

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ****Σχολή Τεχνολογιών Πληροφορικής και
Επικοινωνιών - Τμήμα Πληροφορικής****Πτυχιακή Εργασία**

Τίτλος Πτυχιακής Εργασίας	(Ελληνικά) Android εφαρμογή παιχνιδιού ερωτήσεων και καταγραφής ψηφιακών επιτευγμάτων (Αγγλικά) Android app for quiz games and digital achievements tracking
Ονοματεπώνυμο Φοιτητή	ΜΠΟΜΠΟΤΗΣ ΠΑΝΑΓΙΩΤΗΣ
Πατρώνυμο	ΓΕΩΡΓΙΟΣ
Αριθμός Μητρώου	Π22108
Επιβλέπων	ΕΥΘΥΜΙΟΣ ΑΛΕΠΗΣ, Καθηγητής

Ημερομηνία Παράδοσης **Σεπτέμβριος 2026**

Copyright ©

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν αποκλειστικά τον συγγραφέα και δεν αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Πειραιώς.

Ως συγγραφέας της παρούσας εργασίας δηλώνω πως η παρούσα εργασία δεν αποτελεί προϊόν λογοκλοπής και δεν περιέχει υλικό από μη αναφερόμενες πηγές.

Περιεχόμενα

Copyright ©	2
Περιεχόμενα	3
Περίληψη	5
Εισαγωγή	6
Θεωρητικό Υπόβαθρο: Gamification και Mobile Learning	7
Mobile Learning	7
Η Έννοια του Gamification	7
Ανάλυση Απαιτήσεων Συστήματος	9
Λειτουργικές Απαιτήσεις (Functional Requirements)	9
Μη Λειτουργικές Απαιτήσεις (Non-Functional Requirements)	10
Συγκριτική Ανάλυση και Επιλογή Τεχνολογιών	12
Back-end: Spring Boot (Java) vs Node.js	12
Βάση Δεδομένων: PostgreSQL (RDBMS) vs MongoDB (NoSQL)	12
Front-end: Native Android (Java) vs Cross-Platform (Flutter/React Native)	12
Επισκόπηση Αρχιτεκτονικής	13
Αρχικό στήσιμο εφαρμογής	15
Στήσιμο του back-end	15
Στήσιμο της βάσης δεδομένων	15
Spring Boot (Application Context & Beans)	16
Διαχείριση Δεδομένων με Spring Data JPA	17
Υλοποίηση Συστήματος Ασφάλειας και Αυθεντικοποίησης	18
Η Οντότητα AppUser και η Διεπαφή UserDetails	18
Η Υπηρεσία UserDetailsService και η Ανάκτηση Δεδομένων	18
Διαχείριση Ρόλων και Δικαιωμάτων Πρόσβασης (RBAC)	18
JSON Web Token (JWT): Δομή και Υλοποίηση	19
Παραμετροποίηση Αλυσίδας Ασφαλείας (SecurityFilterChainConfig)	19
Το Φίλτρο JWTAuthenticationFilter	20
Διαδικασία Αυθεντικοποίησης	20
Μηχανισμός Refresh Tokens	20
Διαχείριση Σφαλμάτων και Εξαιρέσεων Ασφαλείας	20
Διάγραμμα της βάσης δεδομένων	21
Κλάσεις για την υλοποίηση της αρχιτεκτονικής	23
Ροές Δεδομένων και Διαγράμματα Αλληλουχίας	24
1. Διαδικασία Ταυτοποίησης Χρήστη (Login Flow)	24
2. Διαδικασία Ανάκτησης Ερωτήσεων Κουίζ	25

3. Διαδικασία Υποβολής Σκορ και Ενημέρωσης Πόντων Εμπειρίας (XP)	26
4. Διαδικασία Ελέγχου και Απόδοσης Ψηφιακών Επιτευγμάτων (Achievements)	27
Annotations	28
Spring Web MVC Annotations	28
Spring Annotations	29
Στήσιμο front-end.....	30
Android UI Components	31
Bottom Navigation Bar και Fragments.....	31
RecyclerView.....	32
Tab layout και ViewPager	33
Εγχειρίδιο χρήσης (για το χρήστη)	34
Εγγραφή και Σύνδεση (Authentication & Onboarding)	35
Ενότητα 1: Επιλογή Κατηγορίας και Διεξαγωγή Κουίζ	37
Ενότητα 2: Παγκόσμιος Πίνακας Κατάταξης (Leaderboard).....	41
Ενότητα 3: Στατιστικά και Ψηφιακά Επιτεύγματα (Achievements)	42
Ενότητα 4: Διαχείριση του προφίλ.....	44
Εγχειρίδιο χρήσης (για το διαχειριστή)	45
Διαχείριση Ερωτήσεων (Questions Management)	45
Εποπτεία και Διαχείριση Χρηστών (User Management)	45
Μελλοντικές επεκτάσεις της εφαρμογής	46
Συμπεράσματα.....	48
Πίνακας Συντμήσεων – Αρκτικόλεξων – Ακρωνυμίων	50
Πίνακας Ορολογίας.....	52
Βιβλιογραφία	53

Περίληψη

English

This thesis focuses on the implementation of an interactive Android application, specifically an educational and entertaining trivia game (Quiz), utilizing the Spring Boot framework in conjunction with Spring Security and Spring Data JPA for the back-end, a PostgreSQL database, and Android UI components provided by the Android SDK for the front-end.

The application allows registered users to select thematic categories and difficulty levels, answer questions under time pressure, and earn experience points (XP). Concurrently, it integrates gamification systems, such as digital achievements and a real-time global leaderboard.

The purpose of the application is to provide a comprehensive, secure, and user-friendly platform that enhances knowledge acquisition through healthy competition, continuous progress tracking, and immediate feedback on player performance.

Ελληνικά

Η παρούσα πτυχιακή επικεντρώνεται στην υλοποίηση μιας διαδραστικής Android εφαρμογής, η οποία αφορά ένα εκπαιδευτικό και ψυχαγωγικό παιχνίδι ερωτήσεων (Quiz), με τη χρήση του framework Spring Boot σε συνεργασία με Spring Security και Spring Data JPA για το back-end, μιας PostgreSQL βάσης δεδομένων και των Android UI components που προσφέρονται από το Android SDK για το front-end.

Η εφαρμογή παρέχει στους εγγεγραμμένους χρήστες τη δυνατότητα να επιλέγουν θεματικές κατηγορίες και επίπεδα δυσκολίας, να απαντούν σε ερωτήσεις υπό την πίεση χρόνου και να κερδίζουν πόντους εμπειρίας (XP). Παράλληλα, ενσωματώνει συστήματα gamification, όπως ψηφιακά επιτεύγματα (achievements) και έναν παγκόσμιο πίνακα κατάταξης (leaderboard) σε πραγματικό χρόνο.

Ο σκοπός της εφαρμογής είναι να προσφέρει μια ολοκληρωμένη, ασφαλή και εύχρηστη πλατφόρμα που ενισχύει την απόκτηση γνώσεων μέσω της υγιούς ανταγωνιστικότητας, της συνεχούς παρακολούθησης της προόδου και της άμεσης ανατροφοδότησης των επιδόσεων των παικτών.

Εισαγωγή

Στη σύγχρονη ψηφιακή εποχή, η απόκτηση γνώσεων και η αυτοαξιολόγηση δεν περιορίζονται πλέον στα στενά πλαίσια της παραδοσιακής εκπαίδευσης. Η μέθοδος της παιχνιδοποίησης (gamification) έχει αποδειχθεί ως ένα από τα πιο ισχυρά εργαλεία για την ενίσχυση της μάθησης και της αφοσίωσης των χρηστών. Υπάρχει, συνεπώς, η ανάγκη για διαδραστικές εφαρμογές που συνδυάζουν την εκπαίδευση με την ψυχαγωγία, προσφέροντας στους χρήστες την ευκαιρία να δοκιμάσουν τις γνώσεις τους σε ένα ανταγωνιστικό αλλά φιλικό περιβάλλον.

Η παρούσα πτυχιακή εργασία στοχεύει να καλύψει αυτές τις ανάγκες μέσω της ανάπτυξης του **Quizly**, μιας ολοκληρωμένης εφαρμογής ερωτήσεων για κινητές συσκευές. Η εφαρμογή επιτρέπει στους χρήστες να επιλέγουν θεματικές κατηγορίες (όπως Ιστορία, Γεωγραφία, Επιστήμη) και επίπεδα δυσκολίας, απαντώντας σε ερωτήσεις πολλαπλής επιλογής. Η ενσωμάτωση ενός χρονόμετρου ανά ερώτηση ενισχύει την πρόκληση, απαιτώντας όχι μόνο γνώση αλλά και ταχύτητα αντίδρασης.

Επιπλέον, η εφαρμογή ενσωματώνει σύγχρονους μηχανισμούς ανταμοιβής. Οι χρήστες, απαντώντας σωστά, συγκεντρώνουν πόντους εμπειρίας (XP) και ξεκλειδώνουν ειδικά ψηφιακά επιτεύγματα (Achievements) ανάλογα με τις επιδόσεις τους. Παράλληλα, ενισχύεται το αίσθημα της κοινότητας και της υγιούς ανταγωνιστικότητας μέσω ενός διαδικτυακού, παγκόσμιου πίνακα κατάταξης (Leaderboard), όπου οι παίκτες μπορούν να συγκρίνουν το συνολικό τους σκορ με άλλους χρήστες σε πραγματικό χρόνο.

Η εφαρμογή έχει κατασκευαστεί χρησιμοποιώντας έναν ισχυρό συνδυασμό τεχνολογιών. Στην πλευρά του back-end, χρησιμοποιήθηκε Java σε συνδυασμό με τα εργαλεία Spring Boot και Spring Data JPA για τη διαχείριση της επιχειρησιακής λογικής (business logic) και των δεδομένων. Η ασφάλεια της εφαρμογής ενισχύεται μέσω του Spring Security και των JSON Web Tokens (JWT), εξασφαλίζοντας την ακεραιότητα των δεδομένων και την ασφαλή ταυτοποίηση των παικτών. Για την αποθήκευση των δεδομένων χρησιμοποιείται το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων PostgreSQL. Η πλευρά του front-end υλοποιήθηκε στο Android Studio με χρήση Java, αξιοποιώντας τα σύγχρονα UI Components του Android SDK και τη βιβλιοθήκη Retrofit για την άμεση και αξιόπιστη επικοινωνία με το RESTful API του εξυπηρετητή.

Ιδιαίτερη έμφαση δόθηκε στην εμπειρία χρήστη και στον σύγχρονο σχεδιασμό (UI), διασφαλίζοντας ότι άτομα κάθε ηλικίας μπορούν εύκολα να περιηγηθούν στο περιβάλλον του παιχνιδιού. Τα ζωντανά χρώματα, οι καθαρές γραμμές και η άμεση οπτική ανατροφοδότηση καθοδηγούν τον χρήστη ομαλά από την εγγραφή του μέχρι την κατάκτηση της κορυφής στην κατάταξη.

Συμπερασματικά, το Quizly γεφυρώνει το χάσμα μεταξύ της μάθησης και της διασκέδασης. Μέσω των προηγμένων χαρακτηριστικών του, της στιβαρής αρχιτεκτονικής του και του σχεδιασμού του με επίκεντρο τον παίκτη, ξεχωρίζει ως ένα ολοκληρωμένο ψηφιακό εργαλείο που αποδεικνύει πώς η τεχνολογία μπορεί να κάνει τη γνώση ένα συναρπαστικό παιχνίδι.

Θεωρητικό Υπόβαθρο: Gamification και Mobile Learning

Η ανάπτυξη εκπαιδευτικών εφαρμογών και παιχνιδιών γνώσεων για κινητές συσκευές δεν βασίζεται μόνο σε τεχνολογικά πρότυπα, αλλά εδράζεται σε σύγχρονες θεωρίες μάθησης και ψυχολογίας. Στην παρούσα ενότητα αναλύονται οι έννοιες του Mobile Learning και του Gamification, οι οποίες αποτελούν τους βασικούς θεωρητικούς πυλώνες πάνω στους οποίους σχεδιάστηκε και αναπτύχθηκε η εφαρμογή **Quizly**.

Mobile Learning

Το m-learning ορίζεται ως η εκπαιδευτική διαδικασία που λαμβάνει χώρα μέσα από φορητές συσκευές, όπως τα κινητά τηλέφωνα και τα tablets. Σε αντίθεση με την παραδοσιακή ηλεκτρονική μάθηση (e-learning), η οποία απαιτεί τη χρήση σταθερών υπολογιστών και συχνά περιορίζει τον χρήστη στον χώρο, η κινητή μάθηση προσφέρει το πλεονέκτημα της «πανταχού παρούσας» πρόσβασης στην πληροφορία.

Τα κύρια χαρακτηριστικά της κινητής μάθησης περιλαμβάνουν:

- **Micro-learning:** Η πληροφορία καταναλώνεται σε μικρές, εύπεπτες «μπουκιές». Τα σύντομα κουίζ γνώσεων αξιοποιούν απόλυτα αυτή τη φιλοσοφία, επιτρέποντας στους χρήστες να μαθαίνουν νέα στοιχεία κατά τη διάρκεια μικρών διαλειμμάτων στην καθημερινότητά τους (π.χ. στις μετακινήσεις).
- **Αυτονομία και Εξατομίκευση:** Ο χρήστης έχει τον απόλυτο έλεγχο του ρυθμού μάθησης, επιλέγοντας ο ίδιος τις θεματικές κατηγορίες και τα επίπεδα δυσκολίας που ταιριάζουν στις ανάγκες και τα ενδιαφέροντά του.

Η Έννοια του Gamification

Η μέθοδος Gamification έχει αποδειχθεί ως ένα από τα πιο ισχυρά εργαλεία για την ενίσχυση της μάθησης και της αφοσίωσης των χρηστών. Ως όρος, η παιχνιδοποίηση αναφέρεται στην ενσωμάτωση στοιχείων και μηχανισμών που συναντώνται παραδοσιακά σε παιχνίδια, σε περιβάλλοντα που δεν αποτελούν παιχνίδια. Ο κύριος στόχος της είναι η αύξηση του εσωτερικού και εξωτερικού κινήτρου των χρηστών, προκειμένου να επιτύχουν συγκεκριμένους στόχους.

Στο πλαίσιο μιας εφαρμογής γνώσεων όπως το Quizly, η παιχνιδοποίηση μετατρέπει την απλή διαδικασία της απάντησης ερωτήσεων σε μια συναρπαστική και διαδραστική εμπειρία. Το σύστημα ενσωματώνει στρατηγικά τα ακόλουθα στοιχεία:

1. Πόντοι Εμπειρίας (Experience Points - XP)

Το σύστημα βαθμολόγησης αποτελεί τον πιο βασικό μηχανισμό ανατροφοδότησης. Όταν οι χρήστες απαντούν σωστά, κερδίζουν πόντους εμπειρίας (XP). Αυτή η άμεση επιβράβευση ενεργοποιεί το σύστημα ντοπαμίνης στον εγκέφαλο του χρήστη, δημιουργώντας μια αίσθηση ικανοποίησης και ενθαρρύνοντάς τον να συνεχίσει να παίζει (και συνεπώς, να μαθαίνει). Οι πόντοι λειτουργούν ως μια μετρήσιμη ένδειξη της συνολικής προόδου και αφοσίωσης.

2. Ψηφιακά Επιτεύγματα (Achievements / Badges)

Η εφαρμογή ενσωματώνει συστήματα gamification, όπως τα ψηφιακά επιτεύγματα. Τα επιτεύγματα λειτουργούν ως εικονικά «μετάλλια» τα οποία ξεκλειδώνονται όταν ο χρήστης εκπληρώσει συγκεκριμένα ορόσημα (π.χ. απάντηση σε 10 συνεχόμενες ερωτήσεις χωρίς λάθος ή συγκέντρωση ενός συγκεκριμένου αριθμού πόντων σε μια θεματική κατηγορία). Η παρουσία κλειδωμένων επιτευγμάτων στο προφίλ του χρήστη δημιουργεί ένα ισχυρό κίνητρο ολοκλήρωσης, ωθώντας τον παίκτη να εξερευνήσει όλες τις πτυχές της εφαρμογής για να τα κατακτήσει.

3. Πίνακες Κατάταξης (Leaderboards)

Η σύγκριση των επιδόσεων ενισχύει το στοιχείο της κοινωνικής αλληλεπίδρασης. Η εφαρμογή διαθέτει έναν παγκόσμιο πίνακα κατάταξης σε πραγματικό χρόνο. Ο πίνακας αυτός καλλιεργεί την υγιή ανταγωνιστικότητα μεταξύ των παικτών. Ο χρήστης βλέπει την τρέχουσα θέση του σε σχέση με άλλους παίκτες, γεγονός που του παρέχει ένα ξεκάθαρο, εξωτερικό κίνητρο να βελτιώσει τις γνώσεις του και την απόδοσή του προκειμένου να αναρριχηθεί στην κορυφή της βαθμολογίας.

4. Χρονικοί Περιορισμοί

Η ενσωμάτωση ενός χρονομέτρου ανά ερώτηση ενισχύει δραματικά την πρόκληση του παιχνιδιού. Η προσθήκη του παράγοντα του χρόνου, απαιτεί από τον χρήστη όχι μόνο να γνωρίζει τη σωστή απάντηση, αλλά να την ανακαλεί ταχύτατα από τη μνήμη του. Αυτή η μηχανική αποτρέπει την αναζήτηση απαντήσεων σε εξωτερικές πηγές κατά τη διάρκεια του κουίζ και εισάγει τον παίκτη σε μια κατάσταση απόλυτης συγκέντρωσης, μεγιστοποιώντας την εκπαιδευτική αξία της διαδικασίας.

Ανάλυση Απαιτήσεων Συστήματος

Η φάση της ανάλυσης απαιτήσεων αποτελεί το κρίσιμότερο στάδιο στον κύκλο ζωής ανάπτυξης λογισμικού. Κατά το στάδιο αυτό, καθορίζονται με σαφήνεια οι λειτουργίες που πρέπει να παρέχει το Quizly, καθώς και οι περιορισμοί υπό τους οποίους οφείλει να λειτουργεί. Η σωστή καταγραφή των απαιτήσεων διασφαλίζει ότι η τελική εφαρμογή θα ανταποκρίνεται πλήρως στις προσδοκίες των χρηστών, διευκολύνοντας παράλληλα τον αρχιτεκτονικό σχεδιασμό του Front-end και του Back-end. Οι απαιτήσεις του συστήματος διακρίνονται σε δύο κύριες κατηγορίες: τις Λειτουργικές και τις Μη Λειτουργικές.

