }     }   ],   "paymentMethod": "string",   "paymentInfo": {     "id": "string",     "status": "string"   },   "itemsPrice": "number",   "discountAmount": "number",   "taxAmount": "number",   "totalAmount": "number",   "orderStatus": "string",   "createdAt": "string",   "updatedAt": "string" }</pre>	<pre>{   "_id": "ObjectId",   "seasonalDiscountMessage": "string",   "categories": ["string"],   "discount": "number",   "applied": "boolean",   "isActive": "boolean" }</pre>
	<pre>gameFile.chunk {   "_id": "ObjectId",   "files_id": "ObjectId",   "n": "number",   "data": "Binary" }</pre>		
	<pre>gameFile.file {   "_id": "ObjectId",   "length": "number",   "chunkSize": "number",   "uploadDate": "Date",   "filename": "string" }</pre>		

Εικόνα 4.8, Αποθήκευση Δεδομένων στη MongoDB- Φυσικό Επίπεδο

## 4.4 Διαγράμματα Περιπτώσεων Χρήσης

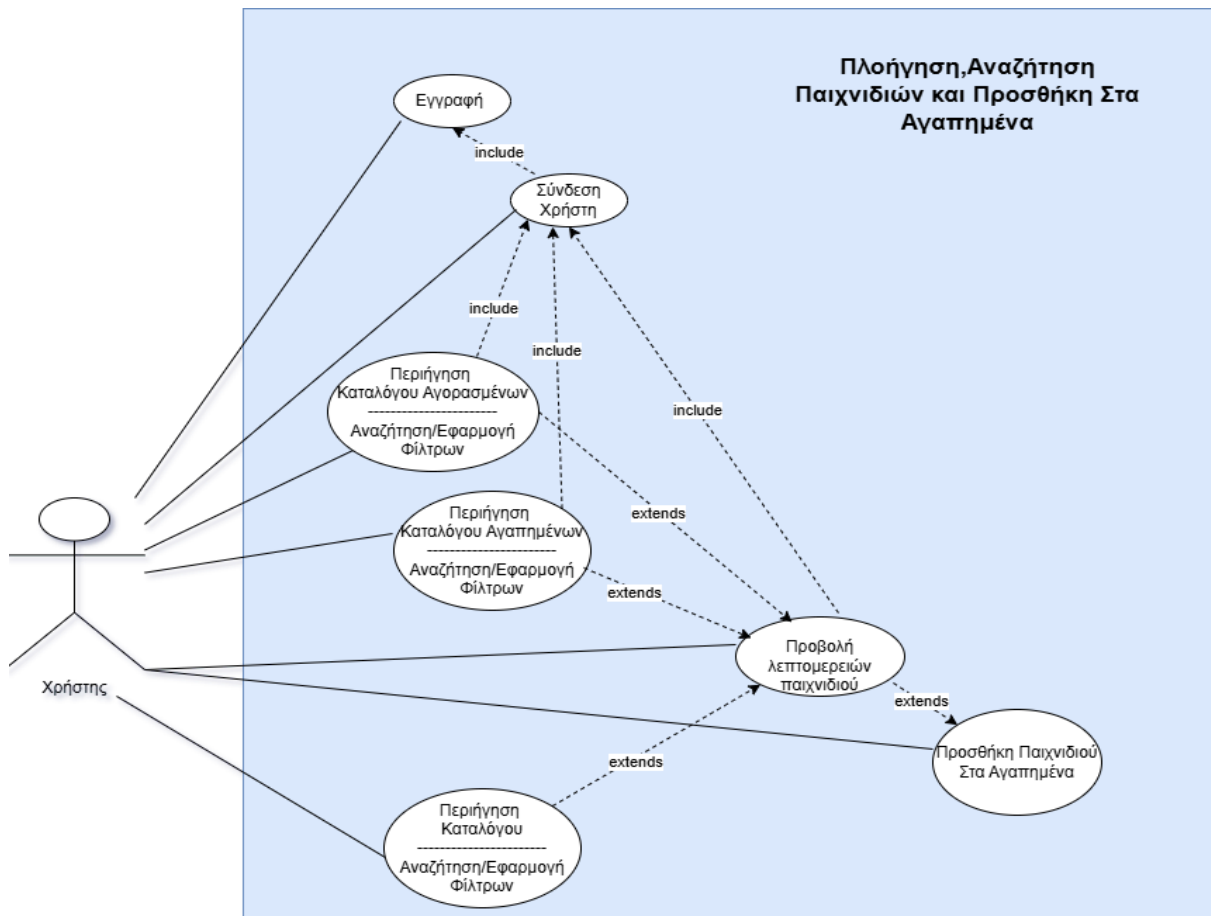
Ένα **Διάγραμμα Περιπτώσεων χρήσης (ΔΠΧ) – Use Case Diagram (UCD)** προτάθηκε από τον Ivar Jacobson το 1986. Μια περίπτωση χρήσης είναι μια μεθοδολογία που χρησιμοποιείται στην ανάλυση συστήματος για να προσδιορίσουμε και να οργανώσουμε τις απαιτήσεις του συστήματος (Aleryani, 2016).

Τα UCD βοήθησαν στην οργάνωση και την αποτύπωση των περιπτώσεων χρήσης. Στη συνέχεια παρουσιάζονται ένα δείγμα από τα διαγράμματα που χρησιμοποιήθηκαν για τις ανάγκες του CheapGames. Επιλέχτηκε οι Περιπτώσεις χρήσης να ομαδοποιηθούν για την καλύτερη αποτύπωση τους.

## Περιπτώσεις Χρήσης

### 1. Πλοήγηση,Αναζήτηση Παιχνιδιών και Προσθήκη Παιχνιδιών Στα Αγαπημένα (Για συνδεδεμένους Χρήστες ανεξάρτητα το ρόλο) (Εικόνα 4.9)

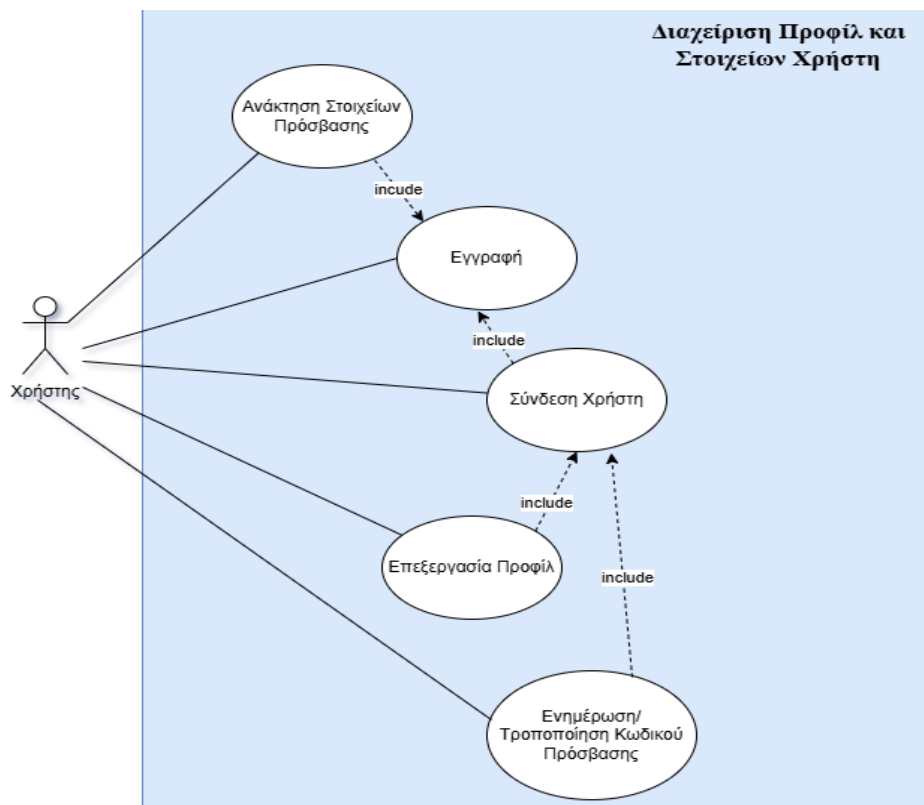
- **PX1:** Περιήγηση στον κατάλογο παιχνιδιών (τίτλος, όριο ηλικίας, τιμή, έκπτωση, σκορ αξιολογήσεων)
- **PX2:** Συνδεση Χρήστη
- **PX3:** Περιήγηση στα αγαπημένα παιχνίδια (τίτλος, όριο ηλικίας, τιμή, έκπτωση, σκορ, πλήθος αξιολογήσεων)
- **PX4:** Περιήγηση στα αγορασμένα παιχνίδια
- **PX5:** Αναζήτηση παιχνιδιών (τίτλος, κατηγορία, φίλτρα τιμής Min/Max, σκορ αξιολογήσεων)
- **PX6:** Προβολή λεπτομερειών παιχνιδιού (περιγραφή, εικόνες, στιγμιότυπα, απαιτήσεις συστήματος, σκορ αξιολογήσεων, αν είναι αγαπημένο, αξιολογήσεις)
- **PX7:** Εγγραφή Χρήστη
- **PX11:** Προσθήκη παιχνιδιού στα Αγαπημένα



Εικόνα 4.9 Διάγραμμα Ομαδοποιημένων Περιπτώσεων χρήσης για Πλοήγηση,Αναζήτηση Παιχνιδιών και Προσθήκη Παιχνιδιών Στα Αγαπημένα (Για Όλους τους Χρήστες)

## 2. Διαχείριση Προφίλ και Στοιχείων Χρήστη και Ανάκτηση Στοιχείων (Εικόνα 4.10)

- ΠΧ2: Σύνδεση Χρήστη
- ΠΧ7: Εγγραφή στο σύστημα (name, email, dateOfBirth, password)
- ΠΧ8: Ανάκτηση στοιχείων πρόσβασης (αποστολή email με σύνδεσμο)
- ΠΧ9: Επεξεργασία προφίλ (τροποποίηση name, email, age, προσθήκη/αλλαγή φωτογραφίας)
- ΠΧ10: Τροποποίηση/Ενημέρωση Κωδικού

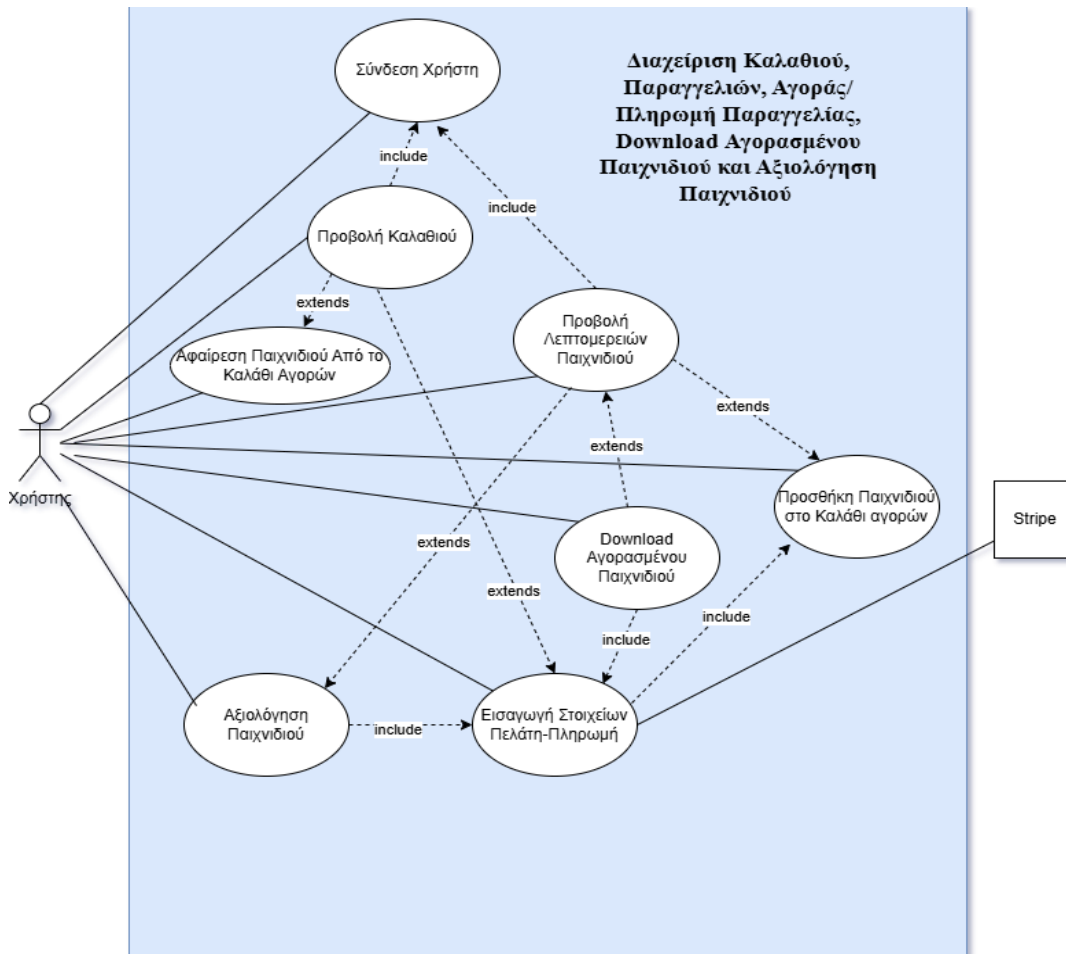


Εικόνα 4.10 Διάγραμμα Ομαδοποιημένων Περιπτώσεων χρήσης για Διαχείριση Προφίλ και Στοιχείων Χρήστη και Ανάκτηση Στοιχείων

## 3. Διαχείριση Καλαθιού, Παραγγελιών, Αγοράς/Πληρωμή Παραγγελίας, Download Αγορασμένου Παιχνιδιού και Αξιολόγηση Παιχνιδιού (Εικόνα 4.11)

- ΠΧ2: Σύνδεση Χρήστη
- ΠΧ6: Προβολή λεπτομερειών παιχνιδιού
- ΠΧ12: Προσθήκη παιχνιδιού στο καλάθι αγορών
- ΠΧ13: Προβολή καλαθιού αγορών (τίτλοι παιχνιδιών, τιμή, συνολική τιμή, πλήθος προϊόντων)
- ΠΧ14: Αφαίρεση παιχνιδιού από το καλάθι

- **PX15:** Εισαγωγή στοιχείων πελάτη (διεύθυνση, τηλέφωνο, χώρα κ.λπ.) και Στοιχείων Κάρτας-Πληρωμή Παραγγελίας
- **PX17:** Download και εθελοντική εισαγωγή χρόνου παιχνίματος αγορασμένου παιχνιδιού
- **PX18:** Αξιολόγηση παιχνιδιού (μόνο αν έχει αγοραστεί από το χρήστη)



**Εικόνα 4.11** Διάγραμμα Ομαδοποιημένων Περιπτώσεων χρήσης για Διαχείριση Καλαθιού, Παραγγελιών, Αγοράς/Πληρωμή Παραγγελίας, Download Αγορασμένου Παιχνιδιού και Αξιολόγηση Παιχνιδιού

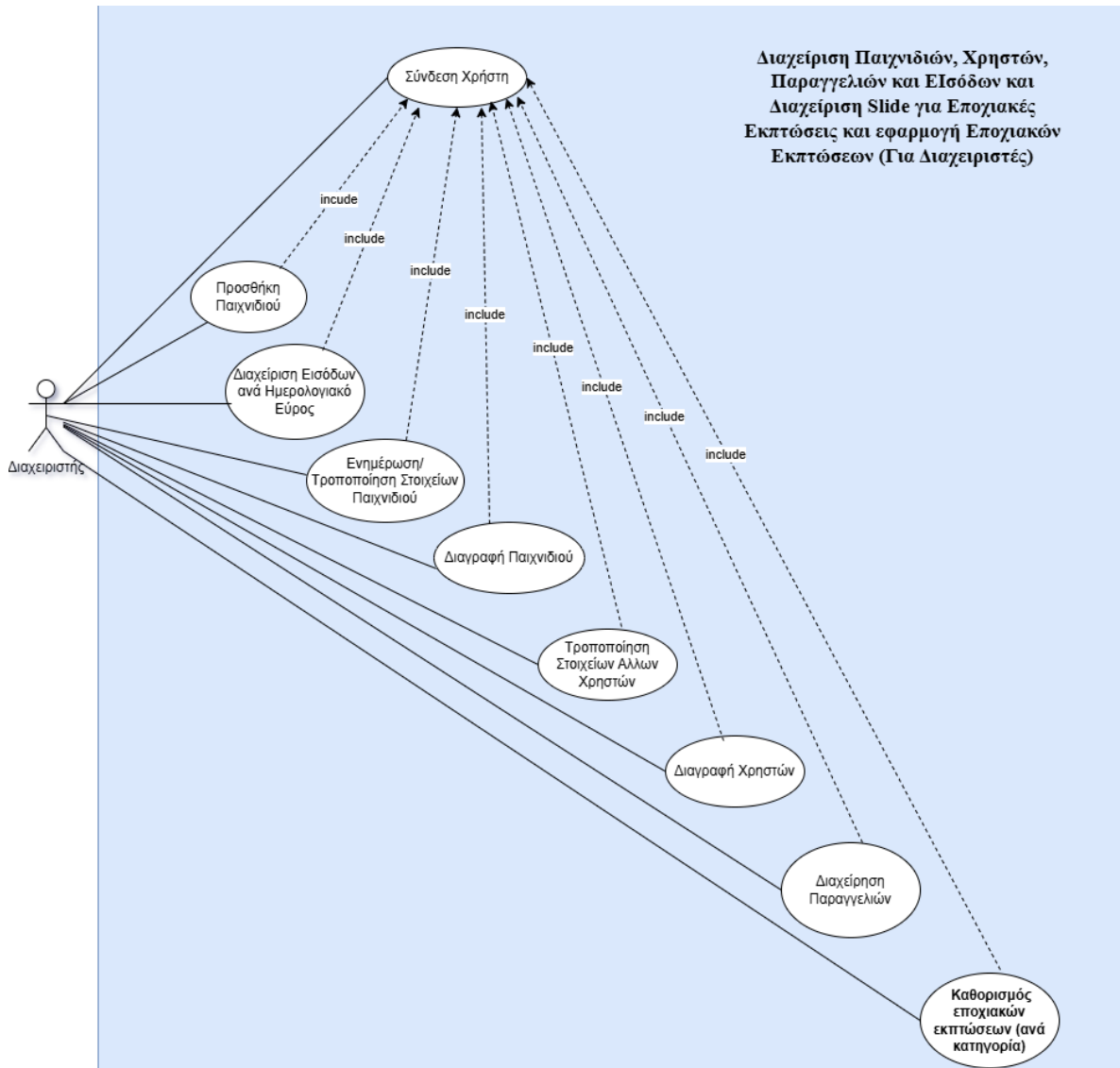
#### 4. Διαχείριση Παιχνιδιών, Χρηστών, Παραγγελιών και Εισόδων και Διαχείριση Slide για Εποχιακές Εκπτώσεις και εφαρμογή Εποχιακών Εκπτώσεων (Για Διαχειριστές) (Εικόνα 4.12)

- **PX19:** Προσθήκη παιχνιδιών (τίτλος, περιγραφή, εικόνες, βίντεο, τιμή, έκπτωση, απαιτήσεις συστήματος, αρχεία παιχνιδιών)
- **PX20:** Ενημέρωση/τροποποίηση Στοιχείων παιχνιδιού

- ΠΧ21: Διαγραφή παιχνιδιού
- ΠΧ22: Τροποποίηση στοιχείων άλλων χρηστών (π.χ αλλαγή ρόλου από χρήστη σε διαχειριστή, διαγραφή)
- ΠΧ23: Διαγραφή χρήστη
- ΠΧ24: Διαχείριση παραγγελιών (έλεγχος, διαγραφή)
- ΠΧ25: Διαχείριση Εσόδων ανα συγκεκριμένο ημερολογιακό διάστημα
- Για το Διάγραμμα περιπτώσεων χρήσης επιλέγεται να υπο-ομαδοποιηθούν οι ΠΧ26,27,28 και 29 οι οποίες αφορούν τις εποχιακές εκπτώσεις για να χωρέσουν στην απεικόνιση του διαγράμματος περιπτώσεων χρήσης

<b>εφαρμογή Εποχιακών Εκπτώσεων</b>	<ul style="list-style-type: none"> <li>• <b>ΠΧ26:</b> Δημιουργία/Ενημέρωση Εποχιακής Εκπτώσης</li> </ul>
	<ul style="list-style-type: none"> <li>• <b>ΠΧ27-</b>Εφαρμογή εποχιακών εκπτώσεων σε συγκεκριμένες κατηγορίες παιχνιδιών</li> </ul>
	<ul style="list-style-type: none"> <li>• <b>ΠΧ28-</b>Επαναφορά εκπτώσεων παιχνιδιών με εφαρμοσμένη την εποχιακή έκπτωση στις αρχικές-προηγούμενες τους τιμές</li> </ul>
	<ul style="list-style-type: none"> <li>• <b>ΠΧ29-</b> Απενεργοποίηση Εμφάνισης Διαφημιστικού Slide</li> </ul>

Πίνακας 4.4 Ομαδοποίηση Περιπτώσεων Χρήσης που αφορούν την Διαχείριση Εποχιακής Εκπτώσης για τις ανάγκες τους διαγράμματος



**Εικόνα 4.12** Διάγραμμα Ομαδοποιημένων Περιπτώσεων Χρήσης για Διαχείριση Παιχνιδιών, Χρηστών, Παραγγελιών και Εισόδων και Διαχείριση Slide για Εποχιακές Εκπτώσεις και εφαρμογή Εποχιακών Εκπτώσεων (Για Διαχειριστές)

## 4.5 Λεκτικές Περιγραφές ΠΧ και Στιγμιότυπα Οθόνης από την Υλοποίηση

Σε αυτό το κεφάλαιο θα δούμε τις λεκτικές περιγραφές των περιπτώσεων χρήσης συνοδευόμενες από τα screenshots της υλοποίησης.

### Περίπτωση Χρήσης Περιήγηση στον κατάλογο παιχνιδιών- Αναγνωριστικό ΠΧ1

#### Όνομα: Περιήγηση Στον Κατάλογο Παιχνιδιών

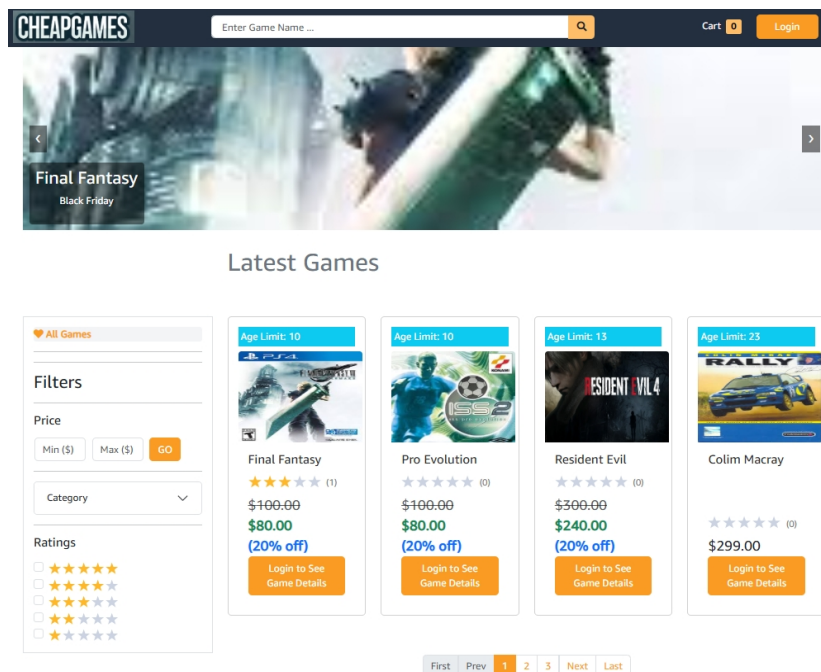
**Περιγραφή:** Ο χρήστης συνδέεται στην σελίδα Cheap Games και βλέπει στην αρχική οθόνη της διαδικτυακής εφαρμογής Cheap Games λογότυπο Cheap Games, Μπάρα αναζήτησης βάση Ονόματος, Κουμπί Login-Σύνδεσης αν δεν είναι συνδεδεμένος ενώ αν είναι βλέπει το όνομα του και κατάλληλο μενού επιλογών ανάλογα τον ρόλο του συνδεδεμένου χρήστη (Αν είναι διαχειριστής πρόσθεση επιλογής Control Panel), Φίλτρα αναζήτησης βάση τιμής, κατηγορίας και Σκορ Αξιολογήσεων, κατάλογος με παιχνίδια σελιδοποιημένα με τα στοιχεία Όριο Ηλικίας, Μία Εικόνα σχετική με το παιχνίδι, Τίτλο ,Σκορ Αξιολογήσεων, Αρχική τιμή Αν υπάρχει, Τελική Τιμή μετά Έκπτωσης, Ποσοστό Έκπτωσης για το καθένα, κουμπιά περιήγησης στις σελίδες του καταλόγου παιχνιδιών και στο τέλος (υποσέλιδο) γενικές πληροφορίες για το Eshop Cheap Games.

#### Βασική ροή γεγονότων:

1. Ο χρήστης συνδέεται στο site
2. Το σύστημα ελέγχει αν ο χρήστης είναι συνδεδεμένος και τον ρόλο του χρήστη στη περίπτωση που είναι

### Εναλλακτική Ροή 1 - Ο χρήστης δεν είναι συνδεδεμένος

**2.α.1** Το σύστημα εμφανίζει την αρχική οθόνη του site λογότυπο Cheap Games, (εικόνα 4.13)



Εικόνα 4.13 Αρχική σελίδα της εφαρμογής Cheap Games για μη συνδεδεμένο χρήστη

### Εναλλακτική Ροή 2 - Ο χρήστης είναι συνδεδεμένος και ο ρόλος του είναι

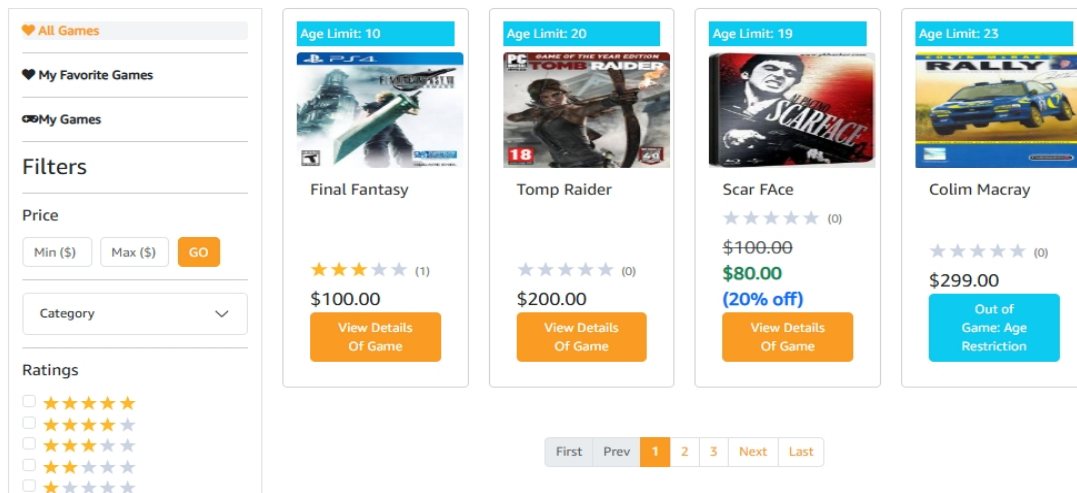


## διαχειριστής

2.α.2 Το σύστημα Το σύστημα εμφανίζει στην αρχική οθόνη του site λογότυπο Cheap Meνού επιλογών που οδηγούν στο Dashboard, στις παραγγελίες, στις Πληροφορίες προφίλ του χρήστη και στην αποσύνδεση του χρήστη, Κουμπιά περιήγησης σε όλα τα παιχνίδια του καταλόγου ή στα αγαπημένα ή στα αγορασμένα (Εικόνα 4.14)



### Latest Games



Εικόνα 4.14 Αρχική σελίδα της εφαρμογής Cheap Games για συνδεδεμένο χρήστη με ρόλο διαχειριστή

**Εναλλακτική Ροή 3- Ο χρήστης είναι συνδεδεμένος και ο ρόλος του είναι απλός εγγεγραμμένος χρήστης, δηλαδή όχι διαχειριστής.**

2.α.3 Το σύστημα εμφανίζει ότι και στην 2.α2 της εναλλακτικής Ροής 2 εκτός από την επιλογή για την Dashboard (Εικόνα 4.15)



Εικόνα 4.15 Απόσπασμα κεφαλίδας Αρχικής σελίδα της εφαρμογής Cheap Games για συνδεδεμένο χρήστη με ρόλο user

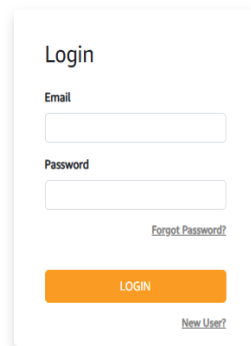
## Περίπτωση Χρήσης Σύνδεση Χρήστη - Αναγνωριστικό ΠΧ2

### Όνομα: Σύνδεση Χρήστη

**Περιγραφή:** Ο χρήστης συνδέεται στη σελίδα Cheap Games και αν δεν είναι συνδεδεμένος από προηγούμενη σύνδεση, επιλέγει το κουμπί login και βλέπει την φόρμα εισαγωγής στοιχείων σύνδεσης όπου και εισάγει τα στοιχεία του.

Βασική Ροή Γεγονότων:

1. Ο χρήστης επιλέγει το κουμπί login
2. Το σύστημα εμφανίζει φόρμα εισαγωγής στοιχείων σύνδεσης (Email και κωδικό) (Εικόνα 4.16)

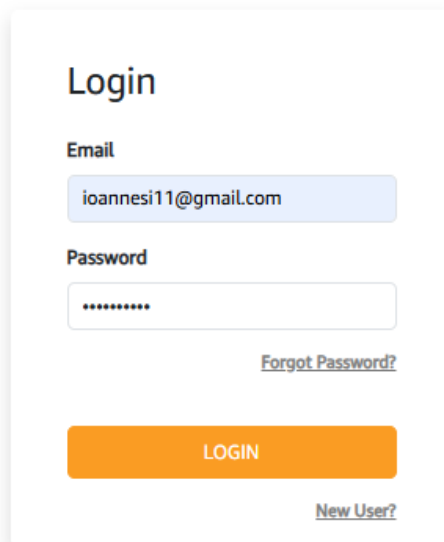
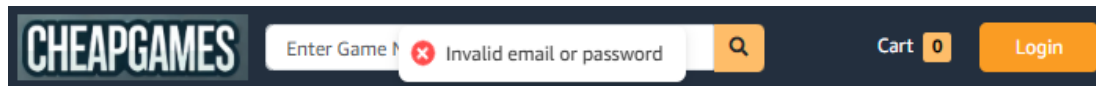


Εικόνα 4.16 Φόρμα Εισαγωγής Στοιχείων Σύνδεσης

3. Το σύστημα ελέγχει τα στοιχεία που εισήγαγε ο χρήστης αν είναι σωστά και στην περίπτωση που είναι ελέγχει το ρόλο του χρήστη
4. Τα στοιχεία είναι σωστά και το σύστημα εμφανίζει την ενημερωμένη αρχική σελίδα..

