ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Εφαρμογής Android ως Επέκταση σε WordPress Εφαρμογές»

Του φοιτητή
Στραπλάκη Κωνσταντίνου
Αρ. Μητρώου: 185280

Επιβλέπων
Μιχαήλ Σαλαμπάση

Ημερομηνία 2026

Τίτλος Δ.Ε. Ανάπτυξη Εφαρμογής Android ως Επέκταση σε WordPress Εφαρμογές
Κωδικός Δ.Ε. 27834
Ονοματεπώνυμο φοιτητή Στραπλάκης Κωνσταντίνος
Ονοματεπώνυμο εισηγητή Μιχαήλ Σαλαμπάση
Ημερομηνία ανάληψης Δ.Ε. 10 - 2023
Ημερομηνία περάτωσης Δ.Ε. 5 - 2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Η επιλογή του θέματος της παρούσας πτυχιακής εργασίας έγινε με γνώμονα την αυξανόμενη ανάγκη για σύγχρονες εφαρμογές κινητών συσκευών που υποστηρίζουν την πρόσβαση σε εκπαιδευτικές υπηρεσίες. Ως φοιτητής του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων, θεώρησα ιδιαίτερα ενδιαφέρον να ασχοληθώ με την ανάπτυξη μιας Android εφαρμογής που συνδέεται με υπάρχον σύστημα WordPress/WooCommerce και προσφέρει λειτουργίες σύνδεσης, προφίλ, μαθημάτων, συνδρομών, καθηγητών, QR πρόσβασης και εσωτερικού ημερολογίου.

Η εργασία αυτή μου έδωσε τη δυνατότητα να εφαρμόσω στην πράξη γνώσεις σχετικά με Kotlin, Jetpack Compose, Material Design, HTTP επικοινωνία, JSON δεδομένα και τοπική αποθήκευση διακριτικών πρόσβασης. Παράλληλα, με βοήθησε να κατανοήσω καλύτερα τις απαιτήσεις μιας εφαρμογής που λειτουργεί ως mobile client πραγματικού συστήματος.

Ελπίζω η παρούσα εργασία να αποτελέσει χρήσιμο παράδειγμα για την ανάπτυξη Android εφαρμογών που αξιοποιούν σύγχρονες τεχνολογίες και συνδέονται με πραγματικές διακομιστικές υπηρεσίες.

Περίληψη

Η παρούσα πτυχιακή εργασία παρουσιάζει την ανάλυση, τον σχεδιασμό, την υλοποίηση και τον έλεγχο μιας εφαρμογής Android με τίτλο Login Management Application. Η εφαρμογή αναπτύχθηκε με Kotlin και Jetpack Compose στο Android Studio και λειτουργεί ως κινητός πελάτης ενός συστήματος που βασίζεται σε προσαρμοσμένη διακομιστική διεπαφή PHP/WordPress. Το αντικείμενο της εφαρμογής είναι η ασφαλής σύνδεση χρηστών, η εμφάνιση στοιχείων προφίλ, η πρόσβαση σε λειτουργίες που σχετίζονται με μαθήματα, παραγγελίες, συνδρομές, καθηγητές, εσωτερικό ημερολόγιο μαθημάτων, QR πρόσβαση και ρυθμίσεις εμφάνισης.

Η εφαρμογή χρησιμοποιεί token-based authentication. Μετά την επιτυχή σύνδεση, το διακριτικό αποθηκεύεται τοπικά με SharedPreferences και αξιοποιείται σε προστατευμένες κλήσεις προς το API. Τα δεδομένα του χρήστη λαμβάνονται από το endpoint προφίλ και εμφανίζονται σε σύγχρονη διεπαφή Compose. Η πλοήγηση πραγματοποιείται μέσω Android Activity κλάσεων και κοινών οθονών Compose, ενώ το πλαϊνό μενού παρέχει πρόσβαση σε βασικές ενότητες όπως Profile, My Orders, My Teachers, My Subscriptions, My Calendar και Settings.

Ιδιαίτερη έμφαση δόθηκε στη σαφή οργάνωση του κώδικα σε Activities, Screens, Repositories, Models, Network και Utilities. Η επικοινωνία με τον διακομιστή υλοποιήθηκε με HttpURLConnection και JSON parsing. Το ημερολόγιο της εφαρμογής είναι εσωτερική προβολή του προγράμματος μαθημάτων που επιστρέφεται από το προσαρμοσμένο API. Δεν πραγματοποιείται ενσωμάτωση με Google Calendar και δεν χρησιμοποιείται Firebase για τη σύνδεση ή την αποθήκευση δεδομένων.

Στην εργασία αναλύονται οι λειτουργικές και μη λειτουργικές απαιτήσεις, η αρχιτεκτονική πελάτη-διακομιστή, η υλοποίηση των επιμέρους οθονών, η διαχείριση σφαλμάτων, οι ροές δεδομένων και η διαδικασία ελέγχου. Παρουσιάζονται επίσης διαγράμματα αρχιτεκτονικής, ροής σύνδεσης, πακέτων και QR λειτουργίας, καθώς και αντιπροσωπευτικά αποσπάσματα κώδικα από το πραγματικό Android project.

«Developing an Android App as a WordPress Plugin»

«Straplakis Konstantinos»

Abstract

This bachelor thesis presents the analysis, design, implementation, and testing of an Android application titled Login Management Application. The application was developed in Android Studio using Kotlin and Jetpack Compose, and it functions as a mobile client for a system based on a custom PHP/WordPress server-side interface. The purpose of the application is to provide secure user login, display profile information, and offer access to features related to courses, orders, subscriptions, teachers, an internal course calendar, QR access, and appearance settings.

The application uses token-based authentication. After a successful login, the token is stored locally using SharedPreferences and is used in protected API calls. User data is retrieved from the profile endpoint and displayed in a modern Compose interface. Navigation is handled through Android Activity classes and shared Compose screens, while the side menu provides access to key sections such as Profile, My Orders, My Teachers, My Subscriptions, My Calendar, and Settings.

Special emphasis was placed on clearly organizing the code into Activities, Screens, Repositories, Models, Network, and Utilities. Communication with the server was implemented using HttpURLConnection and JSON parsing. The application calendar is an internal view of the course schedule returned by the custom API. There is no integration with Google Calendar, and Firebase is not used for login or data storage.