Λειτουργικές Απαιτήσεις (Functional Requirements)

Οι λειτουργικές απαιτήσεις περιγράφουν λεπτομερώς τις ενέργειες και τις συμπεριφορές που το σύστημα πρέπει να είναι σε θέση να εκτελέσει. Ορίζουν το «τι» πρέπει να κάνει το λογισμικό όταν δέχεται συγκεκριμένες εισόδους από τους διαφορετικούς ρόλους (Παίκτης, Διαχειριστής, Σύστημα).

1. Διαχείριση Χρηστών και Ταυτοποίηση (User Management & Authentication)

- Το σύστημα πρέπει να επιτρέπει την εγγραφή νέων χρηστών, απαιτώντας την εισαγωγή βασικών στοιχείων (Όνομα, Επώνυμο, Email, Κωδικός Πρόσβασης).
- Το σύστημα πρέπει να επικυρώνει τη μοναδικότητα του email κατά την εγγραφή, ώστε να μην υπάρχουν διπλότυποι λογαριασμοί στη βάση δεδομένων.
- Το σύστημα πρέπει να επιτρέπει την ασφαλή σύνδεση των χρηστών και την παραγωγή ενός JSON Web Token (JWT) για τη διατήρηση του session.
- Οι χρήστες πρέπει να έχουν τη δυνατότητα επεξεργασίας των προσωπικών τους στοιχείων και ασφαλούς αποσύνδεσης.

2. Λειτουργίες Παιχνιδιού (Gameplay Mechanics)

- Το σύστημα πρέπει να επιτρέπει στον παίκτη να επιλέξει τη θεματική κατηγορία (π.χ. Ιστορία, Γεωγραφία, Επιστήμη) και το επίπεδο δυσκολίας του κουίζ.
- Το σύστημα πρέπει να ανακτά και να προβάλλει τυχαίες ερωτήσεις από τη βάση δεδομένων με βάση τα κριτήρια που επέλεξε ο χρήστης.
- Για κάθε ερώτηση, το σύστημα πρέπει να επιβάλλει χρονικό περιορισμό (αντίστροφη μέτρηση) εντός του οποίου ο παίκτης οφείλει να απαντήσει.
- Το σύστημα πρέπει να αξιολογεί άμεσα την απάντηση (σωστή/λάθος), να παρέχει οπτική ανατροφοδότηση και να υπολογίζει το τρέχον σκορ.
- Με την ολοκλήρωση του κουίζ, το σύστημα πρέπει να αποστέλλει το τελικό σκορ στον server και να ενημερώνει τους συνολικούς Πόντους Εμπειρίας (XP) του παίκτη.

3. Κοινωνική Αλληλεπίδραση και Gamification

- Το σύστημα πρέπει να ελέγχει τα στατιστικά του χρήστη μετά από κάθε παιχνίδι και να ξεκλειδώνει αυτόματα ψηφιακά επιτεύγματα (Achievements) εφόσον πληρούνται τα κριτήρια (π.χ. «Απάντησε 10 ερωτήσεις σωστά»).

- Το σύστημα πρέπει να δημιουργεί και να προβάλλει έναν δυναμικό Παγκόσμιο Πίνακα Κατάταξης (Leaderboard), ταξινομώντας τους χρήστες σε φθίνουσα σειρά με βάση τα συνολικά τους XP.

4. Διαχειριστικές Λειτουργίες (Admin Requirements)

- Ο διαχειριστής πρέπει να έχει τη δυνατότητα προσθήκης, επεξεργασίας και διαγραφής ερωτήσεων μέσω ειδικού περιβάλλοντος (CRUD λειτουργίες).
- Ο διαχειριστής πρέπει να μπορεί να προβάλει τη λίστα όλων των εγγεγραμμένων χρηστών και να προβαίνει σε διαγραφή λογαριασμών (Banning) σε περίπτωση παραβίασης των κανόνων.

Μη Λειτουργικές Απαιτήσεις (Non-Functional Requirements)

Οι μη λειτουργικές απαιτήσεις περιγράφουν τα ποιοτικά χαρακτηριστικά του συστήματος. Δεν εστιάζουν στο «τι» κάνει το σύστημα, αλλά στο «πώς» το κάνει, καλύπτοντας τομείς όπως η ασφάλεια, η απόδοση και η επεκτασιμότητα.

1. Ασφάλεια (Security)

- Η διαχείριση της αυθεντικοποίησης πρέπει να γίνεται αποκλειστικά μέσω ασφαλών JSON Web Tokens (JWT) με μικρή διάρκεια ισχύος, τα οποία θα ανανεώνονται μέσω Refresh Tokens.
- Όλοι οι κωδικοί πρόσβασης των χρηστών πρέπει να κρυπτογραφούνται πριν την αποθήκευσή τους στη βάση δεδομένων (π.χ. μέσω του αλγορίθμου BCrypt), καθιστώντας αδύνατη την ανάκτησή τους σε μορφή απλού κειμένου (plain text).
- Το σύστημα πρέπει να εφαρμόζει έλεγχο πρόσβασης βάσει ρόλων (Role-Based Access Control - RBAC), αποτρέποντας την πρόσβαση σε διαχειριστικά API endpoints από απλούς παίκτες.

2. Απόδοση και Ταχύτητα (Performance & Responsiveness)

- Ο χρόνος απόκρισης του Back-end (REST API) για την ανάκτηση ερωτήσεων δεν πρέπει να ξεπερνά τα 2 δευτερόλεπτα, εξασφαλίζοντας την απρόσκοπτη ροή του παιχνιδιού.
- Το γραφικό περιβάλλον της Android εφαρμογής πρέπει να διατηρεί ρυθμό ανανέωσης 60 καρέ ανά δευτερόλεπτο (60 fps), προσφέροντας ομαλή κύλιση, ειδικά σε οθόνες με μεγάλο όγκο δεδομένων (όπως το Leaderboard μέσω του RecyclerView).

3. Επεκτασιμότητα (Scalability)

- Η αρχιτεκτονική του Back-end πρέπει να είναι εντελώς Stateless. Αυτό εξασφαλίζει ότι το σύστημα (Spring Boot) μπορεί να εξυπηρετήσει ταυτόχρονα χιλιάδες χρήστες χωρίς να υπερφορτώνεται η μνήμη του διακομιστή με πληροφορίες συνεδριών (sessions).
- Ο σχεδιασμός του κώδικα πρέπει να ακολουθεί το μοτίβο N-Tier, διαχωρίζοντας πλήρως την επιχειρησιακή λογική από την παρουσίαση, επιτρέποντας την εύκολη ενσωμάτωση μελλοντικών λειτουργιών (όπως Multiplayer modes) χωρίς ριζικές αλλαγές.

4. Ευχρηστία (Usability)

- Το περιβάλλον διεπαφής (UI) πρέπει να ακολουθεί τα πρότυπα του Material Design 3, προσφέροντας γνώριμα μοτίβα πλοήγησης (όπως το Bottom Navigation Bar) ώστε να ελαχιστοποιείται ο χρόνος εκμάθησης για τους νέους χρήστες.
- Η εφαρμογή πρέπει να παρέχει άμεση και κατανοητή οπτική ανατροφοδότηση (π.χ. πράσινο/κόκκινο χρώμα) σε κάθε αλληλεπίδραση του χρήστη, διατηρώντας την προσοχή του εστιασμένη στο παιχνίδι.

Συγκριτική Ανάλυση και Επιλογή Τεχνολογιών

Η επιλογή της κατάλληλης τεχνολογίας αποτελεί θεμελιώδες βήμα για τη διασφάλιση της βιωσιμότητας, της απόδοσης και της ασφάλειας ενός συστήματος λογισμικού. Για την ανάπτυξη της εφαρμογής **Quizly**, αξιολογήθηκαν διάφορες σύγχρονες τεχνολογίες πριν την τελική τους επιλογή.

Back-end: Spring Boot (Java) vs Node.js

Το Back-end αποτελεί τον πυρήνα του συστήματος, διαχειριζόμενο την επιχειρησιακή λογική και την ταυτοποίηση.

- **Node.js:** Είναι εξαιρετικά γρήγορο σε λειτουργίες I/O (Input/Output) και βασίζεται στην ασύγχρονη, event-driven αρχιτεκτονική. Ωστόσο, επειδή η JavaScript είναι δυναμικά τυποποιημένη, η συντήρηση ενός μεγάλου συστήματος με πολύπλοκες σχέσεις (όπως χρήστες, ρόλοι, βαθμολογίες) μπορεί να οδηγήσει σε σφάλματα/errors.
- **Spring Boot (Η επιλογή μας):** Η Java προσφέρει αυστηρή τυποποίηση, μειώνοντας δραστικά τα σφάλματα κατά το compiling. Το Spring Boot παρέχει ενσωματωμένα και ώριμα υποσυστήματα, όπως το **Spring Security** για διαχείριση JWT και το **Spring Data JPA** για ασφαλή επικοινωνία με τη βάση δεδομένων. Η N-Tier αρχιτεκτονική που επιβάλλει, διασφαλίζει μέγιστη συντηρησιμότητα.

Βάση Δεδομένων: PostgreSQL (RDBMS) vs MongoDB (NoSQL)

Η αποθήκευση των δεδομένων έπρεπε να εγγυάται ακεραιότητα και ταχύτητα.

- **MongoDB (NoSQL):** Λειτουργεί με έγγραφα (document-based) και προσφέρει ευελιξία στο σχήμα (schema-less). Είναι ιδανική για μη δομημένα δεδομένα, αλλά υστερεί όταν υπάρχουν αυστηρές σχέσεις και ανάγκη για ACID (Atomicity, Consistency, Isolation, Durability) συναλλαγές.
- **PostgreSQL (Η επιλογή μας):** Το παιχνίδι απαιτεί σαφείς και αυστηρές συσχετίσεις (π.χ. ένας AppUser συνδέεται με πολλά Score και UserAchievement). Η PostgreSQL, ως ένα από τα ισχυρότερα Συστήματα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων, εξασφαλίζει απόλυτη ακεραιότητα δεδομένων και ταχύτατες αναζητήσεις μέσω πολύπλοκων queries.

Front-end: Native Android (Java) vs Cross-Platform (Flutter/React Native)

Το περιβάλλον διεπαφής έπρεπε να προσφέρει ακαριαία απόκριση.

- **Cross-Platform:** Προσφέρουν το πλεονέκτημα του "γράφω μία φορά, τρέχει παντού" (iOS και Android). Όμως, προσθέτουν ένα επιπλέον επίπεδο μεταξύ του κώδικα και του υλικού της συσκευής, κάτι που ενδέχεται να προκαλέσει frame drop σε βαριές οθόνες.
- **Native Android (Η επιλογή μας):** Η εγγενής ανάπτυξη με Java και το Android SDK προσφέρει τη μέγιστη δυνατή απόδοση. Επιτρέπει την απόλυτη εκμετάλλευση εργαλείων όπως το RecyclerView και το ViewPager2, εξασφαλίζοντας 60 fps κατά την κύλιση στον Παγκόσμιο Πίνακα Κατάταξης, προσφέροντας μια κορυφαία εμπειρία χρήστη.

Επισκόπηση Αρχιτεκτονικής

Η δομή της εφαρμογής Quizly έχει σχεδιαστεί με βάση την Αρχιτεκτονική N-Tier, μια προσέγγιση που κατανέμει τις λειτουργίες του συστήματος σε ξεχωριστά, ανεξάρτητα επίπεδα. Η αρχιτεκτονική αυτή απαρτίζεται από τρεις βασικούς πυλώνες: την εφαρμογή για κινητές συσκευές Android (Client/Front-end), τον κεντρικό εξυπηρετητή (Back-end) που έχει αναπτυχθεί με το Spring Boot, και τη σχεσιακή βάση δεδομένων PostgreSQL για την ασφαλή αποθήκευση της πληροφορίας.

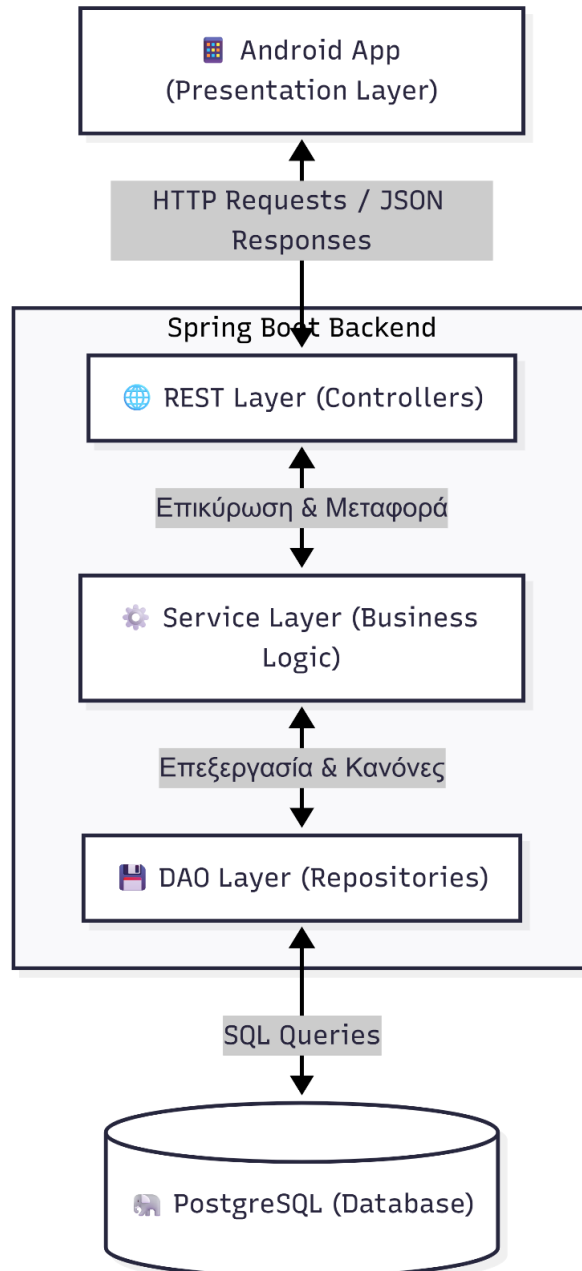
Αναλυτικότερα, το Επίπεδο Παρουσίασης (Presentation Layer) ταυτίζεται με την ίδια την εφαρμογή Android. Αποτελεί τη "βιτρίνα" του συστήματος, το σημείο δηλαδή όπου ο τελικός χρήστης αλληλεπιδρά άμεσα με το παιχνίδι. Μέσω αυτού του επιπέδου, ο παίκτης περιηγείται στις ερωτήσεις, ενημερώνει το προφίλ του, βλέπει τις επιδόσεις του και παρακολουθεί την παγκόσμια κατάταξη (Leaderboard).

Στο κέντρο της αρχιτεκτονικής βρίσκεται ο Back-end Server, ο οποίος λειτουργεί ως ο «εγκέφαλος» της εφαρμογής. Διαχειρίζεται την αμφίδρομη επικοινωνία μεταξύ του κινητού και της βάσης δεδομένων, με τη μορφή HTTP αιτημάτων και απαντήσεων δομημένων σε JSON. Προκειμένου ο κώδικας να παραμένει οργανωμένος και συντηρήσιμος, ο εξυπηρετητής διαχωρίζεται εσωτερικά σε τρία υπο-επίπεδα (separation of concerns):

1. **Επίπεδο REST (Controllers):** Είναι η "πύλη εισόδου" του server. Υποδέχεται τα αιτήματα του Android client, όπως για παράδειγμα τις αιτήσεις για σύνδεση (login), την ανάκτηση νέων ερωτήσεων ή την αποθήκευση ενός νέου σκορ, και τα προωθεί στα ενδότερα του συστήματος.
2. **Επίπεδο Επιχειρησιακής Λογικής (Service Layer):** Εδώ φιλοξενείται ο πυρήνας της λειτουργικότητας του Quizly. Τα Services επεξεργάζονται τα δεδομένα που έλαβαν από τους Controllers, εφαρμόζοντας τους κανόνες του παιχνιδιού (π.χ. υπολογισμός των XP, έλεγχος για την κατάκτηση νέων επιτευγμάτων, διαχείριση της ισχύος των JWT tokens).
3. **Επίπεδο Πρόσβασης Δεδομένων (DAO/Repository Layer):** Είναι ο αποκλειστικός υπεύθυνος για την επικοινωνία με τη βάση δεδομένων. Εκτελεί όλες τις απαραίτητες ενέργειες αποθήκευσης, ανάγνωσης, ενημέρωσης και διαγραφής (CRUD λειτουργίες) στον πίνακα της PostgreSQL.

Στο κατώτερο επίπεδο συναντάμε τη Βάση Δεδομένων (PostgreSQL), η οποία αποτελεί το ασφαλές αποθετήριο του συστήματος. Εκεί τηρούνται οργανωμένα τα δεδομένα των εγγεγραμμένων παικτών, οι βαθμολογίες τους, τα επιτεύγματα που έχουν ξεκλειδώσει και το ιστορικό των ανανεώσεων πρόσβασης (refresh tokens).

Τέλος, η αρχιτεκτονική περιβάλλεται από μια αυστηρή ασπίδα ασφαλείας, υλοποιημένη με το Spring Security και την τεχνολογία JSON Web Tokens (JWT). Αυτό το σύστημα διασφαλίζει ότι μόνο εξουσιοδοτημένοι χρήστες έχουν πρόσβαση στα δεδομένα και τις λειτουργίες της εφαρμογής, προσφέροντας μια ασφαλή και απρόσκοπτη εμπειρία παιχνιδιού.

**Εικόνα 1: Επισκόπηση Αρχιτεκτονική**

Αρχικό στήσιμο εφαρμογής

Στήσιμο του back-end

Για τη δημιουργία και την αρχικοποίηση του βασικού κορμού του συστήματος, αξιοποιήθηκε το διαδικτυακό εργαλείο Spring Initializr. Πρόκειται για την επίσημη πλατφόρμα της κοινότητας του Spring, η οποία επιτρέπει τη γρήγορη παραγωγή ενός έτοιμου προς ανάπτυξη έργου (project) με προρυθμισμένες τις απαραίτητες εξαρτήσεις (dependencies).