### Εναλλακτική Ροή Γεγονότων 1-Ο χρήστης έχει εισάγει λάθος στοιχεία σύνδεσης

3.α.1 Το σύστημα ενημερώνει τον χρήστη ότι έχει εισάγει λάθος στοιχεία σύνδεσης (Εικόνα 4.17)



Εικόνα 4.17, Μήνυμα Ειδοποίησης για εισαγωγή μη εγκυρων στοιχείων σύνδεσης

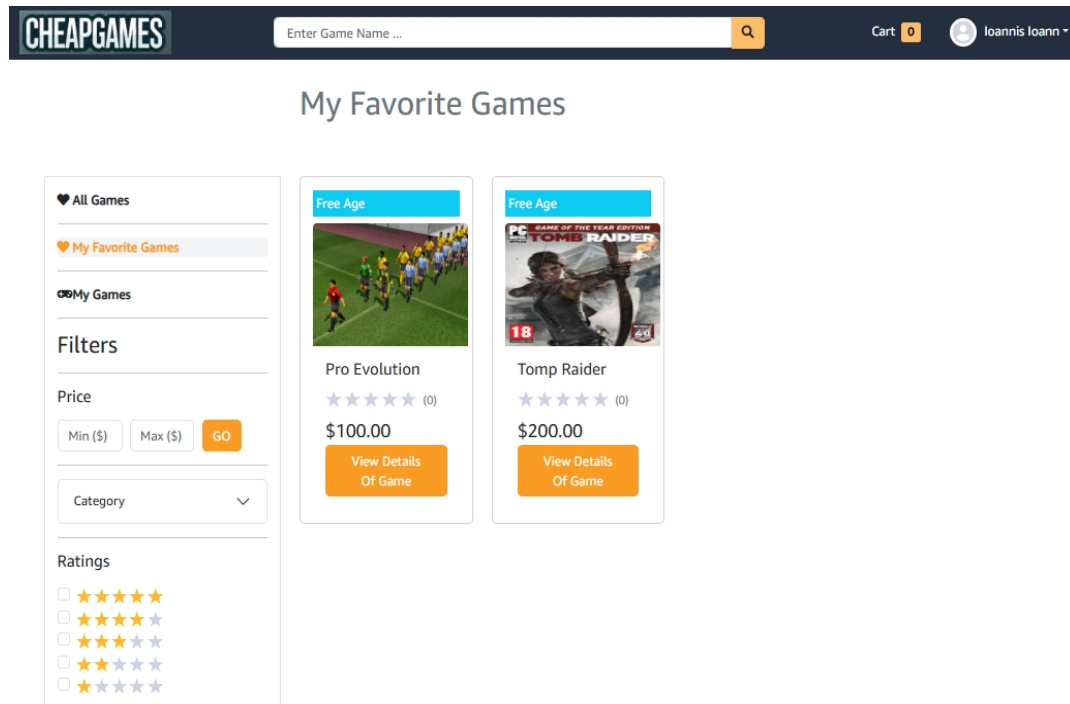
### Περίπτωση Χρήσης Περιήγηση στον κατάλογο αγαπημένων παιχνιδιών - Αναγνωριστικό ΠΧ3

**Όνομα:** Περιήγηση στον κατάλογο αγαπημένων παιχνιδιών

**Περιγραφή:** Ο χρήστης αφού έχει συνδεθεί με τα στοιχεία του εμφανίζεται η επιλογή “My favourite Games” όπου επιλέγοντας την, το σύστημα εμφανίζει τον κατάλογο των αγαπημένων παιχνιδιών.

**Βασική Ροή Γεγονότων:**

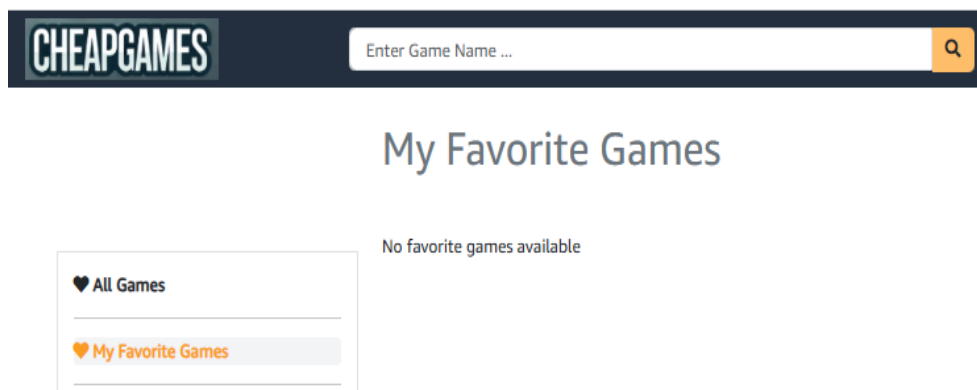
1. Ο χρήστης επιλέγει το κουμπί My Favourite Games
2. Το σύστημα εμφανίζει κατάλογο με τα αγαπημένα παιχνίδια (Εικόνα 4.18)



Εικόνα 4.18 Κατάλογος αγαπημένων παιχνιδιών χρήστη

**Εναλλακτική Ροή Γεγονότων 1-Ο χρήστης δεν έχει προσθέσει κάποιο παιχνίδι στα αγαπημένα**

2.α.1 Το σύστημα ενημερώνει τον χρήστη ότι δεν έχει προσθέσει κάποιο παιχνίδι στα αγαπημένα και ότι ο κατάλογος είναι άδειος (Εικόνα 4.19)



Εικόνα 4.19 Ενημέρωση στον κατάλογο αγαπημένων παιχνιδιών ότι δεν υπάρχουν αγαπημένα παιχνίδια

**Περίπτωση Χρήσης Περιήγηση στον κατάλογο αγορασμένων παιχνιδιών - Αναγνωριστικό ΠΧ4**

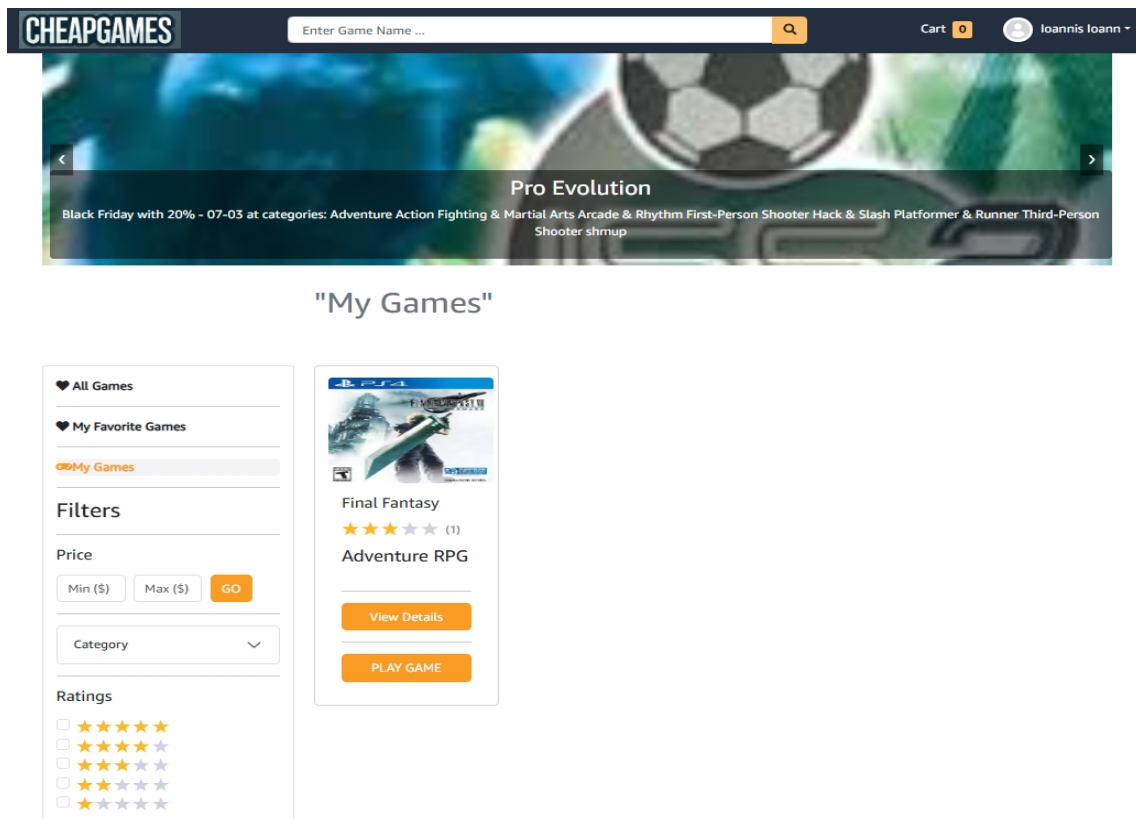
**Όνομα:** Περιήγηση στον κατάλογο αγορασμένων παιχνιδιών

**Περιγραφή:** Ο χρήστης αφού έχει συνδεθεί με τα στοιχεία του εμφανίζεται η επιλογή “My Games” όπου επιλέγοντας την, το σύστημα εμφανίζει τον κατάλογο των αγορασμένων

παιχνιδιών.

#### Βασική Ροή Γεγονότων:

1. Ο χρήστης επιλέγει το κουμπί My Games
2. Το σύστημα εμφανίζει κατάλογο με τα αγορασμένα παιχνίδια και επιλογές View Details και PlayGame για το κάθε παιχνίδι (Εικόνα 4.20)



Εικόνα 4.20 Κατάλογος αγορασμένων παιχνιδιών χρήστη

#### Εναλλακτική Ροή Γεγονότων 1-Ο χρήστης δεν έχει αγοράσει κάποιο παιχνίδι

2.α.1 Το σύστημα ενημερώνει τον χρήστη ότι δεν έχει αγοράσει κάποιο παιχνίδι

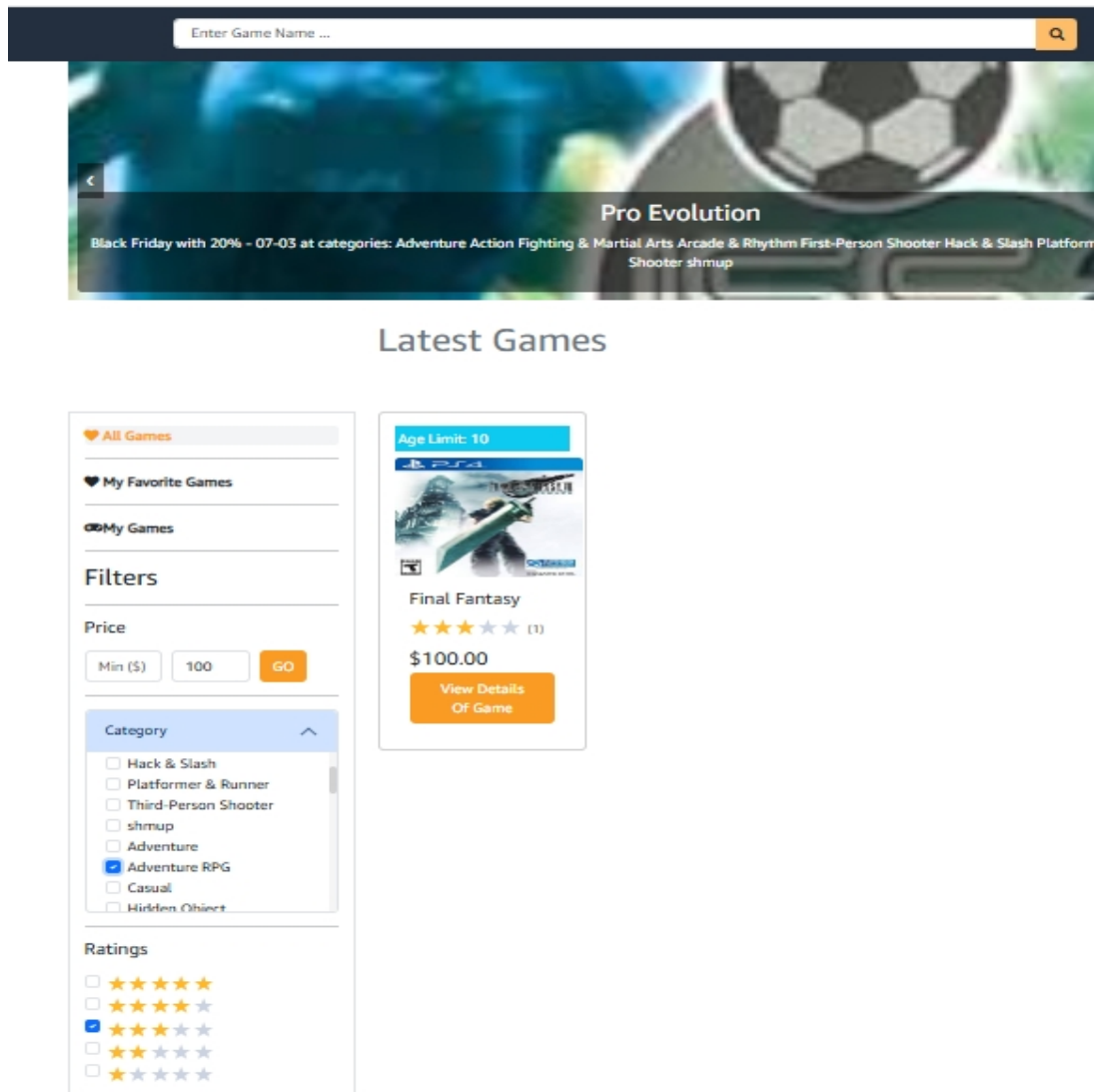
**Περίπτωση χρήσης** Αναζήτηση παιχνιδιών (τίτλος, κατηγορία, φίλτρα τιμής Min/Max, σκορ αξιολογήσεων) -**Αναγνωριστικό ΠΧ5**

**Όνομα:** Αναζήτηση Παιχνιδιών

**Περιγραφή :** Ο χρήστης κάνει τον συνδυασμό που θέλει στα φίλτρα και κάνει την αναζήτηση παιχνιδιών

#### Βασική Ροή Γεγονότων:

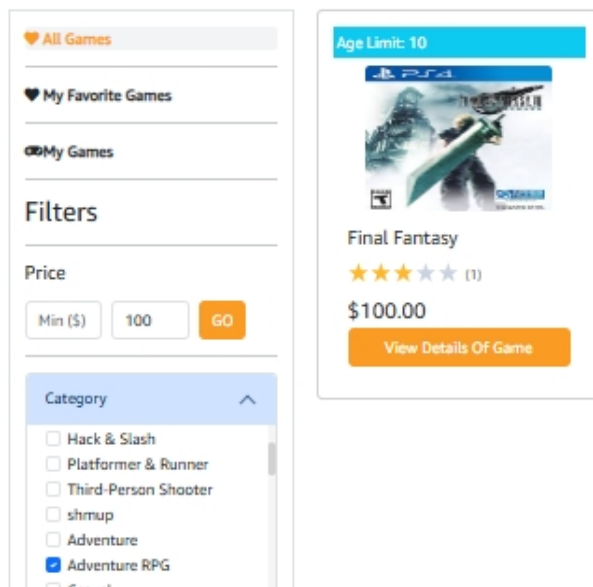
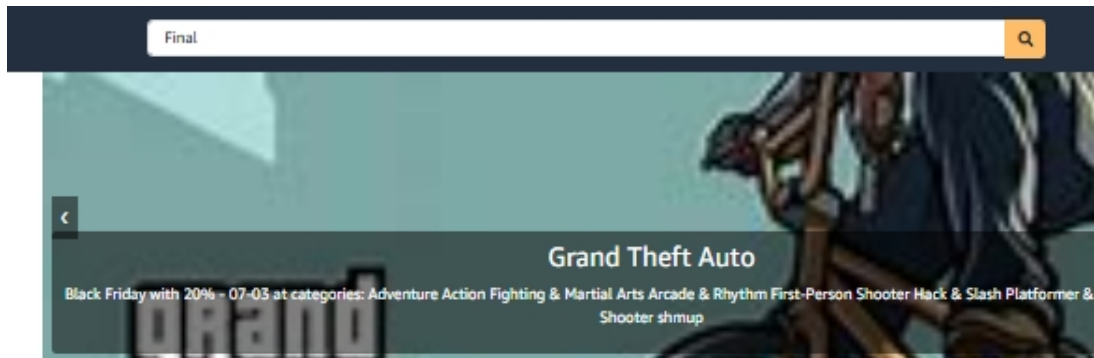
1. Ο χρήστης επιλέγει τον συνδυασμό των φίλτρων που θέλει έχοντας ως επιλογές συνδυασμού την αναζήτηση βάση τίτλου, την αναζήτηση βάση εύρους τιμών, κατηγορία και Min σκορ Αξιολόγησης (Εικόνα 4.21)



Εικόνα 4.21 Στιγμιότυπο Σελίδας Καταλόγου με εισαγωγή Φίλτρων

2. Το Σύστημα εμφανίζει πλήθος παιχνιδιών αν έγινε αναζήτηση και με τίτλο και κατάλογο με τα παιχνίδια που βρέθηκαν (Εικόνα 4.22)





Εικόνα 4.22 Στιγμιότυπο από αναζήτηση στο κατάλογο παιχνιδιών με εισαγωγή τίτλου

**Περίπτωση χρήσης** Προβολή λεπτομερειών παιχνιδιού (περιγραφή, εικόνες, στιγμιότυπα, απαιτήσεις συστήματος, σκορ αξιολογήσεων, αν είναι αγαπημένο, αξιολογήσεις)

#### -Αναγνωριστικό ΠΧ6

**Όνομα:** Προβολή λεπτομερειών παιχνιδιού

**Περιγραφή :** Ο Χρήστης αν είναι συνδεδεμένος έχει τη δυνατότητα να δει περισσότερες λεπτομέρειες για το κάθε παιχνίδι από ότι στο κατάλογο παιχνιδιών κάνοντας κλικ στο τίτλο του παιχνιδιού που μπορεί να βρει στους καταλόγους All Games, My Favourite Games και Mygames




#### **Βασική Ροή Γεγονότων:**

1. Ο χρήστης περιηγείται στους καταλόγους παιχνιδιών και κάνει κλικ σε κάποιο τίτλο παιχνιδιού ή σε κάποια επιλογή View Details (Μόνο για συνδεδεμένους χρήστες)
2. Το σύστημα εμφανίζει το παιχνίδι με περισσότερες πληροφορίες για αυτό (Εικόνα 4.23 και Εικόνα 4.24)



---

---



Iss pRo Age Limit: 8 ♡

675224edb41820058915c619

★★★★★ (1 Reviews)

**\$200** addToCard

Category: **Adventure RPG**

**Description:**

The number of international teams has been increased from the previous release. The teams are still not licensed, although they have their original home, away and goalkeeper kits with emblems and logos resembling their official emblems. In ISS Pro Evolution, for the very first time in the series club teams have been included (there are 16 clubs featured in the game, such as FC Barcelona) along with national teams; however, they could only be played in the new mode Master League, Club teams are named with their respective city names in reference to their real-life equivalents, such as "London" and "Amsterdam" for Arsenal and Ajax, respectively

**Minimum System Requirements:**

- OS: Windows
- Processor: AMD Ryzen
- Memory: 16
- Graphics: Intel
- Storage: Stock

**Recommended System Requirements:**


- OS: Windows
- Processor: Amd Ryzen
- Memory: 16
- Graphics: Intel
- Storage: Stock

Developer: **IoannesI**

Publisher: **IoannesI**

Release Date: **5/12/2024**

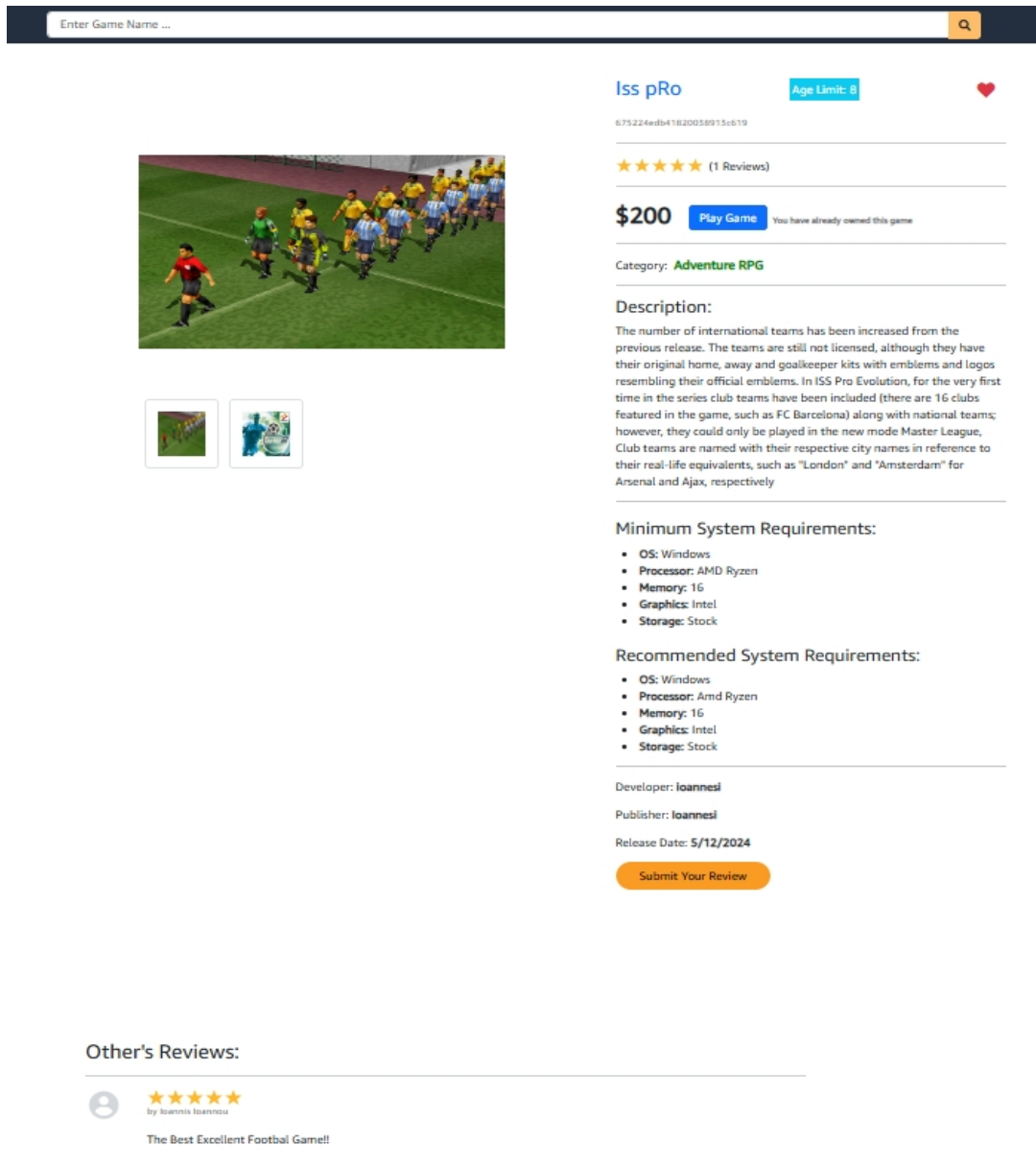
**Other's Reviews:**



★★★★★  
by Ioannis Ioannou

The Best Excellent Football Game!!

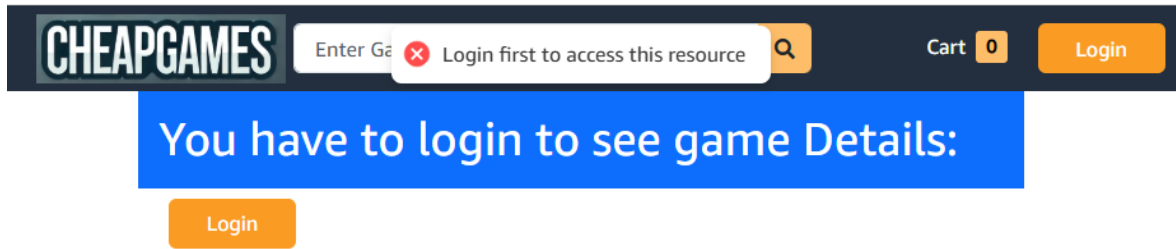
**Εικόνα 4.23** Στιγμιότυπο από λεπτομέρειες παιχνιδιού που δεν έχει αγοραστεί από τον χρήστη



Εικόνα 4.24, Στιγμιότυπο από λεπτομέρειες παιχνιδιού που έχει αγοραστεί

## Εναλλακτική Ροή Γεγονότων 1-Ο χρήστης δεν έχει συνδεθεί στο λογαριασμό του

2.α.1 Το σύστημα εμφανίζει κατάλληλη οθόνη που ενημερώνει τον χρήστη ότι πρέπει να κάνει login για να δει αυτή τη σελίδα και extra επιλογή-κουμπί Login (Εικόνα 4.25)



Εικόνα 4.25, Ενημέρωση χρήστη ότι πρέπει κάνει login για να δει τις λεπτομέρειες ενός παιχνιδιού

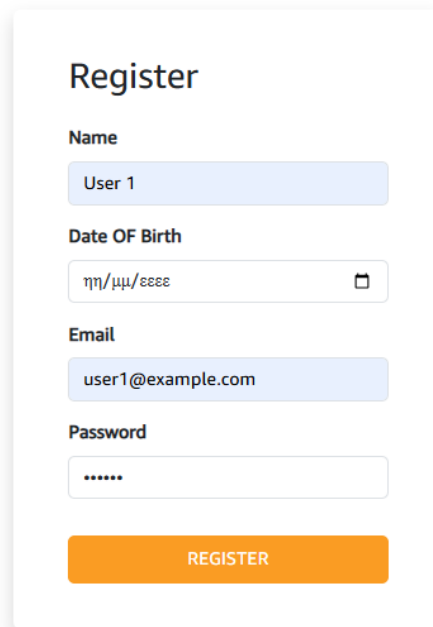
**Περίπτωση χρήσης** Εγγραφή στο σύστημα (name, email, dateOfBirth, password) - Αναγνωριστικό ΠΧ7

**Όνομα:** Εγγραφή στο σύστημα

**Περιγραφή :** Ο χρήστης κάνοντας κλικ στις κατάλληλες επιλογές, εισάγει τα στοιχεία του σε κατάλληλη φόρμα και κάνει εγγραφή-Δημιουργία Λογαριασμού στη Πλατφόρμα

**Βασική Ροή Γεγονότων:**

1. Ο χρήστης κάνει κλικ σε κάποιο login κουμπί
2. Το σύστημα εμφανίζει την οθόνη εισαγωγής στοιχείων σύνδεσης και επιλογές για ανάκτηση στοιχείων πρόσβασης και Δημιουργία Λογαριασμού (New User). (Εικόνα 4.16)
3. Ο χρήστης κάνει κλικ στο κουμπί New User
4. Το σύστημα εμφανίζει κατάλληλη φόρμα εισαγωγής στοιχείων name, dateOfBirth, Email, Password και κουμπί για επικύρωση στοιχείων(Εικόνα 4.26 )



Εικόνα 4.26 Στιγμιότυπο από σελίδα εγγραφής χρήστη

5. Ο χρήστης εισάγει τα στοιχεία του
6. Το σύστημα εμφανίζει την αρχική σελίδα στην οποία ο χρήστης πλέον είναι συνδεδεμένος ως απλός χρήστης (όχι διαχειριστής) έχοντας ένδειξη με το όνομα του και extra επιλογές αποσύνδεσης, προβολής παραγγελιών, προβολής και διαχείρισης προφιλ, και καταλόγους αγαπημένων και αγορασμένων παιχνιδιών (Εικόνα 4.15).

#### Εναλλακτική Ροή Γεγονότων 1- Ο χρήστης δεν έχει εισάγει σωστά δεδομένα

**6.α.1** Το σύστημα εμφανίζει κατάλληλο μήνυμα για το αν κάποιο στοιχείο δεν έχει εισαχθεί σωστά όπως μη σωστή ημερομηνία, email που υπάρχει ήδη, μη αποδεκτός κωδικός

#### Περίπτωση χρήσης Ανάκτηση στοιχείων πρόσβασης (αποστολή email με σύνδεσμο) -Αναγνωριστικό ΠΧ8

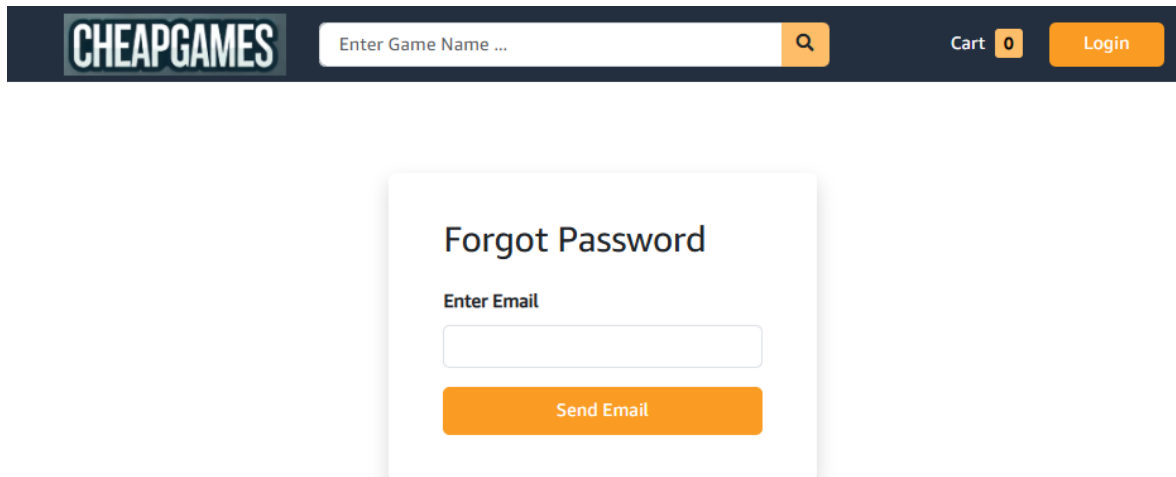
**Όνομα:** Ανάκτηση Στοιχείων

**Περιγραφή :** Ο χρήστης θέλοντας να κάνει ανάκτηση στοιχείων σύνδεσης επιλέγει την ανάκτηση στοιχείων όπου του ζητείται να εισάγει email για να του σταλθεί σύνδεσμος-Link όπου θα εισάγει καινούριο κωδικό πρόσβασης

#### Βασική Ροή Γεγονότων:

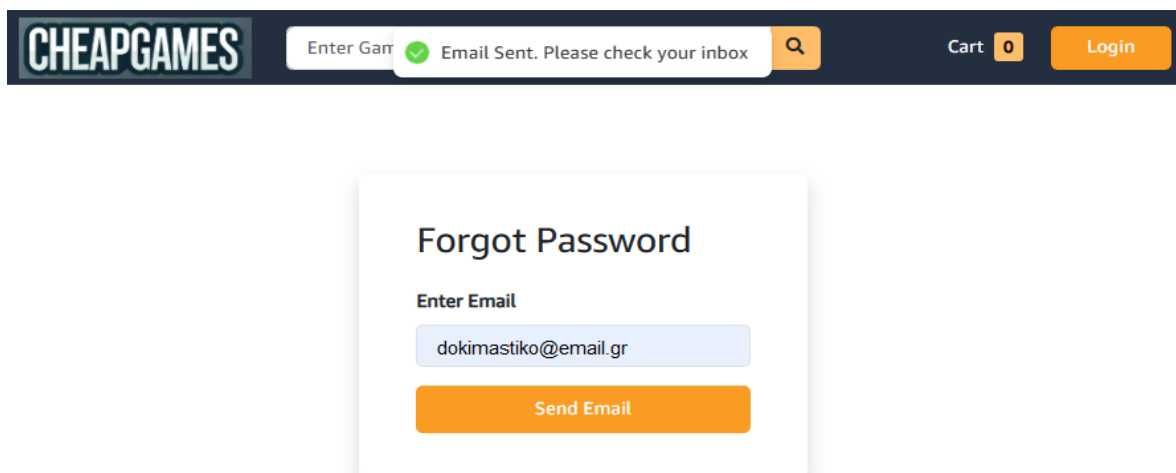
1. Ο χρήστης επιλέγει login στην αρχική οθόνη
2. Το σύστημα εμφανίζει την οθόνη εισαγωγής στοιχείων σύνδεσης και επιλογές για ανάκτηση στοιχείων (Forgot Password) και Δημιουργία καινούριου Λογαριασμού (New User) (Εικόνα 4.16).

3. Ο χρήστης κάνει κλικ στο κουμπί Forgot Password
4. Το σύστημα εμφανίζει κατάλληλη φόρμα εισαγωγής email (Εικόνα 4.27)



Εικόνα 4.27 Στιγμιότυπο από σελίδα ανάκτησης στοιχείων

5. Ο χρήστης εισάγει την διεύθυνση email του
6. Το σύστημα ενημερώνει τον χρήστη με κατάλληλο μήνυμα πως το link στάλθηκε και ο χρήστης λαμβάνει το email με το link στη διεύθυνση του (Εικόνα 4.28)



Εικόνα 4.28 Στιγμιότυπο από εμφάνιση ειδοποίησης αποστολής email με link για ανάκτηση στοιχείων

7. Ο χρήστης κάνει κλικ στον υπερσύνδεσμο-link (Εικόνα 4.29)

**Hi Ioannis Ioannou,**

You recently requested to reset your password for your CheapGames account. Use the button below to reset it. **This password reset is only valid for the next 30 minutes.**

Reset your password

If you did not request a password reset, please ignore this email or [contact support](#) if you have questions.

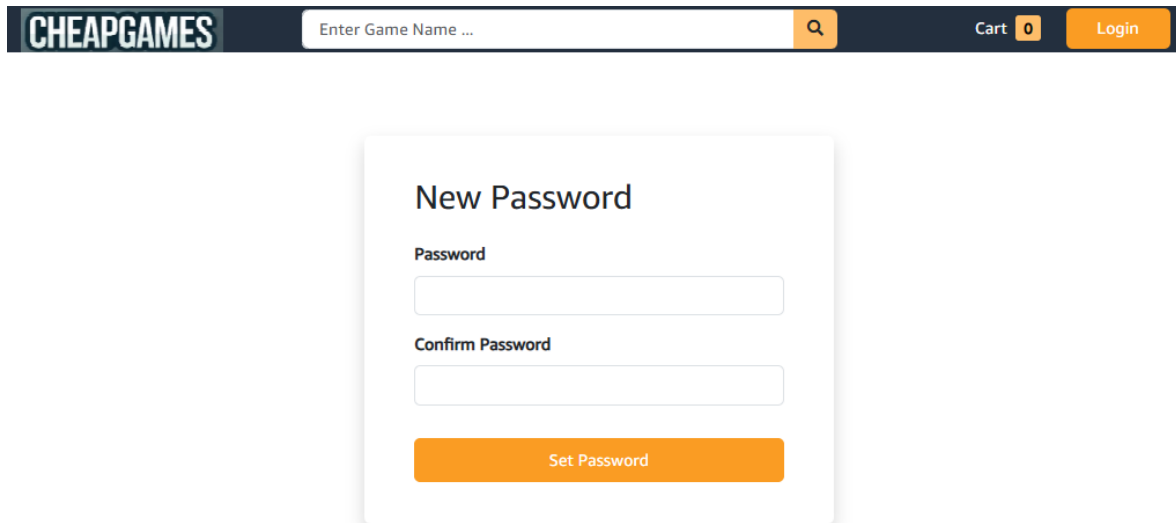
Thanks,  
The CheapGames team

If you're having trouble with the button above, copy and paste the URL below into your web browser.