The thesis analyzes the functional and non-functional requirements, the client-server architecture, the implementation of the individual screens, error handling, data flows, and the testing process. It also presents architecture diagrams, login flow diagrams, package diagrams, QR functionality diagrams, and representative code excerpts from the actual Android project.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

1. Εισαγωγή

1.1.Περιγραφή του θέματος

Η παρούσα εργασία αφορά την ανάπτυξη εφαρμογής Android που λειτουργεί ως κινητός πελάτης για σύστημα διαχείρισης πρόσβασης και υπηρεσιών μαθημάτων. Η εφαρμογή δεν περιορίζεται σε μια απλή φόρμα σύνδεσης, αλλά οργανώνει γύρω από την ταυτοποίηση του χρήστη ένα σύνολο λειτουργιών που σχετίζονται με το προφίλ, τα μαθήματα, τις παραγγελίες, τις συνδρομές, τους καθηγητές, την προβολή ημερολογίου και τη χρήση QR κωδικού.

Το έργο αναπτύχθηκε στο Android Studio με γλώσσα Kotlin. Η διεπαφή χρήστη υλοποιήθηκε κυρίως με Jetpack Compose και Material 3, ακολουθώντας σύγχρονη δηλωτική προσέγγιση σχεδίασης οθονών. Η εφαρμογή επικοινωνεί με προσαρμοσμένα PHP endpoints που βρίσκονται σε διακομιστική υποδομή WordPress/WooCommerce. Με αυτόν τον τρόπο, ο Android πελάτης αξιοποιεί υπάρχοντα δεδομένα χρηστών, παραγγελιών και μαθημάτων, αντί να δημιουργεί ανεξάρτητη βάση δεδομένων μόνο για την εφαρμογή.

Κεντρικό σημείο της εργασίας είναι η σύνδεση του χρήστη με διακριτικό πρόσβασης. Μετά την επιτυχή είσοδο, η εφαρμογή αποθηκεύει το token στη συσκευή και το χρησιμοποιεί για να ανακτήσει προστατευμένα δεδομένα. Η διαδικασία αυτή επιτρέπει στην εφαρμογή να παρέχει εξατομικευμένη εμπειρία, χωρίς ο χρήστης να χρειάζεται να εισάγει ξανά τα στοιχεία του σε κάθε οθόνη.

Η εφαρμογή αποτελεί παράδειγμα πρακτικής σύνδεσης mobile ανάπτυξης με υπάρχον web οικοσύστημα. Το Android project δεν δημιουργήθηκε ως απομονωμένο demo με στατικά δεδομένα, αλλά ως πελάτης που ζητά, επεξεργάζεται και εμφανίζει πραγματικές απαντήσεις JSON. Αυτό αυξάνει την τεχνική αξία της εργασίας, διότι απαιτεί χειρισμό δικτύου, σφαλμάτων, τοπικής κατάστασης, πλοήγησης και ασφάλειας.

1.2.Κίνητρο και πεδίο εφαρμογής

Η χρήση κινητών εφαρμογών στην εκπαίδευση και στη διαχείριση μαθημάτων επιτρέπει στον χρήστη να έχει άμεση πρόσβαση σε πληροφορίες χωρίς να απαιτείται συνεχής χρήση browser. Σε περιβάλλον όπου υπάρχουν ήδη χρήστες, μαθήματα, παραγγελίες και συνδρομές σε WordPress/WooCommerce, η ανάπτυξη Android εφαρμογής προσφέρει ένα επιπλέον επίπεδο ευχρηστίας και ταχύτητας.

Το κίνητρο της εργασίας ήταν να δημιουργηθεί ένας πραγματικός Android πελάτης που δεν λειτουργεί απομονωμένα, αλλά συνδέεται με πραγματικά endpoints. Η εφαρμογή έπρεπε να μπορεί να συνδεθεί στο υπάρχον σύστημα, να εμφανίσει στοιχεία προφίλ, να διαχειριστεί τοπικά την

κατάσταση εμφάνισης, να παρέχει πρόσβαση σε QR κωδικό και να οργανώσει πληροφορίες για παραγγελίες, μαθήματα και καθηγητές.

Το πεδίο εφαρμογής περιορίζεται στην πλευρά του Android client και στα απαραίτητα PHP endpoints που χρησιμοποιούνται από αυτόν. Όσα χαρακτηριστικά αναφέρονται στην παρούσα εργασία αντιστοιχούν σε υλοποιημένες ή άμεσα υποστηριζόμενες λειτουργίες της εφαρμογής. Χαρακτηριστικά που δεν υλοποιήθηκαν, όπως σύνδεση με Google Calendar, Firebase Authentication ή πλήρης offline βάση δεδομένων, αναφέρονται αποκλειστικά ως περιορισμοί ή πιθανές μελλοντικές επεκτάσεις.

1.3.Στόχοι της εργασίας

- υλοποίηση οθόνης σύνδεσης με αποστολή στοιχείων προς custom PHP API,
- λήψη και αποθήκευση token αυθεντικοποίησης στη συσκευή,
- φόρτωση και παρουσίαση πραγματικών στοιχείων προφίλ από προστατευμένο endpoint,
- υλοποίηση σελίδας QR πρόσβασης με ιστορικό χρήσης/μοιράσματος QR,
- προβολή παραγγελιών WooCommerce που συνδέονται με τον χρήστη,
- προβολή συνδρομών/μαθημάτων με βάση ολοκληρωμένες ή ενεργές παραγγελίες,
- προβολή καθηγητών που σχετίζονται με τα μαθήματα του χρήστη,
- υλοποίηση εσωτερικού ημερολογίου μαθημάτων με δεδομένα προγράμματος από το API,
- δημιουργία οθόνης ρυθμίσεων με dark mode, πληροφορίες εφαρμογής, βοήθεια και αποσύνδεση,
- καθαρή οργάνωση κώδικα σε Activities, Screens, Repositories, Models, Network και Utilities.

Οι στόχοι αυτοί ορίζουν την εργασία ως εφαρμογή πρόσβασης και παρουσίασης δεδομένων. Η έμφαση δεν δόθηκε στη δημιουργία νέου backend, αλλά στη σωστή ενσωμάτωση του Android client με το υπάρχον σύστημα και στην ανάπτυξη μιας συνεπούς εμπειρίας χρήστη.