Στο έργο ενσωματώθηκαν στρατηγικά οι εξής βιβλιοθήκες: το Spring Web για την ανάπτυξη των RESTful API endpoints, το Spring Data JPA για την ομαλή διασύνδεση και διαχείριση της βάσης δεδομένων μέσω Object-Relational Mapping (ORM) και το Spring Security για την επιβολή κανόνων ελέγχου πρόσβασης. Επιπρόσθετα, συμπεριλήφθηκαν ο οδηγός της PostgreSQL για την επικοινωνία με τη σχεσιακή βάση, καθώς και η βιβλιοθήκη JWT για τη δημιουργία και διαχείριση των JSON Web Tokens (JWT) που αναλαμβάνουν την αυθεντικοποίηση.

Στήσιμο της βάσης δεδομένων

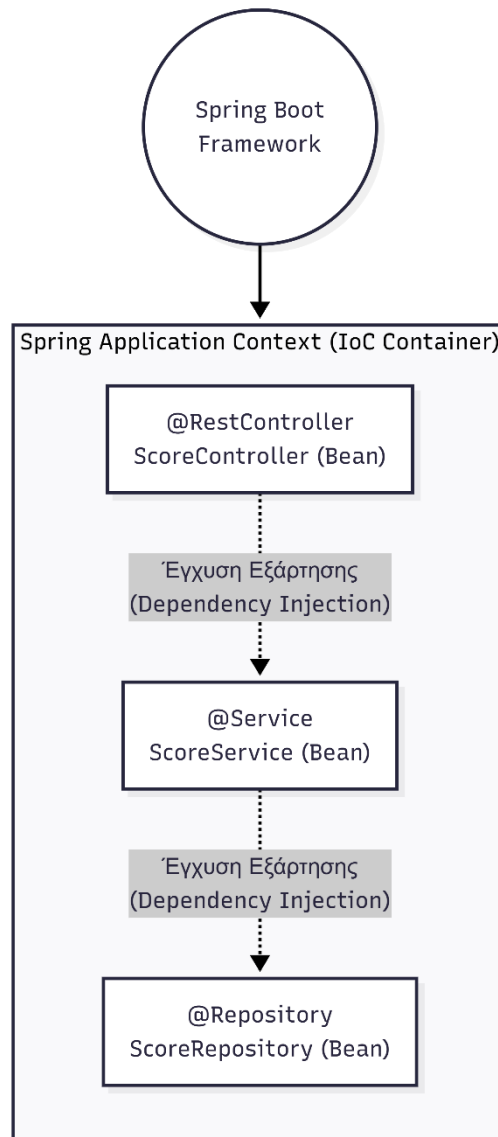
Για την ασφαλή και αποδοτική αποθήκευση των δεδομένων του παιχνιδιού, επιλέχθηκε το Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (RDBMS) PostgreSQL. Το περιβάλλον της βάσης μπορεί να εκτελεστεί αυτόνομα ή μέσα από ένα docker container, εξασφαλίζοντας υψηλή φορητότητα και ευκολία στην ανάπτυξη.

Η σύνδεση της εφαρμογής Spring Boot με την PostgreSQL υλοποιήθηκε μέσω του αρχείου ρυθμίσεων της εφαρμογής. Εκεί καθορίστηκαν οι κρίσιμες παράμετροι σύνδεσης, όπως το URL του εξυπηρετητή της βάσης, το όνομα χρήστη (username) και ο κωδικός πρόσβασης (password). Το Spring Boot αναλαμβάνει αυτόματα να δημιουργήσει τη "γέφυρα" επικοινωνίας, επιτρέποντας στο backend να καταχωρεί τα σκορ, τα νέα προφίλ χρηστών και τις ερωτήσεις απρόσκοπτα.

Spring Boot (Application Context & Beans)

Το Spring Boot αποτελεί το δημοφιλέστερο framework για την ανάπτυξη σύγχρονων, αυτόνομων εφαρμογών Java. Το μεγάλο του πλεονέκτημα έγκειται στο ότι αποκρύπτει την πολυπλοκότητα των configurations, προσφέροντας έτοιμες προεπιλογές (auto-configuration) για τη δημιουργία αξιόπιστων backend συστημάτων.

Ένα από τα ισχυρότερα χαρακτηριστικά του είναι η dependency injection. Στην παραδοσιακή ανάπτυξη λογισμικού, ο προγραμματιστής πρέπει να δημιουργεί συνεχώς νέα στιγμιότυπα κλάσεων (με τη λέξη-κλειδί new). Σε ένα μεγάλο έργο όπως το Quizly, αυτό θα επιβάρυνε σημαντικά τη μνήμη. Το Spring λύνει αυτό το πρόβλημα με τα Beans. Τα Beans είναι αντικείμενα Java των οποίων τον κύκλο ζωής διαχειρίζεται αποκλειστικά το framework. Φυλάσσονται σε έναν κεντρικό "χώρο", το Application Context και όποτε μια κλάση (π.χ. ένας Controller) χρειάζεται πρόσβαση σε μια υπηρεσία (π.χ. το ScoreService), το Spring αναλαμβάνει να του την κάνει inject αυτόματα. Έτσι, εξασφαλίζεται η βέλτιστη χρήση πόρων και η modular αρχιτεκτονική του κώδικα.

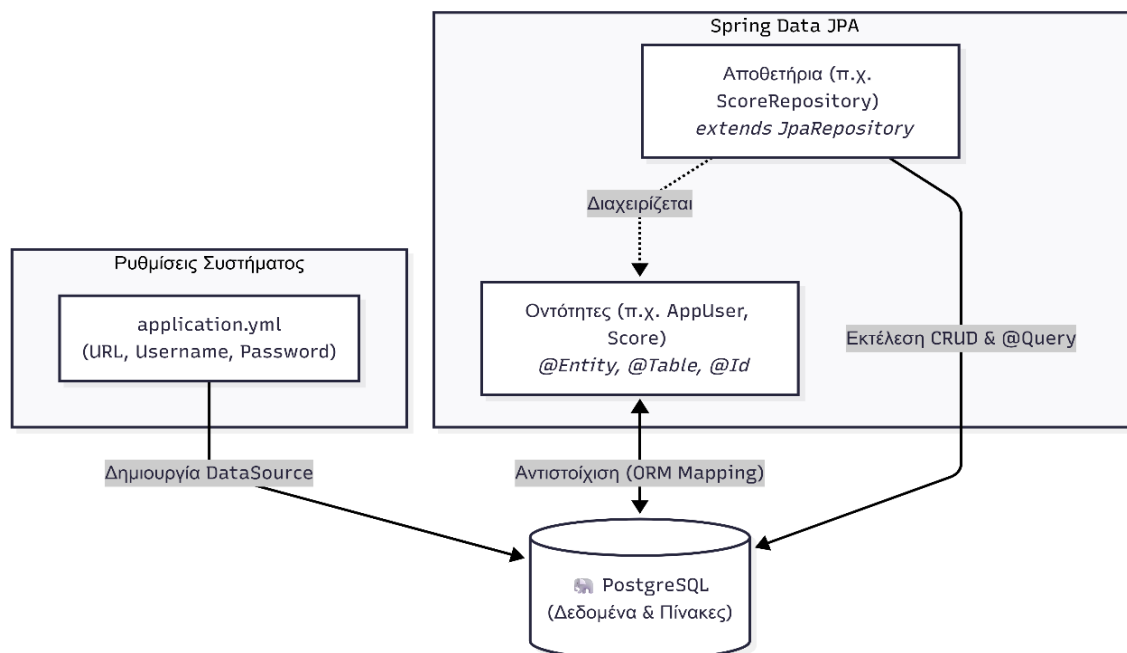


Διαχείριση Δεδομένων με Spring Data JPA

Το Spring Data JPA αποτελεί μια ισχυρή επέκταση που δραματικά απλοποιεί την αλληλεπίδραση του backend με τη βάση δεδομένων. Παρέχει ένα abstraction layer που επιτρέπει στον προγραμματιστή να διαχειρίζεται τα δεδομένα ως αντικείμενα Java (Entities), χωρίς να απαιτείται η συγγραφή περίπλοκου, επαναλαμβανόμενου κώδικα SQL για βασικές λειτουργίες.

Η διαδικασία βασίζεται στο Entity Mapping. Κλάσεις όπως το AppUser, το Question, το Score και το UserAchievement αναπαριστούν τους πίνακες της βάσης. Χρησιμοποιώντας ειδικούς προσδιορισμούς όπως @Entity, @Table, @Id και @Column, υποδεικνύεται στο σύστημα πώς ακριβώς κάθε μεταβλητή της κλάσης θα αντιστοιχηθεί στην αντίστοιχη στήλη του πίνακα της PostgreSQL.

Επιπρόσθετα, το Spring Data JPA προσφέρει τη διεπαφή JpaRepository. Επεκτείνοντας αυτή τη διεπαφή, το σύστημα κληρονομεί αυτόματα έτοιμες μεθόδους για τις βασικές λειτουργίες CRUD, όπως save(), findById() και delete(). Ωστόσο, για τις ιδιαίτερες ανάγκες του Quizly - όπως η δημιουργία του παγκόσμιου πίνακα κατάταξης (Leaderboard) ή ο υπολογισμός των συνολικών πόντων εμπειρίας (XP) ανά θεματική κατηγορία - αξιοποιήθηκε το annotation @Query. Αυτό επέτρεψε τη συγγραφή native SQL queries, όπως αυτά που βρίσκονται στο ScoreRepository, προσφέροντας απόλυτο έλεγχο και ταχύτητα στην άντληση σύνθετων στατιστικών δεδομένων.



Εικόνα 2: Σχεσιακό Μοντέλο Δεδομένων και Διαγράμματα Κλάσεων (UML) της εφαρμογής.

Υλοποίηση Συστήματος Ασφάλειας και Αυθεντικοποίησης

Η διασφάλιση των προσωπικών δεδομένων και η προστασία των API endpoints αποτελούν τον ακρογωνιαίο λίθο της εφαρμογής Quizly. Για τον σκοπό αυτό, αναπτύχθηκε ένα σύνθετο σύστημα ασφαλείας βασισμένο στο πλαίσιο εργασίας Spring Security, το οποίο παραμετροποιήθηκε ώστε να υποστηρίζει μια αρχιτεκτονική χωρίς κατάσταση (stateless) μέσω της τεχνολογίας JSON Web Tokens (JWT).

Η Οντότητα AppUser και η Διεπαφή UserDetails

Προκειμένου το back-end να αναγνωρίζει και να διαχειρίζεται τους χρήστες με ασφάλεια, δημιουργήθηκε η οντότητα AppUser. Η κλάση αυτή δεν αποτελεί απλώς μια αναπαράσταση των δεδομένων στη βάση, αλλά υλοποιεί τη διεπαφή UserDetails του Spring Security. Η υλοποίηση αυτή είναι επιβεβλημένη, καθώς επιτρέπει στο framework να ανακτά με τυποποιημένο τρόπο τις απαραίτητες πληροφορίες για την αυθεντικοποίηση.

Η κλάση AppUser περιλαμβάνει κρίσιμα πεδία όπως το id, το firstName, το lastName και το password. Ως μοναδικό αναγνωριστικό χρήστη (username) επιλέχθηκε το πεδίο email, το οποίο φέρει τον περιορισμό unique στη βάση δεδομένων για την αποφυγή συγκρούσεων. Μέσω των μεθόδων της UserDetails, όπως η getAuthorities(), το σύστημα επιστρέφει τη λίστα των δικαιωμάτων του χρήστη, ενώ οι μέθοδοι isAccountNonExpired(), isAccountNonLocked() και isEnabled() έχουν ρυθμιστεί να επιστρέφουν σταθερά true, διασφαλίζοντας την άμεση πρόσβαση του παίκτη.

Για την επικοινωνία της οντότητας με το επίπεδο αποθήκευσης, χρησιμοποιείται το AppUserRepository, το οποίο επεκτείνει το JpaRepository. Αυτή η προσέγγιση απλοποιεί δραστηριότητες CRUD, καθώς παρέχει έτοιμες μεθόδους για την αναζήτηση χρηστών βάσει email ή ID, εξασφαλίζοντας υψηλή απόδοση και καθαρότητα κώδικα.

Η Υπηρεσία UserDetailsService και η Ανάκτηση Δεδομένων

Η κλάση UserDetailsService (στο πακέτο com.panagiotis.quizly.user) αποτελεί την «καρδιά» της διαδικασίας ταυτοποίησης. Υλοποιεί την ομώνυμη διεπαφή του Spring Security και περιλαμβάνει τη μέθοδο loadUserByUsername(String username). Η μέθοδος αυτή έχει γίνει override ώστε να αναζητά τον χρήστη στη βάση δεδομένων μέσω του UserDataAccessService χρησιμοποιώντας το email του.

Εάν ο χρήστης εντοπιστεί, η υπηρεσία επιστρέφει το αντικείμενο AppUser, το οποίο πλέον περιέχει όλα τα απαραίτητα διαπιστευτήρια. Σε περίπτωση που το email δεν αντιστοιχεί σε κάποιον εγγεγραμμένο παίκτη, εκπέμπεται μια εξαίρεση τύπου UsernameNotFoundException. Αυτή η αρχιτεκτονική επιτρέπει τον πλήρη διαχωρισμό της λογικής αποθήκευσης από τη λογική της ασφάλειας.

Διαχείριση Ρόλων και Δικαιωμάτων Πρόσβασης (RBAC)

Η εξουσιοδότηση στο Quizly βασίζεται στο μοντέλο Role-Based Access Control (RBAC). Η κλάση Role αναπαριστά τα διαφορετικά επίπεδα πρόσβασης, όπως ROLE_USER, ROLE_ADMIN και ROLE_SIMPLE_ADMIN. Η κλάση αυτή υλοποιεί τη διεπαφή GrantedAuthority, η οποία είναι απαραίτητη για να μπορεί το Spring Security να αναγνωρίζει τους ρόλους και να επιβάλλει τους αντίστοιχους περιορισμούς στα API endpoints.

Η κλάση Role περιλαμβάνει το πεδίο authority, το οποίο αποθηκεύει την ονομασία του ρόλου, και συνδέεται με την οντότητα AppUser μέσω μιας σχέσης «πολλά προς πολλά» (ManyToMany). Για τη διαχείριση αυτών των δεδομένων, χρησιμοποιείται το RoleRepository, το οποίο επιτρέπει την ανάκτηση συγκεκριμένων ρόλων μέσω της μεθόδου findByAuthority(). Αυτό είναι ιδιαίτερα χρήσιμο κατά τη διαδικασία της εγγραφής, όπου σε κάθε νέο χρήστη αποδίδεται αυτόματα ο ρόλος ROLE_USER.

JSON Web Token (JWT): Δομή και Υλοποίηση

Λόγω της φύσης της Android εφαρμογής, επιλέχθηκε η χρήση JSON Web Tokens (JWT) για την αυθεντικοποίηση. Το JWT επιτρέπει στον server να παραμένει stateless, δηλαδή να μην χρειάζεται να αποθηκεύει πληροφορίες συνεδρίας στη μνήμη του. Το token αποτελείται από τρία διακριτά μέρη:

- Header: Περιέχει πληροφορίες για τον τύπο του token και το signing algorithm που χρησιμοποιήθηκε.
- Payload: Περιλαμβάνει τα «claims», δηλαδή δεδομένα όπως το email του χρήστη, τον εκδότη (issuer), την ημερομηνία λήξης και τους ρόλους (scopes) του παίκτη.
- Signature: Δημιουργείται με τον συνδυασμό των δύο παραπάνω μερών και ενός κρυφού κλειδιού (Secret Key), διασφαλίζοντας ότι το token δεν έχει τροποποιηθεί.

Η υλοποίηση αυτής της λογικής βρίσκεται στην κλάση JWTUtil. Η μέθοδος issueToken αναλαμβάνει τη δημιουργία του token, ενώ η isValidToken ελέγχει αν το token που στέλνει το κινητό ανήκει στον σωστό χρήστη και αν είναι ακόμη σε ισχύ.

Παραμετροποίηση Αλυσίδας Ασφαλείας (SecurityFilterChainConfig)

Η κλάση SecurityFilterChainConfig αποτελεί το κεντρικό σημείο ελέγχου της ασφάλειας. Μέσω της μεθόδου securityFilterChain, ορίζονται οι εξής κανόνες:

- Απενεργοποίηση CSRF: Καθώς η εφαρμογή λειτουργεί ως RESTful API για κινητές συσκευές, η προστασία από Cross-Site Request Forgery κρίθηκε περιττή.
- Ρύθμιση CORS: Επιτρέπεται η επικοινωνία με την Android εφαρμογή, διασφαλίζοντας την ασφαλή ανταλλαγή πόρων.
- Endpoint Authorization: Ορίστηκε ότι endpoints όπως το /api/v1/auth/login και το registration (POST /api/v1/users) είναι ελεύθερα προσβάσιμα. Αντίθετα, όλα τα υπόλοιπα αιτήματα απαιτούν έγκυρο JWT.
- Stateless Policy: Η διαχείριση συνεδρίας ρυθμίστηκε σε STATELESS, αναγκάζοντας το σύστημα να ελέγχει το token σε κάθε αίτημα ξεχωριστά.

Το Φίλτρο JWTAuthenticationFilter

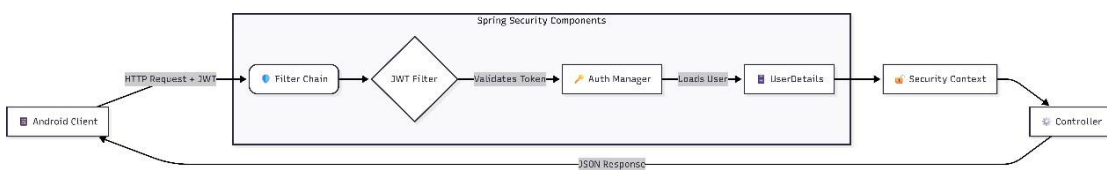
Το JWTAuthenticationFilter παρεμβάλλεται σε κάθε αίτημα HTTP πριν αυτό φτάσει στους Controllers. Το φίλτρο αναζητά την κεφαλίδα "Authorization" και, αν εντοπίσει το πρόθεμα "Bearer", εξάγει το token. Χρησιμοποιώντας την κλάση JWTUtil, ανακτά το email του χρήστη και επαληθεύει την ταυτότητά του μέσω του UserDetailsService. Εφόσον το token είναι έγκυρο, δημιουργείται ένα UsernamePasswordAuthenticationToken το οποίο τοποθετείται στο SecurityContextHolder, δίνοντας την άδεια στο αίτημα να προχωρήσει.



Εικόνα 3: Ροή αυθεντικοποίησης JWT

Διαδικασία Αυθεντικοποίησης

Το παρακάτω διάγραμμα παρέχει μια σαφή επισκόπηση του τρόπου με τον οποίο αλληλεπιδρούν μεταξύ τους τα διάφορα components που συντελούν στην ασφάλεια της εφαρμογής. Δείχνει τη διαδρομή που κάθε αίτημα HTTP ακολουθεί.