<http://localhost:3000/password/reset/4389c86da349aa9c3fe9b6f348630aa6505f18c7>

Εικόνα 4.29, Στιγμιότυπο από Email Ανάκτησης Κωδικού

8. Το link οδηγεί τον χρήστη σε κατάλληλη σελίδα του site με φόρμα εισαγωγής κωδικού, επικύρωσης κωδικού και κουμπιού Ολοκλήρωσης (Εικόνα 4.30)

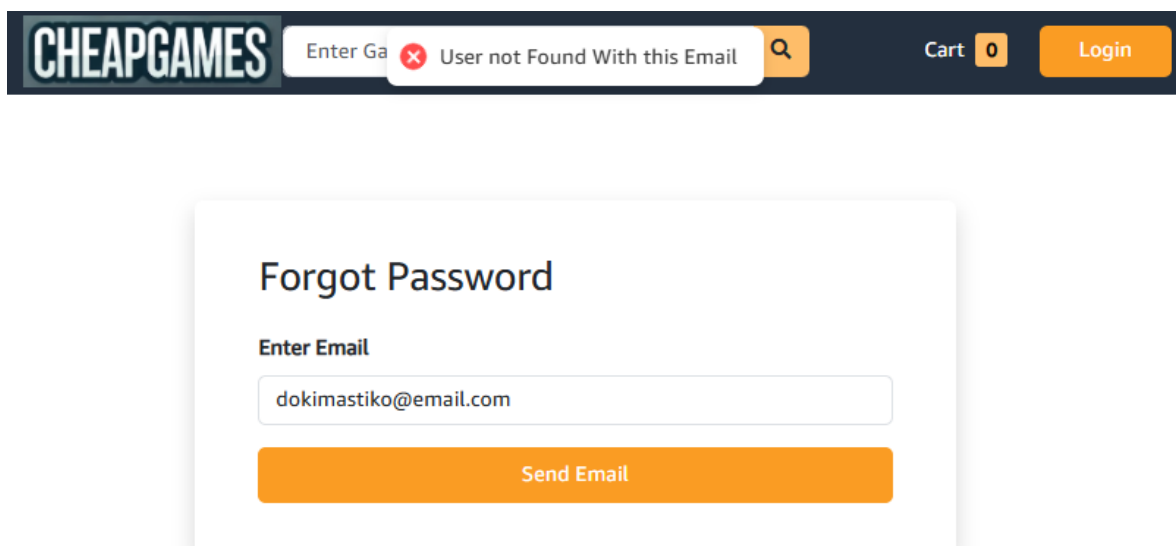


Εικόνα 4.30 Στιγμιότυπο από σελίδα εισαγωγής Νέου Κωδικού

9. Ο χρήστης αφού εισάγει τον καινούριο κωδικό σωστά κάνει κλικ στο Κουμπί Ολοκλήρωσης της διαδικασίας
10. Το σύστημα εμφανίζει την οθόνη Εισαγωγής στοιχείων Σύνδεσης (Εικόνα 4.16)

#### Εναλλακτική Ροή Γεγονότων 1-Ο χρήστης εισάγει email που δεν υπάρχει στη βάση δεδομένων

6.α.1 Το σύστημα ενημερώνει τον χρήστη με κατάλληλο μήνυμα πως δεν υπάρχει χρήστης με τέτοιο email (Εικόνα 4.31).

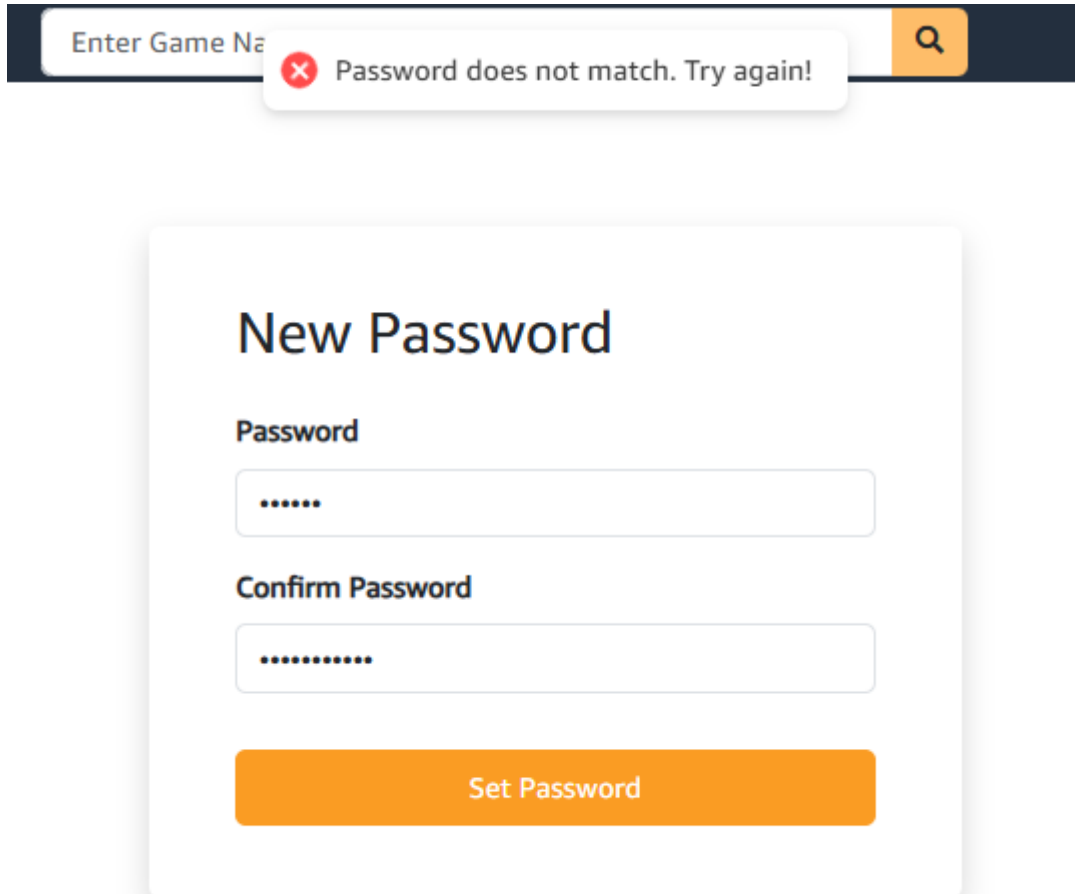


Εικόνα 4.31 Στιγμιότυπο από εμφάνιση ειδοποίησης για μη καταχωρημένο email στη βάση Δεδομένων



## Εναλλακτική Ροή Γεγονότων 2-Ο χρήστης εισάγει μη αποδεκτό κωδικό ή διαφορετικό κωδικό στα δύο σημεία εισαγωγής και επιβεβαίωσης κωδικού

9.α.1 Το σύστημα εμφανίζει κατάλληλο μήνυμα μη αποδεκτού κωδικού ή ότι έχει εισαχθεί λάθος κωδικός στο πεδίο Επιβεβαίωσης Κωδικού (Εικόνα 4.32)



The image shows a web interface for setting a new password. At the top, there is a dark blue header with the text "Enter Game Name" and a search icon. Below this, a white error message box with a red 'x' icon states "Password does not match. Try again!". The main form is titled "New Password" and contains two input fields: "Password" and "Confirm Password". Both fields are filled with dots, indicating masked text. Below the fields is a large orange button labeled "Set Password".

Εικόνα 4.32 Εμφάνιση Ειδοποίησης μη επιτυχούς εισαγωγής Κωδικού

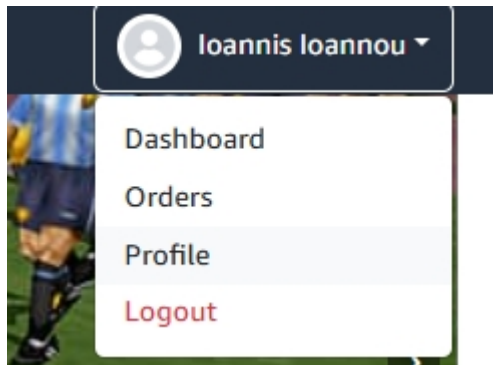
**Περίπτωση χρήσης** Επεξεργασία προφίλ (τροποποίηση name, email, dateOfBirth, προσθήκη/αλλαγή φωτογραφίας)-**Αναγνωριστικό ΠΧ9**

**Ονομα:** Επεξεργασία προφίλ

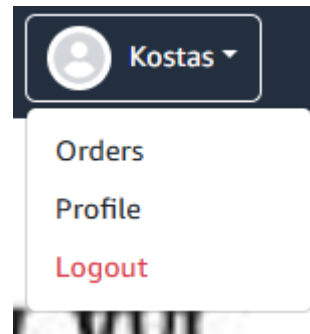
**Περιγραφή :** Ο χρήστης έχοντας κάνει σύνδεση επιλέγει να αλλάξει κάποιο από τα στοιχεία name, email, dateOfBirth.

**Βασική Ροή Γεγονότων:**

1. Ο χρήστης αφού έχει κάνει εισαγωγή στοιχείων σύνδεσης επιλέγει από το μενού επιλογών την επιλογή profile (Εικόνα 4.33 και Εικόνα 4.34).

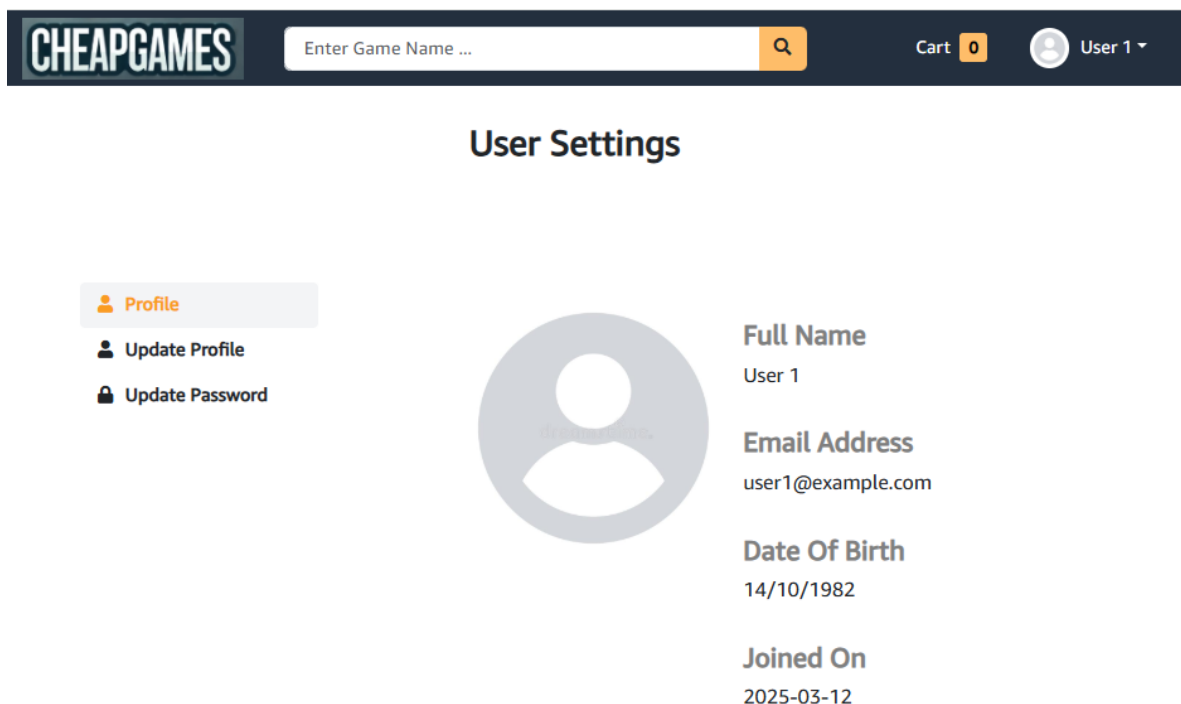


**Εικόνα 4.33** Μενού επιλογών για συνδεδεμένο χρήστη με ρόλο διαχειριστή (admin)



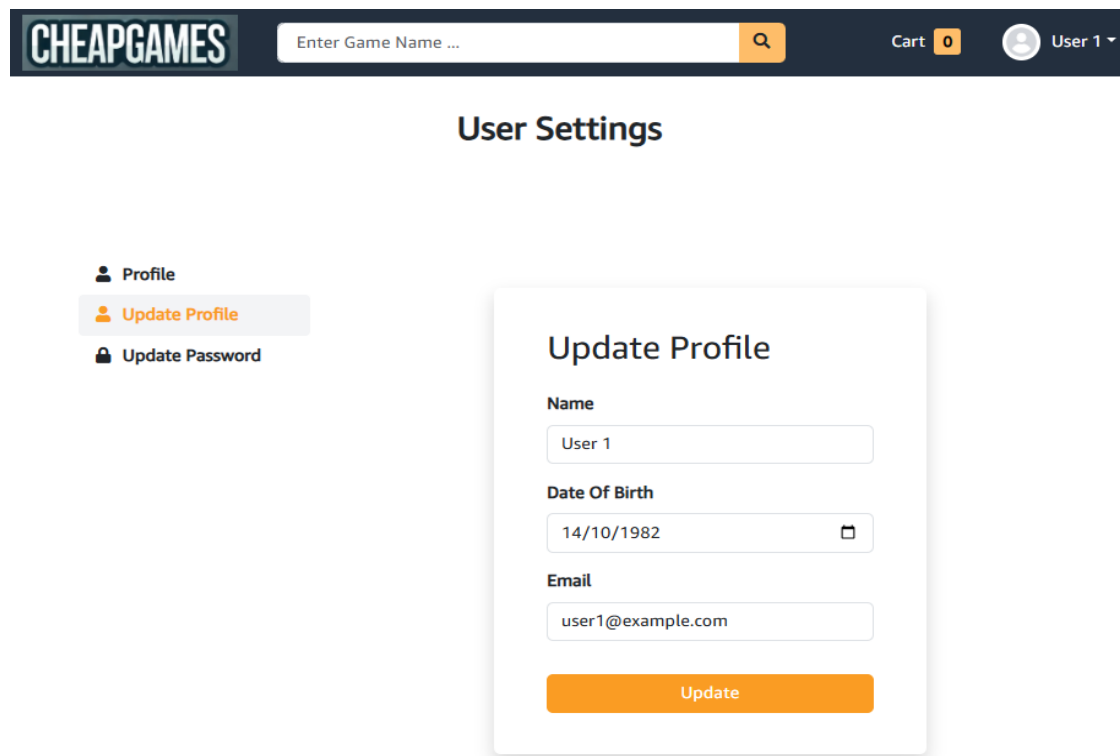
**Εικόνα 4.34** Μενού Επιλογών για συνδεδεμένο χρήστη με ρόλο απλού χρήστη (user)

2. Το σύστημα εμφανίζει στον χρήστη σελίδα με τα στοιχεία του name, email, dateOfBirth, τότε έκανε εγγραφή και μενού επιλογών Τροποποίησης στοιχείων Πρόσβασης (Update Profile), και αλλαγής κωδικού (Update Password) (Εικόνα 4.35).



**Εικόνα 4.35** Μενού Επιλογών Τροποποίησης Προφίλ Χρήστη

3. Ο χρήστης επιλέγει το Update Profile
4. Το σύστημα εμφανίζει προσυμπληρωμένη φόρμα εισαγωγής στοιχείων name, dateOfBirth και email του χρήστη και κουμπί επιβεβαίωσης (Εικόνα 4.36)



**CHEAPGAMES** Enter Game Name ... Cart 0 User 1

### User Settings

- Profile
- Update Profile**
- Update Password

#### Update Profile

**Name**  
User 1

**Date Of Birth**  
14/10/1982

**Email**  
user1@example.com

Update

Εικόνα 4.36 Σελίδα Τροποποίηση Στοιχείων Χρήστη

5. Ο χρήστης τροποποιεί τα στοιχεία που θέλει να αλλάξει και κάνει κλικ στο κουμπί επιβεβαίωσης
6. Το σύστημα εμφανίζει ξανά την οθόνη του γεγονότος 2 μαζί με κατάλληλο μήνυμα ότι η τροποποίηση ήταν επιτυχής

#### Εναλλακτική Ροή Γεγονότων 1- Ο χρήστης εισάγει λάθος κάποιο από τα στοιχεία του λογαριασμού του όπως εισαγωγή email

**6.α.1** Ο χρήστης εισάγει μη σωστά στοιχεία και το σύστημα ενημερώνει τον χρήστη για τα λάθος στοιχεία με κατάλληλα μηνύματα

#### Περίπτωση χρήσης Ενημέρωση/Τροποποίηση Κωδικού Πρόσβασης -Αναγνωριστικό ΠΧ10

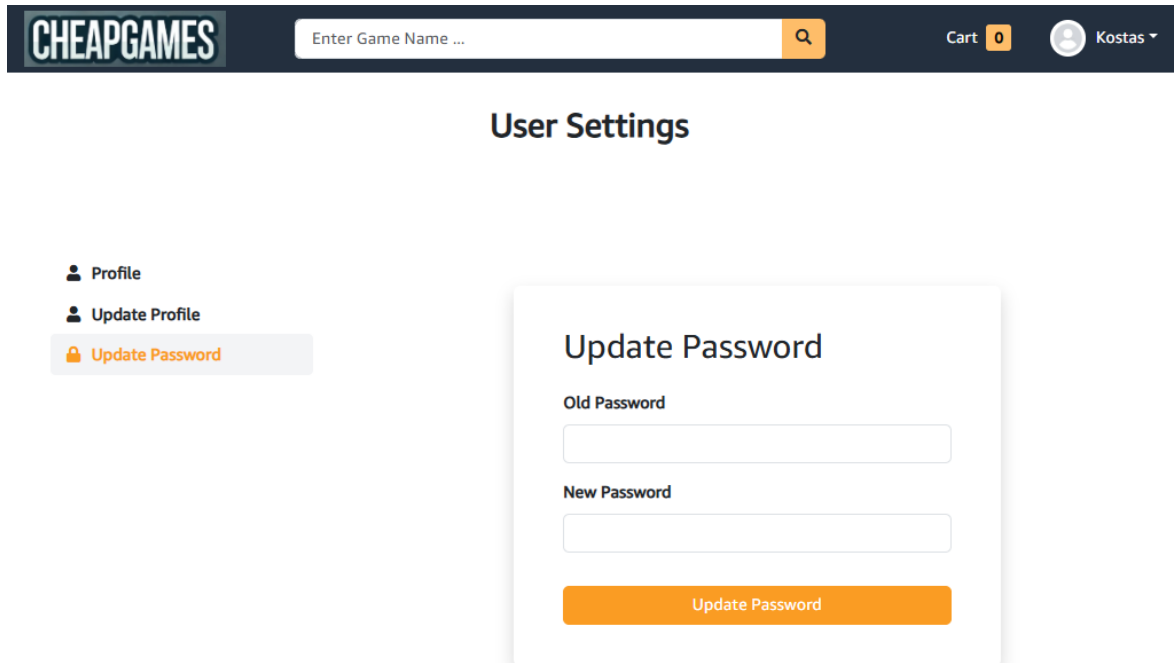
**Όνομα:** Τροποποίηση Κωδικού Πρόσβασης

**Περιγραφή :** Ο χρήστης έχοντας συνδεθεί με τα στοιχεία πρόσβασης στο λογαριασμό του, κάνει κλικ στην επιλογή Profile και στη συνέχεια στην επιλογή Update Password για να αλλάξει τον κωδικό του.

#### Βασική Ροή Γεγονότων:

1. Ο χρήστης αφού έχει κάνει εισαγωγή στοιχείων σύνδεσης, επιλέγει από το μενού επιλογών την επιλογή profile (Εικόνα 4.15)
2. Το σύστημα εμφανίζει στον χρήστη σελίδα με τα στοιχεία του name, email, dateOfBirth, τότε έκανε εγγραφή και μενού επιλογών Τροποποίησης στοιχείων

- Πρόσβασης (Update Profile) και αλλαγής κωδικού (Update Password) (Εικόνα 4.28)
3. Ο χρήστης επιλέγει το Update Password
  4. Το σύστημα εμφανίζει κατάλληλη φόρμα εισαγωγής παλιού κωδικού και καινούριο (Εικόνα 4.37)



Εικόνα 4.37 Σελίδα Ενημέρωσης Κωδικού

5. Ο χρήστης εισάγει τον παλιό και τον καινούριο κωδικό στα πεδία και πατάει στην Επιβεβαίωση (Update Password)
6. Το σύστημα εμφανίζει ξανά την οθόνη του γεγονότος 2 μαζί με κατάλληλο μήνυμα ότι η τροποποίηση ήταν επιτυχής.

#### Εναλλακτική Ροή Γεγονότων 1- Ο χρήστης εισάγει μη σωστό παλιό κωδικό ή αποδεκτούς κωδικούς

- 6.α.1 Το σύστημα εμφανίζει καταλληλα μηνύματα με το λάθος του χρήστη

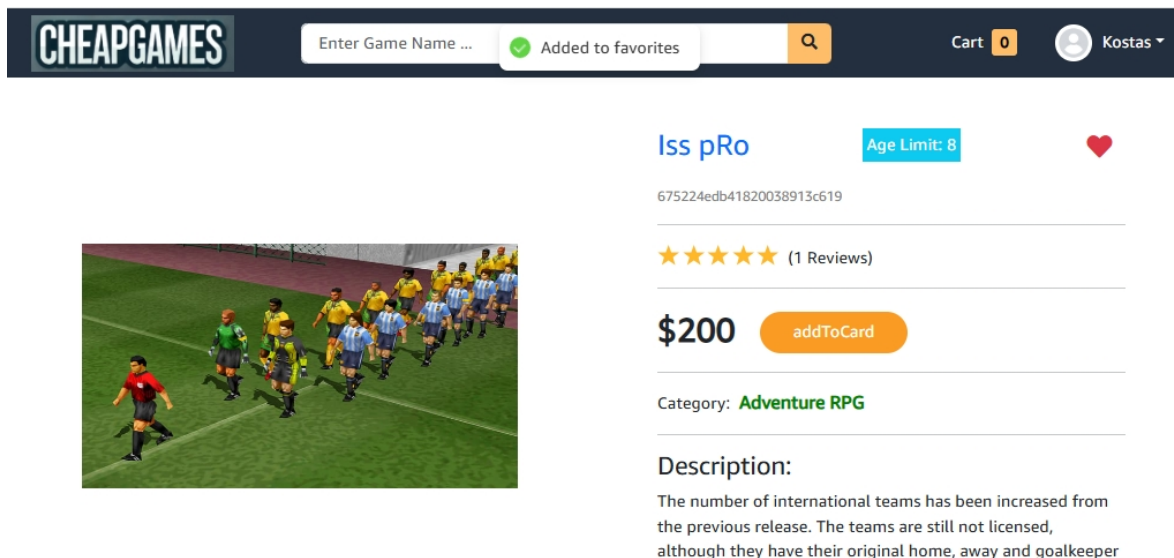
#### Περίπτωση χρήσης Προσθήκη/Αφαίρεση παιχνιδιού από τα αγαπημένα παιχνίδια -Αναγνωριστικό ΠΧ11

**Ονομα:** Προσθήκη/Αφαίρεση αγαπημένου παιχνιδιού

**Περιγραφή :**Ο χρήστης αφού μπει στις λεπτομέρειες κάποιου παιχνιδιού μπορεί να εισάγει ή να εξάγει το συγκεκριμένο παιχνίδι από τα αγαπημένα του

#### Βασική Ροή Γεγονότων:

1. Εκτελείται επιτυχώς η ΠΧ6 Προβολή λεπτομερειών παιχνιδιού (Εικόνα 4.23)
2. Ο χρήστης βλέπει αν το παιχνίδι είναι κλικαρισμενο ως αγαπημενο και αν δεν είναι το κλικάρει για να γίνει ή το κλικάρει για να βγει από το κατάλογο των αγαπημένων
3. Το σύστημα εμφανίζει κατάλληλο μήνυμα Επιτυχούς Πρόσθεσης ή αφαίρεσης παιχνιδιού στα/από αγαπημένα (Εικόνα 4.38)



Εικόνα 4.38 Στιγμιότυπο εμφάνισης ειδοποίησης επιτυχούς πρόσθεσης παιχνιδιού στα αγαπημένα του χρήστη

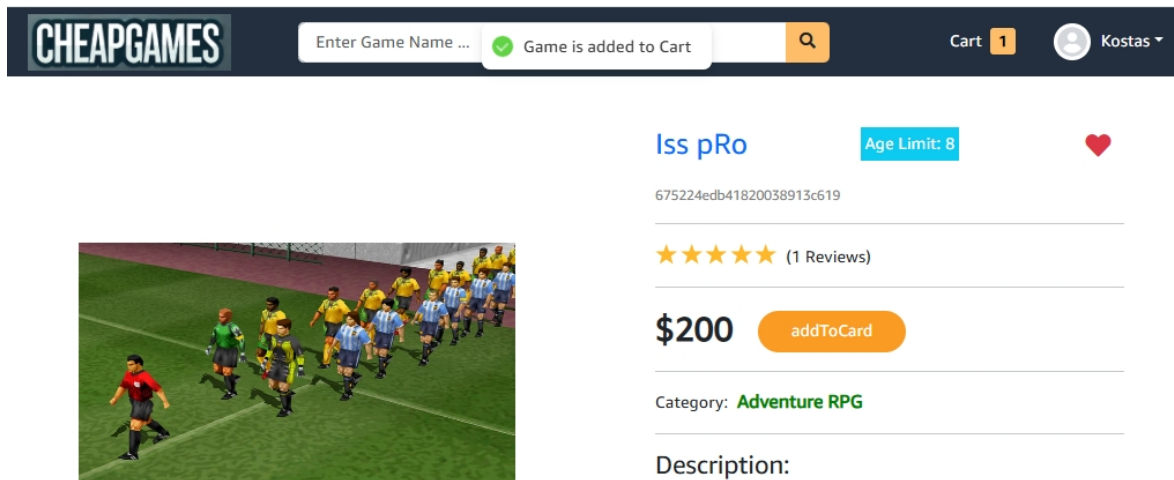
### Περίπτωση χρήσης Προσθήκη παιχνιδιού στο καλάθι αγορών -Αναγνωριστικό ΠΧ12

**Όνομα:** Προσθήκη παιχνιδιού στο καλάθι αγορών

**Περιγραφή :** Ο Χρήστης αφού έχει συνδεθεί με τα στοιχεία σύνδεσης του και έχει εισέλθει στις λεπτομέρειες κάποιου παιχνιδιού μπορεί να προσθέσει το παιχνίδι αυτό στο καλάθι αγορών

#### Βασική Ροή Γεγονότων:

1. Εκτελείται επιτυχώς η ΠΧ6 Προβολή λεπτομερειών παιχνιδιού
2. Το σύστημα αν ο χρήστης δεν έχει αγοράσει το παιχνίδι, εμφανίζει την επιλογή Πρόσθεση παιχνιδιού στο καλάθι αγοράς (add to cart) (Εικόνα 4.23)
3. Ο χρήστης κάνει κλικ στην επιλογή add to cart.
4. ΤΟ σύστημα εμφανίζει κατάλληλο μήνυμα επιτυχούς πρόσθεσης παιχνιδιού στο καλάθι αγορών και ένδειξη με πλήθος προϊόντων που έχουν προστεθεί στο καλάθι (Εικόνα 4.39).



Εικόνα 4.39 Πρόσθεση Παιχνιδιού στο καλάθι αγορών

### Εναλλακτική Ροή Γεγονότων 1 -Ο χρήστης έχει αγοράσει ήδη το παιχνίδι αυτό

2.α.1 Το σύστημα αντί για την επιλογή Πρόσθεσης Παιχνιδιού στο καλάθι αγορών (add to cart) , εμφανίζει την ένδειξη ότι ο χρήστης έχει αγοράσει αυτό το παιχνίδι

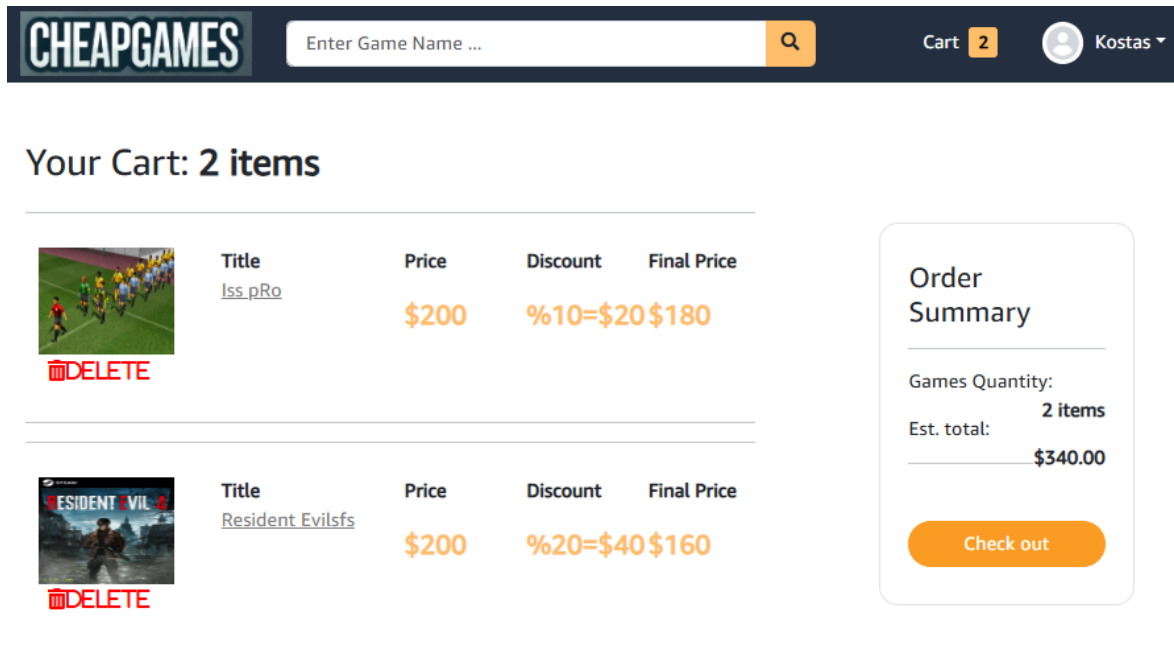
**Περίπτωση χρήσης** Προβολή καλαθιού αγορών (τίτλοι παιχνιδιών, τιμή, συνολική τιμή, πλήθος προϊόντων)-**Αναγνωριστικό ΠΧ13**

**Ονομα:** Προβολή Καλαθιού αγορών

**Περιγραφή :** Ο χρήστης αφού έχει συνδεθεί επιτυχώς στο λογαριασμό του και αφού έχει προσθέσει παιχνίδια στο καλάθι του, κλικάρει στο κουμπί Cart για να δει τα προϊόντα που έχει προσθέσει.