1.4.Οργάνωση εργασίας και όρια ευθύνης

Η εργασία οργανώνεται ως τεχνική παρουσίαση της εφαρμογής Android και της επικοινωνίας της με το υπάρχον σύστημα. Ο διακομιστής αντιμετωπίζεται ως σύνολο διαθέσιμων endpoints που επιστρέφουν JSON απαντήσεις. Η εσωτερική υλοποίηση του WordPress, του WooCommerce και της βάσης δεδομένων δεν αναλύεται ως αυτόνομο αντικείμενο, παρά μόνο στον βαθμό που επηρεάζει τις κλήσεις της εφαρμογής. Σημαντικό όριο της εργασίας είναι ότι το 'My Calendar' αποτελεί εσωτερική προβολή του προγράμματος μαθημάτων. Δεν πραγματοποιήθηκε σύνδεση με Google Calendar ούτε συγχρονισμός με εξωτερικό ημερολόγιο. Αντίστοιχα, η εφαρμογή δεν παρουσιάζεται ως ολοκληρωμένο σύστημα δημιουργίας λογαριασμών, αλλά ως εφαρμογή σύνδεσης υπάρχοντος χρήστη και ανάκτησης των δεδομένων του. Επίσης, δεν χρησιμοποιήθηκε Firebase Authentication για την αυθεντικοποίηση και δεν χρησιμοποιήθηκε Retrofit για τις κλήσεις δικτύου. Οι κλήσεις υλοποιούνται με HttpURLConnection, γεγονός που κάνει τη ροή δικτύου πλήρως ορατή στον κώδικα και κατάλληλη για εκπαιδευτική ανάλυση.

1.5.Δομή της εργασίας

Η εργασία ακολουθεί δομή τεχνικής πτυχιακής: εισαγωγή, θεωρητικό υπόβαθρο, ανάλυση απαιτήσεων, σχεδιασμός συστήματος, υλοποίηση Android, έλεγχος και αποτελέσματα, συμπεράσματα, βιβλιογραφία και παραρτήματα. Στα παραρτήματα συγκεντρώνονται τα διαγράμματα και τα

αποσπάσματα κώδικα, ώστε το κύριο σώμα να παραμένει αναγνώσιμο και η τεχνική τεκμηρίωση να είναι πλήρης.

2. Θεωρητικό Υπόβαθρο και Τεχνολογίες

2.1.Εφαρμογές Android και κινητές τεχνολογίες

Οι εφαρμογές Android αποτελούν σημαντικό πεδίο ανάπτυξης λογισμικού, καθώς προσφέρουν άμεση αλληλεπίδραση με τον χρήστη, πρόσβαση σε υπηρεσίες συσκευής και δυνατότητα επικοινωνίας με απομακρυσμένα συστήματα. Σε ένα σύστημα διαχείρισης μαθημάτων, η κινητή εφαρμογή λειτουργεί ως εύχρηστο σημείο πρόσβασης σε πληροφορίες που ο χρήστης χρειάζεται συχνά, όπως το προφίλ του, τα μαθήματα που παρακολουθεί, το πρόγραμμα και οι παραγγελίες του. Η ανάπτυξη Android απαιτεί προσεκτικό σχεδιασμό ως προς τη διεπαφή, τη διαχείριση κατάστασης, την ασύγχρονη επικοινωνία δικτύου και την αποθήκευση τοπικών δεδομένων. Η παρούσα εφαρμογή χρησιμοποιεί Activity-based πλοήγηση, Compose οθόνες και repository classes για την απομόνωση της δικτυακής επικοινωνίας από τη διεπαφή.

2.2.Android Studio

Το Android Studio χρησιμοποιήθηκε ως κύριο ολοκληρωμένο περιβάλλον ανάπτυξης. Μέσω αυτού έγινε η οργάνωση του Gradle project, η διαχείριση των dependencies, η εκτέλεση της εφαρμογής σε emulator ή φυσική συσκευή και η παρακολούθηση των logs. Το έργο διαθέτει app module με Kotlin και Compose υποστήριξη, ενώ το Gradle αρχείο καθορίζει compileSdk, minSdk, targetSdk και βιβλιοθήκες. Η χρήση του Android Studio διευκόλυνε την ανάπτυξη των οθονών Compose, την προεπισκόπηση του UI και την αποσφαλμάτωση των κλήσεων προς το API. Επιπλέον, μέσω του Logcat ήταν δυνατή η παρακολούθηση των HTTP απαντήσεων που καταγράφονται από τα repositories.

2.3.Kotlin

Η Kotlin επιλέχθηκε ως κύρια γλώσσα ανάπτυξης λόγω της σύγχρονης σύνταξης, της ασφάλειας ως προς null τιμές και της άμεσης υποστήριξης από το Android οικοσύστημα. Στην εφαρμογή χρησιμοποιείται για τις Activity κλάσεις, τα repositories, τα data models, τις utilities και τις Composable συναρτήσεις. Τα data classes αξιοποιούνται για την αναπαράσταση αντικειμένων όπως παραγγελίες, συνδρομές, καθηγητές, μαθήματα και εγγραφές QR ιστορικού. Η καθαρή σύνταξη της Kotlin επιτρέπει την οργάνωση της λογικής με λιγότερο επαναλαμβανόμενο κώδικα σε σχέση με παλαιότερες Java προσεγγίσεις.

2.4.Jetpack Compose και Material 3

Το Jetpack Compose αποτελεί δηλωτικό πλαίσιο ανάπτυξης διεπαφών για Android. Αντί να σχεδιάζεται το UI αποκλειστικά με XML, οι οθόνες περιγράφονται με Composable συναρτήσεις. Στην εφαρμογή χρησιμοποιούνται Composable οθόνες όπως LoginScreen, ProfileScreen, SettingsScreen, OrdersScreen, TeachersScreen, QRScreen και άλλες. Το Material 3 προσφέρει έτοιμα στοιχεία διεπαφής όπως Cards, Buttons, OutlinedTextFields, NavigationDrawerItem, TopAppBar και Dialogs. Η χρήση τους βοήθησε στη δημιουργία σύγχρονου και συνεκτικού περιβάλλοντος χρήστη, με έμφαση στην απλότητα και στην αναγνωσιμότητα.

2.5.Αρχιτεκτονική πελάτη-διακομιστή και REST/PHP endpoints

Η εφαρμογή ακολουθεί λογική πελάτη-διακομιστή. Ο Android client δεν αποθηκεύει μόνιμα όλα τα δεδομένα του συστήματος, αλλά τα ανακτά από τον διακομιστή μέσω HTTP αιτημάτων. Ο διακομιστής παρέχει endpoints όπως login.php, profile.php, orders.php, my_calendar.php, qr_access.php, get_qr_history.php, save_qr_history.php και lesson_details.php. Οι απαντήσεις επιστρέφονται σε JSON μορφή και αναλύονται μέσα στα repositories με JSONObject. Η χρήση custom PHP API ήταν απαραίτητη επειδή το υπάρχον σύστημα βασίζεται σε WordPress/WooCommerce και χρειάστηκε ειδική γέφυρα επικοινωνίας για το Android.