Εικόνα 4: Διαδικασία Αυθεντικοποίησης

Μηχανισμός Refresh Tokens

Για την ενίσχυση της εμπειρίας του χρήστη, υλοποιήθηκε ο μηχανισμός Refresh Tokens. Καθώς τα JWT έχουν σύντομη διάρκεια ζωής, το refresh token επιτρέπει την έκδοση νέων access tokens χωρίς τη μεσολάβηση του χρήστη. Η κλάση RefreshTokenService διαχειρίζεται τη δημιουργία και την αποθήκευση αυτών των tokens στη βάση δεδομένων. Μέσω της μεθόδου verifyExpiration, ο server ελέγχει αν το refresh token έχει λήξει και, αν είναι έγκυρο, προχωρά στην έκδοση νέου JWT, διατηρώντας τη σύνδεση του παίκτη ενεργή.

Διαχείριση Σφαλμάτων και Εξαιρέσεων Ασφαλείας

Για την παροχή σωστής ανατροφοδότησης στην Android εφαρμογή, υλοποιήθηκε ένας κεντρικός μηχανισμός διαχείρισης σφαλμάτων. Η κλάση DelegatedAuthEntryPoint αναχαιτίζει τις αποτυχίες αυθεντικοποίησης και τις προωθεί στον DefaultExceptionHandler. Εκεί, εξαιρέσεις όπως η BadCredentialsException μετατρέπονται σε αντικείμενα ApiError, τα οποία επιστρέφουν στον χρήστη ένα δομημένο JSON response με πληροφορίες για τη φύση του σφάλματος και τον αντίστοιχο κωδικό HTTP (π.χ. 401 Unauthorized).

Διάγραμμα της βάσης δεδομένων

Για την ολοκληρωμένη και λεπτομερή απεικόνιση της αρχιτεκτονικής των δεδομένων της εφαρμογής Quizly, επιλέχθηκε η χρήση ενός διαγράμματος τάξεων UML (Unified Modeling Language). Το διάγραμμα αυτό αποτελεί το πλέον αναγνωρισμένο πρότυπο στη μηχανική λογισμικού για την παροχή μιας σαφούς και δομημένης περιγραφής της στατικής δομής ενός συστήματος. Μέσω αυτού, καθίσταται δυνατή η καταγραφή των τύπων των αντικειμένων, των ιδιοτήτων τους, καθώς και των σύνθετων στατικών σχέσεων που αναπτύσσονται μεταξύ τους κατά τη λειτουργία της εφαρμογής.

Στο πλαίσιο της αντικειμενοστραφούς ανάλυσης, ο όρος κλάση (ή τάξη) αναφέρεται σε ένα λογικό πρότυπο, ένα «blueprint» (προσχέδιο), το οποίο ορίζει τις προδιαγραφές για τη δημιουργία αντικειμένων. Τα αντικείμενα αυτά, τα οποία αποτελούν στιγμιότυπα (instances) των κλάσεων, μοιράζονται κοινές ιδιότητες, συμπεριφορές και μεθόδους, οι οποίες υλοποιούνται κατά τη διάρκεια εκτέλεσης του προγράμματος. Το διάγραμμα τάξεων ανήκει στην ευρύτερη κατηγορία των δομικών διαγραμμάτων (structural diagrams). Η σημασία του έγκειται στο ότι δεν περιορίζεται μόνο στην απεικόνιση των δεδομένων, αλλά αποτυπώνει και τους περιορισμούς που διέπουν τις συνδέσεις μεταξύ των αντικειμένων, διασφαλίζοντας την ακεραιότητα του συστήματος.

Η UML χρησιμοποιεί συγκεκριμένη ορολογία και συμβολισμούς για την περιγραφή των στοιχείων μιας κλάσης:

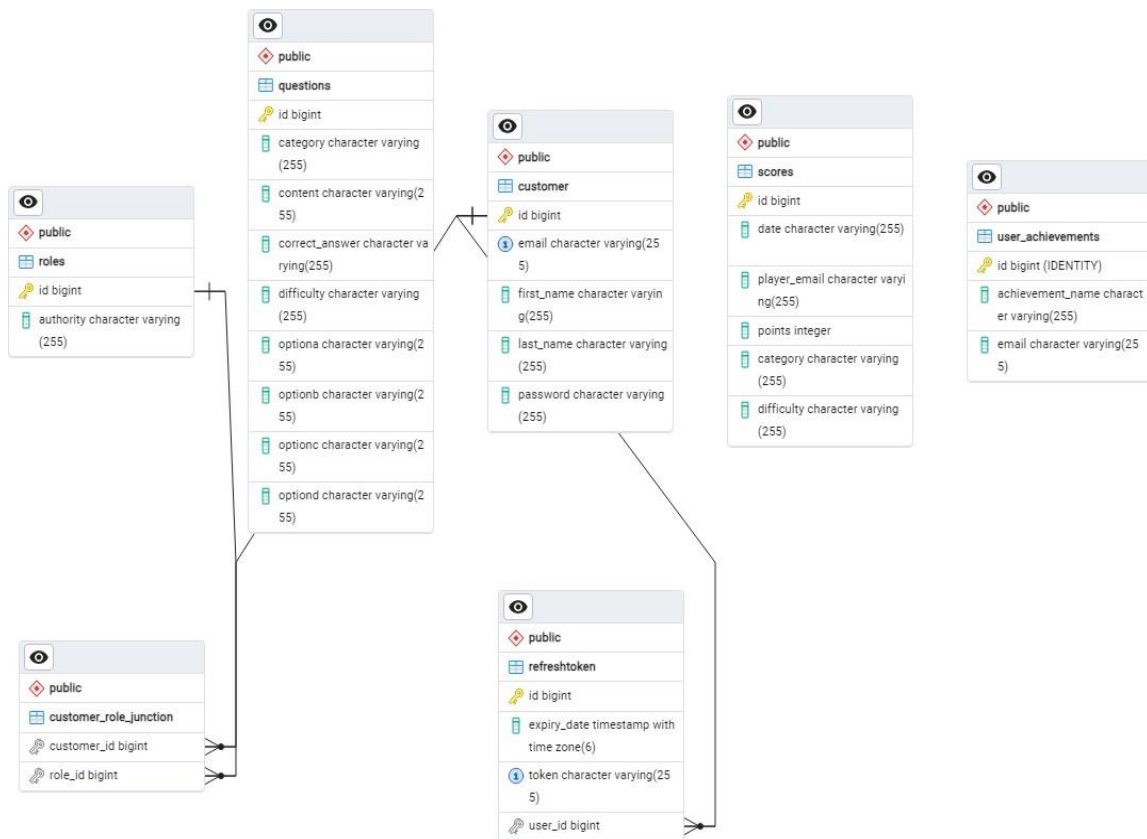
- **Attributes (Ιδιότητες):** Αναφέρονται στα δεδομένα που αποθηκεύονται σε κάθε κλάση (π.χ. το email ενός χρήστη ή το επίπεδο δυσκολίας μιας ερώτησης).
- **Operations (Λειτουργίες):** Αναφέρονται στις μεθόδους και τις λειτουργίες που μπορεί να εκτελέσει η κλάση πάνω στα δεδομένα της.

Η οπτική αναπαράσταση των κλάσεων γίνεται μέσω ορθογωνίων τα οποία χωρίζονται σε τρία διακριτά μέρη. Το ανώτερο τμήμα περιλαμβάνει το όνομα της κλάσης (π.χ. AppUser, Score, Question). Το μεσαίο τμήμα φιλοξενεί τα attributes, ενώ το κατώτερο τμήμα περιέχει τις operations. Για την ενίσχυση της ασφάλειας και της ενθυλάκωσης (encapsulation), τα attributes των κλάσεων στο Quizly φέρουν συνήθως τον προσδιοριστή πρόσβασης private (συμβολίζεται με το μείον «-»), ενώ οι operations είναι public (συμβολίζεται με το συν «+»).

Το διάγραμμα της βάσης δεδομένων για το Quizly αναπτύχθηκε με τη χρήση του IntelliJ IDEA, αξιοποιώντας τις δυνατότητες του εργαλείου για την αυτόματη παραγωγή διαγραμμάτων από τον κώδικα Java. Το αποτέλεσμα απεικονίζει το πλήρες σχήμα της σχεσιακής βάσης δεδομένων PostgreSQL, καταγράφοντας όλες τις οντότητες που υποστηρίζουν το παιχνίδι:

1. **AppUser:** Αποθηκεύει τα στοιχεία ταυτοποίησης και το προφίλ του παίκτη.
2. **Role:** Διαχειρίζεται τα δικαιώματα πρόσβασης (Admin, User).
3. **Question:** Περιέχει το σώμα της ερώτησης, τις εναλλακτικές απαντήσεις και τη θεματική κατηγορία.
4. **Score:** Καταγράφει τις επιδόσεις των χρηστών, τη βαθμολογία και τη χρονική σήμανση της προσπάθειας.
5. **UserAchievement:** Συνδέει τους παίκτες με τα επιτεύγματα που έχουν ξεκλειδώσει κατά την πρόοδό τους.

Επισημαίνεται ότι, για λόγους βελτίωσης της αναγνωσιμότητας και επικέντρωσης στη δομική ουσία της βάσης, οι μέθοδοι πρόσβασης (getters και setters), καθώς και οι constructors των κλάσεων, έχουν παραλειφθεί από την οπτική απεικόνιση του διαγράμματος.

**Εικόνα 5: Database Schema**

Κλάσεις για την υλοποίηση της αρχιτεκτονικής

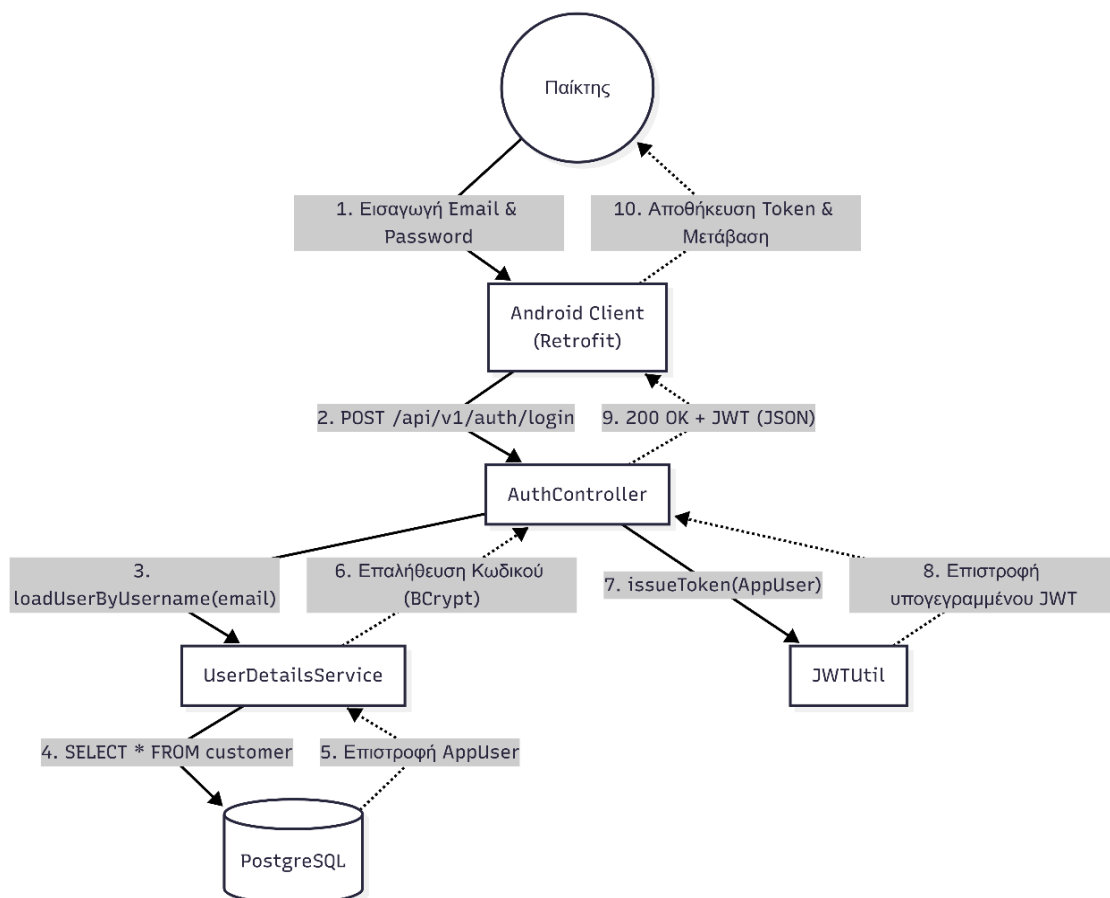
Στο πλαίσιο του Spring Boot project για την εφαρμογή Quizly, υιοθετήθηκε μια αυστηρή N-tier αρχιτεκτονική, η οποία επιτρέπει τον πλήρη διαχωρισμό των ευθυνών (Separation of Concerns). Κάθε δομική οντότητα του συστήματος, όπως οι ερωτήσεις (Question), οι βαθμολογίες (Score) και οι χρήστες (AppUser), υποστηρίζεται από ένα σύνολο αλληλοεξαρτώμενων components που διασφαλίζουν την ομαλή ροή των δεδομένων από τη βάση έως τον τελικό χρήστη.

1. **Controller Layer:** Οι Controllers (π.χ. ScoreController, UserController) αποτελούν το επίπεδο παρουσίασης του back-end. Είναι υπεύθυνοι για την έκθεση των REST endpoints, την υποδοχή των αιτημάτων HTTP και την επιστροφή των απαντήσεων σε μορφή JSON. Η κύρια ευθύνη τους είναι η δρομολόγηση του αιτήματος στο επόμενο επίπεδο, χωρίς να περιλαμβάνουν επιχειρησιακή λογική.
2. **Service Layer:** Το επίπεδο αυτό (π.χ. ScoreService, AppUserService) αποτελεί την "καρδιά" της εφαρμογής. Εδώ υλοποιείται η επιχειρησιακή λογική, όπως ο υπολογισμός των πόντων εμπειρίας (XP) μετά από ένα quiz ή ο έλεγχος των κριτηρίων για την απόδοση ενός επιτεύγματος (Achievement). Τα Services συντονίζουν τις εργασίες και διασφαλίζουν την ακεραιότητα των δεδομένων πριν αυτά καταλήξουν στη βάση.
3. **DataAccessService (DAO):** Λειτουργεί ως ενδιάμεσο επίπεδο αφαίρεσης μεταξύ της επιχειρησιακής λογικής και των αποθετηρίων δεδομένων. Μετασχηματίζει τα δεδομένα και εκτελεί τις CRUD λειτουργίες, εξασφαλίζοντας ότι το Service Layer δεν χρειάζεται να γνωρίζει τις λεπτομέρειες της επικοινωνίας με τη βάση.
4. **JpaRepository Layer:** Πρόκειται για διεπαφές (interfaces) που επεκτείνουν το JpaRepository του Spring Data JPA. Αυτά τα components παρέχουν αυτόματα όλη τη λειτουργικότητα για την αλληλεπίδραση με την PostgreSQL, συμπεριλαμβανομένων σύνθετων αναζητήσεων μέσω μεθόδων όπως η findByAuthority() ή προσαρμοσμένων ερωτημάτων με το annotation @Query.
5. **Records και Mappers:** Για την ασφαλή μεταφορά δεδομένων και την αποφυγή έκθεσης ευαίσθητων πληροφοριών (όπως κωδικοί πρόσβασης), χρησιμοποιούνται αντικείμενα μεταφοράς δεδομένων (DTOs) και κλάσεις χαρτογράφησης όπως η AppUserDTOMapper. Τα Java Records (π.χ. AppUserRegistrationRequest) διασφαλίζουν ότι τα δεδομένα που εισέρχονται στον Controller είναι αμετάβλητα και σωστά δομημένα.

Ροές Δεδομένων και Διαγράμματα Αλληλουχίας

1. Διαδικασία Ταυτοποίησης Χρήστη (Login Flow)

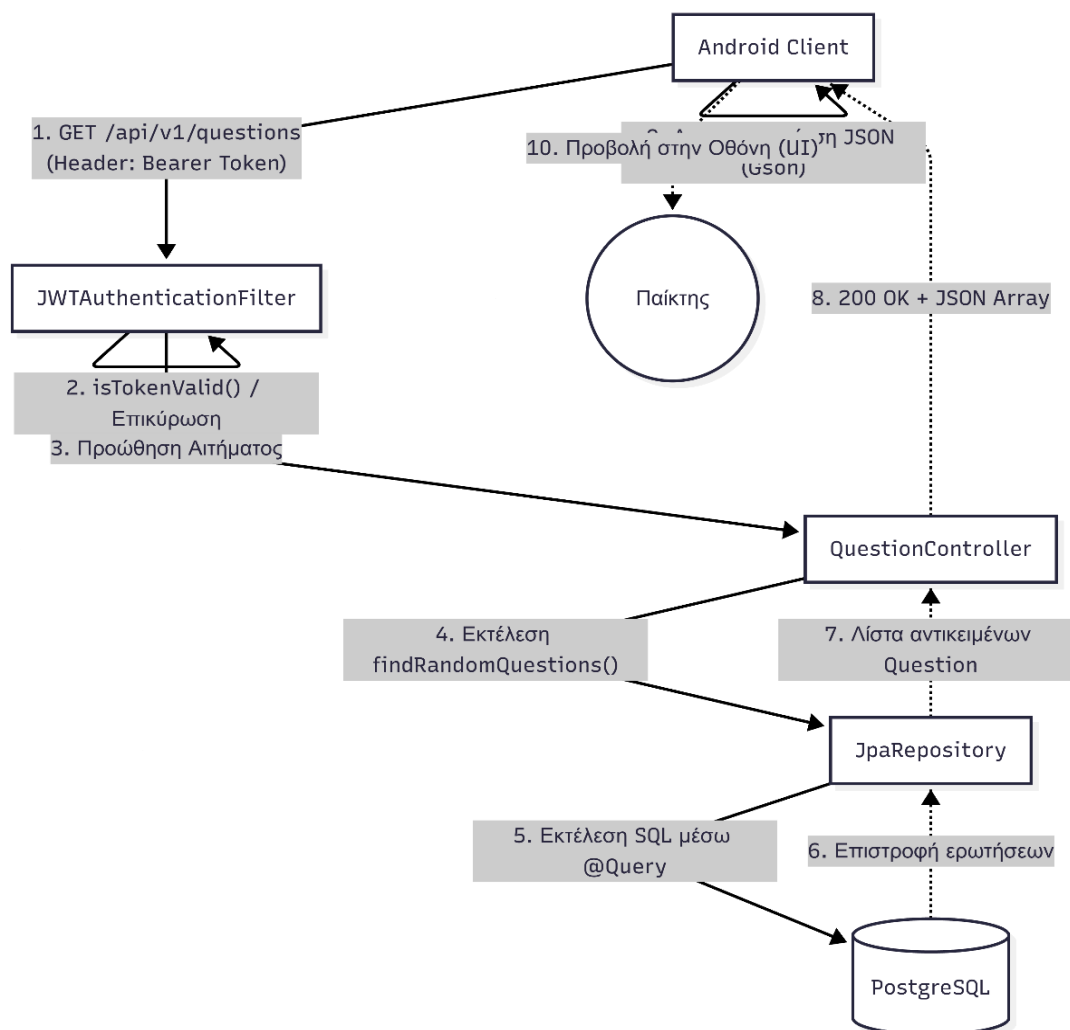
Στο παρακάτω διάγραμμα αλληλουχίας αποτυπώνεται η αυστηρή ροή εκτέλεσης κατά τη διαδικασία της αυθεντικοποίησης του χρήστη. Αρχικά, η συσκευή συλλέγει τα διαπιστευτήρια του παίκτη και, αξιοποιώντας τη βιβλιοθήκη **Retrofit2**, μορφοποιεί ένα HTTP POST αίτημα προς τον εξυπηρετητή. Το αίτημα υποδέχεται ο AuthController, ο οποίος αποτελεί το σημείο εισόδου του REST API. Στη συνέχεια, καλείται η υπηρεσία UserDetailsService, η οποία επικοινωνεί απευθείας με τη βάση δεδομένων **PostgreSQL** για την άντληση των στοιχείων της οντότητας AppUser. Εφόσον ο εισαχθείς κωδικός επαληθευτεί επιτυχώς με τον κρυπτογραφημένο κωδικό της βάσης, το σύστημα καλεί την κλάση JWTUtil προκειμένου να παραγάγει ένα ασφαλές, υπογεγραμμένο JSON Web Token. Η διαδικασία ολοκληρώνεται με την επιστροφή του token στον πελάτη σε μορφή JSON, επιτρέποντας την ασφαλή πλοήγηση στα προστατευμένα τμήματα της εφαρμογής.