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η ΠΧ12 Προσθήκη Παιχνιδιού στο Καλάθι Αγορών
2. Ο χρήστης από οποιαδήποτε φάση του site μπορεί να επιλέγει το κουμπί Cart
3. Το σύστημα εμφανίζει το πλήθος των παιχνιδιών, επιλογή delete για το κάθε παιχνίδι, το τίτλο του κάθε παιχνιδιού, μια μικρή εικόνα που αφορά το παιχνίδι, τη τιμή του κάθε παιχνιδιού, το ποσοστό και ποσό έκπτωσης και την τελική τιμή του παιχνιδιού και περίληψη όλης της παραγγελίας με Συνολική τιμή (Εικόνα 4.40)



Εικόνα 4.40 Καλάθι Αγορών

**Εναλλακτική Ροή Γεγονότων- Ο χρήστης κάνει κλικ χωρίς να έχει προσθέσει κάποιο παιχνίδι στο καλάθι**

**3.α.1** Το Σύστημα εμφανίζει πως ο χρήστης δεν έχει προσθέσει κάποιο παιχνίδι και πως το συνολικό κόστος της παραγγελίας είναι 0 (Εικόνα 4.41)



Εικόνα 4.41 Σελίδα Καλαθιού χωρίς να έχουν προστεθεί παιχνίδια

**Περίπτωση χρήσης Διαγραφή παιχνιδιού από το καλάθι αγορών  
-Αναγνωριστικό PX 14**

**Ονομα:** Διαγραφή παιχνιδιού από το καλάθι αγορών

**Περιγραφή :** Ο χρήστης αφού έχει προσθέσει παιχνίδια στο καλάθι αγορών, αφαιρεί κάποιο-α ή όλα τα παιχνίδια από το καλάθι

**Βασική Ροή Γεγονότων:**

1. Έχει εκτελεστεί επιτυχώς η PX13 Προβολή Καλαθιού Αγορών
2. Ο χρήστης επιλέγει το κουμπί Delete για όποιο παιχνίδι θέλει να αφαιρέσει από το καλάθι



3. Το σύστημα εμφανίζει την λίστα των παιχνιδιών ανανεωμένη χωρίς το παιχνίδι που αφαιρέθηκε

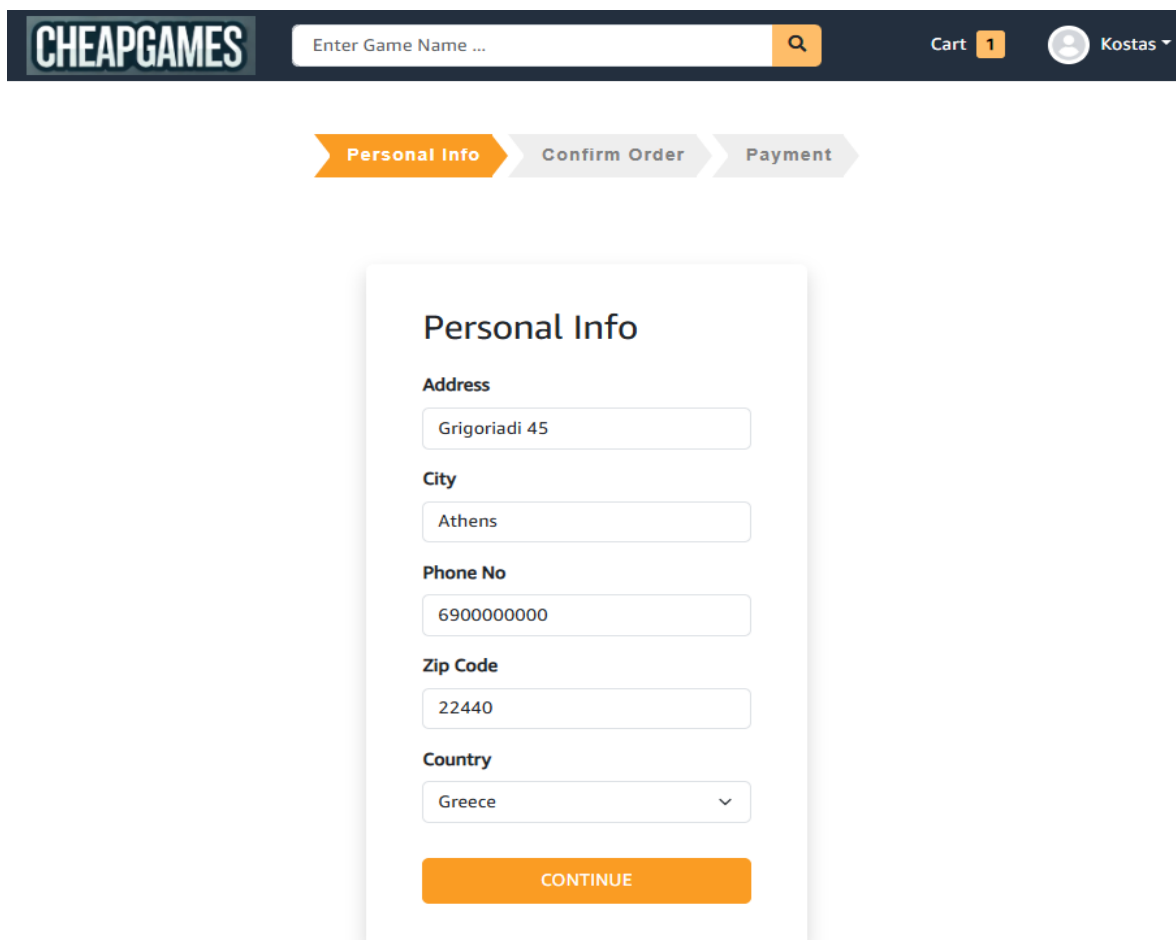
**Περίπτωση χρήσης** Εισαγωγή στοιχείων πελάτη (διεύθυνση, τηλέφωνο, χώρα κ.λπ.) και Στοιχείων Κάρτας-Πληρωμή Παραγγελίας-**Αναγνωριστικό PX15**

**Όνομα:** Εισαγωγή στοιχείων πελάτη (διεύθυνση, τηλέφωνο, χώρα κ.λπ.) και Στοιχείων Κάρτας-Πληρωμή Παραγγελίας

**Περιγραφή :** Ο χρήστης έχει προσθέσει τα παιχνίδια που θέλει στο καλάθι και θέλει να προχωρήσει στη πληρωμή των προϊόντων

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η PX13 (Εικόνα 4.40).
2. Ο χρήστης επιλέγει το κουμπί Check Out.
3. Το σύστημα εμφανίζει στην οθόνη φόρμα εισαγωγής στοιχείων πελάτη (Διεύθυνση, Πόλη, Τηλέφωνο, Ταχ.Κωδ, χώρα και ένα κουμπί Continue (Εικόνα 4.42)).



Εικόνα 4.42 Σελίδα PersonalInfo-Εισαγωγής Στοιχείων χρήστη κατά την αγορά παιχνιδιών

4. Ο χρήστης εισάγει τα στοιχεία του στην φόρμα και επιλέγει το κουμπί Continue

5. Το σύστημα εμφανίζει στην οθόνη Τις Προσωπικές Πληροφορίες του Χρήστη, Λίστα με τα στοιχεία του κάθε παιχνιδιού (Εικόνα, Τίτλος, Αρχική Τιμή, Ποσοστό έκπτωσης Ποσό Εκπτώσης και Τελική Τιμή, Περίληψη Παραγγελίας με Συνολικό Καθαρό Κόστος, Ποσό Φόρων και Συνολικό Ποσό και κουμπί Προώθηση στην Πληρωμή (Εικόνα 4.43)

**CHEAPGAMES** Enter Game Name ... **Cart 1** **Kostas**

**Personal Info** **Confirm Order** **Payment**

**Personal Info**  
 Name: Kostas  
 Phone: 6900000000  
 Address: Grigoriadi 45, Athens, 22440, Greece

**Order Summary**  
 Subtotal: \$160  
 Tax: \$38.4  
 Total: \$198.40

**Your Cart Items:**

Image	Title	Start Price	Discount	Discount Amount	Final Price
	Resident Evilsfs	\$200.00	%20	\$40.00	\$160.00

**Proceed to Payment**

Εικόνα 4.43 Επιβεβαίωση Παραγγελίας

6. Ο χρήστης ελέγχει την παραγγελία του και επιλέγει το κουμπί Proceed to Payment
7. Το σύστημα εμφανίζει φόρμα με επιλογές Πληρωμής (Μόνο Πιστωτική Κάρτα) και κουμπί Continue (Εικόνα 4.44)

**CHEAPGAMES** Enter Game Name ... **Cart 1** **Kostas**

**Personal Info** **Confirm Order** **Payment**

**Select Payment Method**

☒ Card - VISA, MasterCard

**CONTINUE**

Εικόνα 4.44 Επιλογή Τρόπου Πληρωμής

8. Ο χρήστης επιλέγει την πληρωμή με πιστωτική κάρτα και το κουμπί continue
9. Το σύστημα μεταφέρει τον χρήστη στο Ασφαλές περιβάλλον Πληρωμών Stripe όπου ο χρήστης εισάγει τα στοιχεία της κάρτας του και επιλέγει πληρωμή (Εικόνα 4.45)

← TEST MODE

Πληρωμή εμπόρου

**198,40 \$**

	Resident Evil 5	160,00 \$
Μερικό σύνολο		160,00 \$
Sales tax (24%)		38,40 \$
Συνολικό οφειλόμενο ποσό		198,40 \$

Η πληρωμή με κάρτα

Email:

Στοιχεία κάρτας

1234 1234 1234 1234

MM / EE CVC

Όνομα κατόχου κάρτας

Χώρα ή περιοχή

Ασφαλής αποθήκευση των στοιχείων μου για ολοκλήρωση αγοράς με 1 κλικ  
Ταχύτερες πληρωμές σε αυτή την τοποθεσία και οπουδήποτε γίνεται δεκτό το Link.

691 234 5678

link

Με την υποστήριξη της Όροι Απόρρητο

Εικόνα 4.45 Περιβάλλον πληρωμών Stripe (test mode)

10. Το Σύστημα Εμφανίζει στον χρήστη πίνακα με όλες τις παραγγελίες που έχει κάνει και συγκεκριμένα το id για κάθε παραγγελία, Ποσό Παραγγελίας, status για την κάθε παραγγελία (Paid/No Paid) και επιλογή προβολής λεπτομερειών Παραγγελίας (Εικόνα 4.46). Ο χρήστης μπορεί να έχει πρόσβαση στις αγορές του και επιλέγοντας “Orders” στο πτυσόμενο μενού με τις επιλογές του (Εικόνα 4.34)

**CHEAPGAMES** Enter Game Name ...

Cart **0** Kostas

## 1 Orders

Show entries

ID	Amount	Payment Status	Actions
67c597262c62fe8aebcf34d	\$198.4	PAID	

Showing 1 to 1 of 1 entries

Previous **1** Next

Εικόνα 4.46 Σελίδα με τις ολοκληρωμένες αγορές του χρήστη

### Περίπτωση χρήσης Προβολή Λίστας Αγορών και Λεπτομερειών Αγοράς του χρήστη -Αναγνωριστικό ΠΧ16

**Όνομα:** Προβολή Λίστας Αγορών και Λεπτομερειών Αγορών του χρήστη

**Περιγραφή :** Ο χρήστης αφού έχει συνδεθεί μπορεί να δει λίστα με τις αγορές του και επιλέγοντας κάποια αγορά να δει τα αγορασμένα παιχνίδια του

#### Βασική Ροή Γεγονότων:

1. Ο χρήστης κάνει επιτυχώς Login
2. Ο χρήστης στο μενού επιλογών του επιλέγει την επιλογή Orders.
3. Το σύστημα εμφανίζει λίστα με όλες τις αγορές του (Εικόνα 4.46).
4. Ο χρήστης επιλέγει να δει λεπτομέρειες κάποια αγοράς και το σύστημα εμφανίζει τις λεπτομέρειες αγοράς και τα παιχνίδια που συμπεριλαμβάνει (Εικόνα 4.47).

#### Your Order Details

ID	95272f56-198a-4e45-b94e-3cf2abe380b7
Status	Paid
Date	3/11/2025, 10:41:50 PM


#### Personal Info

Name	Ioannis Ioannou
Phone No	6900000000
Address	Grigoriadi 45, Athens, 22440, Greece

#### Payment Info

Status	
Method	
Stripe ID	Null
Amount Paid	\$30.88
Tax Amount Paid	\$5.98
Total Net Worth	\$24.9
Discount Amount	\$5.1

#### Order Items:

	<a href="#">Game 2</a>	Price After discount	Discount
		\$24.9	%17

Εικόνα 4.47 Προβολή Λεπτομερειών Αγοράς Χρήστη

### Περίπτωση χρήσης Download και Καταχώρηση χρόνου παιχνίματος Αγορασμένου Παιχνιδιού

#### -Αναγνωριστικό ΠΧ17

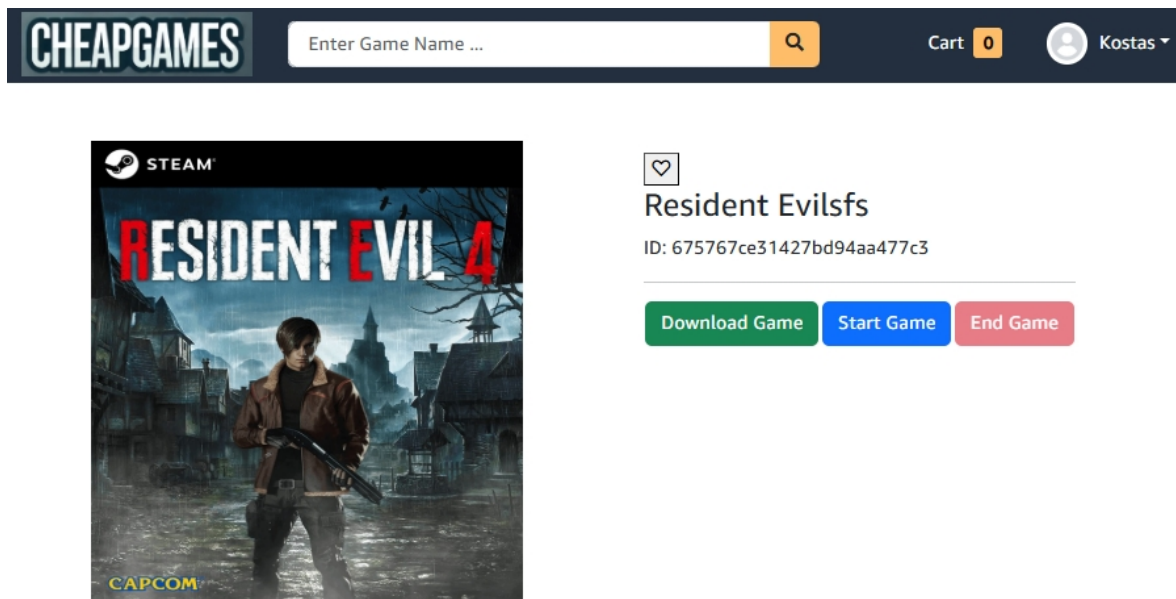
**Όνομα:** Download Αγορασμένου Παιχνιδιού

**Περιγραφή :** Ο χρήστης αφού έχει συνδεθεί και έχει εκτελέσει σωστά την αγορά παιχνιδιού

μπορεί από την αρχική οθόνη να επιλέξει κάποιο από τα αγορασμένα παιχνίδια και να το κάνει Download στον υπολογιστή του/Συσκευή του

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η PX15
2. Ο χρήστης στην αρχική οθόνη επιλέγει τον κατάλογο αγορασμένων παιχνιδιών
3. Εκτελείται η PX4
4. Ο χρήστης επιλέγει στο παιχνίδι που θέλει το κουμπί Play Game
5. Το σύστημα εμφανίζει πληροφορίες για το παιχνίδι και κουμπιά Download, Start Game, End Game (Εικόνα 4.48)



Εικόνα 4.48 Σελίδα Λήψης Αγορασμένου Παιχνιδιού

6. Ο χρήστης επιλέγει το κουμπί Download και το αρχείο παιχνιδιού κατεβαίνει στις λήψεις του χρήστη ή επιλέγει Start Game και EndGame για καταχώρηση χρόνου παιζίματος αγορασμένου παιχνιδιού

**Περίπτωση χρήσης Αξιολόγηση Αγορασμένου Παιχνιδιού**

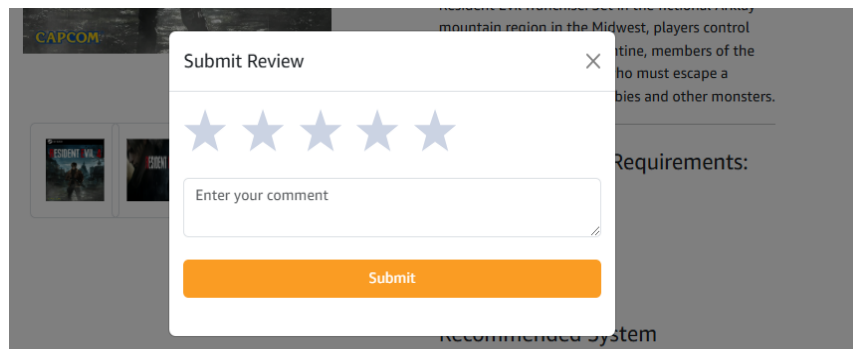
**-Αναγνωριστικό PX18**

**Όνομα:Αξιολόγηση Αγορασμένου Παιχνιδιού**

**Περιγραφή :** Ο χρήστης αφού έχει συνδεθεί μπορεί να μπει στις λεπτομέρειες κάποιου παιχνιδιού από του καταλόγους όλων των παιχνιδιών ή των αγαπημένων ή των αγορασμένων

**Βασική Ροή Γεγονότων:**

1. Εκτελείται Επιτυχώς η PX6 και εμφανίζεται η οθόνη με τις λεπτομέρειες του παιχνιδιού
2. Ο χρήστης έχει αγοράσει το παιχνίδι και επιλέγει το κουμπί Submit Your Review
3. Το σύστημα εμφανίζει φόρμα επιλογής Σκορ και Κειμένου αξιολόγησης (Εικόνα 4.49)



**Εικόνα 4.49** Φόρμα εισαγωγής αξιολόγησης

4. Ο χρήστης επιλέγει το σκορ, εισάγει το κείμενο της αξιολόγησης του και επιλέγει το κουμπί Submit
5. Το σύστημα εμφανίζει κατάλληλο μήνυμα ότι η Κριτική προστέθηκε στις Αξιολογήσεις

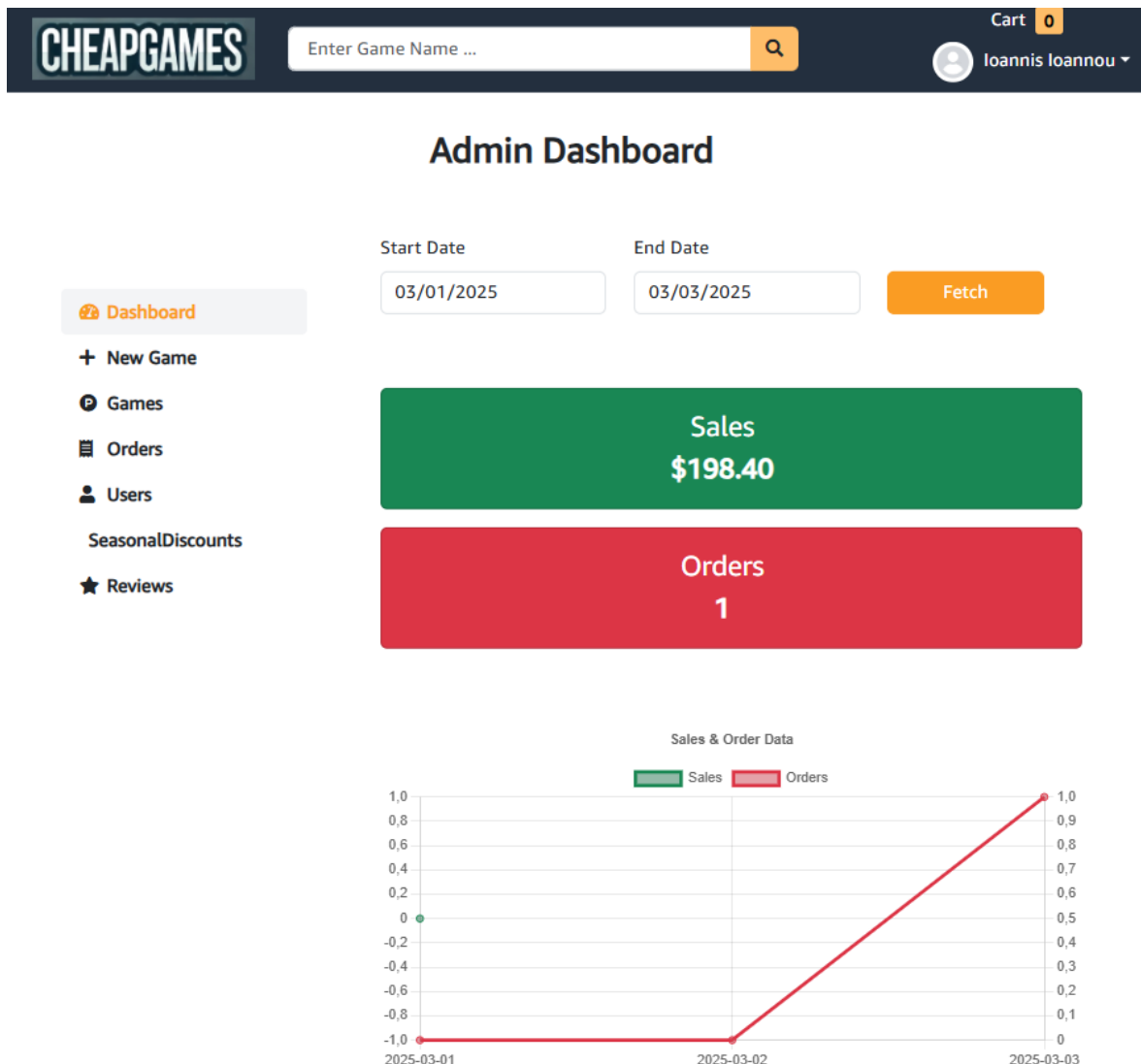
**Περίπτωση χρήσης** Προσθήκη παιχνιδιών (τίτλος, περιγραφή, εικόνες, βίντεο, τιμή, έκπτωση, απαιτήσεις συστήματος, αρχεία παιχνιδιών) - **Αναγνωριστικό PX19**

**Όνομα:** Προσθήκη παιχνιδιών

**Περιγραφή :** Ο Χρήστης αφού συνδεθεί με τα στοιχεία του , το σύστημα ελέγχει το ρόλο του και αν είναι Admin, του εμφανίζει την επιλογή Dashboard όπου πιέγοντας την μεταβιβάζεται στο μενού Dashboard και μπορεί να προσθέσει τα στοιχεία και το αρχείο του καινούριου παιχνιδιού.

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η ΠΧ Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή Dashboard
3. Ο χρήστης κάνει κλικ στο κουμπί Dashboard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της Dashboard (Εικόνα 4.50)



**Εικόνα 4.50** Κονσόλα με μενού επιλογών Διαχειριστή

5. Ο χρήστης επιλέγει το NewGame
6. Η οθόνη εμφανίζει κατάλληλη φόρμα εισαγωγής στοιχείων του παιχνιδιού (Title, Description, Price, Discount, Age Limit, Category, Developer, Publisher, Release Date, Minimum System Requirements, Recommended System Requirements) και Κουμπί Create (Εικόνα 4.51)



Dashboard

+ New Game

Games

Orders

Users

Seasonal Discounts

★ Reviews

## New Game

Title

Description

Price

Discount

Age Limit

Category

Developer

Publisher

Release Date

Action

ηη/μμ/εεεε

Minimum System Requirements

OS

Processor

Memory

Graphics

Storage

Select

Recommended System Requirements

OS

Processor

Memory

Graphics

Storage

Select

CREATE

Εικόνα 4.51 Φόρμα Εισαγωγής Στοιχείων Καινούριου Παιχνιδιού

7. Ο χρήστης εισάγει τα στοιχεία του παιχνιδιού και κάνει κλικ στο κουμπί Create
8. Η οθόνη εμφανίζει πίνακα με όλα τα παιχνίδια που έχουν καταχωρηθεί και επιλογές για το καθένα τροποποίησης στοιχείων, εισαγωγής αρχείων, εισαγωγής εικόνων και μπάρα αναζήτησης βάση τίτλου και κουμπί επικύρωσης (Εικόνα 4.52)



## Admin Dashboard

Dashboard

+ New Game

Games

Orders

Users

SeasonalDiscounts

Reviews

## 6 Games

Show entries

10

Search

ID	Title	Seas.Disc.Active	Discount	Actions
6750cf84178d6564a8eab53f	Final Fantasy...	No		<a href="#">Edit</a> <a href="#">Add</a> <a href="#">View</a> <a href="#">Delete</a>
675224edb41820038913c619	Iss pRo...	No	10	<a href="#">Edit</a> <a href="#">Add</a> <a href="#">View</a> <a href="#">Delete</a>
675767ce31427bd94aa477c3	Resident Evilsfs...	No	20	<a href="#">Edit</a> <a href="#">Add</a> <a href="#">View</a> <a href="#">Delete</a>
6785195147c4eaf3e1756e4a	Final Fantasy...	No	20	<a href="#">Edit</a> <a href="#">Add</a> <a href="#">View</a> <a href="#">Delete</a>
6785198b47c4eaf3e1756e58	Final Fantasy...	No	10	<a href="#">Edit</a> <a href="#">Add</a> <a href="#">View</a> <a href="#">Delete</a>
67c59be22c62fe8aecbcf427	Underground...	No	10	<a href="#">Edit</a> <a href="#">Add</a> <a href="#">View</a> <a href="#">Delete</a>
ID	Title	Seas.Disc.Active	Discount	Actions

Showing 1 to 6 of 6 entries


Previous 1 Next

Εικόνα 4.52 Σελίδα με όλα τα καταχωρημένα παιχνίδια (Admin)

9. Ο χρήστης βρίσκει το παιχνίδι που καταχώρησε στον πίνακα και επιλέγει το κουμπί που αφορά για το παιχνίδι αυτό η εισαγωγή αρχείου
10. Η οθόνη εμφανίζει πληροφορίες για το αρχείο παιχνιδιού αν έχει ανεβάσει κάποιο ο χρήστης ενώ στην περίπτωση που δεν έχει, ενημερώνει τον χρήστη πως δεν έχει ανεβάσει κάποιο αρχείο σχετικό με το παιχνίδι, και τέλος εμφανίζει κουμπιά Chose File και Upload (Εικόνα 4.53)



## Admin Dashboard


 Dashboard

 + New Game

 Games

 Orders

 Users

 Seasonal Discounts

 ★ Reviews

### Game File Info

There is not uploaded File for Underground

#### Upload/Update Game File for Underground

Choose File

Επιλογή αρχείου

Δεν επιλέχθηκε κανένα αρχείο.

Upload

Εικόνα 4.53 Φόρμα Εισαγωγής Αρχείου για παιχνίδι

11. Ο χρήστης κλικάρει το Choose File
12. Ανοίγει η εξερεύνηση αρχείων Του Υπολογιστή του
13. Ο χρήστης επιλέγει το αρχείο που αφορά το παιχνίδι
14. Η οθονη ενημερώνεται με το όνομα του αρχείου προς ανέβασμα και ο χρήστης κάνει κλικ στο Upload
15. Το σύστημα εμφανίζει μήνυμα πως το αρχείο ανέβηκε και οι πληροφορίες αρχείου ενημερώνονται με του καινούριο αρχείου (Εικόνα 4.54)



## Admin Dashboard

 Dashboard

 + New Game

 Games

 Orders

 Users

 Seasonal Discounts

 ★ Reviews

### Game File Info

title:Underground

fileName:9fda30c5c8941b702a55e4539adb7470.zip

fileSize:28746901 bytes

uploadDate:2025-03-03T12:13:22.849Z

contentType:application/x-zip-compressed

#### Upload/Update Game File for Underground

Choose File

Επιλογή αρχείου

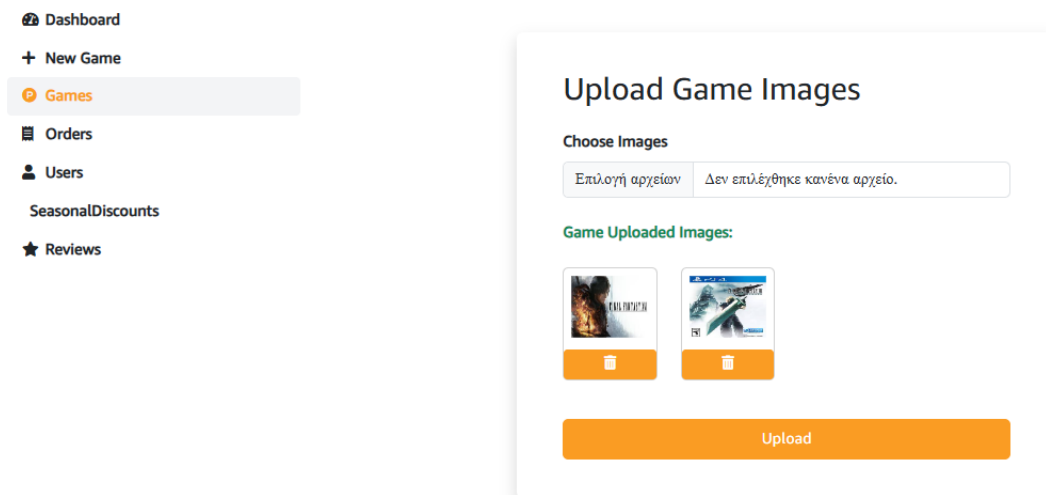
curse-of-the-arrow-win64 (2).zip

Upload

Εικόνα 4.54 Ανεβασμένο αρχείο και πληροφορίες αρχείου

16. Ο διαχειριστής επιλέγει Games στο menu επιλογών της DashBoard
17. Η οθόνη εμφανίζει πίνακα με όλα τα παιχνίδια που έχουν καταχωρηθεί και επιλογές για το καθένα τροποποίησης στοιχείων, εισαγωγής αρχείων, εισαγωγής εικόνων και μπάρα αναζήτησης βάση τίτλου και κουμπί επικύρωσης (Εικόνα 4.52)
18. Ο χρήστης βρίσκει το παιχνίδι που καταχώρησε στον πίνακα και επιλέγει το κουμπί που αφορά για το παιχνίδι αυτό η εισαγωγή/ενημέρωση εικόνων
19. Η οθόνη εμφανίζει εικόνες που τυχόν έχει ανεβάσει ο χρήστης (Εικόνα 4.55)

## Admin Dashboard



Εικόνα 4.55 Ανέβασμα και Διαγραφή Εικόνων Παιχνιδιού

20. ο χρήστης διαγράφει τυχόν ήδη υπάρχουσες φωτογραφίες ή επιλέγει καινούριες φωτογραφίες από τη συσκευή του και επιλέγει το κουμπί upload images
21. Σε κάθε περίπτωση εμφανίζονται μηνύματα επιτυχούς ή μή, διαγραφής ή πρόσθεσης εικόνων και η οθόνη ενημερώνεται με την καινούρια κατάσταση

### Περίπτωση χρήσης Τροποποίηση Στοιχείων Παιχνιδιού-Αναγνωριστικό PX20

**Ονομα:** Τροποποίηση Στοιχείων Παιχνιδιού

**Περιγραφή :** Ο Χρήστης αφού συνδεθεί με τα στοιχεία του , το σύστημα ελέγχει το ρόλο του και αν είναι Admin του εμφανίζει την επιλογή DashBoard επιλέγοντας την μεταβιβάζεται στο μενού DashBoard όπου μπορεί να τροποποιήσει τα στοιχεία του παιχνιδιού

#### Βασική Ροή Γεγονότων:

1. Εκτελείται επιτυχώς η PX2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard

4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της Dashboard
5. Ο χρήστης επιλέγει την καρτέλα Games
6. Η οθόνη εμφανίζει πίνακα με όλα τα παιχνίδια που έχουν καταχωρηθεί και επιλογές για το καθένα τροποποίησης στοιχείων, εισαγωγής αρχείων, εισαγωγής εικόνων και μπάρα αναζήτησης βάση τίτλου και κουμπί επικύρωσης
7. Ο χρήστης κλικάρει το Update για το παιχνίδι του οποίου θέλει να τροποποιήσει τα στοιχεία
8. Το σύστημα εμφανίζει κατάλληλη φόρμα εισαγωγής με όλα τα στοιχεία που αφορούν το παιχνίδι και κουμπί Update (Εικόνα 4.56)

Εικόνα 4.56 Τροποποίηση Στοιχείων Παιχνιδιού

9. Ο χρήστης εισάγει τα στοιχεία μόνο στα πεδία που θέλει να τροποποιήσει και κάνει κλικ στο κουμπί Update
10. Το σύστημα επιστρέφει στην οθόνη με τον Πίνακα που έχει όλα τα παιχνίδια και εμφανίζει μήνυμα πως το παιχνίδι ενημερώθηκε επιτυχώς

### Περίπτωση χρήσης Διαγραφή Παιχνιδιού από τη Βάση-Αναγνωριστικό ΠΧ21

**Όνομα:** Διαγραφή Παιχνιδιού από τη Βάση

**Περιγραφή :** Ο Χρήστης αφού συνδεθεί με τα στοιχεία του , το σύστημα ελέγχει το ρόλο

του και αν είναι Admin του εμφανίζει την επιλογή DashBoard επιλέγοντας την μεταβιβάζεται στο μενού DashBoard όπου μπορεί να διαγράψει κάποιο παιχνίδι και το αρχείο που το αφορά

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η PX2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της DashBoard
5. Ο χρήστης επιλέγει την καρτέλα Games
6. Η οθόνη εμφανίζει πίνακα με όλα τα παιχνίδια που έχουν καταχωρηθεί και επιλογές για το καθένα τροποποίησης στοιχείων, εισαγωγής αρχείων, εισαγωγής εικόνων και Διαγραφή παιχνιδιού και μπάρα αναζήτησης βάση τίτλου και κουμπί επικύρωσης
7. Ο χρήστης βρίσκει το παιχνίδι που θέλει να σβήσει στον πίνακα στον πίνακα ή μέσω της αναζήτησης και κλικάρει το κουμπί Delete που αφορά το παιχνίδι που θέλει να διαγράψει
8. Το σύστημα εμφανίζει κατάλληλο μήνυμα επιτυχούς διαγραφής παιχνιδιού και ο πίνακας ανανεώνεται χωρίς το παιχνίδι που διαγράφηκε (Εικόνα 4.57)

**CHEAPGAMES** Enter Game Name ... Game Deleted Cart 0 Ioannis Ioannou

### Admin Dashboard

- Dashboard
- New Game
- Games**
- Orders
- Users
- SeasonalDiscounts
- Reviews

### 5 Games

Show entries 10 Search

ID	Title	Seas.Disc.Active	Discount	Actions
6750cf84178d6564a8eab53f	Final Fantasy...	No		
675224edb41820038913c619	Iss pRo...	No	10	
675767ce31427bd94aa477c3	Resident Evilsfs...	No	20	
6785195147c4eaf3e1756e4a	Final Fantasy...	No	20	
67c59be22c62fe8aecbcf427	Underground...	No	10	

Showing 1 to 5 of 5 entries Previous 1 Next

**Εικόνα 4.57** Διαγραφή Παιχνιδιού

**Περίπτωση χρήσης Διαχείριση στοιχείων άλλων χρηστών (αλλαγή ρόλου, διαγραφή)-**  
**Αναγνωριστικό PX22**

**Όνομα:** Διαχείριση στοιχείων άλλων χρηστών (αλλαγή ρόλου, διαγραφή)

**Περιγραφή :** Ο διαχειριστής έχει τη δυνατότητα να δει τη λίστα των χρηστών και να επεξεργαστεί τον ρόλο τους (π.χ., από χρήστη σε διαχειριστή) ή να διαγράψει χρήστες από τη βάση δεδομένων.