2.6.Token-based authentication και SharedPreferences

Η ταυτοποίηση βασίζεται σε token που εκδίδεται μετά την επιτυχημένη σύνδεση. Το token αποθηκεύεται σε SharedPreferences με όνομα auth και κλειδί auth_token. Με τον τρόπο αυτό, οι επόμενες οθόνες μπορούν να το ανακτήσουν και να το χρησιμοποιήσουν σε προστατευμένες κλήσεις. Η αποθήκευση token σε SharedPreferences αποτελεί πρακτική λύση για εφαρμογή επίδειξης/πτυχιακή εργασία. Παρόλα αυτά, σε παραγωγικό περιβάλλον θα μπορούσε να εξεταστεί η χρήση πιο ασφαλούς μηχανισμού αποθήκευσης, όπως encrypted preferences.

2.7.QR Code και ZXing

Η εφαρμογή υποστηρίζει QR λειτουργία. Το QR payload λαμβάνεται από το API και εμφανίζεται στον χρήστη μέσα από ειδική οθόνη. Για την παραγωγή/απεικόνιση QR αξιοποιούνται βιβλιοθήκες του οικοσυστήματος ZXing, οι οποίες επιτρέπουν τη δημιουργία κωδικών QR από κείμενο ή JSON δεδομένα. Επιπλέον, η εφαρμογή περιλαμβάνει ιστορικό QR, ώστε να εμφανίζονται παλαιότερες ενέργειες που σχετίζονται με χρήση ή μοίρασμα του QR. Η λειτουργία αυτή συνδέεται με τα endpoints get_qr_history.php και save_qr_history.php.

2.8.Φόρτωση εικόνων με Coil/Glide

Η προβολή εικόνων στο Android απαιτεί μηχανισμούς ασύγχρονης φόρτωσης, caching και σωστής διαχείρισης μνήμης. Στην εφαρμογή χρησιμοποιείται AsyncImage από το Coil Compose για την εμφάνιση της εικόνας προφίλ. Στο Gradle υπάρχουν επίσης εξαρτήσεις σχετικές με Glide/Coil, ώστε να καλύπτονται ανάγκες φόρτωσης εικόνων σε διαφορετικά σημεία του έργου. Η εικόνα προφίλ προέρχεται από το API και εμφανίζεται μέσα σε κυκλικό σχήμα στην οθόνη Profile. Αν το URL είναι κενό, η εφαρμογή χειρίζεται την περίπτωση χωρίς να καταρρέει.

3. Ανάλυση Απαιτήσεων και Σχεδιασμός

3.1.Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν τι πρέπει να μπορεί να κάνει ο χρήστης μέσα από την εφαρμογή. Οι απαιτήσεις που ακολουθούν προκύπτουν από την υλοποιημένη λειτουργικότητα του Android project και από τα διαθέσιμα endpoints που χρησιμοποιούνται από τον κώδικα.

Κωδικός	**Απαίτηση**	**Υλοποιημένη λειτουργία**
FR-01	Σύνδεση χρήστη	LoginActivity, LoginScreen, AuthRepository.login
FR-02	Ανάκτηση προφίλ	AuthRepository.fetchProfile και ProfileActivity
FR-03	Τοπική αποθήκευση token	SharedPreferences auth/auth_token
FR-04	Αποσύνδεση	logout.php ή τοπική διαγραφή token ανά οθόνη
FR-05	Forgot password	ForgotPasswordActivity και forgot_password.php
FR-06	Προβολή QR πρόσβασης	QRActivity, QRRepository, QRScreen
FR-07	Ιστορικό QR	get_qr_history.php και save_qr_history.php
FR-08	Προβολή παραγγελιών	OrdersActivity και OrdersRepository
FR-09	Προβολή συνδρομών/μαθημάτων	SubscriptionsActivity και SubscriptionsRepository
FR-10	Προβολή λεπτομερειών μαθήματος	SubscriptionDetailActivity και lesson_details.php
FR-11	Προβολή καθηγητών	TeachersActivity και TeachersRepository
FR-12	Εσωτερικό ημερολόγιο μαθημάτων	CalendarActivity και my_calendar.php
FR-13	Ρυθμίσεις εμφάνισης	SettingsActivity, SettingsScreen, DarkModeManager
FR-14	Πλαϊνό μενού πλοήγησης	ModalNavigationDrawer σε βασικές οθόνες

Πίνακας 3.1: Λειτουργικές απαιτήσεις της εφαρμογής.

3.2.Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις αφορούν χαρακτηριστικά ποιότητας του συστήματος, όπως ασφάλεια, αξιοπιστία, απόδοση και ευχρηστία. Για την εφαρμογή δόθηκε έμφαση στην απλότητα χρήσης, στη διατήρηση σταθερής λειτουργίας κατά τις δικτυακές κλήσεις και στη συνεπή οπτική εμπειρία.