Εικόνα 6: Διάγραμμα ροής δεδομένων για τη διαδικασία ταυτοποίησης του χρήστη και την ασφαλή έκδοση του JSON Web Token

2. Διαδικασία Ανάκτησης Ερωτήσεων Κουίζ

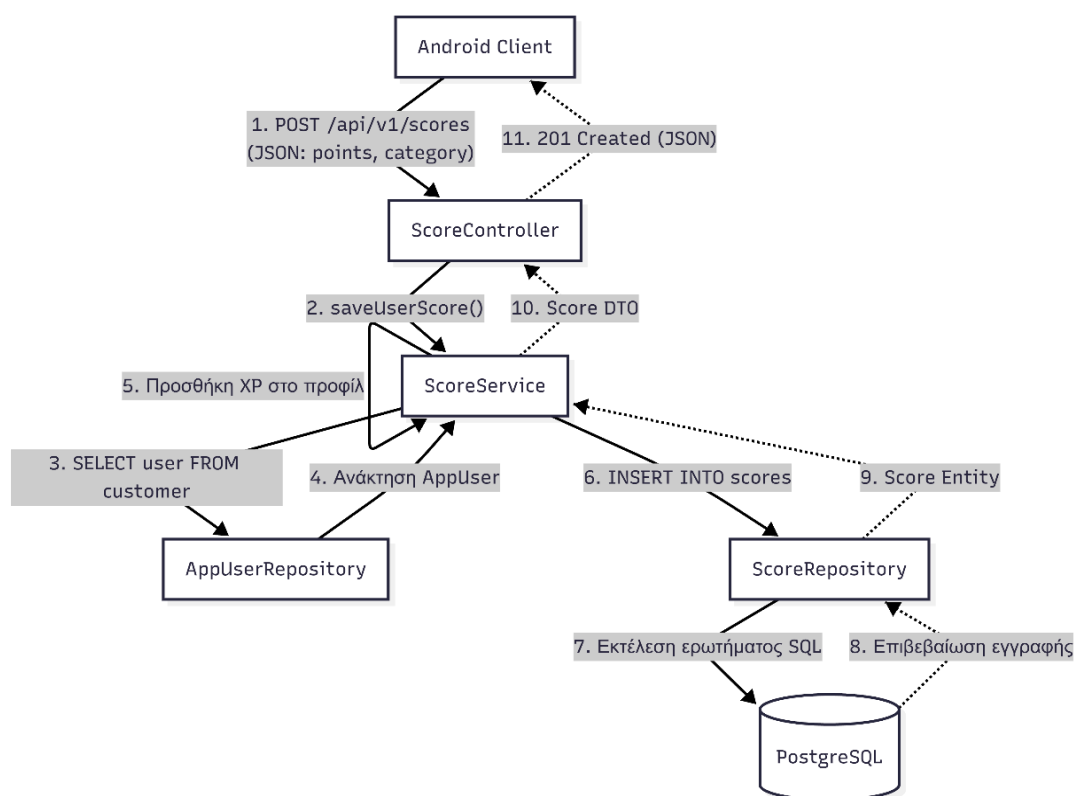
Το δεύτερο διάγραμμα παρουσιάζει τον κύκλο ζωής ενός αιτήματος για την ανάκτηση των δεδομένων του παιχνιδιού, επιδεικνύοντας τον κρίσιμο ρόλο της ασφάλειας σε μια stateless αρχιτεκτονική. Όταν ο χρήστης επιλέξει τις ρυθμίσεις του κουίζ, το κινητό στέλνει ένα HTTP GET αίτημα, ενσωματώνοντας το κρυπτογραφημένο JWT στο Header του αιτήματος. Το JWTAuthenticationFilter αναλαμβάνει να αναχαιτίσει την κλήση και να επικυρώσει την υπογραφή του token πριν επιτρέψει την πρόσβαση στο εσωτερικό επιχειρησιακό επίπεδο. Μόλις το αίτημα εγκριθεί, ο Controller προωθεί την κλήση στο JpaRepository. Το αποθετήριο εκτελεί τα προσαρμοσμένα ερωτήματα SQL (μέσω του annotation @Query) στην PostgreSQL για να αντλήσει τις κατάλληλες ερωτήσεις. Το αποτέλεσμα επιστρέφει ως λίστα αντικειμένων, μετατρέπεται αυτόματα σε μορφότυπο JSON και καταλήγει στο Android Client. Εκεί, τα δεδομένα αποσериοποιούνται και τροφοδοτούν τα Android UI components, ολοκληρώνοντας τον κύκλο.



Εικόνα 7: Διάγραμμα ροής για τον έλεγχο πρόσβασης και την ανάκτηση των ερωτήσεων του κουίζ από τη βάση δεδομένων

3. Διαδικασία Υποβολής Σκορ και Ενημέρωσης Πόντων Εμπειρίας (XP)

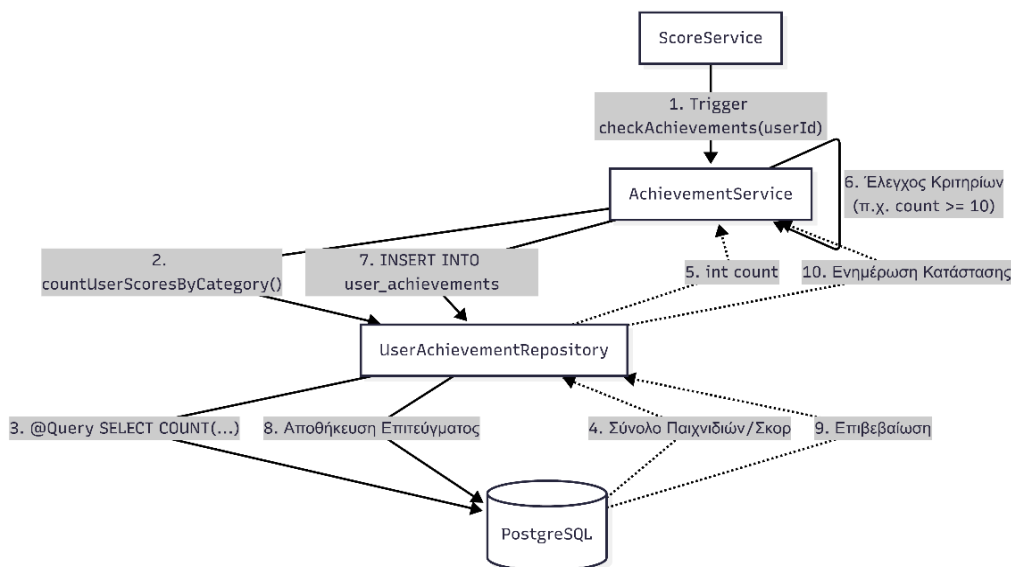
Το τρίτο διάγραμμα απεικονίζει τον μηχανισμό καταγραφής των επιδόσεων μετά την ολοκλήρωση ενός παιχνιδιού. Η συσκευή του χρήστη αποστέλλει ένα αίτημα POST που περιέχει τους πόντους που συγκεντρώθηκαν και τη θεματική κατηγορία του κουίζ. Ο ScoreController παραλαμβάνει τα δεδομένα και τα προωθεί στο ScoreService. Σε αυτό το επίπεδο, εφαρμόζεται η επιχειρησιακή λογική: η υπηρεσία ανακτά το τρέχον προφίλ του παίκτη μέσω του AppUserRepository, αθροίζει τους νέους πόντους στους ήδη υπάρχοντες συνολικούς XP του χρήστη, και δημιουργεί μια νέα εγγραφή βαθμολογίας. Μέσω του ScoreRepository, τα δεδομένα αποθηκεύονται οριστικά στην PostgreSQL. Το back-end επιστρέφει κωδικό επιτυχίας 201, επιβεβαιώνοντας στο Android Client ότι ο Παγκόσμιος Πίνακας Κατάταξης έχει ενημερωθεί.



Εικόνα 8: Διάγραμμα ροής για την υποβολή της τελικής βαθμολογίας ενός κουίζ και την ενημέρωση των XP του χρήστη

4. Διαδικασία Ελέγχου και Απόδοσης Ψηφιακών Επιτευγμάτων (Achievements)

Αυτό το διάγραμμα εστιάζει αποκλειστικά σε εσωτερικές διεργασίες του εξυπηρετητή. Αμέσως μετά την επιτυχή αποθήκευση ενός σκορ, το ScoreService ενεργοποιεί ασύγχρονα το AchievementService. Αυτή η υπηρεσία είναι υπεύθυνη για την αξιολόγηση των κριτηρίων παιχνιδοποίησης. Ζητά από το UserAchievementRepository να υπολογίσει –μέσω προσαρμοσμένων ερωτημάτων SQL– τα στατιστικά του χρήστη (π.χ. πόσα παιχνίδια Ιστορίας έχει παίξει). Εάν διαπιστωθεί ότι πληρούται ένα κριτήριο (όπως η ολοκλήρωση 10 παιχνιδιών), το σύστημα δημιουργεί μια νέα συσχέτιση μεταξύ του AppUser και του αντίστοιχου επιτεύγματος, αποθηκεύοντάς τη στον πίνακα user_achievements της βάσης δεδομένων.



Εικόνα 9: Διάγραμμα ροής του Back-end για τον έλεγχο και την απόδοση νέων Achievements στον χρήστη

Annotations

Η ανάπτυξη εφαρμογών με το Spring Boot βασίζεται σε μεγάλο βαθμό στη χρήση Annotations, τα οποία λειτουργούν ως μεταδεδομένα που καθοδηγούν το framework στην αυτοματοποίηση των ρυθμίσεων. Χάρη σε αυτά, αποφεύγεται η ανάγκη για εκτενείς ρυθμίσεις σε αρχεία XML και μειώνεται δραστικά ο επαναλαμβανόμενος κώδικας.

@SpringBootApplication: Αποτελεί το σημαντικότερο annotation της εφαρμογής και τοποθετείται αποκλειστικά στην κλάση QuizlyApplication. Πρόκειται για έναν συνδυασμό τριών άλλων προσδιορισμών:

- του @Configuration (για ορισμό ρυθμίσεων),
- του @EnableAutoConfiguration (για αυτόματη ρύθμιση των βιβλιοθηκών) και
- του @ComponentScan (για τον αυτόματο εντοπισμό όλων των components του project).

Με την εκκίνηση αυτής της κλάσης, το Spring Boot αρχικοποιεί ολόκληρο το οικοσύστημα της εφαρμογής.

Spring Web MVC Annotations

Αυτή η κατηγορία προσδιορισμών χρησιμοποιείται για τη διαμόρφωση του επιπέδου παρουσίασης και τη διαχείριση των REST API endpoints της εφαρμογής Quizly.

- @Controller & @RestController: Το @Controller ορίζει μια κλάση ως διαχειριστή αιτημάτων ιστού. Στο Quizly χρησιμοποιείται αποκλειστικά το @RestController, το οποίο αποτελεί εξειδίκευση του πρώτου και διασφαλίζει ότι η τιμή επιστροφής κάθε μεθόδου θα σειριοποιείται αυτόματα σε μορφή JSON, καθιστώντας το back-end κατάλληλο για την Android εφαρμογή.
- @ResponseBody: Λέει στο Spring να μετατρέψει την τιμή επιστροφής μιας μεθόδου απευθείας στο σώμα της απάντησης HTTP. Στο project μας, η χρήση του @RestController καθιστά αυτό το annotation προαιρετικό, καθώς εμπεριέχεται σε αυτό.
- @RequestBody: Είναι απαραίτητο για τη λήψη δεδομένων από την Android εφαρμογή. Συνδέει το σώμα (body) της αίτησης HTTP (σε μορφή JSON) με ένα αντικείμενο Java, όπως το AppUserRegistrationRequest κατά την εγγραφή ενός νέου παίκτη.
- @RequestMapping & Εξειδικευμένα Mapping Annotations: Το @RequestMapping καθορίζει το βασικό μονοπάτι (URL) ενός Controller. Για μεγαλύτερη σαφήνεια, χρησιμοποιούνται οι συντομευμένες μορφές του:
 - @GetMapping: Για την ανάκτηση ερωτήσεων ή σκορ.
 - @PostMapping: Για την υποβολή νέων απαντήσεων ή τη σύνδεση χρήστη.
 - @PutMapping: Για την ενημέρωση των στοιχείων του προφίλ.
 - @DeleteMapping: Για τη διαγραφή ενός λογαριασμού ή ιστορικού.
- @RequestParam & @PathVariable: Χρησιμοποιούνται για τη μεταφορά παραμέτρων στο back-end. Το @RequestParam ανακτά τιμές από το query string

του URL, ενώ το `@PathVariable` εξάγει τιμές απευθείας από τη διαδρομή του URL, όπως για παράδειγμα το ID ενός συγκεκριμένου χρήστη (`/api/v1/users/{userId}`).

Spring Annotations

Αυτοί οι προσδιορισμοί αφορούν τη διαχείριση των αντικειμένων (Beans) και την έγχυση εξαρτήσεων (Dependency Injection) εντός του Spring Container.

- `@Bean`: Χρησιμοποιείται για να δηλώσει μια μέθοδο που παράγει ένα αντικείμενο το οποίο θα διαχειρίζεται το Spring. Είναι ζωτικής σημασίας στις κλάσεις ρυθμίσεων ασφαλείας, όπως για τον ορισμό του `PasswordEncoder`.
- `@Component`: Είναι το βασικό annotation που υποδεικνύει στο framework ότι μια κλάση είναι ένα component και πρέπει να συμπεριληφθεί στο `Application Context`.
- `@Service`: Χαρακτηρίζει τις κλάσεις του Service Layer. Αν και λειτουργικά είναι παρόμοιο με το `@Component`, χρησιμοποιείται για να δηλώσει την πρόθεση της κλάσης ως φορέα της επιχειρησιακής λογικής του παιχνιδιού.
- `@Repository`: Χρησιμοποιείται στο επίπεδο πρόσβασης δεδομένων (DAO Layer). Εκτός από τη δήλωση της κλάσης ως component, το `@Repository` παρέχει αυτόματα μηχανισμούς μετάφρασης των εξαιρέσεων (exception translation) που μπορεί να προκύψουν από τη βάση δεδομένων PostgreSQL, διευκολύνοντας τον χειρισμό των σφαλμάτων SQL.

Στήσιμο front-end

Η ανάπτυξη της πλευράς του πελάτη (client-side) για την εφαρμογή Quizly πραγματοποιήθηκε στο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) Android Studio, το οποίο αποτελεί το επίσημο και πλέον αξιόπιστο εργαλείο της Google για τη δημιουργία εφαρμογών στο λειτουργικό σύστημα Android. Κεντρικός άξονας κατά τη φάση του σχεδιασμού και της υλοποίησης του front-end ήταν η παροχή μιας εξαιρετικά responsive, σταθερής και διαισθητικής εμπειρίας χρήστη (UserExperience - UX). Η διεπαφή έπρεπε να προσαρμόζεται δυναμικά και χωρίς αλλοιώσεις σε ένα ευρύ φάσμα συσκευών με διαφορετικές αναλύσεις, DPI και λόγους διαστάσεων οθόνης.

Για την επίτευξη μιας καθαρής και συντηρήσιμης αρχιτεκτονικής κώδικα, εφαρμόστηκε η θεμελιώδης αρχή του διαχωρισμού των ευθυνών (Separation of Concerns). Η οπτική αναπαράσταση και η διάταξη των στοιχείων της διεπαφής χρήστη (User Interface - UI) περιγράφονται πλήρως μέσα από δηλωτικά αρχεία δομημένης γλώσσας XML. Αντίθετα, η επιχειρησιακή λογική της διεπαφής, η διαχείριση της ροής της εφαρμογής, ο έλεγχος των αλληλεπιδράσεων και η ασύγχρονη επικοινωνία με το RESTful API του back-end πραγματοποιούνται μέσω του προγραμματιστικού κώδικα της εφαρμογής.