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η PX2 Σύνδεση Χρήστη

2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της DashBoard
5. Ο χρήστης επιλέγει την καρτέλα Users
6. Η οθόνη εμφανίζει πίνακα με όλους τους χρήστες που έχουν κάνει εγγραφή και επιλογές για το καθένα τροποποίησης στοιχείων και διαγραφής αυτού και μπάρα αναζήτησης βάση τίτλου και κουμπί επικύρωσης αναζήτησης (Εικόνα 4.58)



















### Admin Dashboard

Dashboard  
+ New Game  
Games  
Orders  
**Users**  
SeasonalDiscounts  
Reviews

## 9 Users

Show entries  
10

Search

ID	Name	Email	Role	Actions
66da7cb166aef8ff296b6950	I	grigoris@gmail.com	admin	 
66da7dfd0b79f13f41a0f76d	Ioannis	dokimastiko@email.com	admin	 
66da7e070b79f13f41a0f76f	Ioannis	dokimastiko2@email.com	user	 
66dda6ff7204bb0f86e77af6	Ioannis Ioannou	dokimastiko3@email.com	admin	 
66e35b74a29b959dfdb3d23a	Despoina	despoina@gmail.com	user	 
66e4c18a3853910e33184f6d	Michel	michel@gmail.com	user	 
66e4c5ec3853910e33184fc9	Kostas	kostas@gmail.com	user	 
6752056be2718d106f37a526	John	papapetroyu@gmail.com	user	 
676995617dae1497d7e76a3a	Ioannis	papaoyu@gmail.com	user	 
ID	Name	Email	Role	Actions

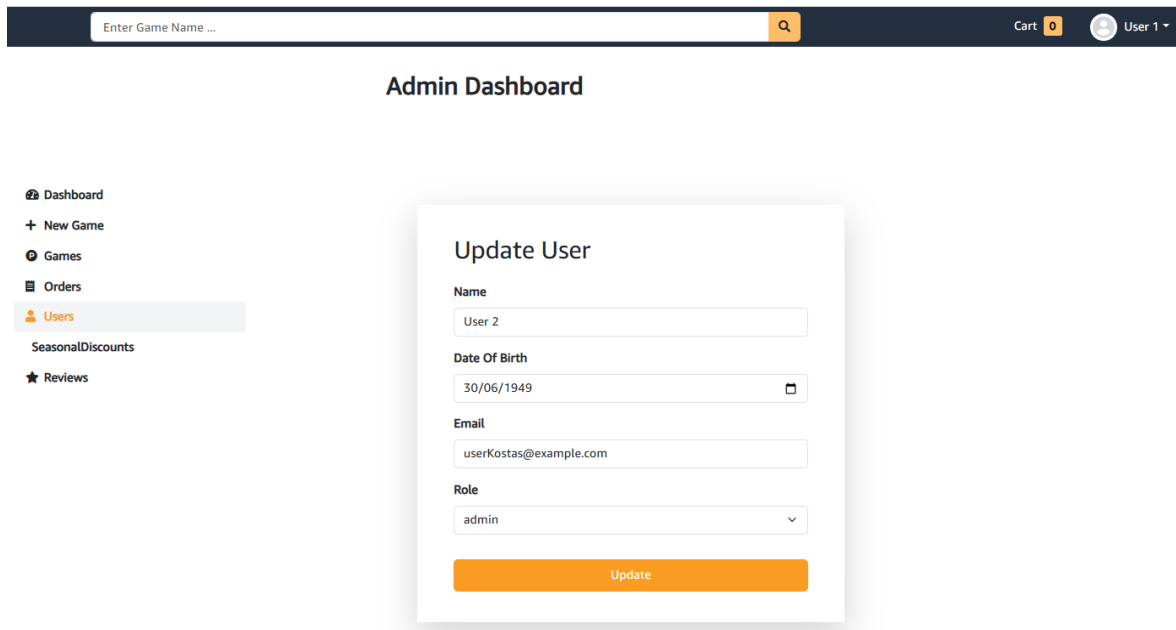
Showing 1 to 9 of 9 entries

Previous
1
Next

Εικόνα 4.58 Λίστα Όλων των Χρηστών

7. Ο διαχειριστής βρίσκει τον χρήστη που θέλει στον πίνακα η αναζητώντας κάποιο από τα στοιχεία του Id, Name, Email και user ή admin για Role και κάνει κλικ στο κουμπί Update
8. Το σύστημα εμφανίζει κατάλληλη φόρμα εισαγωγής στοιχείων ενημερωμένη με τα ήδη καταχωρημένα στοιχεία του χρήστη (Εικόνα 4.59)





The screenshot shows the Admin Dashboard with a sidebar on the left containing links to Dashboard, New Game, Games, Orders, Users (highlighted), Seasonal Discounts, and Reviews. The main content area displays the 'Update User' form. The form has the following fields: Name (text input with 'User 2'), Date Of Birth (date picker with '30/06/1949'), Email (text input with 'userKostas@example.com'), and Role (dropdown menu with 'admin' selected). An orange 'Update' button is at the bottom of the form.

Εικόνα 4.59 Ενημέρωση στοιχείων χρήστη

9. Ο χρήστης αλλάζει τα στοιχεία που θέλει και κάνει κλικ στο κουμπί Update
10. Το σύστημα εμφανίζει τον ενημερωμένο πίνακα με τους χρήστες του γεγονότος 6 και κατάλληλο μήνυμα επιτυχούς τροποποίησης

### Περίπτωση χρήσης Διαγραφή Χρήστη-Αναγνωριστικό PX23

#### Όνομα: Διαγραφή Χρήστη

**Περιγραφή :** Ο διαχειριστής έχει τη δυνατότητα να δει τη λίστα των χρηστών και να διαγράψει το λογαριασμό κάποιου χρήστη

#### Βασική Ροή Γεγονότων:

1. Εκτελείται επιτυχώς η PX2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της DashBoard
5. Ο χρήστης επιλέγει την καρτέλα Users
6. Η οθόνη εμφανίζει πίνακα με όλους τους χρήστες που έχουν κάνει εγγραφή και επιλογές για το καθένα τροποποίησης στοιχείων και διαγραφής αυτού και μπάρα αναζήτησης και κουμπί επικύρωσης αναζήτησης (Εικόνα 4.58)
7. Ο χρήστης βρίσκει τον χρήστη που θέλει στον πίνακα η αναζητώντας κάποιο από τα στοιχεία του Id, Name, Email και user ή admin για Role και κάνει κλικ στο κουμπί Delete
8. Το σύστημα εμφανίζει τον ενημερωμένο πίνακα με τους χρήστες του γεγονότος 6 και κατάλληλο μήνυμα επιτυχούς διαγραφής χρήστη

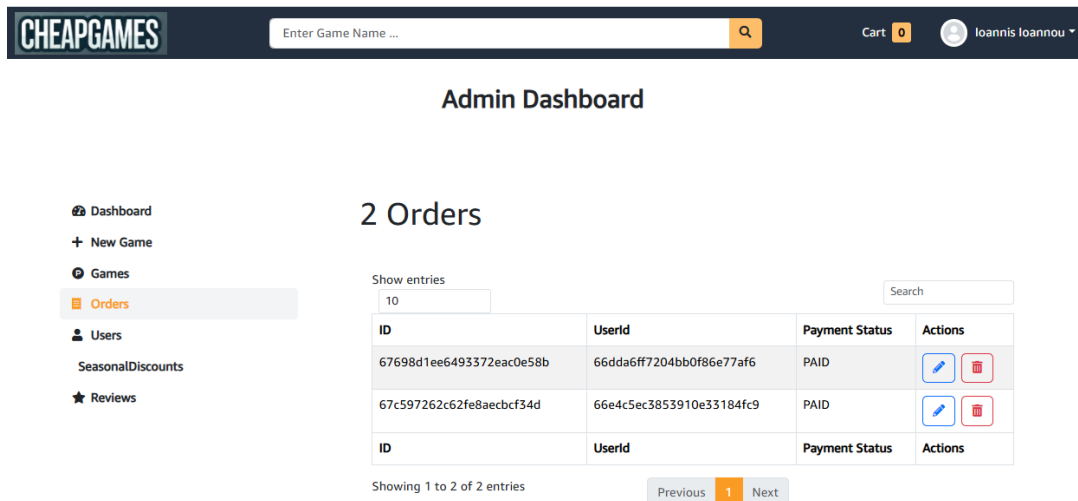
### Περίπτωση χρήσης Διαχείριση παραγγελιών (έλεγχος, διαγραφή) -Αναγνωριστικό PX24

**Όνομα:** Διαχείριση παραγγελιών (έλεγχος, διαγραφή)

**Περιγραφή :** Ο διαχειριστής έχει τη δυνατότητα να δει όλες τις παραγγελίες σε έναν πίνακα με τα αναγνωστικά ID, userId και payment status ενώ επιλέγοντας μία παραγγελία μπορεί να δει τα προϊόντα τον πελάτη και όλα τα στοιχεία της παραγγελίας





**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η ΠΧ2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της DashBoard
5. Ο χρήστης επιλέγει την καρτέλα Orders
6. Η οθόνη εμφανίζει πίνακα με όλες τις παραγγελίες με πεδία Id, userId και payment status που έχουν γίνει και επιλογές για τη κάθε μία τροποποίησης στοιχείων και διαγραφής αυτού και μπάρα αναζήτησης και κουμπί επικύρωσης αναζήτησης (Εικόνα 4.60)



**Admin Dashboard**

2 Orders

ID	UserId	Payment Status	Actions
67698d1ee6493372eac0e58b	66dda6ff7204bb0f86e77af6	PAID	 
67c597262c62fe8aecbcf34d	66e4c5ec3853910e33184fc9	PAID	 

Showing 1 to 2 of 2 entries

Previous 1 Next

**Εικόνα 4.60** Λίστα όλων των Αγορών

7. Ο χρήστης επιλέγει την παραγγελία που θελει βρίσκοντας την στον πίνακα για να δει περισσότερες λεπτομέρειες (Εικόνα 4.61) ή για να την διαγράψει κάνοντας κλικ στο κουμπί Delete

+ New Game	
🎮 Games	
📦 Orders	
👤 Users	
🏷️ Seasonal Discounts	
★ Reviews	

ID	2ff09059-3d03-4f91-99a5-4c37384b5ac3
Status	Paid
Date	3/11/2025, 11:33:57 PM


Personal Info

Name	Ioannis Ioannou
Email	ioannes11@gmail.com
userId	fb688c65-d821-475c-8404-4ea016189737
Phone No	6900000000
Address	Grigoriadi 45, Athens, 22440, Greece

Payment Info

Status	paid
Method	Card
Stripe ID	pi_3R1aQvCOzKWKwI2t0UTb0nzD
Amount Paid	\$33.48
Tax Amount Paid	\$6.48
Total Net Worth	\$27
Discount Amount	\$0

Order Items:

	Game 1	Price After discount	Discount %0
		\$27	

Εικόνα 4.61 Πληροφορίες αγοράς

8. Αν ο χρήστης διαγράψει μία παραγγελία ο πίνακας ανανεώνεται και τον βλέπουμε χωρίς την διεγραμμένη παραγγελία

**Περίπτωση χρήσης** Διαχείριση Εισόδων ανά συγκεκριμένο ημερολογιακό διάστημα

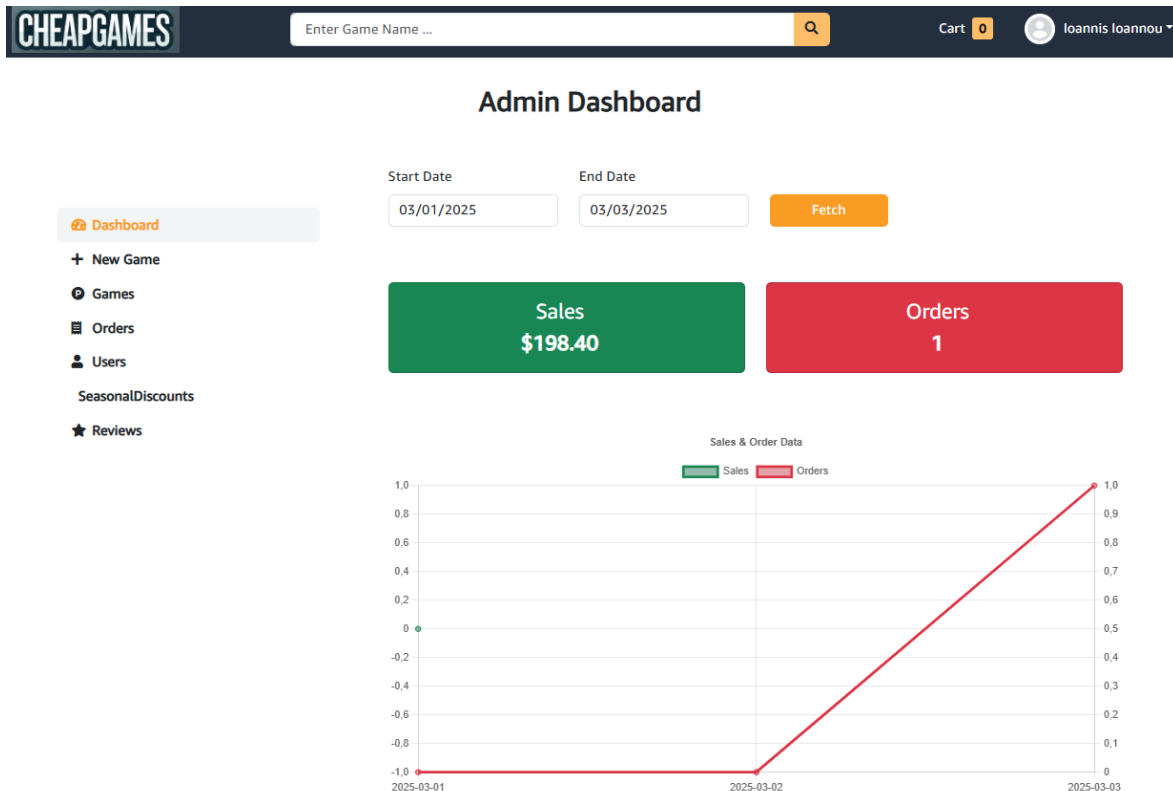
**-Αναγνωριστικό ΠΧ25**

**Όνομα: Διαχείριση Εισόδων**

**Περιγραφή :** Ο Χρήστης έχει τη δυνατότητα να δει σύνολο για τις παραγγελίες και τα έσοδα επιλέγοντας συγκεκριμένο ημερολογιακό εύρος

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η ΠΧ Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard
4. Η οθόνη εμφανίζει το μενού επιλογών της DashBoard, φόρμα εισαγωγής StartDate και EndDate, κουμπί επικύρωσης Πλήθος παραγγελιών και Σύνολο Εισόδων και γράφημα Αξόνων με Πωλήσεις και Παραγγελίες (Εικόνα 4.62)



Εικόνα 4.62 Σύνολο Εσόδων και Αγορών με γράφημα

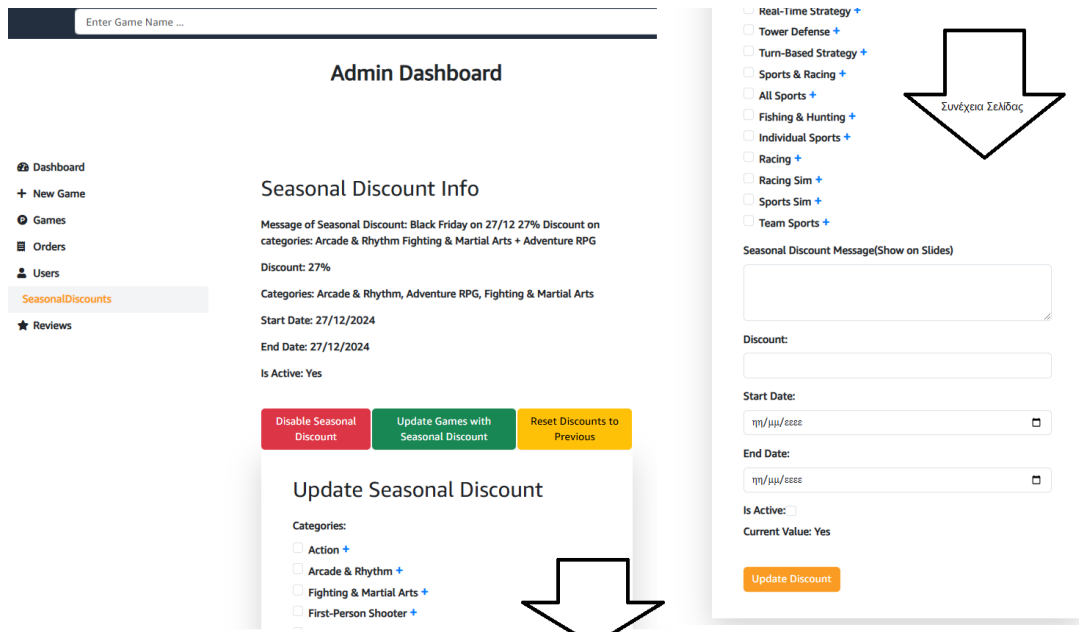
### Περίπτωση χρήσης Δημιουργία/Ενημέρωση Εποχιακής Εκπτώσης -Αναγνωριστικό PX26

**Όνομα:** Δημιουργία/Ενημέρωση Εποχιακής Εκπτώσης

**Περιγραφή :** Ο διαχειριστής έχει τη δυνατότητα να ενημερώνει/αποθηκεύει τα στοιχεία μιας εποχιακής έκπτωσης και να ενεργοποιεί αυτόματα την εμφάνιση στους χρήστες μέσω ενός διαφημιστικού slide καθώς και να την απενεργοποιεί αν είναι **ενεργοποιημένη**.

#### Βασική Ροή Γεγονότων:

1. Εκτελείται επιτυχώς η PX2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της DashBoard,
5. Ο χρήστης επιλέγει από το μενού Seasonal Discounts
6. Η Οθόνη εμφανίζει την καταχωρημένη Εποχιακή Έκπτωση αν υπάρχει (Προωθητικό μήνυμα, Ποσοστό Εποχιακής Έκπτωσης, Κατηγορίες Παιχνιδιών που αφορά η εποχιακή Εκπτωση, Εύρος Ημερομηνίας που αφορά η εποχιακή έκπτωση και αν είναι ενεργή δηλαδή αν εμφανίζεται) , κουμπί απενεργοποίησης της διαφήμισης, Κουμπί ενεργοποίησης των εκπτώσεων της εποχιακής έκπτωσης, Κουμπί επαναφοράς των εκπτώσεων στην προηγούμενη μορφή τους και φόρμα εισαγωγής/ενημέρωσης στοιχείων εποχιακής έκπτωσης και κουμπί update. (Εικόνα 4.63)



**Εικόνα 4.63** Ρύθμιση Εποχιακής Εκπτώσης και Slide με διαφημίσεις

7. Ο χρήστης εισάγει τα στοιχεία στην φόρμα( Προωθητικό μήνυμα slide, κατηγορίες που θα πάρουν έκπτωση, έκπτωση, Ημερομηνία Εκίνησης, Ημερομηνία Λήξης, Επιλέγει YES στο πεδίο IsActive (checkbox δηλαδή να ενεργοποιηθεί ή όχι η διαφήμιση) και επιλέγει το κουμπί UpdateDiscount.
8. Η σελίδα ενημερώνεται με τα καινούρια στοιχεία της εποχιακής έκπτωσης, εμφανίζει κατάλληλο μήνυμα επιτυχούς ενημέρωσης και το slide της διαφήμισης εμφανίζεται αυτόματα στις οθόνες των πελατών.

### **Περίπτωση χρήσης** Εφαρμογή Εποχιακών Εκπτώσεων στις καταχωρημένες κατηγορίες παιχνιδιών -**Αναγνωριστικό ΠΧ27**

**Όνομα:**Εφαρμογή εποχιακών εκπτώσεων σε συγκεκριμένες κατηγορίες παιχνιδιών

**Περιγραφή :** Ο διαχειριστής έχει καταχωρήσει εποχιακή έκπτωση η οποία εμφανίζεται στους χρήστες με Slide και έχει τη δυνατότητα να ενημερώσει τα παιχνίδια των κατηγοριών που αφορά η εποχιακή έκπτωση όταν φτάσει η ημερομηνία που ξεκινάει η εποχιακή έκπτωση.

#### **Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η ΠΧ2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή DashBoard
3. Ο χρήστης κάνει κλικ στο κουμπί DashBoard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της DashBoard,
5. Ο χρήστης επιλέγει από το μενού Seasonal Discounts
6. Η Οθόνη εμφανίζει την καταχωρημένη Εποχιακή Έκπτωση αν υπάρχει (Προωθητικό μήνυμα, Ποσοστό Εποχιακής Έκπτωσης, Κατηγορίες Παιχνιδιών που αφορά η εποχιακή Εκπτωση, Εύρος Ημερομηνίας που αφορά η εποχιακή έκπτωση και αν είναι ενεργή δηλαδή αν εμφανίζεται) , κουμπί απενεργοποίησης της διαφήμισης, Κουμπί

ενεργοποίησης των εκπτώσεων της εποχιακής έκπτωσης, Κουμπί επαναφοράς των εκπτώσεων στην προηγούμενη μορφή τους και φόρμα εισαγωγής/ενημέρωσης στοιχείων εποχιακής έκπτωσης και κουμπί update.

7. Ο χρήστης επιλέγει την ενεργοποίηση της ενημέρωσης των παιχνιδιών με το ποσοστό της εποχιακής έκπτωσης που έχει καταχωρήσει στο βήμα 7 όταν φτάσει η ημερομηνία των εκπτώσεων.
8. Το σύστημα εμφανίζει μήνυμα επιτυχούς ενημέρωσης όλων των παιχνιδιών που ανήκουν στις κατηγορίες που έχουν καταχωρηθεί στην εποχιακή έκπτωση

**Περίπτωση χρήσης** Επαναφορά εκπτώσεων παιχνιδιών με εφαρμοσμένη την εποχιακή έκπτωση στις αρχικές-προηγούμενες τους τιμές -**Αναγνωριστικό PX28**

**Όνομα:**Επαναφορά εκπτώσεων παιχνιδιών με εφαρμοσμένη την εποχιακή έκπτωση στις αρχικές-προηγούμενες τους τιμές

**Περιγραφή :** Ο διαχειριστής έχει ενημερώσει τα παιχνίδια των κατηγοριών που αφορά η εποχιακή έκπτωση με την εποχιακή έκπτωση και επαναφέρει τις εκπτώσεις αυτών των παιχνιδιών στην αρχική-προηγούμενη τους τιμή.

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η PX2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή Dashboard
3. Ο χρήστης κάνει κλικ στο κουμπί Dashboard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της Dashboard,
5. Ο χρήστης επιλέγει από το μενού Seasonal Discounts
6. Η Οθόνη εμφανίζει την καταχωρημένη Εποχιακή Έκπτωση αν υπάρχει (Προωθητικό μήνυμα, Ποσοστό Εποχιακής Έκπτωσης, Κατηγορίες Παιχνιδιών που αφορά η εποχιακή Έκπτωση, Εύρος Ημερομηνίας που αφορά η εποχιακή έκπτωση και αν είναι ενεργή δηλαδή αν εμφανίζεται) , κουμπί απενεργοποίησης της διαφήμισης, Κουμπί ενεργοποίησης των εκπτώσεων της εποχιακής έκπτωσης, Κουμπί επαναφοράς των εκπτώσεων στην προηγούμενη μορφή τους και φόρμα εισαγωγής/ενημέρωσης στοιχείων εποχιακής έκπτωσης και κουμπί update.
9. Ο χρήστης επιλέγει την επαναφορά των εκπτώσεων στην αρχική τους τιμή όταν φτάσει η ημερομηνία των εκπτώσεων και εμφανίζεται κατάλληλο μήνυμα επιτυχούς επαναφοράς των εκπτώσεων.

**Περίπτωση χρήσης** Απενεργοποίηση Εμφάνισης Διαφημιστικού Slide με καταχωρημένα στοιχεία εποχιακής έκπτωσης -**Αναγνωριστικό PX29**

**Όνομα:**Απενεργοποίηση Εμφάνισης Διαφημιστικού Slide

**Περιγραφή :** Ο διαχειριστής έχει ενημερώσει τα παιχνίδια των κατηγοριών που αφορά η εποχιακή έκπτωση με την εποχιακή έκπτωση και επαναφέρει τις εκπτώσεις αυτών των παιχνιδιών στην αρχική-προηγούμενη τους τιμή.

**Βασική Ροή Γεγονότων:**

1. Εκτελείται επιτυχώς η PX2 Σύνδεση Χρήστη
2. Το σύστημα ελέγχει το ρόλο του χρήστη και αν είναι διαχειριστής εμφανίζει την επιλογή Dashboard
3. Ο χρήστης κάνει κλικ στο κουμπί Dashboard
4. Η οθόνη εμφανίζει στην οθόνη το μενού επιλογών της Dashboard,
5. Ο χρήστης επιλέγει από το μενού Seasonal Discounts
6. Η Οθόνη εμφανίζει την καταχωρημένη Εποχιακή Έκπτωση αν υπάρχει (Προωθητικό



μήνυμα, Ποσοστό Εποχιακής Έκπτωσης, Κατηγορίες Παιχνιδιών που αφορά η εποχιακή Έκπτωση, Εύρος Ημερομηνίας που αφορά η εποχιακή έκπτωση και αν είναι ενεργή δηλαδή αν εμφανίζεται) , κουμπί απενεργοποίησης της διαφήμισης, Κουμπί ενεργοποίησης των εκπτώσεων της εποχιακής έκπτωσης, Κουμπί επαναφοράς των εκπτώσεων στην προηγούμενη μορφή τους και φόρμα εισαγωγής/ενημέρωσης στοιχείων εποχιακής έκπτωσης και κουμπί update.

7. Ο χρήστης επιλέγει την απενεργοποίηση της διαφήμισης, εμφανίζεται μήνυμα επιτυχίας και το slide με την διαφήμιση παύει να εμφανίζεται από τις οθόνες των χρηστών
8. Ο χρήστης ανανεώνει τη σελίδα και εμφανίζεται η καταχωρημένη εποχιακή έκπτωση με το Πεδίο IsActive No (δηλαδή δεν εμφανίζεται η διαφήμιση).

## 4.6 Καταγραφή Endpoints

Στο κεφάλαιο 2.4, παρουσιάστηκε η αρχιτεκτονική Rest κατά την οποία ο server δέχεται HTTP αιτήματα από τον client σε διάφορες διαδρομές ανάλογα του τι “πρέπει” να συμβεί οι οποίες καθορίζονται από τους σχεδιαστές-developers. Επίσης παρουσιάστηκε ο όρος URI και συχνά αναφερόμενος κι ως endpoint με τους οποίους εννοούμε τις διευθύνσεις-διαδρομές στις οποίες ο server δέχεται HTTP αιτήματα για να στείλει ή να λάβει δεδομένα προς τον client και να εκτελέσει απαραίτητες λειτουργίες που έχουν να κάνουν με το σκοπό του αιτήματος και τη διαχείριση της βάσης δεδομένων. Τα endpoints, τα δεδομένα, οι μέθοδοι και το τι θέλουμε να εκτελείται, θα πρέπει να καταγραφούν πριν προχωρήσουμε στην υλοποίηση. Εχοντας τα διαγράμματα περιπτώσεων χρήσης και τις λεκτικές περιγραφές τους από το κεφάλαιο 4 έχουμε τον οδηγό για να απεικονίσουμε το τι ακριβώς θέλουμε να συμβαίνει, από ποιο σημείο της διαδικτυακής μας εφαρμογής και τις ροές δεδομένων μεταξύ Server-Client. Για αυτόν τον λόγο, δημιουργήθηκε ένας πίνακας που περιέχει τις εξής στήλες:

- Κωδικός Περίπτωσης Χρήσης-Ονομασία Περίπτωσης Χρήσης (Βασική Ροή/Εναλλακτική Ροή/βήμα) από κεφάλαιο 4.4
- Τα routes(διαδρομές στις οποίες "ακούει" ο server)
- Τη Μέθοδο (GET, POST, PUT, DELETE, PATCH)
- Τα δεδομένα που στέλνει ο Client στον Server
- Ονομα Controller
- Τα δεδομένα που στέλνει ο Server στον Client
- Την εικόνα του screenshot για να ξέρουμε από ποια σελίδα του site γίνεται αυτή η ενέργεια.

Ο πίνακας A.1 παρουσιάζεται στο παράρτημα Α σελ. 138.

Στον πίνακα έχει επιλεγθεί να παρουσιαστούν και τα ονόματα των controller στα οποία ορίζονται λειτουργίες όπως πχ ανάκτηση από τη βάση συγκεκριμένων δεδομένων για να σταλθούν ως απάντηση στον client. Οι ονομασίες αυτές θα είναι ίδιες όπως θα οριστούν και μέσα στον κώδικα.

Κατά τη λειτουργία του server, οι καθορισμένες διαδρομές (routes) συνδυάζονται με τη διεύθυνση του server για να σχηματίσουν τα τελικά endpoints. Σε κάποια routes θα πρέπει να παρατηρηθεί η χρήση του συμβόλου “:” όπως το route “/games/:id” που αντιστοιχεί στη ΠΧ 6 Προβολή Λεπτομερειών παιχνιδιού (Παράρτημα Α σελ.138, Πίνακας Α.1 γραμμή 7). Μέσω του συμβόλου “:” δηλώνουμε ένα δυναμικό route το οποίο καθορίζεται από τη τιμή



του πεδίου που ακολουθεί. Δηλαδή ανάλογα την τιμή που αντικαθιστά ο client στο πεδίο αυτό δημιουργείται και το αντίστοιχο route. Με αυτό το τρόπο αν κάνει ένα αίτημα στο route `games/acbfd14` όπου “acbfd14” η τιμή του id κάποιου παιχνιδιού στη βάση, ο server θα επιστρέψει πληροφορίες για αυτό το παιχνίδι. Σημειώνεται ότι η θύρα του server δεν περιλαμβάνεται στον πίνακα, καθώς αυτή καθορίζεται κατά τη φάση υλοποίησης του server. Να σημειωθεί επίσης ότι οι περιπτώσεις χρήσης οι οποίες αφορούν τη διαχείριση του καλαθιού του χρήστη (Γραμμές 17,18 και 19 Παράρτημα Α Πίνακας Α1 Σελ. 138) δεν περιλαμβάνουν ροές δεδομένων μεταξύ server και client καθώς αφορούν αποκλειστικά το frontend.

## **5. Υλοποίηση Backend**

### **5.1 Υλοποίηση Backend με MERN Stack**

Στο κεφάλαιο θα δούμε μια σύντομη περιγραφή από την υλοποίηση του backend στη περίπτωση της MERN. Σκοπός αυτού του κεφαλαίου είναι να αναδειχθεί η σημαντικότητα προηγούμενων βημάτων στην ανάλυση και σχεδίαση όπως η καταγραφή των endpoints και ο σημαντικός ρόλος που έχουν κατά την υλοποίηση.