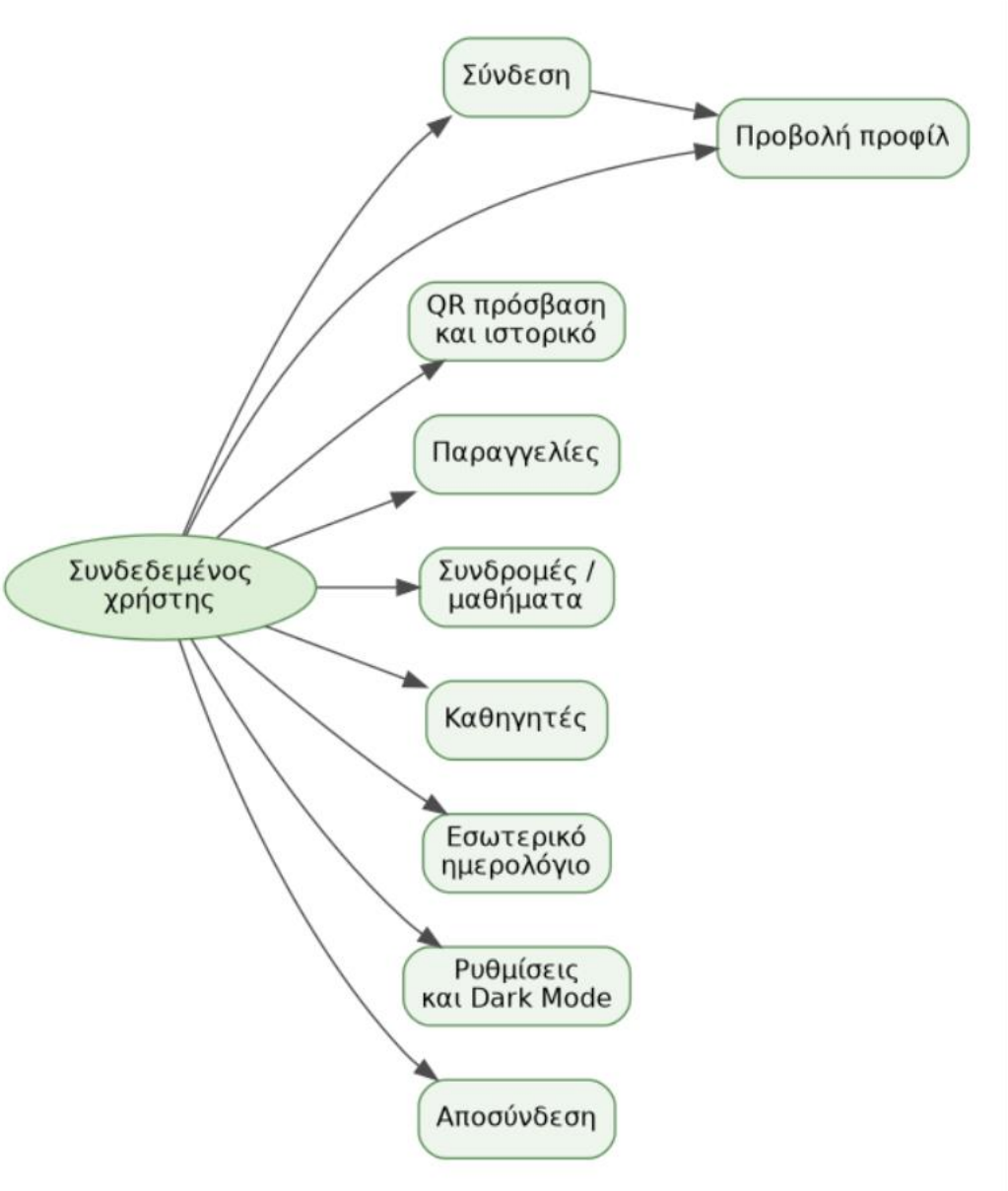
Κατηγορία	**Περιγραφή**	**Τρόπος κάλυψης**
Ασφάλεια	Πρόσβαση μόνο με έγκυρο token	Αποστολή token σε Authorization Bearer ή X-Auth-Token ανά endpoint
Ευχρηστία	Καθαρό UI και απλή πλοήγηση	Material 3, cards, drawer navigation, κουμπιά ενεργειών
Συντηρησιμότητα	Διαχωρισμός λογικής ανά πακέτο	Repositories, Activities, Screens, Models, Utilities
Απόδοση	Αποφυγή δικτυακών κλήσεων στο κύριο νήμα	Χρήση Thread μέσα στα repositories και επιστροφή στο UI thread
Αξιοπιστία	Χειρισμός αποτυχιών API	Callbacks onError, Toast μηνύματα, τερματισμός οθόνης όπου λείπει token
Προσαρμοστικότητα	Υποστήριξη light/dark εμφάνισης	DarkModeManager και επανεκκίνηση Activity

Πίνακας 3.2: Μη λειτουργικές απαιτήσεις.

3.3.Περιπτώσεις χρήσης

Οι βασικές περιπτώσεις χρήσης οργανώνονται γύρω από έναν συνδεδεμένο χρήστη. Ο χρήστης ξεκινά από την οθόνη σύνδεσης, εισάγει username και password και, αν τα στοιχεία είναι σωστά, μεταφέρεται στην οθόνη προφίλ. Από εκεί μπορεί να ανοίξει QR κωδικό, παραγγελίες, καθηγητές, συνδρομές, ημερολόγιο ή ρυθμίσεις. Ο χρήστης μπορεί επίσης να αποσυνδεθεί από κάθε κύρια ενότητα.

Το διάγραμμα περιπτώσεων χρήσης παρουσιάζει τις λειτουργίες όπως έχουν υλοποιηθεί. Δεν περιλαμβάνει δημιουργία μαθημάτων από καθηγητή, εξωτερικό Google Calendar ή Firebase login, διότι αυτά δεν αποτελούν μέρος του πραγματικού scope της εφαρμογής.