Η διαχείριση του κύκλου ζωής των βιβλιοθηκών και η αυτοματοποίηση της διαδικασίας μεταγλώπτισης και δημιουργίας του τελικού αρχείου εγκατάστασης (APK) ανατέθηκαν στο σύστημα δόμησης Gradle. Μέσω του αρχείου ρυθμίσεων build.gradle, ενσωματώθηκαν και παραμετροποιήθηκαν στρατηγικά οι απαραίτητες εξωτερικές εξαρτήσεις (dependencies). Αυτές περιλαμβάνουν τη βιβλιοθήκη Retrofit2 για τη διαχείριση των δικτυακών κλήσεων HTTP, τη βιβλιοθήκη Gson για την αυτόματη σειριοποίηση και αποσειριοποίηση των δεδομένων JSON, καθώς και τη βιβλιοθήκη Material Components για την υιοθέτηση σύγχρονων σχεδιαστικών προτύπων.

Android UI Components

Η αρχιτεκτονική του γραφικού περιβάλλοντος στο λειτουργικό σύστημα Android βασίζεται σε μια αυστηρά ιεραρχική δομή αντικειμένων, η οποία οργανώνεται ως ένα δέντρο από οπτικά στοιχεία. Η βάση αυτής της ιεραρχίας αποτελείται από δύο θεμελιώδεις κλάσεις του Android SDK: την κλάση View και την κλάση ViewGroup.

- **View:** Αντιπροσωπεύει το βασικό δομικό στοιχείο με το οποίο αλληλεπιδρά ο χρήστης ή από το οποίο αντλεί πληροφορίες. Πρόκειται για ορθογώνιες περιοχές της οθόνης που αναλαμβάνουν τη σχεδίαση των γραφικών στοιχείων και τη διαχείριση των events. Στο Quizly, τυπικά παραδείγματα αποτελούν τα TextView για την προβολή των κειμένων των ερωτήσεων, τα EditText για την εισαγωγή των διαπιστευτηρίων κατά τη σύνδεση, και τα Button ή MaterialButton για την επιλογή των απαντήσεων.
- **ViewGroup:** Αποτελεί μια εξειδικευμένη κλάση View η οποία λειτουργεί ως ένας αόρατος container άλλων Views ή ViewGroups. Η κύρια ευθύνη του είναι ο καθορισμός της γεωμετρικής διάταξης, της στοίχισης και της κατανομής του διαθέσιμου χώρου στα θυγατρικά στοιχεία.

Ο σχεδιασμός των οθονών βασίστηκε κατά κύριο λόγο στο ConstraintLayout. Η επιλογή αυτή έγινε για λόγους βελτιστοποίησης της απόδοσης, καθώς το ConstraintLayout επιτρέπει τη δημιουργία σύνθετων και ευέλικτων διατάξεων με επίπεδη δομή (flat view hierarchy), εξαλείφοντας την ανάγκη για πολλαπλές, εμφωλευμένες (nested) διατάξεις (όπως LinearLayout ή RelativeLayout). Με τον τρόπο αυτό, μειώνεται δραστικά ο χρόνος υπολογισμού των διαστάσεων και της σχεδίασης από τον επεξεργαστή γραφικών της συσκευής, αποτρέποντας τις καθυστερήσεις στην ανανέωση των καρτέ.

Όλα τα οπτικά στοιχεία σχεδιάστηκαν ακολουθώντας τις επίσημες προδιαγραφές του Material Design 3, ενσωματώνοντας σύγχρονες χρωματικές παλέτες, καθαρές γραμματοσειρές, στρογγυλεμένες γωνίες και διακριτικές σκίες που προσδίδουν αίσθηση βάθους.

Bottom Navigation Bar και Fragments

Για την υλοποίηση ενός σύγχρονου και εύχρηστου συστήματος πλοήγησης, επιλέχθηκε η υιοθέτηση της αρχιτεκτονικής Single-Activity Architecture. Σύμφωνα με αυτό το πρότυπο σχεδίασης, η εφαρμογή δεν εκκινεί μια νέα αυτόνομη Δραστηριότητα (Activity) για κάθε διαφορετική οθόνη, καθώς κάτι τέτοιο θα απαιτούσε σημαντικούς πόρους από το σύστημα και θα προκαλούσε οπτικές διακοπές κατά τη μετάβαση. Αντίθετα, χρησιμοποιείται μία κεντρική MainActivity, η οποία φιλοξενεί έναν κεντρικό FragmentContainerView που λειτουργεί ως δυναμικός καμβάς.

Πάνω σε αυτόν τον καμβά εναλλάσσονται τα Fragments. Ένα Fragment αντιπροσωπεύει μια αυτόνομη, modular ενότητα της διεπαφής χρήστη, η οποία διαθέτει τον δικό της ξεχωριστό κύκλο ζωής (lifecyle) αλλά είναι υποχρεωτικό να φιλοξενείται από μια Activity. Για τη διαχείριση της εναλλαγής τους, χρησιμοποιείται ο FragmentManager του Android framework, ο οποίος εκτελεί ασύγχρονες συναλλαγές για την απόκρυψη, την αντικατάσταση ή την προσθήκη στοιχείων στην οθόνη.

Η πλοήγηση μεταξύ των κύριων λειτουργικών ενοτήτων του Quizly ελέγχεται από ένα Bottom Navigation Bar τοποθετημένο στο κάτω μέρος της οθόνης. Το component αυτό περιλαμβάνει διακριτά εικονίδια και ετικέτες κειμένου για τις βασικές σελίδες: την Αρχική Οθόνη (HomeFragment), τον Παγκόσμιο Πίνακα Κατάταξης (LeaderboardFragment) και το Προφίλ του Χρήστη (ProfileFragment).

Όταν ο χρήστης πιάξει ένα στοιχείο της μπάρας πλοήγησης, πυροδοτείται ένας listener (OnItemSelectedListener), ο οποίος δίνει εντολή στον FragmentManager να αντικαταστήσει το τρέχον Fragment με το επιλεγμένο. Η προσέγγιση αυτή εξασφαλίζει εξαιρετικά γρήγορες μεταβάσεις, καθώς τα Fragment είναι πολύ πιο ελαφριά αντικείμενα στη μνήμη σε σχέση με τα Activity, επιτρέποντας παράλληλα τη διατήρηση της κατάστασης της κεντρικής δραστηριότητας χωρίς απώλεια δεδομένων.

RecyclerView

Η προβολή δυναμικών δεδομένων που οργανώνονται σε εκτενείς λίστες - όπως οι εκατοντάδες βαθμολογίες των παικτών στον πίνακα κατάταξης ή η συλλογή των ξεκλειδωμένων και κλειδωμένων ψηφιακών μεταλλίων (UserAchievements) - αποτελεί μια απαιτητική διαδικασία για τους πόρους μιας κινητής συσκευής. Για την αντιμετώπιση αυτού του προβλήματος, ενσωματώθηκε το προηγμένο στοιχείο RecyclerView.

Το RecyclerView αντικαθιστά τα παλαιότερα και μη αποδοτικά στοιχεία, όπως το ListView, εισάγοντας έναν εξελιγμένο μηχανισμό ανακύκλωσης και επαναχρησιμοποίησης των οπτικών στοιχείων. Η θεμελιώδης διαφορά του έγκειται στο σχεδιαστικό μοτίβο ViewHolder.

Όταν μια λίστα περιλαμβάνει μεγάλο αριθμό εγγραφών, το RecyclerView δεν δημιουργεί ένα ξεχωριστό αντικείμενο View για κάθε εγγραφή. Αντίθετα, δημιουργεί μόνο έναν περιορισμένο αριθμό Views, ο οποίος επαρκεί για να καλυφθεί η ορατή περιοχή της οθόνης, συν ελάχιστα επιπλέον στοιχεία για λόγους buffer. Καθώς ο χρήστης εκτελεί μια κίνηση κύλισης (scroll) και μια εγγραφή βγαίνει εκτός του ορατού πεδίου, το αντίστοιχο View δεν καταστρέφεται. Αντίθετα, το αντικείμενο ViewHolder διατηρεί την αναφορά (reference) στα γραφικά στοιχεία της διάταξης και το View «ανακυκλώνεται», παίρνοντας τη θέση του στο κάτω ή στο πάνω μέρος της λίστας για να εμφανίσει τα δεδομένα της επόμενης εγγραφής που εισέρχεται στην οθόνη.

Η λειτουργία του RecyclerView βασίζεται στη στενή συνεργασία τριών βασικών συνιστωσών:

1. **LayoutManager:** Καθορίζει τον τρόπο διάταξης των στοιχείων (π.χ. LinearLayoutManager για κάθετη ή οριζόντια λίστα, ή GridLayoutManager για πλέγμα).
2. **Adapter:** Αποτελεί τη γέφυρα μεταξύ της πηγής δεδομένων (π.χ. μιας λίστας από αντικείμενα ScoreDTO που ελήφθησαν από το backend) και του RecyclerView. Είναι υπεύθυνος για τη δημιουργία των ViewHolders (onCreateViewHolder) και για τη σύνδεση των δεδομένων με τα οπτικά στοιχεία (onBindViewHolder).
3. **ViewHolder:** Διατηρεί τις αναφορές για τα επιμέρους sub-views (όπως TextView για το username και TextView για το score), αποφεύγοντας τη συνεχή και κοστοβόρα κλήση της μεθόδου findViewById().

Με αυτή την αρχιτεκτονική, η εφαρμογή Quizly επιτυγχάνει ακαριαία απόκριση και απολύτως ομαλή κύλιση (60 frames per second), ελαχιστοποιώντας την κατανάλωση της κεντρικής μνήμης (RAM) και της μπαταρίας της συσκευής.

Tab layout και ViewPager

Σε συγκεκριμένες οθόνες της εφαρμογής, όπου η πληροφορία έπρεπε να οργανωθεί σε διακριτές υποκατηγορίες χωρίς όμως να διασπάται η ενότητα της τρέχουσας σελίδας, χρησιμοποιήθηκε ο συνδυασμός των στοιχείων TabLayout και ViewPager2. Η συγκεκριμένη υλοποίηση προσφέρει ένα εξαιρετικά δημοφιλές και λειτουργικό μοτίβο πλοήγησης, επιτρέποντας στον χρήστη να μεταβαίνει μεταξύ διαφορετικών προβολών δεδομένων είτε πιέζοντας τις ορατές οριζόντιες καρτέλες (tabs) στο επάνω μέρος, είτε εκτελώντας μια φυσική χειρονομία ολίσθησης προς τα αριστερά ή τα δεξιά.

- ViewPager2: Αποτελεί ένα σύγχρονο widget του Android Jetpack, το οποίο βασίζεται εσωτερικά στην αρχιτεκτονική του RecyclerView. Είναι υπεύθυνο για τον χειρισμό των χειρονομιών ολίσθησης και για την εκτέλεση animations κατά την αλλαγή των σελίδων. Η διαχείριση των περιεχομένων του γίνεται μέσω ενός εξειδικευμένου Adapter, του FragmentStateAdapter. Ο Adapter αυτός αναλαμβάνει να δημιουργεί, να καταστρέφει ή να αποθηκεύει την κατάσταση των Fragments που αντιστοιχούν σε κάθε σελίδα, διασφαλίζοντας ότι μόνο τα απαραίτητα Fragments παραμένουν ενεργά στη μνήμη.
- TabLayout: Λειτουργεί ως ο οπτικός οδηγός του χρήστη. Τοποθετείται συνήθως στο επάνω μέρος της οθόνης και εμφανίζει τις διαθέσιμες κατηγορίες.

Η σύνδεση και ο συγχρονισμός μεταξύ των δύο αυτών στοιχείων επιτυγχάνεται μέσω της κλάσης TabLayoutMediator. Ο Mediator αναλαμβάνει να ευθυγραμμίσει τη συμπεριφορά τους: όταν ο χρήστης κάνει swipe στο ViewPager2, το TabLayout μετακινεί αυτόματα τον οπτικό δείκτη στην αντίστοιχη καρτέλα. Αντίστροφα, όταν ο χρήστης επιλέγει μια καρτέλα, το ViewPager2 κυλάει αυτόματα στη σωστή σελίδα.

Στην εφαρμογή Quizly, η χρησιμότητα αυτού του σχήματος είναι εμφανής στην οθόνη των στατιστικών και των βαθμολογιών. Επιτρέπει τον διαχωρισμό του περιεχομένου σε διακριτές θεματικές ενότητες, όπως για παράδειγμα «Ιστορικό Προσπαθειών» και «Αναλυτικά Στατιστικά ανά Κατηγορία», προσφέροντας μια απόλυτα συμπαγή, καθαρή και οργανωμένη δομή χωρίς την ανάγκη φορτώματος νέων οθονών.

Εγχειρίδιο χρήσης (για το χρήστη)

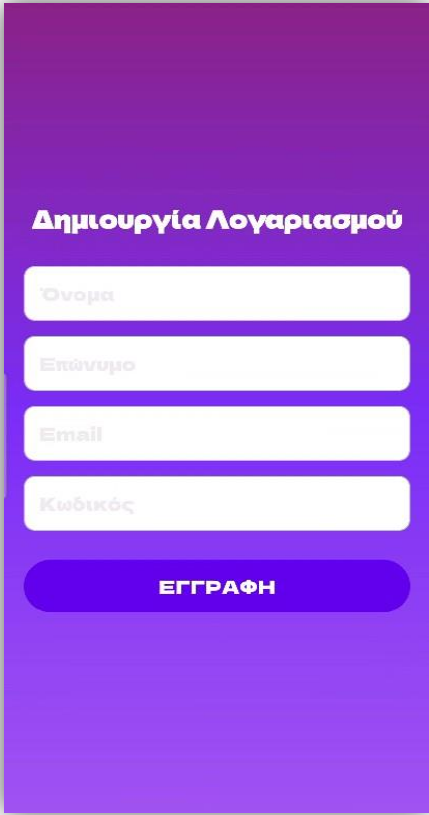
Το παρόν κεφάλαιο λειτουργεί ως ένας ολοκληρωμένος και αναλυτικός οδηγός περιήγησης, σχεδιασμένος για να αποτυπώσει πλήρως την εμπειρία του τελικού χρήστη (User Experience - UX) κατά την αλληλεπίδρασή του με το γραφικό περιβάλλον (UI) της εφαρμογής Quizly. Στόχος είναι η λεπτομερής χαρτογράφηση της διαδρομής του χρήστη, από την αρχική εγκατάσταση και ταυτοποίηση, έως την πλήρη αξιοποίηση των μηχανισμών gamification.

Κατά τον σχεδιασμό της εφαρμογής, δόθηκε τεράστια έμφαση στη δημιουργία μιας διαισθητικής διεπαφής (intuitive UI) η οποία ελαχιστοποιεί το γνωστικό φορτίο του παίκτη. Αυτό επιτεύχθηκε μέσω της συνεπούς χρήσης οπτικών ενδείξεων, σαφών τυπογραφικών ιεραρχιών και άμεσης ανατροφοδότησης (feedback) σε κάθε ενέργεια. Το εγχειρίδιο χωρίζεται στις βασικές λειτουργικές ενότητες, αναλύοντας τα οπτικά στοιχεία και τη ροή της πληροφορίας.

Εγγραφή και Σύνδεση (Authentication & Onboarding)

Η πρώτη επαφή του χρήστη με την πλατφόρμα του Quizly πραγματοποιείται μέσω της οθόνης ταυτοποίησης. Καθώς το οικοσύστημα του παιχνιδιού βασίζεται στην εξατομικευμένη παρακολούθηση της προόδου, η δημιουργία ενός ατομικού προφίλ αποτελεί αυστηρό προαπαιτούμενο. Η διεπαφή σε αυτό το στάδιο είναι μινιμαλιστική, επιτρέποντας στον χρήστη να εστιάσει αποκλειστικά στην εισαγωγή των διαπιστευτηρίων του.

1. **Διαδικασία Εγγραφής (Sign Up):** Εάν ο χρήστης εκκινεί την εφαρμογή για πρώτη φορά, κατευθύνεται στην οθόνη δημιουργίας λογαριασμού. Εκεί συναντά μια καθαρή φόρμα εισαγωγής δεδομένων, αποτελούμενη από πεδία κειμένου (EditText ή TextInputLayout κατά τα πρότυπα του Material Design). Η φόρμα απαιτεί τη συμπλήρωση του ονόματος, του επωνύμου, μιας διεύθυνσης ηλεκτρονικού ταχυδρομείου (email) και ενός κωδικού πρόσβασης.
 - ο **Οπτική Ανατροφοδότηση και Επικύρωση:** Κατά την πληκτρολόγηση, η εφαρμογή εκτελεί τοπικούς ελέγχους. Εάν ο χρήστης παραλείψει ένα πεδίο ή εισάγει μη έγκυρο email (χωρίς το σύμβολο '@'), το περίγραμμα του πεδίου χρωματίζεται κόκκινο και εμφανίζεται ένα διακριτικό μήνυμα σφάλματος ακριβώς κάτω από τη γραμμή εισαγωγής.
 - ο Με την επιτυχή συμπλήρωση και την πίεση του κουμπιού «Εγγραφή», ένα κυκλικό animation φόρτωσης υποδεικνύει την επικοινωνία με το back-end, διατηρώντας τον χρήστη ενήμερο.



Εικόνα 10: Sign Up Screen

2. **Διαδικασία Σύνδεσης (Login):** Για τους ήδη εγγεγραμμένους χρήστες, η οθόνη Login απαιτεί μόνο το email και τον κωδικό πρόσβασης. Για την ενίσχυση της χρηστικότητας, το πεδίο του κωδικού συνοδεύεται από ένα εικονίδιο (toggle), το οποίο επιτρέπει την προσωρινή εμφάνιση του κωδικού για την αποφυγή τυπογραφικών λαθών.
- ο **Διαχείριση Σφαλμάτων Σύνδεσης:** Εάν τα διαπιστευτήρια είναι λανθασμένα, η εφαρμογή αναχαιτίζει το σφάλμα (status code 401) από τον server και προβάλλει ένα αναδυόμενο μήνυμα τύπου Snackbar στο κάτω μέρος της οθόνης, ενημερώνοντας τον χρήστη για το πρόβλημα χωρίς να διακόπτει τη ροή του. Εφόσον η είσοδος είναι επιτυχής, η μετάβαση στην κεντρική οθόνη γίνεται με ένα ομαλό animation (fade-in). Το εκδοθέν JWT αποθηκεύεται κρυπτογραφημένα, εξασφαλίζοντας ότι στις επόμενες εκκινήσεις της εφαρμογής ο χρήστης θα συνδέεται αυτόματα.