#### **5.1.1 Δομή Φακέλων και Αρχείων του Backend (MERN)**

Το backend της διαδικτυακής εφαρμογής CheapGames οργανώθηκε σε φακέλους και αρχεία ακολουθώντας την αρχιτεκτονική Rest. Μία σωστή δομή του backend είναι σημαντική για τη διατήρηση της επεκτασιμότητας, της καθαρότητας του κώδικα και της σωστής λειτουργίας και επικοινωνίας με το frontend. Στην εικόνα 5.1 μπορούμε να δούμε την δομή των φακέλων και των αρχείων του backend η οποία εξήχθη σε ένα αρχείο backend\_structure.txt μέσω του Visual Studio Code.

```

backend > backend_structure.txt
1  Folder PATH listing for volume Windows8_OS
2  Volume serial number is F4DA-01BD
3  C:..
4  |   app.js ----->Εκκίνηση του Express server,Φόρτωση των απαραίτητων middleware
5  |   |   Εκκίνηση της Σύνδεσης με τη βάση δεδομένων,
6  |   |   Εισαγωγή και χρήση των routes του API,
7  |   |   Διαχείριση σφαλμάτων και unhandled exceptions.
8  |   ---config ----->Ρυθμίσεις και Παραμετροποίηση
9  |   |   config.env --> Περιέχει τις ευαίσθητες και περιβαλλοντικές μεταβλητές του backend
10 |   |   dbConnect.js--> Διαχειρίζεται τη σύνδεση με τη MongoDB μέσω του Mongoose.
11 |   |   gridFsConfig.js--> Ρυθμίζει το GridFS για την αποθήκευση αρχείων στη MongoDB.
12 |   |   multerConfig.js--> Περιέχει τη ρύθμιση του Multer για μεταφόρτωση αρχείων
13 |   |
14 |   ---controllers -----> Φάκελος με όλα τα controller
15 |   |   authControllers.js -----> controllers που αφορούν τον χρήστη(σύνδεση,
16 |   |   |   εγγραφή,προσθήκη στα αγαπημένα, κλπ.)
17 |   |   gameControllers.js -----> controllers που αφορούν τα παιχνίδια(ανάκτηση παιχνιδιών,
18 |   |   |   λεπτομέρειες παιχνιδιού κλπ.)
19 |   |   orderControllers.js -----> controllers που αφορούν τις παραγγελίες
20 |   |   |   (ανάκτηση παραγγελιών, λεπτομέρειες παραγγελίας κλπ.)
21 |   |   paymentControllers.js -----> controllers που αφορούν την πληρωμή
22 |   |   |   (Σύνδεση με Stripe για επεξεργασία πληρωμών, δημιουργία παραγγελίας κλπ )
23 |   |
24 |   ---middlewares -----> Middleware για έλεγχο και διαχείριση σφαλμάτων
25 |   |   auth.js -----> Middleware για τον έλεγχο authentication με JWT και Authorization
26 |   |   catchAsyncErrors.js -----> Middleware για τη διαχείριση ασύγχρονων σφαλμάτων
27 |   |   errors.js -----> Κεντρική διαχείριση σφαλμάτων
28 |   |   uploadCheckGame.js -----> uploadCheckGame.js → Έλεγχος αρχείων που ανεβάζουν οι χρήστες
29 |   |
30 |   ---models -----> Σχήματα Μοντέλα(Mongoose)
31 |   |   game.js -----> Σχήμα για τα παιχνίδια και εξαγωγή Μοντέλου
32 |   |   order.js -----> Σχήμα για τις παραγγελίες και εξαγωγή μοντέλου
33 |   |   seasonalDiscount.js -----> Σχήμα για εποχιακή έκπτωση και εξαγωγή μοντέλου
34 |   |   user.js -----> Σχήμα για χρήστες και εξαγωγή μοντέλου
35 |   |
36 |   ---routes ----->Διαδρομές API (RESTful Routes)
37 |   |   auth.js -----> διαδρομές για αιτήματα HTTP από clients που αφορούν τους χρήστες
38 |   |   games.js -----> διαδρομές για αιτήματα HTTP από clients που αφορούν τα παιχνίδια
39 |   |   orders.js -----> διαδρομές για αιτήματα HTTP από clients που αφορούν τις παραγγελίες
40 |   |   payment.js -----> διαδρομές για αιτήματα HTTP από clients που αφορούν τις πληρωμές
41 |   |
42 |   ---utils -----> χρήσιμες κλάσεις και μέθοδοι
43 |   |   apiFilters.js -----> Φίλτρα για ταξινόμηση και αναζήτηση
44 |   |   apiFilters2.js -----> Φίλτρα για ταξινόμηση και αναζήτηση
45 |   |   cloudinary.js -----> Διαχείριση αποθήκευσης εικόνων στο Cloudinary
46 |   |   emailTemplates.js -----> Προκαθορισμένο email template για αποστολή email ανάκτησης στοιχείων
47 |   |   errorHandler.js -----> Κλάση για δημιουργία κατάλληλων status Code και message ως απόκριση
48 |   |   |   σε περίπτωση σφάλματος(πχ σε κάποιο controller)
49 |   |   sendEmail.js -----> Μέθοδοι αποστολής email επιβεβαίωσης στους χρήστες
50 |   |   sendToken.js -----> Δημιουργία και αποστολή JWT Tokens στους χρήστες
51 |

```

Εικόνα 5.1 Δομή Φακέλου backend

### 5.1.2 Ρύθμιση του EntryPoint Αρχείου

Σε αυτό το κεφάλαιο θα δούμε την διαμόρφωση του αρχείου που αφορά το entryPoint για την εκκίνηση του Server της διαδικτυακής εφαρμογής CheapGames κατά την υλοποίηση με MERN STACK. Το αρχείο αυτό ονομάστηκε app.js και δημιουργήθηκε μέσα στο φάκελο app.js.

Το app.js είναι το κεντρικό αρχείο του backend, καθώς:

- Εκκινεί τον Express Server, επιτρέποντας την αλληλεπίδραση με τους clients.
- Συνδέει τον server με τη βάση δεδομένων MongoDB.
- Ρυθμίζει ποια endpoints είναι διαθέσιμα για αιτήματα από clients.
- Περιέχει μηχανισμούς διαχείρισης σφαλμάτων unhandled promise rejections, διασφαλίζοντας τη σταθερότητα της εφαρμογής.

Αρχικά στο αρχείο app.js εισάγονται οι απαραίτητες βιβλιοθήκες και συναρτήσεις που απαιτούνται για τη λειτουργία του server (Εικόνα 5.2). Η συνάρτηση connectDataBase χρησιμοποιείται για την σύνδεση του Server με τη βάση δεδομένων και η υλοποίηση της στο αρχείο “./config/dbConnect.js” παρουσιάζεται επόμενο κεφάλαιο.

```
backend > JS app.js > ...
1 | // Εισαγωγή απαιτούμενων βιβλιοθηκών και modules
2 | import express from "express";
3 | // Δημιουργία instance του Express
4 | const app= express();
5 | import bodyParser from 'body-parser';
6 | import dotenv from 'dotenv';
7 | import cookieParser from "cookie-parser";
8 | // Εισαγωγή της συνάρτησης για τη σύνδεση με τη βάση δεδομένων
9 | import { connectDataBase } from "./config/dbConnect.js";
10 | // Middleware για τη διαχείριση λαθών
11 | import errorMiddleware from "./middlewares/errors.js"
12
```

Εικόνα 5.2 Εισαγωγή Βιβλιοθηκών και συναρτήσεων στο αρχείο app.js

Στο αρχείο app.js υλοποιήθηκε :

- μία ρύθμιση για το όριο των json που μπορεί να δεχθεί ο server ως 10Mb για λόγους ασφάλειας όπως επιθέσεις DOS Attacks (Γραμμή 23-25 στην εικόνα 5.3)
- Ρύθμιση για την επεξεργασία των δεδομένων που στέλνονται από τον εξωτερικό πάροχο Stripe αποκλειστικά ως json (Γραμμή 32 στην εικόνα 5.3)
- Ρύθμιση για την χρήση cookies ώστε να μπορεί ο server να στέλνει cookies
- Εισαγωγή των αρχείων των routes και ορισμός των endpoints στα οποία μπορεί να δέχεται αιτήματα ο Server (Γραμμές 37-45, Εικόνα 5.3). Ορίστηκε το πρόθεμα “api/v1/” και αυτό σημαίνει ότι κατά τη χρήση των routes για την υλοποίηση της δυνατότητας αποστολής αιτημάτων από το frontend, θα πρέπει να χρησιμοποιείται μπροστά από κάθε route. Ο ορισμός του προθέματος αποτελεί κοινή πρακτική για λόγους οργάνωσης, επέκτασης (πχ καινούρια έκδοση “api/v2/” της διαδικτυακής

- εφαρμογής) καθώς και αποτροπής συγκρούσεων
- Χρήση Ενδιάμεσου Κώδικα (Middleware) για την διαχείριση των σφαλμάτων.
  - Εκκίνηση του Server

```
22 connectDataBase();
23 app.use(
24   express.json({
25     limit: "10mb", // Όριο μεγέθους δεδομένων JSON
26     verify: (req, res, buf) => {
27       req.rawBody = buf.toString();
28     },
29   })
30 );
31 // Χρησιμοποιείται για την επεξεργασία raw JSON δεδομένων
32 app.use(bodyParser.raw({ type: 'application/json' }));
33 // Ενεργοποίηση του cookie-parser για χρήση cookies
34 app.use(cookieParser());
35
36 // Εισαγωγή και χρήση των αρχείων με τα routes
37 import gameRoutes from "./routes/games.js"
38 import authRoutes from "./routes/auth.js"
39 import orderRoutes from "./routes/orders.js"
40 import paymentRoutes from "./routes/payment.js"
41 //Καθορισμός των endpoints
42 app.use("/api/v1", gameRoutes);
43 app.use("/api/v1", authRoutes);
44 app.use("/api/v1", orderRoutes);
45 app.use("/api/v1", paymentRoutes);
46
47 //Using error middleware
48 app.use(errorMiddleware);
49 // Εκκίνηση του server
50 const server = app.listen(process.env.Port, () => {
51   console.log(`Server started on Port: ${process.env.Port} in ${process.env.NODE_ENV} mode`);
52 });
```

Εικόνα 5.3 Απόσπασμα κώδικα από αρχείο App.js

### 5.1.3 Σύνδεση Server με τη MongoDB μέσω Mongoose και μηχανισμός GridFs

Για τη σύνδεση του Server με τη βάση Δεδομένων MongoDB μέσω του ODM (Object Data Modeling) Mongoose, θα πρέπει αρχικά να έχουμε εγκαταστήσει το πακέτο mongoose που είναι απαραίτητο για την ενσωμάτωση του ODM. Όπως και στις άλλες εξαρτήσεις, γίνεται μέσω του terminal και της εντολής `npm install mongoose` (npmjs, 2024).

Για τη σύνδεση του server με τη βάση δεδομένων MongoDB, δημιουργήθηκε το αρχείο `dbConnect.js` (Εικόνα 5.4) μέσα στο φάκελο `config` (εικόνα 5.1).

```
backend > config > JS dbConnect.js > ...
1 import mongoose from 'mongoose'; //Εισάγουμε το πακέτο mongoose με το import
2
3 export const connectDataBase = () => { //μέθοδος ConnectDataBase
4   let DB_URI = ""; //Δημιουργία μεταβλητής για τη διεύθυνση
5   //θα συνδεθεί ο server
6
7
8   if (process.env.NODE_ENV === 'DEVELOPMENT') { //Περίπτωση npm run dev
9     DB_URI = process.env.DB_LOCAL_DEV_URI; //Η μεταβλητή παίρνει την διεύθυνση δοκιμαστικής βάσης
10  }
11  if (process.env.NODE_ENV === 'PRODUCTION') { //Περίπτωση npm run prod
12    DB_URI = process.env.DB_LOCAL_PROD_URI; //Η μεταβλητή παίρνει την διεύθυνση βάσης παραγωγής
13  }
14  mongoose.set('strictQuery', false)
15  mongoose.connect(DB_URI, { //Σύνδεση με τη βάση μέσω mongoose
16
17  }) //Ακολουθούν μηνύματα επιτυχούς σύνδεσης ή σφάλματος
18  .then((con) => {
19    console.log('MongoDB Database connected with HOST: ${con?.connection?.host}');
20  })
21  .catch((err) => {
22    console.error('Database connection error: ${err.message}');
23  });
24  };
```

Εικόνα 5.4 Αρχείο dbConnect.js - Σύνδεση με τη βάση MongoDB μέσω Mongoose

- **GridFs**

Ο μηχανισμός GridFs χρησιμοποιείται για ανέβασμα αρχείων μεγαλύτερων των 16MB στη βάση MongoDB. Κατά τη χρήση του GridFS ένα αρχείο διαιρείται σε μικρότερα κομμάτια, τα οποία αποθηκεύονται σε ξεχωριστά documents ενώ οι σχετικές πληροφορίες για αυτό αποθηκεύονται σε ξεχωριστή συλλογή (gameFiles.files, σχήμα). Τα πακέτα που συνθέτουν το αρχείο και οι πληροφορίες του αρχείου αποτελούν ένα bucket. Για την υλοποίηση αυτού του μηχανισμού δημιουργήθηκε το αρχείο gridFsCongig.js μέσα στο φάκελο config. Στην εικόνα 5.5 μπορείτε να δείτε τον κώδικα υλοποίησης και τις απαραίτητες βιβλιοθήκες οι οποίες εγκαταστάθηκαν μέσω της εντολή npm install όπως και στην περίπτωση των άλλων εξαρτήσεων ενημερώνοντας το αρχείο package.json.

```
backend > config > JS gridFsConfig.js > ...
1 import mongoose from 'mongoose'; // Εισαγωγή της βιβλιοθήκης mongoose για σύνδεση με τη MongoDB
2 import pkg from 'mongodb'; // Εισαγωγή του πακέτου mongodb για χρήση του GridFSBucket
3 const { GridFSBucket } = pkg; // Απόσπαση (destructuring) της κλάσης GridFSBucket από το πακέτο
4
5 let gfs; // Δήλωση μεταβλητής για αποθήκευση του instance του GridFSBucket
6
7 // Αρχικοποίηση του GridFSBucket
8 // Λαμβάνουμε τη σύνδεση με τη βάση δεδομένων από το mongoose
9 const conn = mongoose.connection;
10 conn.once('open', () => { // Όταν η σύνδεση ανοίξει, αρχικοποιούμε το GridFSBucket
11   // Δημιουργία νέου instance του GridFSBucket με βάση τη βάση δεδομένων που παρέχεται από τη σύνδεση
12   gfs = new GridFSBucket(conn.db, {
13     // Ορισμός του ονόματος του bucket για αποθήκευση αρχείων (π.χ. αρχεία παιχνιδιών)
14     bucketName: 'gameFiles',
15   });
16   // Εμφάνιση μηνύματος επιβεβαίωσης στην κονσόλα
17   console.log('GridFSBucket initialized');
18 });
19 // Εξαγωγή του instance του GridFSBucket για χρήση σε άλλα μέρη της εφαρμογής
20 export { gfs };
```

Εικόνα 5.5 Αρχείο GridFs

#### 5.1.4 Διαχείριση Σφαλμάτων (Error Handling)

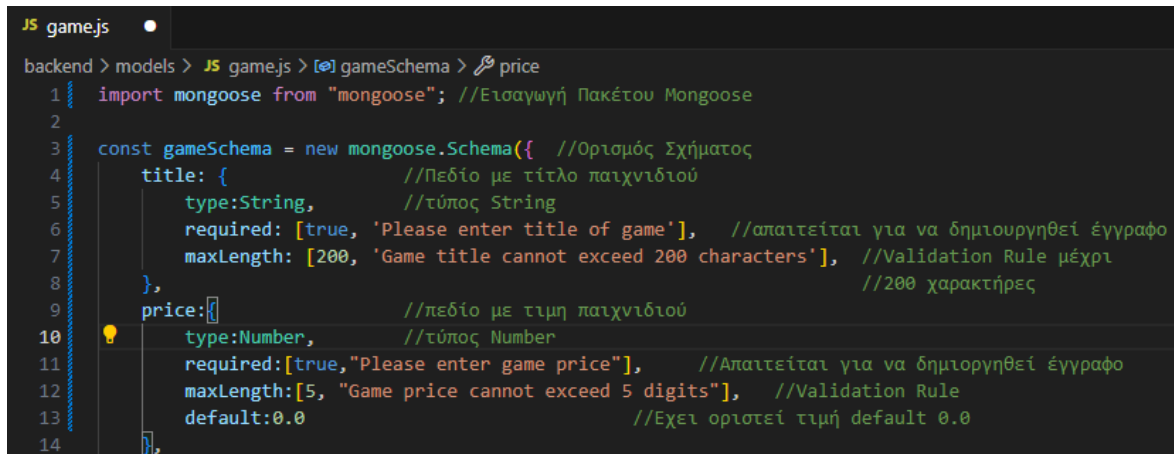
Κατά την υλοποίηση του backend ενός REST API, πρέπει να αποφασιστεί ο τρόπος επιστροφής των σφαλμάτων στον client κατά την εκτέλεση ενός HTTP request. Η επιστροφή των σφαλμάτων γίνεται συνήθως μέσω του HTTP status code που συμπεριλαμβάνεται στην απόκριση του server. Η περιγραφή της αιτίας του σφάλματος μπορεί να περιλαμβάνεται στο body της απόκρισης ως JSON αντικείμενο, το οποίο περιέχει ένα πεδίο message για αυτό το σκοπό. Αυτή η προσέγγιση διευκολύνει την ανάγνωση του σφάλματος από τον χρήστη και επιτρέπει καλύτερη διαχείριση των προβλημάτων που προκύπτουν στο frontend (Subramanian, 2017, p. 85). Για την διαχείριση των σφαλμάτων επιλέχθηκε η παραπάνω προσέγγιση.

#### 5.1.5 Ορισμός Μοντέλων Δεδομένων με Mongoose

Για να υπάρξει η δομή των δεδομένων στο φυσικό επίπεδο όπως παρουσιάστηκε στην Εικόνα 4.8 στο κεφάλαιο 4.3.4, δημιουργήθηκε ο φάκελος με όνομα “models” και ένα αρχείο για κάθε συλλογή εκτός από τη συλλογή gameFile.chunks και gameFiles.files οι οποίες δημιουργούνται αυτόματα μέσω του μηχανισμού GrifFs. Τα αρχεία μοντέλων είναι 4, game.js , user.js, order.js και seasonalDiscount.js και ορίστηκαν με τη χρήση του ODM Mongoose. Επίσης σε κάθε μοντέλο με την χρήση της Mongoose, ορίστηκαν κανόνες επικύρωσης (Validation Rules) των δεδομένων ώστε να αποτρέπεται η λανθασμένη καταχώρηση στοιχείων. Στην εικόνα 5.6 μπορούμε να δούμε ένα απόσπασμα από τον ορισμό του μοντέλου game όπου υλοποιείται ένα Validation Rule με την χρήση του Mongoose στο πεδίο title ώστε αν γίνει προσπάθεια για καταχώρηση πάνω από 200 χαρακτήρες να επιστρέφεται ανάλογο μήνυμα. Επίσης στην εικόνα 5.6 στο πεδίο μπορούμε να δούμε ότι



μπορεί να γίνει ορισμός των δεδομένων που απαιτούνται για τη δημιουργία ενός εγγράφου με ορίζοντας το “require” ως true ή False.



```
JS game.js
backend > models > JS game.js > gameSchema > price
1 import mongoose from "mongoose"; //Εισαγωγή Πακέτου Mongoose
2
3 const gameSchema = new mongoose.Schema({ //Ορισμός Σχήματος
4   title: {                                //Πεδίο με τίτλο παιχνιδιού
5     type:String,                          //τύπος String
6     required: [true, 'Please enter title of game'], //απαιτείται για να δημιουργηθεί έγγραφο
7     maxLength: [200, 'Game title cannot exceed 200 characters'], //Validation Rule μέχρι
8   },                                     //200 χαρακτήρες
9   price: {                                //Πεδίο με τιμή παιχνιδιού
10    type:Number,                          //τύπος Number
11    required:[true, "Please enter game price"], //Απαιτείται για να δημιουργηθεί έγγραφο
12    maxLength:[5, "Game price cannot exceed 5 digits"], //Validation Rule
13    default:0.0                          //Έχει οριστεί τιμή default 0.0
14  },
15 }
```

Εικόνα 5.6 Στιγμιότυπο από αρχείο models/game.js -Validation Rule

### 5.1.6 Routes και Middlewares Αυθεντικοποίησης και Εξουσιοδότησης

Στο κεφάλαιο 4.4 παρουσιάστηκε η καταγραφή των endpoints στον πίνακα Α.1 (Παράρτημα Α, σελ. 138), όπου περιγράφονται οι διαδρομές (routes) στις οποίες ανταποκρίνεται ο server. Κατά την υλοποίηση του backend, τα routes οργανώθηκαν σε τέσσερα αρχεία (auth.js, games.js, orders.js και payment.js), καθένα εκ των οποίων εξυπηρετεί διαφορετικές λειτουργίες. Στην εικόνα 5.7 μπορείτε να δείτε ένα απόσπασμα από την υλοποίηση του αρχείου auth.j. Για τη προστασία της πρόσβασης στις λειτουργίες του API, χρησιμοποιήθηκαν middlewares αυθεντικοποίησης και εξουσιοδότησης σε συγκεκριμένα routes βάσει της καταγραφής της πρόσβασης που έγινε στον πίνακα 4.1. Τα middleware isAuthenticatedUser και isAuthorizeRole τα οποία μπορούν να εντοπιστούν στο απόσπασμα από τον κώδικα της εικόνας 5.7 υλοποιούν τη λογική που παρουσιάστηκε στο κεφάλαιο 2.5 σχετικά με την αυθεντικοποίηση βασισμένη σε κώδικό και Token και εξουσιοδότηση βασισμένη στο ρόλο του χρήστη. Επίσης για κάθε route γίνεται η αντιστοίχιση της λειτουργίας στο οποίο αντιστοιχεί όπως επίσης σχεδιάστηκε στον πίνακα Α1 στο παράρτημα Α σελ. 138.

```
//αντιστοίχιση controller εγγραφής στο route
router.route("/register").post(registerUser);
//αντιστοίχιση controller σύνδεσης στο route
router.route("/login").post(loginUser);
//αντιστοίχιση controller αποσύνδεσης στο route
router.route("/logout").get(logout);
//αντιστοίχιση controller ανάκτησης στοιχείων στο route
router.route("/password/forgot").post(forgotPassword);
//αντιστοίχιση controller ανάκτησης στοιχείων στο route-Μεσολαβεί isAuthenticated
router.route("/password/reset/:token").put(resetPassword);
//αντιστοίχιση controller ανάκτησης προφίλ χρήστη στο route-Μεσολαβεί isAuthenticated
router.route("/me").get(isAuthenticatedUser, getUserProfile);
router.route("/me/update").put(isAuthenticatedUser, updateProfile);
router.route("/password/update").put(isAuthenticatedUser, updatePassword);
router.route("/favourites")
.get(isAuthenticatedUser, getAllFavoriteGames)
.post(isAuthenticatedUser, addFavoriteGame)
.delete(isAuthenticatedUser, removeFavoriteGame);
router.route("/purchased_games").get(isAuthenticatedUser, getPurchasedGames);
router.route("/purchased_games").put(isAuthenticatedUser, updateTimePlayed);

router
.route("/admin/users")
.get(isAuthenticatedUser, authorizeRoles("admin"), allUsers);
router
.route("/admin/users/:id")
.get(isAuthenticatedUser, authorizeRoles("admin"), getUserDetails)
.put(isAuthenticatedUser, authorizeRoles("admin"), updateUser)
.delete(isAuthenticatedUser, authorizeRoles("admin"), deleteUser)

export default router;
```

Εικόνα 5.7 Κώδικας για αντιστοίχιση route, controllers, και προστασία controller με middlewares isAuthenticatedUser και authorizeRoles

### 5.1.7 Controllers

Στο προηγούμενο Κεφάλαιο 5.1.6 παρουσιάστηκε η αντιστοίχιση των controllers σε μια διαδρομή. Σε αυτό το κεφάλαιο θα παρουσιαστεί σε μια σύντομη περιγραφή η υλοποίηση των controllers. Για τις ανάγκες των controllers δημιουργήθηκε ο φάκελος controllers μέσα στο φάκελο backend και μέσα στον φάκελο controllers τέσσερα αρχεία:

- authControllers.js: Περιέχει τα controllers στις γραμμές 2, 4, 5, 6 (/favourites και /purchasedGames), 8, 9, 10, 11, 12, 13, 14, 15, 16, 25, 37, 38, 39, 40 του πίνακα Α.1 Παραρτήματος Α Σελ. 163 τα οποία αφορούν κυρίως την διαχείριση των χρηστών, τις λειτουργίες σύνδεσης, εγγραφής και ανάκτησης στοιχείων καθώς και διαχείριση αγαπημένων και αγορασμένων παιχνιδιών του χρήστη.
- gameControllers.js: Περιέχει τα controllers στις γραμμές 1,6 ("/games"), 7, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 46, 47, 48, 49, 50, 51 του πίνακα Α.1 Παραρτήματος Α Σελ. 163 τα οποία αφορούν την διαχείριση των παιχνιδιών, των

αρχείων τους και των reviews.

- `orderControllers.js`: Περιέχει τα controllers στις γραμμές 22, 23, 41, 42, 43, 44 του πίνακα A.1 στο Παραρτήμα Α Σελ. 163 τα οποία σχετίζονται με την διαχείριση των παραγγελιών.
- `paymentControllers.js`: Περιέχει τα controllers που σχετίζονται με τη διαχείριση των πληρωμών μέσω Stripe, στις γραμμές 20, 21 του πίνακα A.1 Παραρτήματος Α Σελ. 163.

Σε κάθε αρχείο controller γίνεται εισαγωγή των απαραίτητων μοντέλων από τα αρχεία που παρουσιάστηκαν στο κεφάλαιο 4 ώστε να προγραμματιστεί η διαχείριση των δεδομένων στις συλλογές της βάσης δεδομένων MongoDB. Σε κάθε αρχείο υπάρχουν συναρτήσεις οι οποίες έχουν όνομα, αντιστοιχούν σε ένα controller και εξαγονται ώστε να μπορούν να χρησιμοποιηθούν από τα αρχεία routes.

## 5.2 Υλοποίηση Backend με PERN Stack

Η δομή των φακέλων που βρίσκονται μέσα στο backend κατά την υλοποίηση με PERN stack δεν έχει καμία διαφορά από την υλοποίηση με MERN καθώς δεν χρειάστηκε κάποιος καινούριος φάκελος (Εικόνα 5.8).

```
3 C:.\n4   app.js\n5   └── config\n6       config.env\n7       dbConnect.js\n8       multerConfig.js\n9   └── controllers\n10       authControllers.js\n11       gameControllers.js\n12       orderControllers.js\n13       paymentControllers.js\n14   └── middlewares\n15       auth.js\n16       catchAsyncErrors.js\n17       errors.js\n18       uploadCheckGame.js\n19   └── models\n20       favouriteGame.js\n21       game.js\n22       gameFile.js\n23       image.js\n24       order.js\n25       orderGame.js\n26       purchasedGame.js\n27       relationships.js\n28       review.js\n29       seasonalDiscount.js\n30       user.js\n31   └── routes\n32       auth.js\n33       games.js\n34       orders.js\n35       payment.js\n36   └── utils\n37       apiFilters.js\n38       apiFilters2.js\n39       cloudinary.js\n40       emailTemplates.js\n41       errorHandler.js\n42       sendEmail.js\n43       sendToken.js
```

Εικόνα 5.8, Δομή Φάκελων και αρχείων Backend (PERN Stack)

Υπάρχουν ίδια αρχεία και διαφορές στις δύο υλοποιήσεις με τα πιο σημαντικά να περιγράφονται παρακάτω:

- Το αρχείο config.env έχει τις ευαίσθητες μεταβλητές ακριβώς με τον ίδιο τρόπο που υπάρχουν και στην υλοποίηση με MERN
- Το περιεχόμενο του αρχείου dbConnect.js άλλαξε ώστε να εφαρμόζει τη σύνδεση με το PostgreSQL βάση μέσω Sequelize (Εικόνα 5.9)

```
backend > config > JS dbConnect.js > ...
1 //Εισαγωγή Βιβλιοθηκών
2 import { Sequelize } from 'sequelize';
3 import dotenv from 'dotenv';
4
5 dotenv.config({ path: 'backend/config/config.env' });
6 //Μεταβλητή για δημιουργία instance
7 let sequelize;
8 //Ελεγχος για Περιβάλλον Development ή Production
9 if (process.env.NODE_ENV === 'DEVELOPMENT') {
10   sequelize = new Sequelize(process.env.DB_LOCAL_URI, {
11     dialect: 'postgres',
12     logging: false, // Αν δεν θέλεις να βλέπεις τα logs
13   });
14 } else if (process.env.NODE_ENV === 'PRODUCTION') {
15   sequelize = new Sequelize(process.env.DB_PROD_URI, {
16     dialect: 'postgres',
17     logging: false, // Αν δεν θέλεις να βλέπεις τα logs
18   });
19 }
20 //Συνάρτηση για σύνδεση στη βάση
21 export const connectDatabase = async () => {
22   try {
23     await sequelize.authenticate(); // Ελέγχουμε αν η σύνδεση είναι επιτυχής
24     console.log('PostgreSQL Database connected successfully!');
25   } catch (error) {
26     console.error('Unable to connect to the database:', error);
27     process.exit(1); // Αν δεν μπορεί να συνδεθεί η εφαρμογή σταματά
28   }
29 };
30
31 // Εξάγουμε το `sequelize` για να το χρησιμοποιήσουμε στα μοντέλα
32 export { sequelize };
33
```

Εικόνα 5.9 Σύνδεση με Βάση Δεδομένων μέσω Sequelize

- Δεν δημιουργήθηκε το αρχείο GridFs καθώς δεν χρησιμοποιήθηκε κάποιος μηχανισμός για την κοπή του αρχείου του παιχνιδιού σε πακέτα. Στην PERN υλοποίηση (PostgreSQL), το αρχείο αποθηκεύεται ως BLOB/Bytea στο column fileData του πίνακα fileGames, άρα δεν χρειάζεται GridFS.
- Ο φάκελος models πλέον θα έχει 11 αρχεία, ένα για όλους τους πίνακες όπως παρουσιάζονται και στο ΜΟΣ του σχήματος της εικόνα 4.4 του κεφαλαίου 4 και ένα με όνομα relationship.js στο οποίο ορίζονται οι σχέσεις και οι πολλαπλότητες. Τα δεδομένα στην εικόνα 4.7 του κεφαλαίου 4.3.4 που εμφανίζονται στο φυσικό επίπεδο είναι αποτέλεσμα του ορισμού τους μέσα στα αρχεία models. Στην εικόνα 5.10 παρουσιάζεται ένα αποσπάσμα από την υλοποίηση των μοντέλου user με το sequelize ενώ στην εικόνα 5.11 παρουσιάζεται ένα απόσπασμα του αρχείου relationships.js στο οποίο μπορείτε να δείτε την υλοποίηση των σχέσεων.

```
backend > models > JS user.js > ...
1 import { DataTypes } from 'sequelize';
2 import { sequelize } from '../config/dbConnect.js'; // Σύνδεση με τη βάση
3 import bcrypt from 'bcryptjs'; // Χρήση import για bcrypt
4 import crypto from 'crypto'; // Χρήση import για crypto
5 import jwt from 'jsonwebtoken'; // Χρήση import για jwt
6 // Ορισμός του Μοντέλου Χρήστη
7 const User = sequelize.define('User', {
8   id: {
9     type: DataTypes.UUID,
10    defaultValue: DataTypes.UUIDV4,
11    primaryKey: true,
12  },
13  name: {
14    type: DataTypes.STRING,
15    allowNull: false,
16    validate: {
17      len: [1, 50],
18    },
19  },
20  email: {
21    type: DataTypes.STRING,
22    allowNull: false,
23    unique: true,
24    validate: {
25      isEmail: true,
26    },
27  },
28  password: {
29    type: DataTypes.STRING,
30    allowNull: false,
31    validate: {
32      len: [6, 255],
33    },
34  },
35  age: {
36    type: DataTypes.INTEGER,
37    allowNull: false,
38  },
39  }, {
40    hooks: {
41      beforeCreate: async (user) => {
42        // Κρυπτογράφηση κωδικού πριν την αποθήκευση
43        if (user.password) {
44          const hashedPassword = await bcrypt.hash(user.password, 10);
45          user.password = hashedPassword;
46        }
47      },
48      beforeUpdate: async (user) => {
49        // Κρυπτογράφηση κωδικού πριν την ενημέρωση (Update)
50        if (user.password) {
51          const hashedPassword = await bcrypt.hash(user.password, 10);
52          user.password = hashedPassword;
53        }
54      },
55    },
56  }, {
57    // Μέθοδος για τη δημιουργία JWT
58    prototype: {
59      getJwtToken: function () {
60        return jwt.sign({ id: this.id }, process.env.JWT_SECRET, {
61          expiresIn: process.env.JWT_EXPIRES_TIME,
62        });
63      },
64      // Μέθοδος για σύγκριση του κωδικού
65      comparePassword: async function (enteredPassword) {
66        return await bcrypt.compare(enteredPassword, this.password);
67      },
68      // Μέθοδος για την αναγέννηση του token για reset password
69      getResetPasswordToken: function () {
70        const resetToken = crypto.randomBytes(20).toString('hex');
71        // Χρησιμοποιούμε το SHA256 για να κάνουμε hash το resetToken
72        this.resetPasswordToken = crypto
73          .createHash('sha256')
74          .update(resetToken)
75          .digest('hex');
76        // Ορίζουμε το χρονικό όριο του token
77        this.resetPasswordExpire = Date.now() + 30 * 60 * 1000;
78        return resetToken;
79      },
80    },
81  },
82  }, {
83    export default User;
84  }
85  );
```