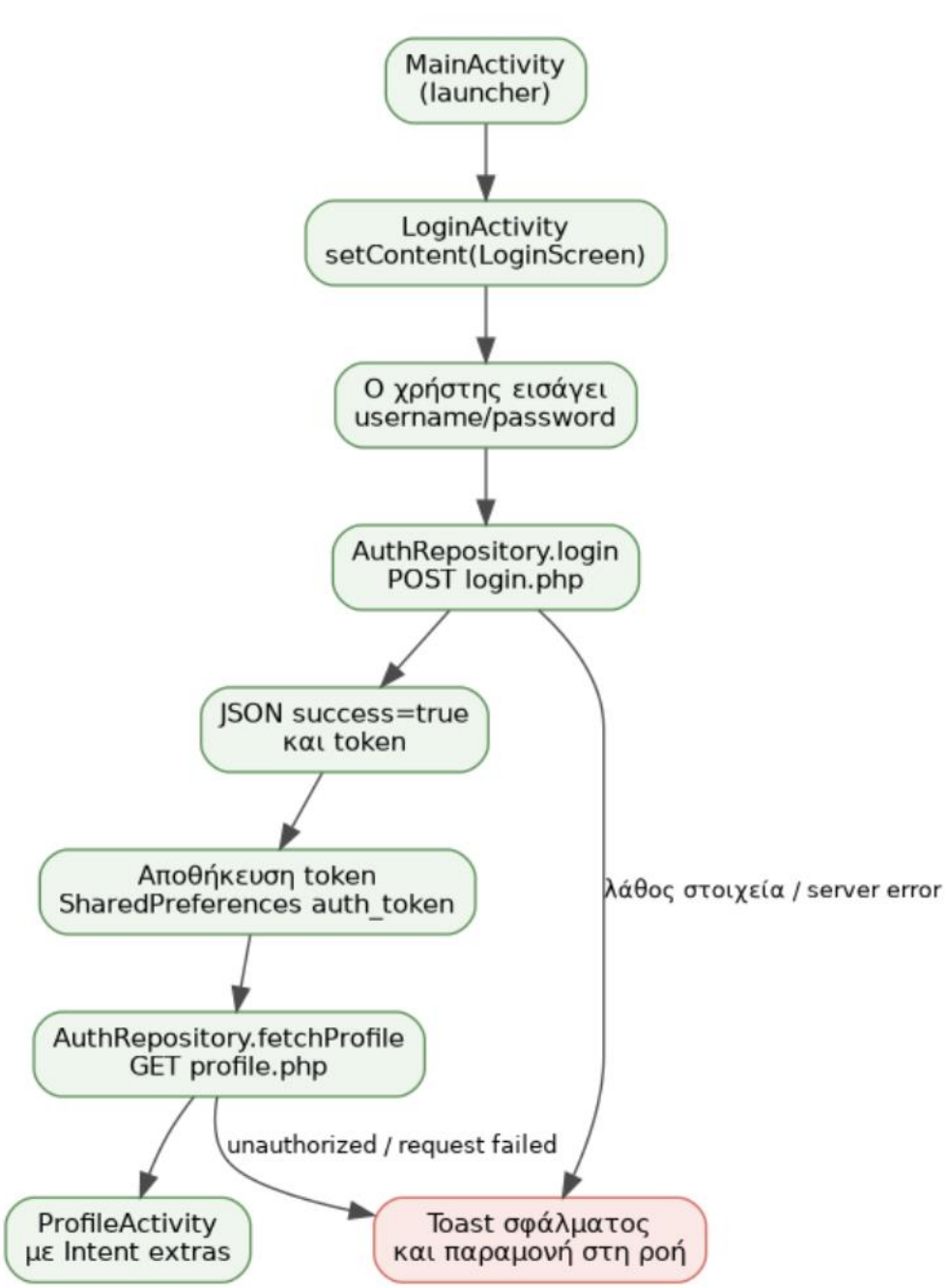


Σχήμα 3.1: Διάγραμμα περιπτώσεων χρήσης της εφαρμογής.

3.4.Ροή σύνδεσης

Η ροή σύνδεσης αποτελεί την πρώτη και σημαντικότερη ροή της εφαρμογής. Η MainActivity λειτουργεί ως launcher και μεταφέρει άμεσα τον χρήστη στη LoginActivity. Στη συνέχεια, η LoginActivity εμφανίζει τη LoginScreen. Όταν ο χρήστης πατήσει το κουμπί Login, καλείται η μέθοδος handleLogin, η οποία χρησιμοποιεί το AuthRepository για αποστολή HTTP POST προς το login endpoint.

Αν η απάντηση του διακομιστή είναι επιτυχής, λαμβάνεται token, αποθηκεύεται στις SharedPreferences και ακολουθεί κλήση fetchProfileWithToken. Η δεύτερη κλήση ανακτά τα στοιχεία του χρήστη και τα μεταφέρει με Intent extras στην ProfileActivity. Αν οποιοδήποτε βήμα αποτύχει, εμφανίζεται μήνυμα Toast και ο χρήστης παραμένει στη σχετική οθόνη.



Σχήμα 3.2: Ροή σύνδεσης και φόρτωσης προφίλ.

3.5.Όρια υλοποίησης

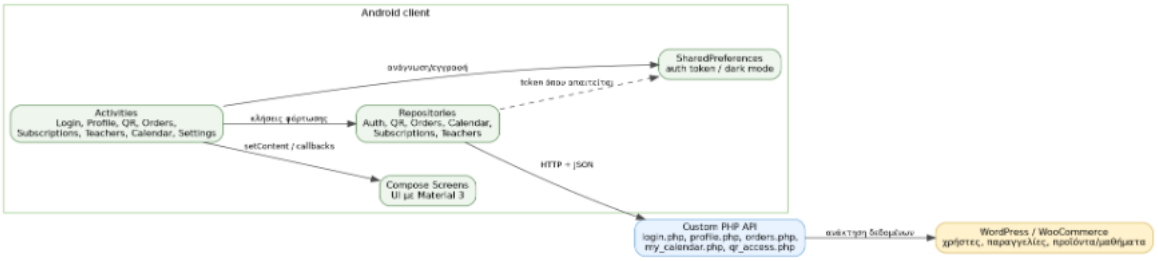
Για να είναι σαφές το πραγματικό εύρος του έργου, καταγράφονται και όσα δεν υλοποιήθηκαν ως μέρος της παρούσας εφαρμογής. Δεν υλοποιήθηκε σύνδεση με Google Calendar. Η οθόνη My Calendar είναι εσωτερικό ημερολόγιο/προβολή προγράμματος μαθημάτων που χρησιμοποιεί δεδομένα από το endpoint my_calendar.php. Δεν χρησιμοποιήθηκε Firebase για login ή βάση δεδομένων. Δεν χρησιμοποιήθηκε Retrofit· οι κλήσεις υλοποιήθηκαν με HttpURLConnection. Δεν παρουσιάζεται ξεχωριστό teacher dashboard για δημιουργία μαθημάτων από καθηγητή. Η εφαρμογή λειτουργεί ως client υπάρχοντος συστήματος και όχι ως αυτόνομο backend.

4. Διακομιστής και Διεπαφές API

4.1.Ρόλος του διακομιστή

Ο διακομιστής παρέχει τα δεδομένα που χρειάζεται η εφαρμογή Android. Η εφαρμογή δεν περιέχει απευθείας σύνδεση με βάση δεδομένων και δεν αποθηκεύει πλήρη πληροφορία μαθημάτων ή παραγγελιών τοπικά. Αντίθετα, καλεί endpoints που επιστρέφουν JSON απαντήσεις. Αυτή η προσέγγιση διατηρεί την εφαρμογή ελαφριά και επιτρέπει στο υπάρχον WordPress/WooCommerce σύστημα να παραμένει η κύρια πηγή αλήθειας.

Τα endpoints συγκεντρώνονται στην κλάση ApiClient. Εκεί ορίζεται η BASE_URL και οι σταθερές LOGIN_URL, PROFILE_URL, LOGOUT_URL, ORDERS_URL, CALENDAR_URL, QR_ACCESS_URL, QR_HISTORY_URL, SAVE_QR_HISTORY_URL και FORGOT_PASSWORD_URL. Υπάρχει επίσης βοηθητική συνάρτηση lessonDetailsUrl(productId), η οποία δημιουργεί URL για λεπτομέρειες συγκεκριμένου μαθήματος.



Σχήμα 4.1: Αρχιτεκτονική πελάτη-διακομιστή.

4.2.Κύρια endpoints

Endpoint	Χρήση στην εφαρμογή	Repository/Activity
login.php	Σύνδεση χρήστη και έκδοση token	AuthRepository / LoginActivity
profile.php	Ανάκτηση στοιχείων προφίλ	AuthRepository / ProfileActivity
logout.php	Αποσύνδεση από διακομιστή	AuthRepository / ProfileActivity
forgot_password.php	Αίτημα επαναφοράς κωδικού	AuthRepository / ForgotPasswordActivity
orders.php	Ανάκτηση παραγγελιών χρήστη	OrdersRepository / OrdersActivity
orders.php	Εξαγωγή μαθημάτων και καθηγητών από παραγγελίες	SubscriptionsRepository / TeachersRepository
my_calendar.php	Ανάκτηση προγράμματος μαθημάτων	CalendarRepository / CalendarActivity
qr_access.php	Ανάκτηση QR payload	QRRepository / QRActivity
get_qr_history.php	Ανάκτηση ιστορικού QR	QRRepository / QRActivity

save_qr_history.php	Αποθήκευση εγγραφής QR ιστορικού	QRRepository / QRActivity
lesson_details.php?product_id=...	Λεπτομέρειες μαθήματος	SubscriptionDetailActivity

Πίνακας 4.1: Κύρια endpoints που χρησιμοποιεί η εφαρμογή.