Εικόνα 11: Login In screen

Ενότητα 1: Επιλογή Κατηγορίας και Διεξαγωγή Κουίζ

Αυτή η ενότητα αποτελεί τον πυρήνα της διαδραστικότητας του Quizly. Μετά την επιτυχή ταυτοποίηση, ο παίκτης μεταφέρεται στην Αρχική Οθόνη (HomeFragment), η οποία λειτουργεί ως το κέντρο ελέγχου (dashboard) του παιχνιδιού.

- Εξερεύνηση Θεματικών Ενοτήτων: Το κεντρικό τμήμα της οθόνης καταλαμβάνεται από μια δυναμική λίστα ή ένα πλέγμα (GridLayout) με τις διαθέσιμες κατηγορίες ερωτήσεων (π.χ. Ιστορία, Τεχνολογία, Αθλητισμός, Κινηματογράφος). Κάθε κατηγορία αναπαρίσταται από μια καλοσχεδιασμένη κάρτα (MaterialCardView) που περιλαμβάνει έναν χαρακτηριστικό τίτλο, ένα σχετικό εικονίδιο και ίσως μια ένδειξη του επιπέδου δυσκολίας (Εύκολο, Μεσαίο, Δύσκολο). Ο χρήστης μπορεί να περιηγηθεί οπτικά σε αυτές, κάνοντας κάθετη κύλιση.



Εικόνα 12: Main Screen



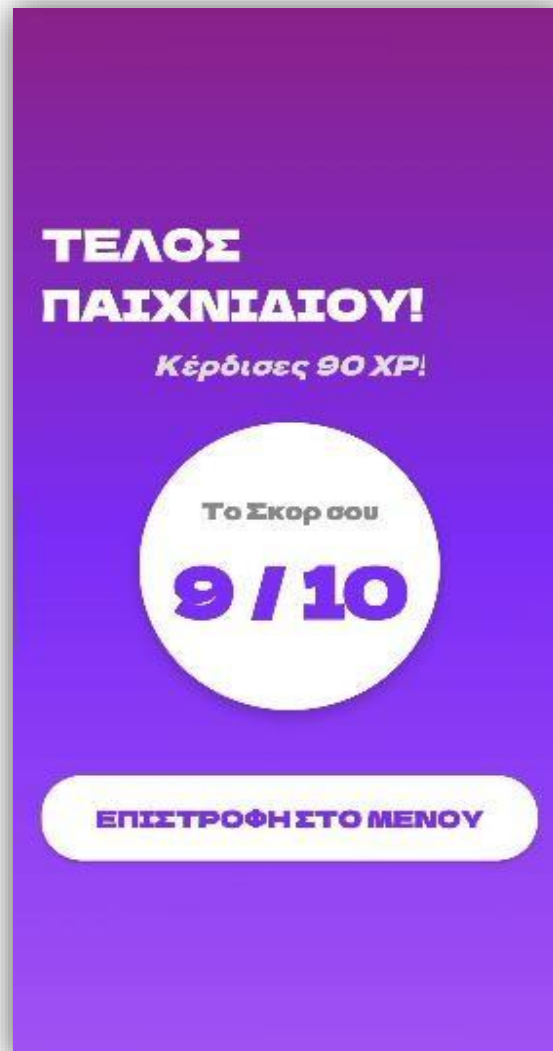
Εικόνα 13: Quiz Settings Screen

- Η Εμπειρία του Παιχνιδιού (In-Game UI): Επιλέγοντας μια κατηγορία, ο χρήστης μεταβαίνει στο καθαυτό περιβάλλον του κουίζ. Ο σχεδιασμός της οθόνης του παιχνιδιού είναι αυστηρά εστιασμένος στην ερώτηση, αφαιρώντας κάθε περιττό στοιχείο που θα μπορούσε να αποσπάσει την προσοχή.
 - Στο επάνω μέρος δεσπόζει η εκφώνηση της ερώτησης με έντονη, ευανάγνωστη γραμματοσειρά. Ακριβώς από πάνω, μια μπάρα προόδου γεμίζει σταδιακά, υποδεικνύοντας στον παίκτη σε ποια ερώτηση του γύρου βρίσκεται (π.χ. Ερώτηση 3 από 10).
 - Στο κέντρο, οι τέσσερις πιθανές απαντήσεις παρουσιάζονται ως μεγάλα, ορθογώνια κουμπιά, εξασφαλίζοντας εύκολη αλληλεπίδραση ακόμα και σε συσκευές με μικρή οθόνη.
- Άμεση Οπτική Ανατροφοδότηση: Μόλις ο χρήστης επιλέξει μια απάντηση, η εφαρμογή αντιδρά ακαριαία. Εάν η απάντηση είναι σωστή, το κουμπί χρωματίζεται με πράσινο χρώμα, επιβεβαιώνοντας την επιτυχία. Εάν είναι λανθασμένη, το επιλεγμένο κουμπί γίνεται κόκκινο, ενώ ταυτόχρονα το σύστημα φωτίζει με πράσινο χρώμα τη σωστή επιλογή, παρέχοντας εκπαιδευτική αξία στον παίκτη.
- Σύνοψη και Υπολογισμός Σκορ: Μετά την ολοκλήρωση της τελευταίας ερώτησης, ο παίκτης οδηγείται στην οθόνη αποτελεσμάτων (ResultsFragment). Εδώ, ένα εντυπωσιακό



Εικόνα 14: Quiz screen

UI τον ενημερώνει για τις επιδόσεις του: εμφανίζεται το συνολικό σκορ, ο αριθμός των εύστοχων απαντήσεων και οι Πόντοι Εμπειρίας (XP) που προστέθηκαν στο προφίλ του. Από αυτή την οθόνη, παρέχονται κουμπιά για άμεση επανάληψη του κουίζ («Play Again») ή επιστροφή στο κεντρικό μενού.



Εικόνα 15: Results Screen

Ενότητα 2: Παγκόσμιος Πίνακας Κατάταξης (Leaderboard)

Μέσω της μπάρας πλοήγησης στο κάτω μέρος της οθόνης (Bottom Navigation Bar), ο χρήστης μπορεί ανά πάσα στιγμή να μεταβεί στον Παγκόσμιο Πίνακα Κατάταξης (LeaderboardFragment). Αυτή η οθόνη αποτελεί τον βασικό άξονα του κοινωνικού ανταγωνισμού εντός του Quizly.

- **Δομή και Ιεραρχία:** Η οθόνη προβάλλει τους κορυφαίους παίκτες παγκοσμίως, ιεραρχημένους βάσει της συνολικής τους βαθμολογίας.
- **Πλοήγηση στα Δεδομένα:** Η τεράστια λίστα των χρηστών διαχειρίζεται μέσω ενός αποδοτικού RecyclerView. Ο χρήστης μπορεί να κάνει scroll προς τα κάτω για να εξερευνήσει τη λίστα, με τα δεδομένα να φορτώνουν ομαλά χωρίς να προκαλούν «πάγωμα» στην οθόνη.
- **Εντοπισμός Προσωπικής Θέσης:** Για να μην χάνεται ο παίκτης μέσα στο πλήθος των ονομάτων, η εγγραφή που αφορά τον δικό του λογαριασμό εντοπίζεται αυτόματα από το σύστημα και είτε χρωματίζεται με διαφορετικό φόντο (π.χ. διακριτικό μπλε ή κίτρινο), είτε "καρφιτσώνεται" στο κάτω μέρος της οθόνης, ώστε να γνωρίζει ανά πάσα στιγμή την απόστασή του από την κορυφή.



Εικόνα 16: Leaderboard Screen

Ενότητα 3: Στατιστικά και Ψηφιακά Επιτεύγματα (Achievements)

Η παρακολούθηση της ατομικής προόδου αποτελεί τον βασικότερο μηχανισμό διατήρησης του ενδιαφέροντος στην εφαρμογή. Η ενότητα αυτή φιλοξενείται στο προφίλ του χρήστη και προσφέρει μια βαθιά ανάλυση της πορείας του, χωρισμένη σε δύο διακριτές οπτικές κάρτες (Tabs) χρησιμοποιώντας τον μηχανισμό του ViewPager2.

1. Αναλυτικά Στατιστικά: Στην πρώτη καρτέλα, ο παίκτης έρχεται σε επαφή με τα δεδομένα της απόδοσής του. Μέσω εύπεπτων οπτικών στοιχείων (όπως κυκλικές μπάρες προόδου ή γραφήματα), παρουσιάζεται μια ανάλυση των επιδόσεών του ανά επιμέρους κατηγορία και επίπεδο δυσκολίας καθώς και τα συνολικά XP του χρήστη. Αυτή η ανατροφοδότηση λειτουργεί ως εργαλείο αυτοβελτίωσης, επιτρέποντας στον χρήστη να κατανοήσει τα δυνατά και τα αδύναμα σημεία του.



Εικόνα 17: Statistics Screen

2. Το Σύστημα Επιτευγμάτων: Στη δεύτερη καρτέλα προβάλλεται η συλλογή των ψηφιακών εμβλημάτων του παίκτη. Κάθε επίτευγμα (π.χ. «Προθέρμανση» ή «Ο Χρονοταξιδιώτης») αναπαρίσταται ως μια ανεξάρτητη κάρτα.
- Για να ενισχυθεί το αίσθημα της εξερεύνησης και της προσμονής, τα επιτεύγματα που δεν έχουν κατακτηθεί ακόμα εμφανίζονται με γκρι ή ημιδιαφανή χρώματα (locked state).
 - Αντίθετα, όσα έχουν ξεκλειδωθεί (unlocked state) προβάλλονται με ζωντανά χρώματα. Εάν ο χρήστης πατήσει πάνω σε μια κάρτα επιτεύγματος, ένα AlertDialog ή BottomSheet μπορεί να αναδυθεί, εξηγώντας αναλυτικά τις προϋποθέσεις κατάκτησής του και το status του.

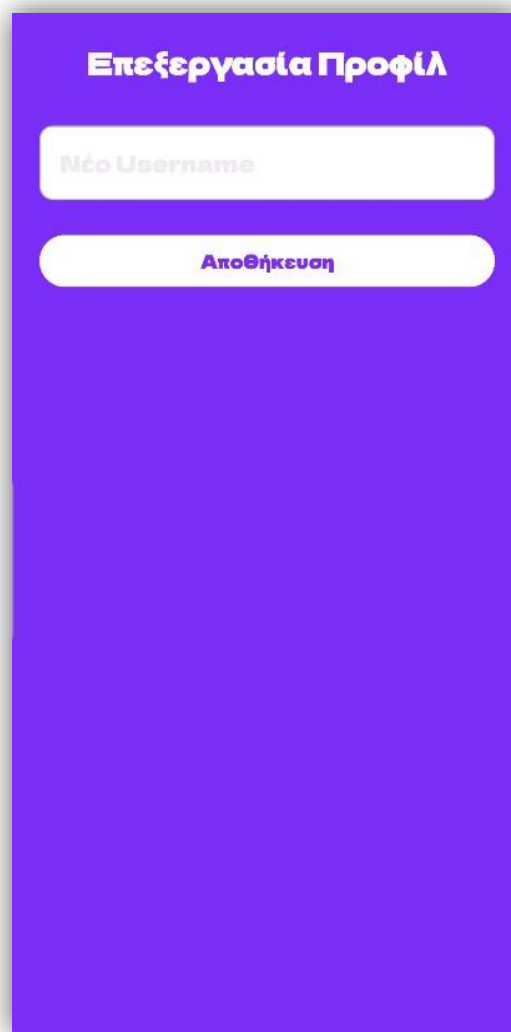


Εικόνα 18: Achievement Screen

Ενότητα 4: Διαχείριση του προφίλ

Η τελευταία ενότητα ολοκληρώνει την εμπειρία του χρήστη παρέχοντας τον απόλυτο έλεγχο πάνω στα προσωπικά του δεδομένα. Αυτή η λειτουργικότητα εδράζεται στο ProfileFragment, όπου η έμφαση δίνεται στην ασφάλεια και την προσωποποίηση.

- **Επεξεργασία Στοιχείων (Edit Profile):** Στην κορυφή της οθόνης εμφανίζεται ένα Avatar του χρήστη μαζί με τα βασικά του στοιχεία (Όνομα και Email). Μέσω ενός κουμπιού "Επεξεργασία", τα πεδία κειμένου ξεκλειδώνουν (από view-only μετατρέπονται σε editable). Ο χρήστης μπορεί να τροποποιήσει το όνομά του και να πατήσει αποθήκευση. Η εφαρμογή, σε αυτό το σημείο, εμφανίζει μια ένδειξη φόρτωσης και έπειτα από την επιτυχή επικοινωνία με τον server, εμφανίζει ένα Toast επιβεβαίωσης ("Τα στοιχεία ενημερώθηκαν επιτυχώς").
- **Μηχανισμός Αποσύνδεσης (Logout):** Δίπλα από το κουμπί για την επεξεργασία, βρίσκεται το κουμπί Αποσύνδεσης. Εάν ο χρήστης το πατήσει, η εφαρμογή διαγράφει με ασφάλεια το JWT από τη συσκευή, εκκαθαρίζει τη στοίβα πλοήγησης του Android, και τον επιστρέφει στην αρχική οθόνη εισόδου, κλειδώνοντας την πρόσβαση στα δεδομένα του παιχνιδιού.



Εικόνα 19: Edit Profile Screen

Εγχειρίδιο χρήσης (για το διαχειριστή)

Η εύρυθμη λειτουργία μιας δυναμικής πλατφόρμας όπως το Quizly δεν βασίζεται μόνο στην εμπειρία του τελικού παίκτη, αλλά και στα ισχυρά εργαλεία διαχείρισης που παρέχονται στο παρασκήνιο. Το εγχειρίδιο διαχειριστή αφορά τους χρήστες στους οποίους έχει αποδοθεί ο αναβαθμισμένος ρόλος `ROLE_ADMIN` ή `ROLE_SIMPLE_ADMIN` στη βάση δεδομένων. Μέσω αυτών των ρόλων, το σύστημα ασφαλείας "ξεκλειδώνει" την πρόσβαση σε προστατευμένα API endpoints, επιτρέποντας την πλήρη εποπτεία και τροποποίηση των δεδομένων του συστήματος.

Καθώς η παρούσα υλοποίηση εστιάζει στην αρχιτεκτονική του back-end και στο Android client, η διαχείριση της εφαρμογής δύναται να πραγματοποιηθεί μέσω κλήσεων στο REST API (χρησιμοποιώντας εργαλεία όπως το Postman ή το Swagger UI) ή μέσω μιας διακριτής διαχειριστικής οθόνης (Admin Dashboard) εντός της εφαρμογής.

Διαχείριση Ερωτήσεων (Questions Management)

Ο πυρήνας του παιχνιδιού είναι η δεξαμενή των ερωτήσεων. Ο διαχειριστής έχει την αποκλειστική εξουσία να εμπλουτίζει και να συντηρεί αυτή τη δεξαμενή μέσω των λειτουργιών CRUD (Create, Read, Update, Delete).

- Προσθήκη νέου περιεχομένου: Μέσω ενός HTTP POST αιτήματος στο endpoint `/api/v1/questions`, ο διαχειριστής μπορεί να εισάγει νέες ερωτήσεις. Το σώμα του αιτήματος (JSON payload) πρέπει να περιλαμβάνει το κείμενο της ερώτησης, τις τέσσερις πιθανές απαντήσεις, τον δείκτη της σωστής επιλογής (π.χ. `correctAnswerIndex`), την κατηγορία και τον βαθμό δυσκολίας. Το σύστημα επικυρώνει τα δεδομένα και τα αποθηκεύει στην PostgreSQL.
- Επεξεργασία και Διαγραφή: Σε περίπτωση που διαπιστωθεί κάποιο τυπογραφικό λάθος ή μια ερώτηση θεωρηθεί παρωχημένη, ο διαχειριστής μπορεί να εκτελέσει μια κλήση PUT για την ενημέρωσή της ή μια κλήση DELETE για την οριστική απομάκρυνσή της από τη βάση, αποτρέποντας την εμφάνισή της σε μελλοντικά κουίζ.

Εποπτεία και Διαχείριση Χρηστών (User Management)

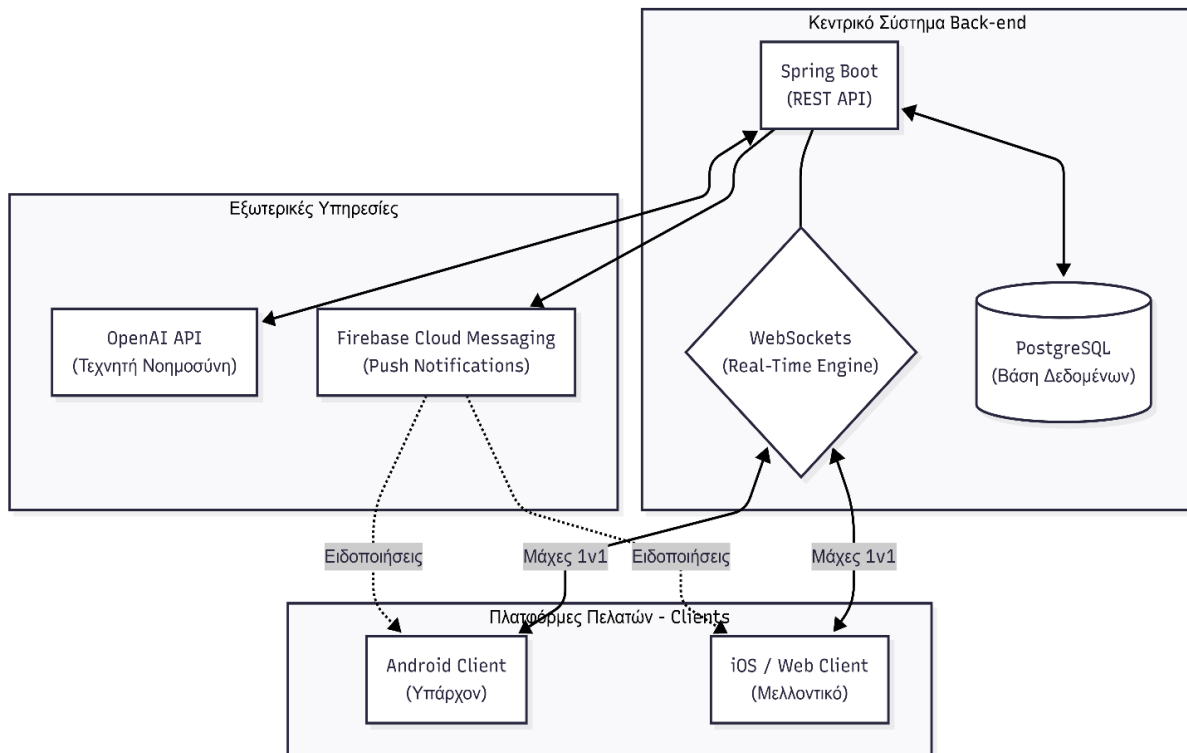
Η διατήρηση ενός υγιούς και δίκαιου ανταγωνιστικού περιβάλλοντος είναι ζωτικής σημασίας.