Εικόνα 5.10 Αποσπάσματα κώδικα Αρχείου User Model-Κάποια πεδία και μέθοδοι για το Password

```
backend > models > JS relationships.js > [⌘] setupRelationships
1 // Εισαγωγή των μοντέλων
2 import Game from './game.js';
3 import User from './user.js';
4 import Order from './order.js';
5 import OrderGame from './orderGame.js';
6 import PurchasedGame from './purchasedGame.js';
7 import Image from './image.js';
8 import Review from './review.js';
9 import FavouriteGame from './favouriteGame.js';
10 import GameFile from './gameFile.js';
11 const setupRelationships = () => {
12   // Σχέσεις μεταξύ Game και άλλων μοντέλων
13   Game.hasMany(PurchasedGame, { foreignKey: 'gameId', as: 'purchasedGames', onDelete: 'CASCADE' });
14   PurchasedGame.belongsTo(Game, { foreignKey: 'gameId', as: 'game' });
15
16   Game.hasMany(Image, { foreignKey: 'gameId', as: 'images', onDelete: 'CASCADE' });
17   Image.belongsTo(Game, { foreignKey: 'gameId' });
18
19   Game.hasMany(Review, { foreignKey: 'gameId', as: 'reviews', onDelete: 'CASCADE' });
20   Review.belongsTo(Game, { foreignKey: 'gameId', as: 'game' });
21
22   Game.belongsToMany(User, { through: PurchasedGame, as: 'owners', foreignKey: 'gameId' });
23   User.belongsToMany(Game, { through: PurchasedGame, as: 'ownedGames', foreignKey: 'userId' });
24
25   // Σχέσεις μεταξύ User και άλλων μοντέλων
26   User.hasMany(PurchasedGame, { foreignKey: 'userId', as: 'userPurchases', onDelete: 'CASCADE' });
27   PurchasedGame.belongsTo(User, { foreignKey: 'userId', as: 'user' });
28
29   User.hasMany(Review, { foreignKey: 'userId', as: 'reviews' });
30   Review.belongsTo(User, { foreignKey: 'userId', as: 'user' });
31
32   User.hasMany(Order, { foreignKey: 'userId', as: 'orders' });
33   Order.belongsTo(User, { foreignKey: 'userId', as: 'user' });
34
35   // Σχέσεις μεταξύ Order και Game μέσω OrderGame
36   Order.belongsToMany(Game, { through: OrderGame, as: 'games', foreignKey: 'orderId' });
37   Game.belongsToMany(Order, { through: OrderGame, as: 'orders', foreignKey: 'gameId' });
38
39   Order.hasMany(OrderGame, { foreignKey: 'orderId', as: 'orderItems', onDelete: 'CASCADE' });
40   OrderGame.belongsTo(Order, { foreignKey: 'orderId', as: 'order' });
41
42   Game.hasMany(OrderGame, { foreignKey: 'gameId', as: 'gameOrders', onDelete: 'CASCADE' });
43   OrderGame.belongsTo(Game, { foreignKey: 'gameId', as: 'game' });
}
```

Εικόνα 5.11 Απόσπασμα από αρχείο relationships.js

- Το κάθε αρχείο controller (authControllers.js , gameControllers.js, orderControllers και paymentControllers.js) περιέχει τα ίδια controller όπως και στην υλοποίηση με MERN. Ωστόσο στη PERN, λόγω της διάσπασης των δεδομένων σε περισσότερους πίνακες (10) από ότι στη MERN όπου χρησιμοποιήθηκαν μόνο τέσσερις συλλογές με ενσωματώσεις πινάκων και υποεγγράφων, κάποια controller χρειάζονται πρόσβαση σε περισσότερα μοντέλα με αποτέλεσμα να εισάγονται σε κάθε αρχείο περισσότερα μοντέλα. Επίσης χρησιμοποιούνται οι μέθοδοι του Sequelize για τις αναζητήσεις, ευρέσεις, διαγραφή δεδομένων κ.α που εκτελούνται από τα controllers. Στην εικόνα 5.12 βλέπετε στο απόσπασμα από το orderController.js, το controller newOrder στο οποίο χρησιμοποιήθηκε η μέθοδος createOrder για την δημιουργία παραγγελίας κατά την αποστολή αιτήματος επιτυχής πληρωμής χρήστη από τον εξωτερικό πάροχο Stripe



```

1  import catchAsyncErrors from "../middlewares/catchAsyncErrors.js";
2  import Order from "../models/order.js";
3  import User from "../models/user.js";
4  import Game from "../models/game.js";
5  import OrderGame from "../models/orderGame.js";
6  import PurchasedGame from "../models/purchasedGame.js";
7  import ErrorHandler from "../utils/errorHandler.js";
8  import { Op, fn, col, literal } from 'sequelize'; //Sequelize Operators
9
10 // Δημιουργία νέας παραγγελίας => /api/v1/orders/new
11 export const newOrder = catchAsyncErrors(async (req, res, next) => {
12   const {
13     personalInfo, // Προσωπικές πληροφορίες χρήστη
14     orderItems, // Αντικείμενα που περιλαμβάνονται στην παραγγελία
15     itemsPrice, // Συνολικό κόστος αντικειμένων
16     taxAmount, // Φόρος
17     totalAmount, // Συνολικό ποσό πληρωμής
18     paymentMethod, // Τρόπος πληρωμής
19     paymentInfo, // Πληροφορίες πληρωμής
20   } = req.body;
21
22   // Έλεγχος αν όλα τα απαιτούμενα πεδία υπάρχουν στο αίτημα
23   if (!personalInfo || !orderItems || !itemsPrice || !taxAmount || !totalAmount || !paymentMethod) {
24     return next(new ErrorHandler('Please provide all required fields', 400));
25   }
26
27   // Ανάκτηση του userId από το session ή το JWT token
28   const userId = req.user._id;
29
30
31   // Έλεγχος αν ο χρήστης είναι συνδεδεμένος
32   if (!userId) {
33     return next(new ErrorHandler('User not authenticated', 401));
34   }
35
36   // Δημιουργία νέας εγγραφής παραγγελίας στη βάση δεδομένων
37   const order = await Order.create({
38     personalInfo, // Αποθήκευση προσωπικών πληροφοριών
39     orderItems, // Αντικείμενα της παραγγελίας
40     itemsPrice, // Κόστος αντικειμένων
41     taxAmount, // Φόρος
42     totalAmount, // Συνολικό ποσό
43     paymentMethod, // Μέθοδος πληρωμής
44     paymentInfo, // Πληροφορίες πληρωμής
45     userId, // Σύνδεση της παραγγελίας με τον χρήστη που την έκανε
46   });
47
48   // Επιστροφή της δημιουργημένης παραγγελίας ως απάντηση
49   res.status(201).json({
50     success: true,
51     order,
52   });
53 });

```

Εικόνα 5.12 Απόσπασμα από αρχείο orderController.js

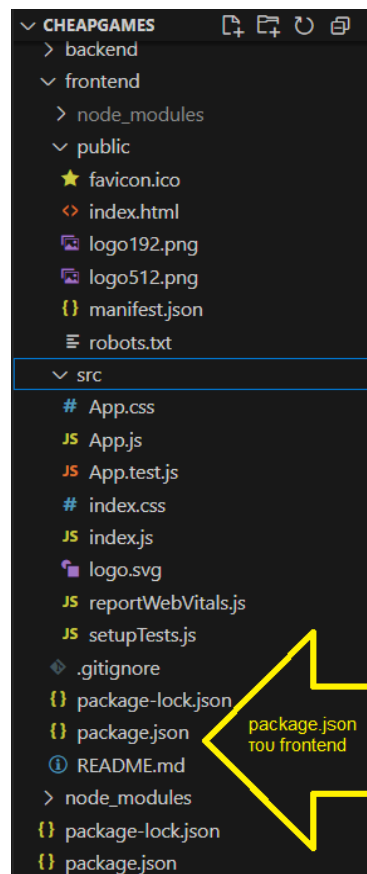
- Το entry point app.js προσαρμόστηκε στη χρήση Sequelize αντί για Mongoose.
- Τα αρχεία routes είναι ίδια με την υλοποίηση με MERN.

## 6. Υλοποίηση FrontEnd

### 6.1 Ρυθμίσεις και Δομή φακέλου Frontend

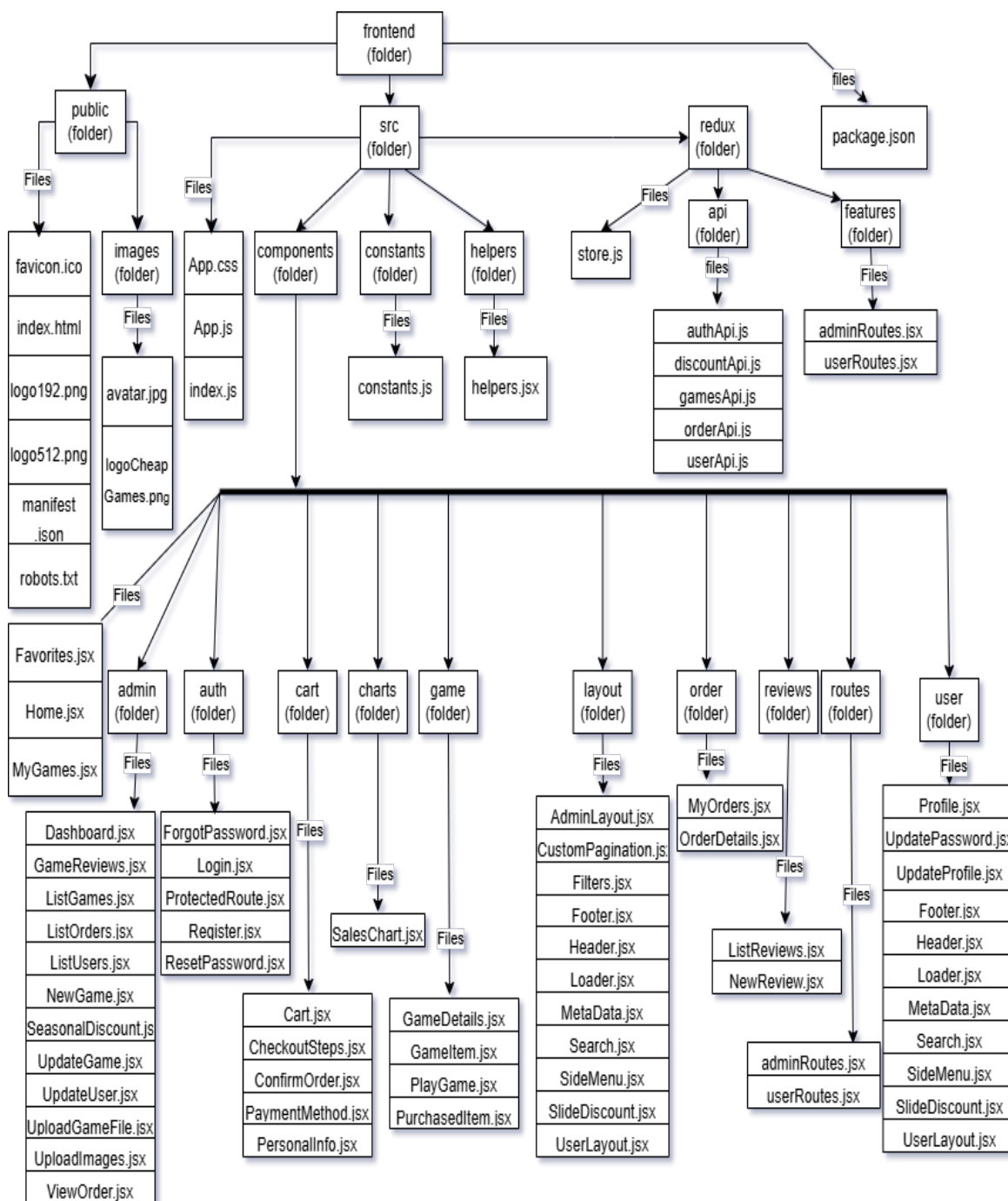
Αρχικά για την υλοποίηση του frontend, δημιουργήθηκε ο φάκελος ο frontend.

Στο φάκελο frontend μέσω της εντολής “npx create-react-app .” στο terminal δημιουργήθηκε ένα πρότυπο φακέλων και αρχείων (Εικόνα 6.1) πάνω στο οποίο μπορούμε να εργαστούμε και να χτίσουμε το frontend της εφαρμογής μας βασισμένο στη React. Η (Create React App, 2024). Η παραπάνω εντολή συγκεκριμενοποιεί ότι η εγκατάσταση αυτή αφορά μόνο το φάκελο frontend και όχι το υπόλοιπο project.



Εικόνα 6.1 Δημιουργία React Προτύπου για Δημιουργία Web Api

Στη συνέχεια εφαρμόστηκαν αλλαγές και διαμορφώσεις στα ήδη υπάρχοντα. Φυσικά υπάρχουν διαφορετικές προσεγγίσεις ως προς τη διαμόρφωση των αρχείων και τη δομή των φακέλων τις οποίες μπορεί να ακολουθήσει κάποιος developer, ανάλογα πάντα τους σκοπούς και στόχους. Η τελική μορφή της δομής των φακέλων και των αρχείων παρουσιάζεται στο σχήμα 6.2.



Εικόνα 6.2 Δομή Φακέλων και Αρχείων Frontend

Όπως και στην υλοποίηση του backend, οι βασικές ρυθμίσεις και βιβλιοθήκες που εγκαταστάθηκαν μέσω npm και terminal του visual studio code και χρησιμοποιήθηκαν για την υλοποίηση του frontend αποτυπώνονται στο αρχείο package.json (Εικόνα 6.3). Το αρχείο είναι ακριβώς ίδιο και στην υλοποίηση με MERN και στην υλοποίηση PERN.

```
frontend > {} package.json > ...
1  {}
2    "name": "frontend",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@reduxjs/toolkit": "^2.2.7",
7      "@testing-library/jest-dom": "^5.17.0",
8      "@testing-library/react": "^13.4.0",
9      "@testing-library/user-event": "^13.5.0",
10     "countries-list": "^3.1.1",
11     "mdbreact": "^5.2.0",
12     "react": "^18.3.1",
13     "react-chartjs-2": "^5.2.0",
14     "react-datepicker": "^7.3.0",
15     "react-dom": "^18.3.1",
16     "react-helmet": "^6.1.0",
17     "react-hot-toast": "^2.4.1",
18     "react-js-pagination": "^3.0.3",
19     "react-redux": "^9.1.2",
20     "react-router-dom": "^6.26.2",
21     "react-scripts": "5.0.1",
22     "react-star-ratings": "^2.3.0",
23     "web-vitals": "^2.1.4"
24   },
25   > Debug
26   "scripts": {
27     "start": "react-scripts start",
28     "build": "react-scripts build",
29     "test": "react-scripts test",
30     "eject": "react-scripts eject"
31   },
```

Εικόνα 6.3 Απόσπασμα Κώδικα από αρχείο package.json

Από τις βιβλιοθήκες που εγκαταστάθηκαν, κάποιες έχουν να κάνουν με μηχανισμούς εμφάνισης έτοιμων γραφικών στοιχείων μέσα στις σελίδες όπως οι react-star-ratings (αστέρια ως εμφάνιση σκορ) , react-hot-toast (Εμφάνιση ειδοποιήσεων) και οι react-js-pagination (μηχανισμός για σελιδοποίηση). Βιβλιοθήκες σαν αυτές είναι ένα δυνατό ατού της react καθώς μειώνουν την πολυπλοκότητα του κώδικα και διευκολύνουν την εργασία των developers. Εκτός από τις βιβλιοθήκες που αφορούν περισσότερο το UI, κάποιες χρησιμοποιούνται περισσότερο για την λειτουργία της σελίδας:

- react & react-dom: Αποτελεί απαραίτητο εργαλείο καθώς μέσω αυτού γίνεται η φόρτωση των στοιχείων
- react-router-dom: Χρησιμοποιείται για τη διαχείριση της πλοήγησης στις σελίδες της εφαρμογής, επιτρέποντας τη δημιουργία δυναμικών routes.

- @reduxjs/toolkit & react-redux: Χρησιμοποιήθηκε για την διαχείριση των δεδομένων στη μεριά το client. Μέσω αυτή της βιβλιοθήκης μειώνεται η πολυπλοκότητα του τρόπου που ορίζονται τα αιτήματα προς τον Server μέσα στο κώδικα συγκεντρώνοντας τα σε αρχεία και φακέλους ενώ επίσης μπορεί να γίνεται ορισμός των δεδομένων που είναι διαθέσιμα σε κάθε σελίδα κατά την πλοήγηση του χρήστη.

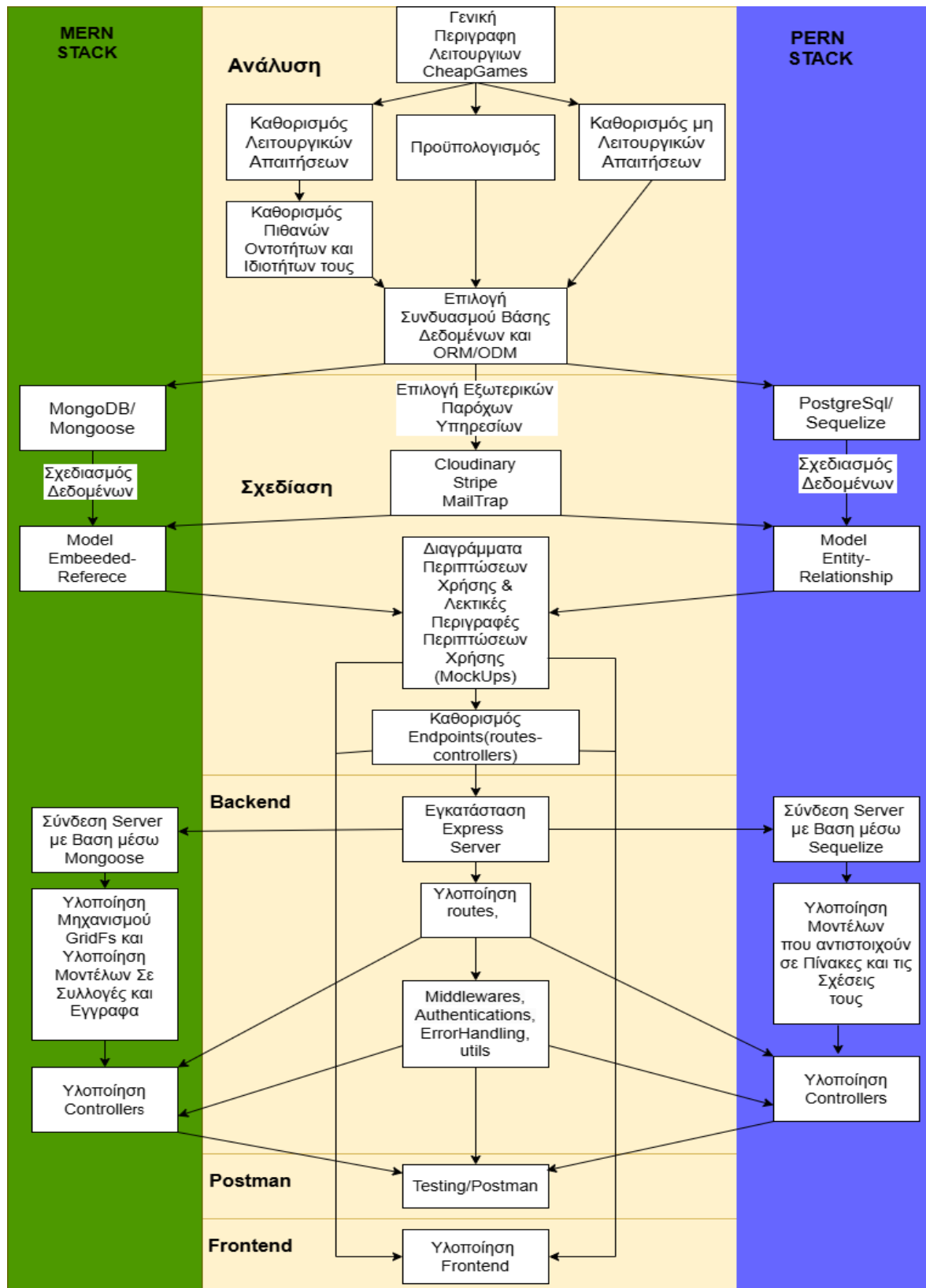
Στο παράρτημα Β Σελ.144 παρουσιάζεται ο πίνακας Β.1 με τα ονόματα του αρχείων και των φακέλων και μία σύντομη περιγραφή για το κάθε ένα.

## 7. Συμπεράσματα

### 7.1 Ανασκόπηση Σταδίων Ανάπτυξης

Στο σχήμα της εικόνας 7.1 βλέπουμε όλα τα στάδια από την ανάλυση μέχρι και την υλοποίηση ξεχωρίζοντας από τα βήματα που αποτελούν καθαρά κομμάτι για τη κάθε στοίβα ξεκινώντας από τη γενική περιγραφή του Cheap Games. Ο μεσαίος διάδρομος αποτελείται από τα κοινά σημεία των δύο στοιβών ενώ οι δύο ακριανές από αποκλειστικά βήματα της MERN και PERN αντίστοιχα. Στην σχήμα της εικόνας 7.1 έχουν συμπεριληφθεί ο προϋπολογισμός και η καταγραφή μη λειτουργικών απαιτήσεων ως στοιχεία της ανάλυσης όπως συμβαίνει σε όλα τα έργα. Μετά την γενική περιγραφή του CheapGames, Ο καθορισμός των λειτουργικών απαιτήσεων μας οδήγησε στον καθορισμό των οντοτήτων και ιδιοτήτων τους και στην συνέχεια στην επιλογή του συνδυασμού της βάσης και ORM ή ODM. Ο σχεδιασμός δεδομένων έγινε για το κάθε stack λαμβάνοντας υπόψη και τους εξωτερικούς παρόχους όπως το Cloudinary. Το επόμενο στάδιο μετά την σχεδίαση των δεδομένων ήταν οι περιγραφές των λειτουργικών απαιτήσεων και σχέδια MockUps για την εμφάνιση της εφαρμογής. Με τον καθορισμό των endpoints, ξεκίνησε η υλοποίηση του backend η οποία θα μπορούσε να γίνει και ταυτόχρονα με τον frontend ή και μετά την υλοποίηση του frontend. Οι λειτουργίες για το κάθε controller υλοποιήθηκαν λαμβάνοντας υπόψη την δομή των δεδομένων στη κάθε βάση και η δοκιμή τους με την χρήση του Postman αποτέλεσε επίσης ένα ξεχωριστό στάδιο.





Εικόνα 7.1 Στάδια από την Ανάλυση ως και την Υλοποίηση

## 7.2 Συμπεράσματα Ανάλυσης και Σχεδίασης

Τα συμπεράσματα όπως προέκυψαν από την ανάλυση και σχεδίαση είναι τα εξής:

- **Ανάλυση Απαιτήσεων:** Κατά την ανάλυση και σχεδίαση του CheapGames, δόθηκε ιδιαίτερη έμφαση στις λειτουργικές απαιτήσεις, καθώς αυτές καθορίζουν άμεσα τις δυνατότητες που προσφέρονται στους χρήστες και διαχειριστές της εφαρμογής. Κατά την υλοποίηση και ολοκλήρωση του Backend, με τη χρήση του Postman έγινε έλεγχος ώστε τα αιτήματα Client και Server να λειτουργούν και να επιστέφουν τα σωστά δεδομένα αλλά θα πρέπει να γίνει μια αναλυτικότερη έρευνα για την δοκιμή του συστήματος σε υψηλό φόρτο αιτημάτων και επισκεπτών λαμβάνοντας υπόψη και μη λειτουργικές απαιτήσεις. Επίσης δεν λήφθηκε υπόψη στο βαθμό που πρέπει η απόδοση των συνδυασμών των τεχνολογιών που χρησιμοποιήθηκαν όσον αφορά την αποθήκευση και την ανάκτηση των αρχείων των παιχνιδιών με τις δύο στοίβες. Η χρήση του ORM sequelize δεν αξιοποιεί στο μέγιστο τις δυνατότητες της PostgreSQL ως προς την αποθήκευση των αρχείων καθώς μπορεί να χρησιμοποιήσει μόνο το τύπο Bytea και η μέθοδος GridFs στην περίπτωση της MERN αν και δείχνει αρκετά λειτουργική και συμβατική με το ODM Mongoose, δεν μετρήθηκε η απόδοση της ώστε να έχουμε μια ολοκληρωμένη άποψη για την απόδοση της εφαρμογής. Η επιλογή εξωτερικών παρόχων, όπως το Cloudinary για αποθήκευση εικόνων ή το MongoDB Atlas για διαχείριση βάσεων δεδομένων, επηρεάζει τον προϋπολογισμό και την αρχιτεκτονική της εφαρμογής.
- **Σχεδιασμός Δεδομένων:** Στο εννοιολογικό επίπεδο, κατά την σχεδίαση των δεδομένων και με τις δύο στοίβες ήταν απαραίτητο να ορίσουμε τις βασικές οντότητες και τις σχέσεις μεταξύ τους κάτι που έγινε με την χρήση ενός απλοποιημένου Μοντέλου Οντοτήτων-Συσχετίσεων και είναι αποδεκτό και στις δύο περιπτώσεις. Επίσης και στο λογικό επίπεδο η ανάπτυξη του μοντέλου που αναπτύχθηκε στο εννοιολογικό επίπεδο συμπληρώνοντας πληροφορίες όπως οντότητες και ιδιότητες αυτών καθώς και την πολλαπλότητα των σχέσεων αποτέλεσε ένα κοινό βήμα. Αυτό που διαφοροποίησε σε μεγάλο βαθμό το Data Modeling με τις δύο στοίβες είναι ο τρόπος που έγινε η μετάβαση στο φυσικό επίπεδο όπου στη περίπτωση της MERN η προσέγγιση έχει να κάνει με μια σειρά από ερωτήματα όπως αυτά που προτείνει η MongoDB για τη λήψη αποφάσεων μεταξύ ενσωματωμένων εγγράφων (embedded documents) και αναφορών (references), επιτρέποντας μια πιο διαισθητική και προσαρμοστική διαδικασία (MongoDB, 2024). Αντίθετα, στην περίπτωση της PERN, η χρήση μιας σχεσιακής βάσης δεδομένων όπως η PostgreSQL έχοντας ως βήμα τη κανονικοποίηση των δεδομένων, προσφέρει μεγαλύτερη αξιοπιστία και έλεγχο της κατασκευής.

## 7.3 Συμπεράσματα Υλοποίησης Backend και Frontend

Τα συμπεράσματα όπως προέκυψαν από την υλοποίηση του backend και του frontend συνοψίζονται στον πίνακα 7.1.

<b>Backend</b>	<ul style="list-style-type: none"> <li>Κοινά Στοιχεία: Η εγκατάσταση του Express server, η διαχείριση των routes, middlewares και βοηθητικών αρχείων ήταν κοινά και στις δύο στοίβες.</li> <li>Διαφορές: Στη MERN, υλοποιήθηκαν 4 μοντέλα δεδομένων και ο μηχανισμός GridFs για διαχείριση αρχείων. Στην PERN, υλοποιήθηκαν 10 μοντέλα που αντιστοιχούν σε πίνακες, με τη χρήση του ORM Sequelize για διαχείριση των σχέσεων. τα controllers στη MERN σχεδιάστηκαν ώστε να διαχειρίζονται δεδομένα σε μορφή JSON, αξιοποιώντας την ευελιξία της MongoDB μέσω του ODM Mongoose. Στη PERN, αν και η λογική των controllers ως προς τις λειτουργίες είναι ίδια με την MERN, έπρεπε να λαμβάνονται υπόψη οι σχέσεις μεταξύ των πινάκων.</li> </ul>
<b>Frontend</b>	Το frontend υλοποιήθηκε με βάση τις περιπτώσεις χρήσης και ήταν κοινό και για τις δύο στοίβες, παρά τις διαφορές στην υλοποίηση του backend.

Πίνακας 7.1 Συμπεράσματα από Υλοποίηση backend και frontend

## 7.4 Ευελιξία και Ταχύτητα

Η ευελιξία που προσφέρεται κατά την ανάπτυξη αλλά και την συντήρηση ενός έργου είναι ένα σημαντικό κριτήριο σύγκρισης και επιλογής ανάμεσα στις τεχνολογίες που θα χρησιμοποιηθούν.

Κατά την ανάπτυξη του CheapGames με τις δύο τεχνολογικές στοίβες προέκυψαν αλλαγές σε αρχικές αποφάσεις ή διόρθωση λαθών με αποτέλεσμα την πρόσθεση ή αφαίρεση πεδίων στα μοντέλα των δεδομένων. Στη περίπτωση της MongoDB, η πρόσθεση ή αφαίρεση πεδίων σε μια συλλογή ή έγγραφο, ήταν πολύ πιο εύκολη χωρίς να επηρεάζονται τα ήδη καταχωρημένα δεδομένα εκείνη τη στιγμή. Αυτό οφείλεται στο ότι η MongoDB δεν απαιτεί ένα αυστηρά προκαθορισμένο σχήμα δεδομένων αλλά φορτώνεται κάθε φορά “δυναμικά” δηλαδή ανάλογα τις υπάρχουσες συνθήκες και ρυθμίσεις. Αυτή η ευελιξία όχι μόνο επιτάχυνε την υλοποίηση, αλλά μπορεί να διευκολύνει και τη μετέπειτα συντήρηση μιας εφαρμογής, καθώς επιτρέπει την άμεση προσαρμογή σε νέες απαιτήσεις χωρίς πολύπλοκες διαδικασίες

Αντίθετα στη περίπτωση της PostgreSQL (PERN), κατά την υλοποίηση του έργου αν έχουμε δημιουργήσει μία σχέση-πίνακα με κάποια πεδία και έχουμε καταχωρήσει δεδομένα σε αυτόν τον πίνακα και θέλουμε να σβήσουμε ή να προσθέσουμε κάποιο πεδίο θα πρέπει ή να σβήσουμε τα ήδη υπάρχοντα δεδομένα ώστε να καθοριστεί ξανά το σχήμα ή να χρησιμοποιήσουμε migrations (Μεταναστεύσεις). Τα migrations είναι ένας τρόπος να διαχειριζόμαστε καταστάσεις των δεδομένων και να κρατάμε ιστορικό αυτών και στο στάδιο της ανάπτυξης αλλά στα στάδια λειτουργίας και συντήρησης.

Στην περίπτωση του CheapGames όπου ήταν απαραίτητη η ευελιξία ως προς τις τροποποιήσεις και η ταχύτητα ανάπτυξης για την ολοκλήρωση του έργου, η ανάπτυξη με MERN αποδείχθηκε ταχύτερη καθώς δεν χρειάστηκε να διαγράφονται τα δεδομένα σε κάθε τροποποίηση ή το extra βήμα των migrations. Ωστόσο, αυτό δεν σημαίνει ότι τα migrations δεν αποτελούν πλεονέκτημα σε άλλες περιπτώσεις διαδικτυακών εφαρμογών όπου οι

αλλαγές είναι πολύπλοκες και χρειάζεται καταγραφή των εκδόσεων των δεδομένων. Παρόλο που η MERN στοίβα προσφέρει ευελιξία, η PERN στοίβα παρέχει καλύτερο έλεγχο δεδομένων, περισσότερη συνέπεια και ευκολότερη διαχείριση των πολύπλοκων σχέσεων, κάτι που μπορεί να κάνει τη μακροπρόθεσμη συντήρηση πιο αξιόπιστη.

## 7.5 Διαχείριση Διαγραφής Δεδομένων

- MERN: Η διαγραφή ενός χρήστη διαγράφει τα ενσωματωμένα δεδομένα του, αλλά αφήνει "ορφανά" έγγραφα αγορών τα οποία βρίσκονται σε άλλη συλλογή χωρίς να δηλώνεται σε ποιον χρήστη ανήκουν. Η έλλειψη ενσωματωμένων εντολών και κανόνων απαιτεί πρόσθετη λογική για τη σωστή διαγραφή σχετικών δεδομένων που δεν βρίσκονται εντός ενός εγγράφου. Κατά την ανάλυση και σχεδίαση με MERN, θα πρέπει να δίνεται έμφαση και προσοχή στη περίπτωση που τα δεδομένα που διαγράφονται αφορούν αναφορές σε άλλα έγγραφα.
- PERN: Η χρήση εντολών όπως "cascade" και "setNull" διευκολύνει τη διαχείριση των διαγραφών, προσφέροντας περισσότερες επιλογές ως προς τον έλεγχο.