4.3.Μορφή αιτημάτων και απαντήσεων

Η σύνδεση πραγματοποιείται με POST αίτημα τύπου application/x-www-form-urlencoded. Τα πεδία username και password κωδικοποιούνται με URLEncoder πριν σταλούν στον διακομιστή. Η απάντηση αναμένεται να περιέχει πεδίο success και, σε περίπτωση επιτυχίας, αντικείμενο data με token. Η εφαρμογή ελέγχει το success και είτε καλεί onSuccess(token) είτε εμφανίζει μήνυμα σφάλματος.

Οι προστατευμένες κλήσεις χρησιμοποιούν επικεφαλίδες Authorization: Bearer token ή X-Auth-Token, ανάλογα με το endpoint. Τα profile, orders και calendar endpoints χρησιμοποιούν Bearer token, ενώ τα QR endpoints χρησιμοποιούν X-Auth-Token. Η διαφοροποίηση αυτή αποτυπώνεται στα αντίστοιχα repositories και αποτελεί μέρος του συμβολαίου του API.

Η ανάλυση των απαντήσεων γίνεται με JSONObject. Για λίστες δεδομένων, όπως orders, lessons ή history, η εφαρμογή διαβάζει JSON arrays και δημιουργεί λίστες από Kotlin model objects. Η μετατροπή αυτή βρίσκεται στα repositories, ώστε οι Composable οθόνες να λαμβάνουν ήδη έτοιμα δεδομένα παρουσίασης.

4.4.Χειρισμός σφαλμάτων δικτύου

Κάθε repository εκτελεί τις δικτυακές κλήσεις μέσα σε ξεχωριστό Thread ώστε να μην μπλοκάρει το UI thread. Η απάντηση διαβάζεται είτε από inputStream είτε από errorStream, ανάλογα με τον HTTP κωδικό. Σε περίπτωση εξαίρεσης, η εφαρμογή καταγράφει το σφάλμα και καλεί onError με κατανοητό μήνυμα. Στην πλευρά του Activity, οι αλλαγές στο UI εκτελούνται μέσω runOnUiThread.

Η σχεδίαση αυτή δεν είναι τόσο εξελιγμένη όσο μια υλοποίηση με coroutines, ViewModel και Retrofit, αλλά είναι καθαρή και πλήρως ορατή στον κώδικα. Για εκπαιδευτικούς σκοπούς βοηθά στην κατανόηση του πώς δημιουργείται μία HTTP σύνδεση, πώς γράφεται το request body, πώς επιλέγεται stream απάντησης και πώς μετατρέπεται η απάντηση σε αντικείμενα Kotlin.

4.5.Ασφάλεια και περιορισμοί

Η χρήση token περιορίζει την πρόσβαση σε προσωπικά δεδομένα. Παρόλα αυτά, στο AndroidManifest είναι ενεργό το usesCleartextTraffic, επειδή το περιβάλλον ανάπτυξης/δοκιμών χρησιμοποιεί HTTP endpoint. Σε παραγωγικό περιβάλλον, η χρήση HTTPS θεωρείται απαραίτητη. Επίσης, η αποθήκευση token σε απλές SharedPreferences είναι επαρκής για εκπαιδευτικό σκοπό, αλλά σε τελική παραγωγή θα μπορούσε να αντικατασταθεί με κρυπτογραφημένη αποθήκευση.

Οι περιορισμοί αυτοί δεν αναιρούν τη λειτουργικότητα της εφαρμογής, αλλά καταγράφονται ώστε να είναι σαφές ποια στοιχεία αποτελούν εκπαιδευτική υλοποίηση και ποια θα έπρεπε να ενισχυθούν πριν από παραγωγική διάθεση.

5. Υλοποίηση Εφαρμογής Android

5.1.Δομή έργου

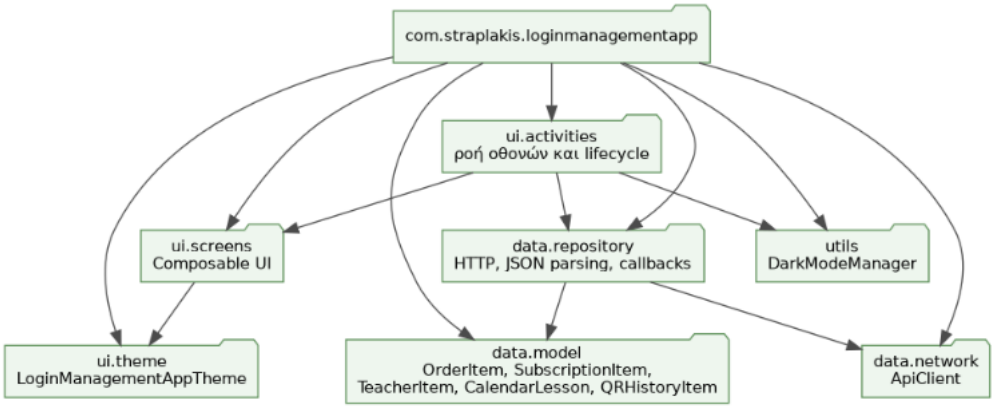
Η εφαρμογή έχει οργανωθεί σε πακέτα που χωρίζουν τις ευθύνες. Το πακέτο ui.activities περιλαμβάνει τις Activity κλάσεις που ξεκινούν τις οθόνες και χειρίζονται τη βασική ροή. Το πακέτο ui.screens περιλαμβάνει τις Composable συναρτήσεις που εμφανίζουν τη διεπαφή. Το πακέτο data.repository περιέχει τη δικτυακή λογική, ενώ το data.model περιέχει τα αντικείμενα δεδομένων που χρησιμοποιούνται για την παρουσίαση πληροφοριών. Το data.network περιέχει την κλάση ApiClient και το utils περιέχει βοηθητικές λειτουργίες, όπως το DarkModeManager.

Ο διαχωρισμός αυτός επιτρέπει την πιο εύκολη συντήρηση. Αν αλλάξει κάποιο endpoint, η αλλαγή αφορά κυρίως το αντίστοιχο repository. Αν αλλάξει κάποια οθόνη, η αλλαγή αφορά κυρίως το screen. Αν αλλάξει η μορφή ενός αντικειμένου, η αλλαγή αφορά κυρίως το model.