- Προβολή Στατιστικών Πλατφόρμας: Ο διαχειριστής έχει πρόσβαση σε endpoints που επιστρέφουν συγκεντρωτικά δεδομένα, όπως τον συνολικό αριθμό εγγεγραμμένων χρηστών, τα παιχνίδια που παίζονται καθημερινά και τον μέσο όρο βαθμολογίας ανά κατηγορία.
- Διαγραφή / Αποκλεισμός (Banning): Εάν εντοπιστεί κακόβουλη δραστηριότητα από έναν λογαριασμό (π.χ. προσπάθεια αλλοίωσης του σκορ μέσω αυτοματοποιημένων scripts), ο διαχειριστής έχει το δικαίωμα να αναστείλει τη λειτουργία του λογαριασμού ή να τον διαγράψει οριστικά (DELETE `/api/v1/users/{userId}`). Η ενέργεια αυτή καταστρέφει παράλληλα τα συνδεδεμένα JWT, αποκόπτοντας άμεσα τον χρήστη από το παιχνίδι.

Μελλοντικές επεκτάσεις της εφαρμογής

Ο σχεδιασμός και η υλοποίηση του Quizly πραγματοποιήθηκαν με γνώμονα την επεκτασιμότητα. Η αποσυνδεδεμένη αρχιτεκτονική μεταξύ του Spring Boot REST API και του Android Client αφήνει τεράστια περιθώρια για την προσθήκη νέων, καινοτόμων λειτουργιών. Ορισμένες από τις σημαντικότερες μελλοντικές βελτιώσεις περιλαμβάνουν:

1. Multiplayer / Real-Time Battles (Μάχες σε Πραγματικό Χρόνο): Η τρέχουσα έκδοση βασίζεται σε ασύγχρονο ανταγωνισμό (μέσω του Leaderboard). Στο μέλλον, η εφαρμογή μπορεί να ενσωματώσει την τεχνολογία WebSockets. Αυτό θα επιτρέπει τη δημιουργία «δωματίων» (lobbies) όπου δύο ή περισσότεροι παίκτες θα διαγωνίζονται ταυτόχρονα στις ίδιες ερωτήσεις, βλέποντας την πρόοδο του αντιπάλου τους σε πραγματικό χρόνο.
2. Παραγωγή Ερωτήσεων μέσω Τεχνητής Νοημοσύνης (AI Integration): Για να αποφευχθεί η εξάντληση της δεξαμενής των ερωτήσεων, το back-end θα μπορούσε να διασυνδεθεί με εξωτερικά Large Language Models (π.χ. OpenAI API). Μέσω κατάλληλων prompts, το σύστημα θα μπορεί να παράγει δυναμικά, αστείρευτες και μεταφρασμένες ερωτήσεις σε πραγματικό χρόνο, ταξινομημένες ανά δυσκολία.
3. Σύστημα Ειδοποιήσεων (Push Notifications): Με την ενσωμάτωση του Firebase Cloud Messaging (FCM), η εφαρμογή θα μπορεί να στέλνει στοχευμένες ειδοποιήσεις στις συσκευές των χρηστών. Για παράδειγμα, μια ειδοποίηση "Έχασες την 1η θέση στο Leaderboard! Μπες να παίξεις!" θα αύξανε δραματικά τη διατήρηση και το engagement των παικτών.
4. Cross-Platform: Δεδομένου ότι το API είναι ανεξάρτητο από την πλατφόρμα παρουσίασης, ένα επόμενο φυσικό βήμα είναι η ανάπτυξη ενός iOS Client (π.χ. με χρήση Swift) ή μιας Web εφαρμογής (π.χ. με React ή Angular), επιτρέποντας σε χρήστες διαφορετικών συστημάτων να παίζουν και να ανταγωνίζονται στην ίδια βάση δεδομένων.



Εικόνα 20: Πιθανή Μελλοντική αρχιτεκτονική του συστήματος

Συμπεράσματα

Η εκπόνηση της παρούσας πτυχιακής εργασίας και η ανάπτυξη της εφαρμογής Quizly αποτέλεσαν μια πολυεπίπεδη πρόκληση, η οποία συνδύασε με επιτυχία ετερογενείς τεχνολογίες σε ένα ενιαίο, λειτουργικό και ασφαλές προϊόν. Η μετάβαση από τη θεωρητική σχεδίαση στην πρακτική υλοποίηση ενός Full-Stack συστήματος ανέδειξε τη σημασία της σωστής αρχιτεκτονικής λογισμικού.

Σε επίπεδο Back-end, η επιλογή του Spring Boot αποδείχθηκε ιδανική. Επιβεβαιώθηκε η δύναμη της αρχιτεκτονικής N-Tier, η οποία επέτρεψε τον απόλυτο διαχωρισμό της επιχειρησιακής λογικής από τον χειρισμό των δεδομένων. Η χρήση του Spring Security σε συνδυασμό με τα JSON Web Tokens (JWT) κατέδειξε πως ένα σύστημα μπορεί να παραμένει εντελώς stateless, διασφαλίζοντας παράλληλα τα προσωπικά δεδομένα των χρηστών με σύγχρονες μεθόδους κρυπτογράφησης. Επιπλέον, η επικοινωνία με τη σχεσιακή βάση PostgreSQL μέσω του Spring Data JPA (Hibernate) εξάλειψε την ανάγκη συγγραφής πολύπλοκων ερωτημάτων SQL, προστατεύοντας την εφαρμογή από επιθέσεις τύπου SQL Injection.

Σε επίπεδο Front-end, η ανάπτυξη στο περιβάλλον του Android προσέφερε πολύτιμες γνώσεις γύρω από τη διαχείριση του κύκλου ζωής των οπτικών στοιχείων. Η υιοθέτηση της αρχιτεκτονικής Single-Activity με τη χρήση Fragments, σε συνδυασμό με το εξαιρετικά αποδοτικό component RecyclerView, απέδειξε ότι είναι απολύτως εφικτή η δημιουργία μιας ομαλής, απρόσκοπτης εμπειρίας χρήστη, ακόμη και όταν απαιτείται η διαχείριση μεγάλου όγκου δεδομένων από τον server. Η ενσωμάτωση της βιβλιοθήκης Retrofit2 διευκόλυνε δραστικά την κατανάλωση των REST APIs, αυτοματοποιώντας τη διαδικασία (de)serialization των JSON δεδομένων.

Πέρα από την τεχνική αρτιότητα, η υλοποίηση του Quizly πέτυχε απόλυτα τον αρχικό της στόχο: τη δημιουργία ενός ελκυστικού περιβάλλοντος μάθησης μέσω της παιχνιδοποίησης (gamification). Η απρόσκοπτη επικοινωνία μεταξύ του Android Client και του Spring Boot Back-end επέτρεψε την άμεση ανατροφοδότηση του χρήστη, στοιχείο κρίσιμο για τη διατήρηση της προσοχής του. Ο μηχανισμός υπολογισμού των πόντων εμπειρίας (XP) και η δυναμική ενημέρωση του Παγκόσμιου Πίνακα Κατάταξης (Leaderboard) λειτουργήσαν ακριβώς όπως σχεδιάστηκαν, προσφέροντας ισχυρά εξωτερικά κίνητρα στους παίκτες. Αποδείχθηκε στην πράξη ότι η ενσωμάτωση στοιχείων ανταγωνισμού σε ένα εκπαιδευτικό εργαλείο μπορεί να μετατρέψει μια τυπική διαδικασία αξιολόγησης γνώσεων σε μια ευχάριστη και εθιστική εμπειρία.

Κατά τη διάρκεια της ανάπτυξης, προέκυψαν σημαντικές αρχιτεκτονικές και προγραμματιστικές προκλήσεις, η επίλυση των οποίων προσέφερε πολύτιμη τεχνογνωσία. Μια από τις μεγαλύτερες προκλήσεις ήταν η διασφάλιση της ορθής διαχείρισης της κατάστασης στην Android εφαρμογή, ειδικά κατά την εναλλαγή των οθονών και τη διαχείριση του χρονόμετρου στις ερωτήσεις. Επιπλέον, ο συγχρονισμός των ασύγχρονων κλήσεων δικτύου μέσω της βιβλιοθήκης Retrofit απαιτούσε προσεκτικό χειρισμό των exceptions, ώστε η εφαρμογή να μην καταρρέει σε περιπτώσεις αστάθειας του δικτύου ή λήξης του JWT token. Η υλοποίηση του μηχανισμού Refresh Token ήταν κομβική, καθώς διασφάλισε ότι ο χρήστης παραμένει συνδεδεμένος χωρίς να διακόπτεται η εμπειρία του παιχνιδιού.

Εξίσου σημαντικό ρόλο στην επιτυχή ολοκλήρωση του συστήματος διαδραμάτισε η διαδικασία του ελέγχου και του debugging. Η μετάβαση από το περιβάλλον ανάπτυξης σε συνθήκες που προσομοιώνουν την παραγωγή ανέδειξε τη σημασία της σχολαστικής επαλήθευσης των δεδομένων. Η δημιουργία ανεξάρτητων API endpoints επέτρεψε τη μεμονωμένη δοκιμή της επιχειρησιακής λογικής του εξυπηρετητή (π.χ. μέσω εργαλείων όπως το Postman), πριν καν αυτά ενσωματωθούν στο γραφικό περιβάλλον του κινητού. Αυτός ο

διαχωρισμός απέτρεψε τη δημιουργία στενά συνδεδεμένου κώδικα.

Σε προσωπικό και ακαδημαϊκό επίπεδο, η εκπόνηση της συγκεκριμένης πτυχιακής εργασίας γεφύρωσε το χάσμα ανάμεσα στη θεωρητική ακαδημαϊκή γνώση και τις πραγματικές απαιτήσεις της βιομηχανίας λογισμικού. Η εξοικείωση με την ανάγνωση και κατανόηση εκτενών μηνυμάτων σφάλματος, η διαχείριση εξαρτήσεων μέσω του Maven και του Gradle, καθώς και η κατανόηση των αρχών σχεδίασης RESTful υπηρεσιών, αποτελούν εφόδια ανεκτίμητης αξίας. Το έργο αυτό ανέδειξε ότι η ανάπτυξη λογισμικού δεν περιορίζεται μόνο στη συγγραφή κώδικα, αλλά απαιτεί συνολική αρχιτεκτονική αντίληψη, σχεδιασμό εμπειρίας χρήστη και αυστηρή διαχείριση ασφάλειας.

Συμπερασματικά, το Quizly δεν αποτελεί απλώς μια ακαδημαϊκή άσκηση, αλλά έναν πλήρως παραγωγικό σκελετό. Οι τεχνολογίες, τα μοτίβα σχεδίασης (design patterns) και οι πρακτικές καθαρού κώδικα (clean code) που εφαρμόστηκαν, ανταποκρίνονται στα σύγχρονα βιομηχανικά πρότυπα, θέτοντας γερά θεμέλια για τη δημιουργία αξιόπιστων, scalable και εμπορικά βιώσιμων εφαρμογών λογισμικού.

Πίνακας Συντμήσεων – Αρκτικόλεξων – Ακρωνυμίων

Ακρωνύμιο / Σύντμηση	Πλήρης Όρος (Αγγλικά)	Επεξήγηση (Ελληνικά)
API	Application Programming Interface	Διεπαφή Προγραμματισμού Εφαρμογών
CRUD	Create, Read, Update, Delete	Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή (Βασικές λειτουργίες βάσεων δεδομένων)
DAO	Data Access Object	Αντικείμενο Πρόσβασης Δεδομένων
DTO	Data Transfer Object	Αντικείμενο Μεταφοράς Δεδομένων
FCM	Firebase Cloud Messaging	Υπηρεσία της Google για αποστολή ειδοποιήσεων (Push Notifications)
HTTP	Hypertext Transfer Protocol	Πρωτόκολλο Μεταφοράς Υπερκειμένου
IDE	Integrated Development Environment	Ολοκληρωμένο Περιβάλλον Ανάπτυξης (π.χ. Android Studio, IntelliJ IDEA)
JPA	Java Persistence API	Διεπαφή της Java για διαχείριση σχεσιακών δεδομένων
JSON	JavaScript Object Notation	Ελαφρύς μορφότυπος ανταλλαγής δεδομένων
JWT	JSON Web Token	Πρότυπο για την ασφαλή μετάδοση πληροφοριών μεταξύ μερών ως αντικείμενο JSON

ORM	Object-Relational Mapping	Αντιστοίχιση Αντικειμένων-Σχέσεων
RBAC	Role-Based Access Control	Έλεγχος Πρόσβασης Βάσει Ρόλων
RDBMS	Relational Database Management System	Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (π.χ. PostgreSQL)
REST	Representational State Transfer	Αρχιτεκτονικό στυλ για την ανάπτυξη διαδικτυακών υπηρεσιών
SQL	Structured Query Language	Δομημένη Γλώσσα Ερωτημάτων
UI	User Interface	Διεπαφή Χρήστη / Γραφικό Περιβάλλον
UX	User Experience	Εμπειρία Χρήστη
XP	Experience Points	Πόντοι Εμπειρίας (Μηχανισμός παιχνιδιού)

Πίνακας Ορολογίας

Όρος	Επεξήγηση
Back-end	Το τμήμα της εφαρμογής που εκτελείται στον εξυπηρετητή και δεν είναι άμεσα ορατό στον χρήστη. Διαχειρίζεται την επιχειρησιακή λογική, την ασφάλεια και την επικοινωνία με τη βάση δεδομένων.
Dependency Injection	Σχεδιαστικό πρότυπο όπου τα αντικείμενα δεν δημιουργούν τα ίδια τις εξαρτήσεις τους, αλλά τους παρέχονται από μια εξωτερική πηγή.
Endpoint	Η μοναδική κατάληξη ενός URL σε ένα RESTful API, μέσω της οποίας οι εφαρμογές μπορούν να επικοινωνήσουν με τον εξυπηρετητή για να ανακτήσουν ή να τροποποιήσουν δεδομένα.
Framework	Ένα πλαίσιο ανάπτυξης λογισμικού που παρέχει έτοιμες βιβλιοθήκες, εργαλεία και κανόνες αρχιτεκτονικής, διευκολύνοντας και επιταχύνοντας τη διαδικασία προγραμματισμού (π.χ. Spring Boot).
Front-end / Client	Το τμήμα της εφαρμογής με το οποίο αλληλεπιδρά άμεσα ο τελικός χρήστης. Στο συγκεκριμένο έργο, αφορά την εγγενή Android εφαρμογή.
Gamification	Η πρακτική της ενσωμάτωσης στοιχείων και μηχανισμών παιχνιδιού σε περιβάλλοντα που δεν αποτελούν παιχνίδια, με σκοπό την αύξηση της συμμετοχής και της παρακίνησης του χρήστη.
Leaderboard	Ο Παγκόσμιος Πίνακας Κατάταξης, ο οποίος εμφανίζει δυναμικά τη βαθμολογία των παικτών της πλατφόρμας σε φθίνουσα σειρά, ενισχύοντας την υγιή ανταγωνιστικότητα.
Micro-learning	Μια εκπαιδευτική προσέγγιση σύμφωνα με την οποία η πληροφορία και η μάθηση παρέχονται σε σύντομες, εξαιρετικά εστιασμένες και εύπεπτες ενότητες.
N-Tier Architecture	Αρχιτεκτονική η οποία κατανέμει τις λειτουργίες ενός συστήματος λογισμικού σε ξεχωριστά και ανεξάρτητα επίπεδα.
Stateless	Αρχιτεκτονική δικτύου όπου ο εξυπηρετητής δεν διατηρεί στη μνήμη του καμία πληροφορία για τον πελάτη μεταξύ των αιτημάτων. Κάθε αίτημα πρέπει να περιέχει όλες τις απαραίτητες πληροφορίες για την εξυπηρέτησή του.

Βιβλιογραφία

1. Politou, Eugenia, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. 2022. "Privacy and Data Protection Challenges in the Distributed Era." Springer.
2. Sigala, Effrosyni, Efthimios Alepis, and Constantinos Patsakis. 2020. "Measuring the Quality of Street Surfaces in Smart Cities Through Smartphone Crowdsensing." In 2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA), 1–8. IEEE.
3. Alonistioti, Nancy, Evangelia Aikaterini Tsichrintzi, Konstantina Chrysafiadi, and Efthimios Alepis. 2023. "Requirements for Fuzzy Logic in Personalisation of Fire Emergency Alerts." In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–8. IEEE.
4. Douladiris, Anargyros, and Efthimios Alepis. 2023. "Covid-19 New Cases Correlation Analysis: Weather Conditions, Citizen Traffic and Vaccination Statistics Impact in NARX Estimated Regressions in Attica, Greece." In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–7. IEEE.
5. Virvou, Maria, Efthimios Alepis, George A Tsihrintzis, and Lakhmi C Jain. 2020. "Machine Learning Paradigms: Advances in Learning Analytics." Machine Learning Paradigms: Advances in Learning Analytics, 1–5.
6. Michail, Tselepatiotis, and Efthimios Alepis. 2023. "Design of Real-Time Multiplayer Word Game for the Android Platform Using Firebase and Fuzzy Logic." In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–8. IEEE.
7. Bloch, J. (2018). *Effective Java (3rd Edition)*. Addison-Wesley Professional.
8. Walls, C. (2022). *Spring in Action (6th Edition)*. Manning Publications.
9. Spring Framework Documentation. (2023). *Spring Boot Reference Guide*. Διαθέσιμο στο: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
10. Google Developers. (2023). *Android Developers Guide: UI Overview & Fragments*. Διαθέσιμο στο: <https://developer.android.com/guide>
11. Square, Inc. (2023). *Retrofit: A type-safe HTTP client for Android and Java*. Διαθέσιμο στο: <https://square.github.io/retrofit/>
12. IETF (Internet Engineering Task Force). (2015). *RFC 7519: JSON Web Token (JWT)*. Διαθέσιμο στο: <https://datatracker.ietf.org/doc/html/rfc7519>
13. PostgreSQL Global Development Group. (2023). *PostgreSQL Documentation*.

Διαθέσιμο στο: <https://www.postgresql.org/docs/>

14. Material Design. (2023). *Material Design 3 Guidelines*. Διαθέσιμο στο: <https://m3.material.io/>