## 7.6 Επιλογή Στοιβάς MERN vs PERN

Η έρευνα για την υλοποίηση του CheapGames, έδειξε ότι ένα e-commerce site μπορεί να υλοποιηθεί και με τις δύο στοίβες MERN και PERN. Επίσης ανέδειξε τα σημεία ελέγχου ή σημεία σύγκρισης στα οποία μπορούμε να εστιάσουμε για να επιλέξουμε την μία στοίβα ή την άλλη κρίνοντας τα χαρακτηριστικά τους.

Στον πίνακα 7.2, περιγράφονται τα κριτήρια επιλογής όπως αυτά αναδείχθηκαν από την υλοποίηση με τις δύο στοίβες.

Κριτήριο	MERN	PERN
<b>Απαιτούμενες Γνώσεις Προγραμματισμού</b>	Απαιτείται μόνο γνώση JavaScript, καθώς όλες οι τεχνολογίες της στοίβας χρησιμοποιούν αυτή τη γλώσσα.	Εκτός από τη γνώση JavaScript, απαιτείται κατανόηση της SQL για τη διαχείριση της σχεσιακής βάσης δεδομένων PostgreSQL.
<b>Ευελιξία στο Σχεδιασμό Δεδομένων</b>	Η MongoDB επιτρέπει εύλεκτο σχήμα, διευκολύνοντας την ενσωμάτωση εγγράφων και αναφορών χωρίς αυστηρούς κανόνες σχέσεων.	Η PostgreSQL απαιτεί κανονικοποίηση και αυστηρό έλεγχο σχέσεων, προσφέροντας μεγαλύτερη αξιοπιστία και έλεγχο.
<b>Ταχύτητα Ανάπτυξης</b>	Η ευελιξία της MongoDB επιταχύνει την ανάπτυξη, καθώς οι αλλαγές στα μοντέλα δεδομένων δεν απαιτούν περίπλοκες διαδικασίες.	Οι αλλαγές στα σχήματα δεδομένων απαιτούν χρήση migrations ή διαγραφή δεδομένων, επιβραδύνοντας την ανάπτυξη.
<b>Συντήρηση και Επεκτασιμότητα</b>	Η δυναμική φύση του σχήματος της MongoDB διευκολύνει την	Η αυστηρή δομή της PostgreSQL προσφέρει

	προσαρμογή σε νέες απαιτήσεις, καθιστώντας τη συντήρηση πιο ευέλικτη.	συνέπεια και ευκολότερη διαχείριση πολύπλοκων σχέσεων, καθιστώντας τη συντήρηση πιο αξιόπιστη μακροπρόθεσμα.
<b>Διαχείριση Πολύπλοκων Δεδομένων</b>	Η ευέλικτη δομή της MongoDB μπορεί να οδηγήσει σε πολυπλοκότητα κατά τη διαχείριση πολύπλοκων σχέσεων μεταξύ δεδομένων	Η PostgreSQL, με την αυστηρή κανονικοποίηση και τους σαφείς ορισμούς σχέσεων, διευκολύνει τη διαχείριση σύνθετων δομών δεδομένων.

Πίνακας 7.2 Κριτήρια Επιλογής Στοιβάς

Βάση του πίνακα 7.2, θα μπορούσαμε να περιγράψουμε την επιλογή της στοιβάς ως μία ιεράρχηση των αναγκών που υπάρχουν για ένα έργο.

Όσο πιο πολύπλοκες είναι οι σχέσεις και αυστηρός ο σχεδιασμός μεταξύ των δεδομένων τόσο πιο πολύ αυξάνεται η καταλληλότητα της PERN ενώ όταν χρειάζεται ευελιξία και ακανόνιστο σχήμα δεδομένων προτιμάται η MERN. Με την χρήση της MERN μπορούμε να εφαρμόζουμε εύκολα μελλοντικές αλλαγές αλλά με την PERN έχουμε τη δυνατότητα να έχουμε ένα σταθερό σχήμα δεδομένων όπου ελαχιστοποιούνται τα λάθη βάσης της αυστηρής σχεδίασης.

## Βιβλιογραφία

### Βιβλία

- Brown, E. (2014). Web Development with Node and Express. Sebastopol, CA: O'Reilly Media.
- Chodorow, K. (2013). MongoDB: The Definitive Guide (2η έκδοση). Sebastopol, CA: O'Reilly Media.
- Daniele, D. (2022). Supercharging Node.js Applications with Sequelize. Birmingham: Packt Publishing Ltd.
- Date, C. J. (1990). An Introduction to Database Systems, Volume II. Reading, MA: Addison-Wesley Publishing Company. ISBN: 0-201-51381-1.
- Douglas, K. & Douglas, S. (2003). PostgreSQL: A Comprehensive Guide to Building, Programming and Administering PostgreSQL Databases. Sams Publishing.
- Elmasri, R. & Navathe, S. B. (2016). Fundamentals of Database Systems (7η έκδοση). Pearson.
- Richardson, L. & Amundsen, M. (2013). RESTful Web APIs. Sebastopol, CA: O'Reilly Media.
- Simsion, G. & Witt, G., 2005. Data Modeling Essentials. 3rd ed. San Francisco: Morgan Kaufmann
- Schmidt, J. W. & Brodie, M. L. (Επιμ.) (1983). Relational Database Systems: Analysis and Comparison. Νέα Υόρκη: Springer-Verlag.
- Shama, S. (2018). Full-Stack React Projects. Birmingham: Packt Publishing.
- Subramanian, V. (2017). Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. Νέα Υόρκη: Apress.
- Chapman, N. & Chapman, J. (2012). Authentication and Authorization on the Web. Scotland: MacAvon Media.

### Άρθρα

- Aleryani, A.Y. (2016) 'Comparative Study between Data Flow Diagram and Use Case Diagram', *International Journal of Scientific and Research Publications*, 6(3), σ. 124.
- Barnes, J. M. (2007). 'Object-Relational Mapping as a Persistence Mechanism for Object-Oriented Applications', *Mathematics, Statistics, and Computer Science Honors Projects*, 6.
- Ireland, C., Bowers, D., Newton, M. & Waugh, K. (2009). 'Understanding Object-Relational Mapping: A Framework-Based Approach', *International Journal on Advances in Software*, 2(2&3), σσ. 201–213.
- Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M. & Luo, B. (2023). 'SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review', *Big Data and Cognitive Computing*, 7(97), σσ. 2-44.
- Panta, P., Rajawat, A. S., Goyal, S. B., Bedi, P., Verma, C., Raboaca, M. S. & Enescu, F. M. (2023). 'Authentication and authorization in modern web apps for data security using Node.js and role of dark web', *Procedia Computer Science*, 215, σσ. 781–790.
- Singh, G., Javed, M. & Dhaliwal, B.K. (2022). 'Full Stack Web Development: Vision, Challenges and Future Scope', *International Research Journal of Engineering and Technology (IRJET)*, 9(4), σ. 3083



## Διαδικτυακές πηγές

- Abhisri. (2024, Ιανουάριος 14). 'Why Node.js, Sequelize and Express.js render a good backend development architecture'. Medium. Διαθέσιμο σε: <https://medium.com/@Abhisri04/why-node-js-sequelize-and-express-js-render-a-good-backend-development-architecture-94812b9c330d> (Ημερομηνία πρόσβασης: 8 Φεβρουαρίου 2025).
- Create React App (2024). 'Getting Started'. Διαθέσιμο σε: <https://create-react-app.dev/docs/getting-started/> (Ημερομηνία πρόσβασης: 1 Μαρτίου 2025).
- Express.js (2024). 'Installing Express.js'. Διαθέσιμο σε: <https://expressjs.com/en/starter/installing.html> (Ημερομηνία πρόσβασης: 19 Φεβρουαρίου 2025).
- Git (n.d.). 'About Git'. Διαθέσιμο σε: <https://git-scm.com/about/> (Ημερομηνία πρόσβασης: 16 Φεβρουαρίου 2025).
- GitHub Docs (n.d.). 'GitHub Documentation'. Διαθέσιμο σε: <https://docs.github.com> (Ημερομηνία πρόσβασης: 16 Φεβρουαρίου 2025).
- Hall, J. (2022). 'Getting Started With MongoDB & Mongoose', MongoDB Developer Center. Διαθέσιμο σε: <https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/> (Ημερομηνία πρόσβασης: 8 Φεβρουαρίου 2025).
- Karlsson, J. (2022). 'MongoDB Schema Design: Data Modeling Best Practices'. Διαθέσιμο σε: <https://www.mongodb.com/developer/products/mongodb/schema-design-best-practices/> (Ημερομηνία πρόσβασης: 19 Φεβρουαρίου 2025).
- Microsoft (n.d.). 'Getting started with Visual Studio Code'. Διαθέσιμο σε: <https://code.visualstudio.com/docs/getstarted/getting-started> (Ημερομηνία πρόσβασης: 16 Φεβρουαρίου 2025).
- MongoDB (n.d.). 'JSON and BSON'. Διαθέσιμο σε: <https://www.mongodb.com/resources/basics/json-and-bson> (Ημερομηνία πρόσβασης: 7 Μαρτίου 2025).
- Node.js (n.d.). 'About Node.js'. Διαθέσιμο σε: <https://nodejs.org/en/about/> (Ημερομηνία πρόσβασης: 7 Μαρτίου 2025).
- Oracle (n.d.). 'What is a Database?'. Διαθέσιμο σε: <https://www.oracle.com/database/what-is-database/> (Ημερομηνία πρόσβασης: 14 Ιανουαρίου 2025).
- Postman (n.d.). 'What is Postman?'. Διαθέσιμο σε: <https://www.postman.com/> (Ημερομηνία πρόσβασης: 16 Φεβρουαρίου 2025).
- Sequelize (n.d.). 'Sequelize Documentation'. Διαθέσιμο σε: <https://sequelize.org> (Ημερομηνία πρόσβασης: 28 Ιανουαρίου 2025).



## Παράρτημα Α - Πίνακας Α.1 Καθορισμός EndPoints

A.	Αναγνωριστικό ΠΧ-Ονομασία ΠΧ(Βασική Ροή/Εναλλακτική Ροή/Βήμα)	Route	Μέθοδος	Data(Client προς Server)	Controller	Data/message/status Code(Server προς Client)	Εικόνα
1	ΠΧ1-Περιήγηση Στον Κατάλογο Παιχνιδιών	/games	Get	-	getGames	Διαθέσιμα Παιχνίδια Για Αγορά σελιδοδοποιημένα ανά 4,200	4.13
2	ΠΧ2-Σύνδεση Χρήστη (Βασική Ροή)	/login	Post	{email,Password}	loginUser	token, στοιχεία χρήστη,200	4.16
3	ΠΧ2-Σύνδεση Χρήστη (Εναλλακτική Ροή 1)	/login	Post	{email,Password}	loginUser	'invalid Email or Password',Status Code:401	4.9
4	ΠΧ3-Περιήγηση στον κατάλογο αγαπημένων παιχνιδιών	/favourites	Get		getAllFavoriteGames	Αγαπημένα Παιχνίδια του Χρήστη σελιδοδοποιημένα ανά 4,200	4.10
5	ΠΧ4-Περιήγηση στον κατάλογο αγορασμένων παιχνιδιών	/purchasedGames	Get		getPurchasedGames	Αγορασμένα Παιχνίδια Του Χρήστη,200	4.12
6	ΠΧ5-Αναζήτηση Παιχνιδιών	/games ή /favourites ή /purchasedGames (+παράμετροι link)	Get		getGames ή getAllFavoriteGames ή getPurchasedGames	Παιχνίδια Βάση της Αναζήτησης και των Φίλτρων,200	4.13
7	ΠΧ6-Προβολή λεπτομερειών παιχνιδιού	/games/:id	Get		getGameDetails	Στοιχεία παιχνιδιού που έχει id ίδιο με τη τιμή στο πεδίο id του route,200	4.23 και 4.24
8	ΠΧ7-Εγγραφή στο σύστημα	/register	Post	{name,email,age,password}	registerUser	token, στοιχεία χρήστη	4.26
9	ΠΧ8-Ανάκτηση	/password/	Post	{email}	forgotPass	'email sent to	4.27

	Στοιχείων(βήμα 5)	forgot			word	user's email',200	
10	ΠΧ8-Ανάκτηση Στοιχείων(βήμα 8)	password/reset/:token	Post	{password}	resetPassword	token, στοιχεία χρήστη,200	4.30
11	ΠΧ8-Ανάκτηση Στοιχείων(Εναλλακτική Ροή 1)	/password/forgot	Post	email	forgotPassword	'User not Found With this Email', 404	4.31
12	ΠΧ8-Ανάκτηση Στοιχείων(Εναλλακτική Ροή 2)	password/reset/:token	Post	{password}	resetPassword	error message, 500	4.32
13	ΠΧ9-Επεξεργασία προφίλ	me/update	Put	{name,email,password}	updateProfile	ενημερωμένα στοιχεία Χρήστη,200	4.36
14	ΠΧ10-Τροποποίηση Κωδικού Πρόσβασης	/password/update	Put	{oldPassword, password}	updatePassword	Status Code 200	4.37
15	ΠΧ10-Τροποποίηση Κωδικού Πρόσβασης(Εναλλακτική Ροή 1_	/password/update	Put	{oldPassword, password}	updatePassword	'Old Password is incorrect', 400	
16	ΠΧ11 Ονομα: Προσθήκη/Αφαίρεση αγαπημένου παιχνιδιού	favourites	Post ή delete	{gameId}	addFavoriteGame ή removeFavoriteGame	"Game added to favourites",200 ή "Game removed from favourites",200	4.38
17	ΠΧ12-Προσθήκη παιχνιδιού στο καλάθι αγορών	Εκτελείται στο FrontEnd-Client					
18	ΠΧ13-Προβολή Καλαθιού αγορών	Εκτελείται στο FrontEnd-Client					
19	ΠΧ14-Διαγραφή παιχνιδιού από το καλάθι αγορών	Εκτελείται στο FrontEnd-Client					
20	ΠΧ15-Εισαγωγή στοιχείων πελάτη (διεύθυνση, τηλέφωνο, χώρα κ.λπ.) και Στοιχείων Κάρτας-Πληρωμή Παραγγελίας	/payment/checkout_session	Post	orderItems (gameId, title, image, price, discount),	stripeCheckoutSession	session.url (URL για το Stripe Checkout)	4.44

	(βήμα 7)			personalInfo (address, country), itemsPrice, discountAmount			
21	PX15-Εισαγωγή στοιχείων πελάτη (διεύθυνση, τηλέφωνο, χώρα κ.λπ.) και Στοιχείων Κάρτας-Πληρωμή Παραγγελίας (βήμα 7)	/payment/webhook	Post	(από Stripe προς Server) session.id, payment_intent, metadata (order details)	stripeWebhook	{ success: true } ή { success: false, error: error.message }	4.45
22	PX16- Προβολή όλων των αγορών και λεπτομέρειες αγοράς του χρήστη	/me/orders	get	userId	myOrders	Όλες τις αγορές(200)	4.46
23	PX16- Προβολή όλων των αγορών και λεπτομέρειες αγοράς του χρήστη	/orders/:id	getOrderDetails	orderId	getOrderDetails	Λεπτομέρειες παραγγελίας, 200	4.47
24	PX17-Download Αγορασμένου Παιχνιδιού	/games/download/:id	get		downloadGameFile	Αρχείο Παιχνιδιού, 200	4.48
25	PX17-Εθελοντική Καταχώρηση χρόνου παιξίματος Αγορασμένου Παιχνιδιού	/purchased_games	put	timePlayed	updateTimePlayed	"Time played updated successfully", 200	4.48
26	PX18-Αξιολόγηση Αγορασμένου Παιχνιδιού	/reviews	put	{comment, rating, gameId }	createGameReview	'Review created successfully', 200	4.49
27	PX19-Προσθήκη παιχνιδιών (Βήμα 7)	/admin/games	post	{Στοιχεία Παιχνιδιού }	newGame	στοιχεία Παιχνιδιού, 200	4.51
28	PX19-Προσθήκη παιχνιδιών (Βήμα 8)	/admin/games	get		getAdminGames	λίστα με τα παιχνίδια, 200	4.52
29	PX19-Προσθήκη παιχνιδιών (Βήμα 9)	/admin/games/:id/game_file_info	get		getGameFileInfo	fileInfo, 200	4.53

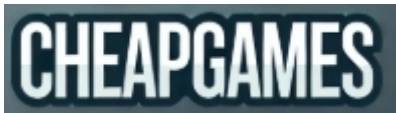
30	ΠΧ19-Προσθήκη παιχνιδιών (Βήμα 14)	/admin/games/:id/upload	post	gameFile	uploadGameFile	'File uploaded successfully and linked to the game', 200	4.54
31	ΠΧ19-Προσθήκη παιχνιδιών (Βήμα 17)	/admin/games	get		getAdminGames	λίστα με τα παιχνίδια, 200	4.52
32	ΠΧ19-Προσθήκη παιχνιδιών (Βήμα 19)	/games/:id	get		getGameDetails	gameDetails, 200	4.55
33	ΠΧ19-Προσθήκη παιχνιδιών (Βήμα 20) Διαγραφή εικόνων	/admin/games/:id/delete_image	delete	imgId	deleteGameImage	'Image deleted successfully', 200	4.55
34	ΠΧ19-Προσθήκη παιχνιδιών (Βήμα 20) Προσθήκη Εικόνων	/admin/games/:id/upload_images	post	imageFile	uploadGameImages	gameDetails, 200	4.55
35	ΠΧ20-Τροποποίηση Στοιχείων Παιχνιδιού	/admin/games/:id	put	{Πεδία και στοιχεία που θέλουμε να ενημερώσουμε}	updateGame	{Ενημερωμένο παιχνίδι, 200}	4.56
36	ΠΧ21-Διαγραφή Παιχνιδιού από τη Βάση	/games/games/:id	delete		deleteGame	"Game and associated file deleted successfully", 200	4.57
37	ΠΧ22-Διαχείριση στοιχείων άλλων χρηστών (Βήμα 5)	/admin/users	get		allUsers	Λίστα με όλους τους χρήστες	4.58
38	ΠΧ22-Διαχείριση στοιχείων άλλων χρηστών (Βήμα 8)	/admin/users/:id	get		getUserDetails	Στοιχεία Χρήστη	4.59
39	ΠΧ22-Διαχείριση στοιχείων άλλων χρηστών (Βήμα 9)	/admin/users/:id	put	{name, email, role, dateOfBirth}	updateUser	Ενημερωμένα Στοιχεία Χρήστη, 200	4.59

40	ΠΧ23-Διαγραφή Χρήστη	/admin/users/:id	delete		deleteUser	success,true	4.58
41	ΠΧ24 Όνομα: Διαχείριση παραγγελιών (έλεγχος, διαγραφή) (Βήμα 5)	/admin/orders	get		allOrders	{orders}	4.60
42	ΠΧ24 Όνομα: Διαχείριση παραγγελιών (έλεγχος, διαγραφή) (Βήμα 7)	/orders/:id	get		viewOrder	{order}	4.61
43	ΠΧ24 Όνομα: Διαχείριση παραγγελιών (έλεγχος, διαγραφή) (Βήμα 7)	admin/orders/:id	delete		deleteOrder	'Order deleted and user's purchased games updated successfully', 200	4.60
44	ΠΧ25 - Διαχείριση Εισόδων	/admin/get_sales	get		getSales	{totalSales, totalNumOrders, sales}	4.62
45	ΠΧ26-Καθορισμός εποχιακών εκπτώσεων σε συγκεκριμένες κατηγορίες (Βήμα 6)	/games/seasonal/discount	get		getSeasonalDiscountDetails	seasonalDiscounts	4.63
46	ΠΧ26-Καθορισμός εποχιακών εκπτώσεων σε συγκεκριμένες κατηγορίες (Βήμα 6)	/games/seasonal/discount	get		getSeasonalDiscountDetails	seasonalDiscountDetails	4.63
47	ΠΧ26-Δημιουργία/Ενημέρωση Εποχιακής Εκπτώσης	/admin/games/seasonal/discount	get		getSeasonalDiscountDetails	seasonalDiscountDetails, 200	4.63
48	ΠΧ26-Δημιουργία/Ενημέρωση Εποχιακής Εκπτώσης	/admin/games/seasonal/discount	put	{seasonalDiscountMessage, categories, discount, startDate, endDate, isActive}	applySeasonalDiscount	"Seasonal Discount updated and applied successfully", 200	4.63
49	ΠΧ27 Όνομα:Εφαρμογή εποχιακών εκπτώσεων σε συγκεκριμένες κατηγορίες παιχνιδιών	/admin/games/seasonal/discount/updateGamesWithSeasonalDiscount	patch		updateGamesWithSeasonalDiscount	"You updated all games of categories with seasonal discount.", 200	4.63

		nt					
50	ΠΧ28-Επαναφορά εκπτώσεων παιχνιδιών με εφαρμοσμένη την εποχιακή έκπτωση στις αρχικές-προηγούμενες τους τιμές	/admin/games/seasonal/discount/resetdiscountstoPrevious	patch		resetdiscountstoPrevious	"All games discounts have been reset and seasonal discount is now disabled.",200	4.63
51	ΠΧ29 - Απενεργοποίηση Εμφάνισης Διαφημιστικού Slide	/admin/games/seasonal/discount/disable	patch		disableSeasonalDiscount	"Seasonal discount has been disabled successfully",200	4.63

Πίνακας Α.1 Καθορισμός EndPoints

## Παράρτημα Β - Πίνακας Β.1 Φάκελοι και Αρχεία Frontend με Περιγραφές

Α Σ	Φάκελος/Αρχείο	Περιγραφή
1	Public (folder)	Περιέχει αρχεία που είναι διαθέσιμα σε όλους τους χρήστες και δεν επεξεργάζονται από τη React
2	favicon.ico (file)	Το εικονίδιο που θέλουμε να εμφανίζεται στην καρτέλα του browser
3	index.html (file)	Το κύριο HTML αρχείο στο οποίο φορτώνονται όλες οι σελίδες που φτιάχνουμε μέσω των component. Σε αυτό το αρχείο προστέθηκε η Bootstrap μέσω link και των οδηγιών που υπάρχουν στο επίσημο site <a href="https://getbootstrap.com/docs/4.4/getting-started/introduction/">https://getbootstrap.com/docs/4.4/getting-started/introduction/</a> ώστε η διαδικτυακή εφαρμογή CheapGames να είναι responsive
4	logo192.png και logo512.png (files)	Αυτές οι εικόνες συμπεριλαμβάνονται στην διαμόρφωση κατά την αρχική εγκατάσταση της React και αντιστοιχούν στις εικόνες που εμφανίζονται όταν η διαδικτυακή εφαρμογή αποθηκεύεται σε μία συσκευή. Διατηρήθηκαν οι δύο εικόνες που δείχνουν το Logo της React σε διαφορετικά μεγέθη
5	manifest.json (file):	Περιέχει στοιχεία και ρυθμίσεις που χρησιμοποιούν οι browsers για την εμφάνιση της εφαρμογής. Διατηρήθηκαν οι Default ρυθμίσεις και στοιχεία.
6	robots.txt (file):	Σε αυτό το αρχείο καταγράφονται οι μηχανές αναζήτησης και τα sites που επιτρέπεται ή δεν επιτρέπεται να είναι ανιχνεύσιμο το CheapGames
7	Images/ (folder)	Περιέχει εικόνες που χρησιμοποιούνται στην εφαρμογή, όπως η default_avatar.jpg και logoCheapGames.png (Εικόνα 6.4)
		 <p>Εικόνα 6.4 CheapGames logo</p>
8	src (folder)	Περιέχει τα αρχεία οργανωμένα σε φακέλους που έχουν να κάνουν με το περιεχόμενο και εμφάνιση κάθε σελίδας του Site στον Client.
9	App.css (file)	Περιέχει τους κανόνες Css που εφαρμόζονται σε κάθε σελίδα του site
10	App.js(file)	Περιέχει δομικά στοιχεία (Εικόνα 6.4) όπως πχ. κεφαλίδες, υποσέλιδα, στοιχεία πλοήγησης των οποίων το περιεχόμενο τους καθορίζεται σε άλλα αρχεία. Επίσης σε αυτό το αρχείο μπορούμε να ορίσουμε τις διευθύνσεις της ιστοσελίδας, το τι θα εμφανίζεται σε κάθε σελίδα. Αυτός ο ορισμός μπορεί να γίνει απευθείας μέσα στο αρχείο app.js ή



		<p>μέσω εισαγωγής άλλων αρχείων κάτι που συνηθίζεται μειώνοντας την πολυπλοκότητα του κώδικα. Στην υλοποίηση αυτή προτιμήθηκε ο δεύτερος τρόπος καθώς τα routes της ιστοσελίδας χωρίστηκαν ανάλογα αν είναι διαθέσιμα για όλους τους χρήστες ή τους Admin και ορίστηκαν σε δύο αρχεία “userRoutes” και “adminRoutes”.</p>  <pre> frontend &gt; src &gt; JS App.js &gt; ... 1   // Εισαγωγή των βασικών CSS styles 2   import './App.css'; 3   // React Router για διαχείριση διαδρομών (routes) 4   import {BrowserRouter as Router, Routes,} from 'react-router-dom'; 5   // Εισαγωγή του Header και του Footer από άλλα αρχεία 6   import Header from './components/layout/Header'; 7   import Footer from './components/layout/Footer'; 8   //Βιβλιοθήκη για εμφάνιση ειδοποιήσεων 9   import {Toaster} from 'react-hot-toast' 10   //Στα αρχεία που γίνονται import ορίζονται οι διαδρομές 11   import useUserRoutes from './components/routes/userRoutes'; 12   import useAdminRoutes from './components/routes/adminRoutes'; 13   function App() { 14     const userRoutes = useUserRoutes(); 15     const adminRoutes = useAdminRoutes(); 16     return ( 17       &lt;Router&gt; 18         &lt;div className="App"&gt; 19           &lt;Toaster position="top-center" /&gt; 20           &lt;Header /&gt; 21           &lt;div className="container"&gt; 22             &lt;Routes&gt; 23               {userRoutes} 24               {adminRoutes} 25             &lt;/Routes&gt; 26           &lt;/div&gt; 27           &lt;Footer /&gt; 28         &lt;/div&gt; 29       &lt;/Router&gt; 30     ); 31   } 32   33   34   export default App; </pre> <p>Εικόνα 6.5 Απόσπασμα Από κώδικα αρχείου App.js</p>
11	index.js (file)	Το αρχείο αυτό συνδέει το αρχείο index.html με το App.js.
12	components (folder)	περιέχει όλα τα React components, οργανωμένα σε υποφακέλους ανάλογα με τη λειτουργία τους.
13	admin/ (folder):	Περιέχει τα components που σχετίζονται με το dashboard του διαχειριστή
14	Dashboard.jsx (file)	Κεντρική σελίδα του admin panel
15	GameReviews.jsx (file)	Διαχείριση κριτικών παιχνιδιών.
16	ListGames.jsx (file)	Λίστα με όλα τα διαθέσιμα παιχνίδια.
17	ListOrders.jsx (file)	Διαχείριση παραγγελιών.

18	ListUsers.jsx (file)	Διαχείριση χρηστών.
19	NewGame.jsx (file)	Φόρμα δημιουργίας νέου παιχνιδιού.
20	SeasonalDiscount.jsx (file)	Εφαρμογή εποχιακών εκπτώσεων
21	UpdateGame.jsx (file)	Ενημέρωση πληροφοριών παιχνιδιού.
22	UpdateUser.jsx (file)	Ενημέρωση στοιχείων χρήστη
23	UploadGameFile.jsx (file)	Διαχείριση αρχείων παιχνιδιών
24	UploadImages.jsx (file)	Διαχείριση εικόνων παιχνιδιών.
25	ViewOrder.jsx (file)	Προβολή λεπτομερειών μιας παραγγελίας.
26	components/auth/ (folder)	Περιέχει τα components που σχετίζονται με τη διαχείριση λογαριασμού
27	ForgotPassword.jsx (file)	Φόρμα για ανάκτηση κωδικού
28	Login.jsx (file)	Σελίδα σύνδεσης χρήστη
29	ProtectedRoute.jsx (file)	Ελέγχει αν ένας χρήστης έχει πρόσβαση σε μια σελίδα.
30	Register.jsx (file)	Σελίδα εγγραφής νέου χρήστη
31	ResetPassword.jsx (file)	Φόρμα επαναφοράς κωδικού.
32	components/cart/ (folder)	Περιέχει τα components που σχετίζονται με το καλάθι και τις αγορές
33	Cart.jsx (file)	Σελίδα προβολής των προϊόντων στο καλάθι.
34	CheckoutSteps.jsx (file)	Διαδικασία πληρωμής βήμα-βήμα.
35	ConfirmOrder.jsx (file)	Σελίδα επιβεβαίωσης παραγγελίας.
36	PaymentMethod.jsx (file)	Επιλογή τρόπου πληρωμής
37	PersonalInfo.jsx (file)	Εισαγωγή προσωπικών στοιχείων για την αγορά
38	components/game/ (folder)	Περιέχει τα components που σχετίζονται με την προβολή και διαχείριση παιχνιδιών
39	GameDetails.jsx (file)	Προβολή λεπτομερειών παιχνιδιού
40	GameItem.jsx (file)	Component εμφάνισης ενός παιχνιδιού σε λίστα
41	PlayGame.jsx (file)	Επιτρέπει στους χρήστες να παίξουν αγορασμένα παιχνίδια
42	PurchasedItem.jsx (file)	Λίστα αγορασμένων παιχνιδιών
43	components/layout/ (folder)	Περιλαμβάνει components που διαμορφώνουν την εμφάνιση της εφαρμογής
44	AdminLayout.jsx (file)	Layout για τις admin σελίδες.
45	CustomPagination.jsx (file)	Μηχανισμός pagination

46	Filters.jsx (file)	Φίλτρα για αναζητήσεις παιχνιδιών
47	Footer.jsx (file)	Υποσέλιδο
48	Header.jsx (file)	Κεφαλίδα (logo, μπάρα αναζήτησης, κουμπό Login ή μενού επιλογών
49	Loader.jsx (file)	Component εμφάνισης loading
50	MetaData.jsx (file)	Τίτλος που εμφανίζεται στη καρτέλα
51	Search.jsx (file)	Μπάρα αναζήτησης στη Κεφαλίδα
52	SideMenu.jsx (file)	Πλευρικό μενού για το Admin DashBoard
53	SlideDiscount.jsx (file)	Προβολή Εποχιακής Έκπτωσης
54	UserLayout.jsx (file)	Layout για κονσόλα ρυθμίσεων του χρήστη
55	redux/ (folder)	Ο φάκελος redux περιλαμβάνει όλες ρυθμίσεις για την αποθήκευση των δεδομένων στο client και οργανώνει όλα τα αιτήματα σε αρχεία
56	store.js (file)	Το κεντρικό store που ενώνει όλα τα reducers.
57	api/ (folder)	Περιέχει τα αρχεία με τα αιτήματα προς τον Server
58	authApi.js (file)	Αιτήματα σχετικά με login , logout
59	discountApi.js (file)	Αιτήματα σχετικά με διαχείριση εποχιακής έκπτωσης
60	gamesApi.js (file)	Αιτήματα σχετικά με τα παιχνίδια (ανακτηση, δημιουργία, κλ)
61	orderApi.js (file)	Αιτήματα σχετικά με αγορές
63	userApi.js (file)	Αιτήματα σχετικά με τους χρήστης(Ανάκτησης όλων των χρηστών, ανάκτηση προφιλ κλ)
63	features(folder)	Περιέχει τα Redux slices
64	cartSlice.js (file)	Ορίζει μια κατάσταση για τα δεδομένα του καλαθιού ώστε να είναι διαθέσιμα κατά την πλοήγηση του χρήστη στις σελίδες
65	userSlice.js (file)	Ορίζει μια κατάσταση για τα δεδομένα του χρήστη όπως αν είναι συνδεδεμένος στη σελίδα , το όνομα του για να εμφανίζεται κατά τη πλοήγηση του στη σελίδα
66	constants/(folder)	(Σταθερές τιμές)
67	constants.js(file)	Περιέχει σταθερές τιμές όπως οι διαθέσιμες κατηγορίες παιχνιδιών.
68	helpers/ (folder)	(Βοηθητικές Συναρτήσεις)
69	helpers.jsx(file)	Περιέχει βοηθητικές συναρτήσεις για την εφαρμογή των φίλτρων στα αιτήματα και εύρεση ποσού φόρου , συνολικού ποσού πληρωμής
70	package.json (file)	Περιέχει όλες τις βιβλιοθήκες και ρυθμίσεις για την εκτέλεση της React εφαρμογής. Οποια βιβλιοθήκη εγκαταστάθηκε, προστίθεται στο

	package.json του frontend.
--	----------------------------

Πίνακας Β.1 Φάκελοι και Αρχεία Frontend με Περιγραφές

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