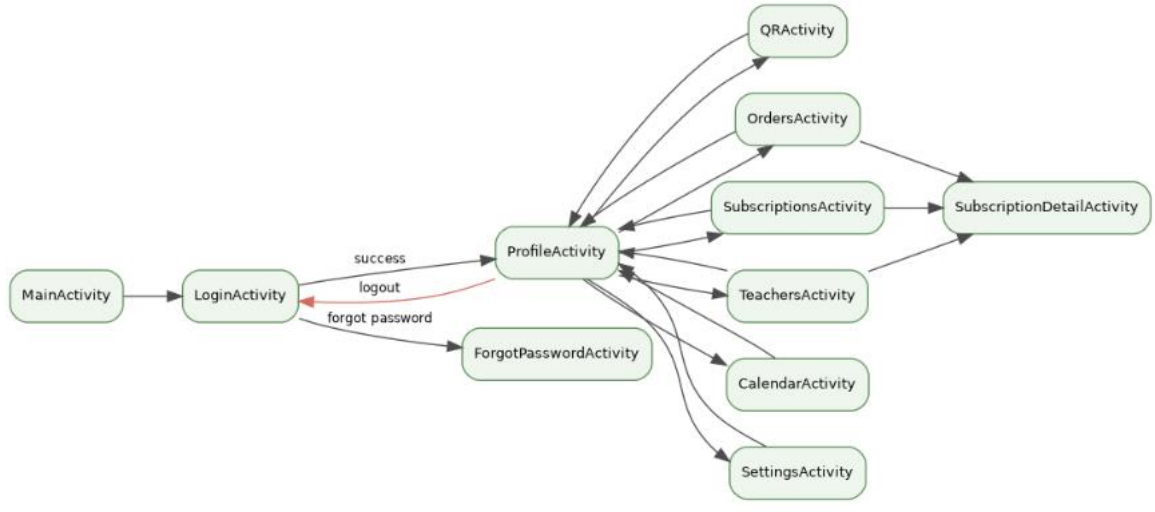
Κλάση	**Ρόλος**
MainActivity	Launcher Activity που μεταφέρει στη LoginActivity
LoginActivity	Οθόνη σύνδεσης και ανάκτηση προφίλ μετά το login
ProfileActivity	Κεντρική οθόνη προφίλ και πλοήγηση προς βασικές ενότητες
QRActivity	Φόρτωση QR payload, ιστορικού και δυνατότητα share
OrdersActivity	Προβολή παραγγελιών χρήστη
SubscriptionsActivity	Προβολή ενεργών/σχετικών συνδρομών μαθημάτων
TeachersActivity	Προβολή καθηγητών που σχετίζονται με τα μαθήματα
CalendarActivity	Εσωτερικό ημερολόγιο προγράμματος μαθημάτων

SettingsActivity	Ρυθμίσεις εφαρμογής και dark mode
AuthRepository	Login, fetchProfile, logout, forgotPassword
QRRepository	QR access, QR history, save QR history
OrdersRepository	Ανάκτηση WooCommerce παραγγελιών
CalendarRepository	Ανάκτηση μαθημάτων ημερολογίου
DarkModeManager	Αποθήκευση και ανάκτηση προτίμησης dark mode

Πίνακας 5.1: Ενδεικτικές κλάσεις και ρόλος τους.



Σχήμα 5.1: Διάγραμμα πακέτων Android project.



Σχήμα 5.2: Διάγραμμα πλοήγησης Activity κλάσεων.

5.2.Εκκίνηση εφαρμογής

Η MainActivity έχει περιορισμένη ευθύνη. Κατά την onCreate εκκινεί τη LoginActivity και τερματίζει τον εαυτό της με finish. Αυτό σημαίνει ότι η πραγματική αρχική οθόνη της εφαρμογής είναι η οθόνη

σύνδεσης. Η απλότητα αυτή βοηθάει ώστε η εφαρμογή να έχει ξεκάθαρη αρχική ροή και να μη διατηρεί περιττή Activity στο back stack.

Η επιλογή αυτή έχει πρακτικό πλεονέκτημα: ο χρήστης δεν βλέπει ενδιάμεση οθόνη και δεν μπορεί να επιστρέψει σε άδεια MainActivity. Όλη η ροή ξεκινά από τη LoginActivity, όπου βρίσκεται η πρώτη πραγματική αλληλεπίδραση.

5.3.Οθόνη σύνδεσης

Η LoginActivity ορίζει ένα AuthRepository και εμφανίζει τη LoginScreen μέσα από setContent. Η LoginScreen περιλαμβάνει πεδίο username, πεδίο password, κουμπί Login και επιλογή Forgot Password. Η εμφάνιση της οθόνης βασίζεται σε Card, OutlinedTextField, Icons και Button του Material 3. Το φόντο χρησιμοποιεί κάθετο gradient που συνδυάζει το primary χρώμα με το background, δημιουργώντας πιο σύγχρονη αίσθηση από μια απλή λευκή σελίδα.

Όταν ο χρήστης πατήσει Login, η LoginScreen καλεί το onLoginClick και η LoginActivity εκτελεί handleLogin. Η μέθοδος αυτή περνάει username και password στο AuthRepository.login. Σε επιτυχία, αποθηκεύεται το token στις SharedPreferences και αμέσως καλείται fetchProfileWithToken, ώστε η επόμενη οθόνη να έχει πραγματικά δεδομένα προφίλ.

Η οθόνη σύνδεσης αποτελεί παράδειγμα καθαρού διαχωρισμού μεταξύ UI και logic. Η Composable δεν γνωρίζει πώς γίνεται το HTTP αίτημα. Απλώς συλλέγει τα δεδομένα και καλεί callback. Η Activity συντονίζει τη ροή και το repository εκτελεί το δικτυακό τμήμα.

5.4.Forgot Password

Η λειτουργία Forgot Password συνδέεται με ξεχωριστή Activity και endpoint forgot_password.php. Ο χρήστης μπορεί να εισάγει αναγνωριστικό στοιχείο, όπως username ή email, και η εφαρμογή αποστέλλει αίτημα στον διακομιστή. Το AuthRepository.forgotPassword χειρίζεται την αποστολή form-urlencoded δεδομένων και αναμένει απάντηση με success, message και, όπου παρέχεται, reset_url. Η λειτουργία αυτή διευκολύνει την ανάκτηση πρόσβασης χωρίς να απαιτείται άμεση παρέμβαση από διαχειριστή.

5.5.Οθόνη προφίλ

Η ProfileActivity λαμβάνει από το Intent τα στοιχεία username, email, firstname, lastname και profile_picture. Στη συνέχεια εμφανίζει την ProfileScreen μέσα στο LoginManagementAppTheme. Η οθόνη προφίλ αποτελεί το κεντρικό σημείο μετά τη σύνδεση. Εμφανίζει την εικόνα προφίλ με AsyncImage, το ονοματεπώνυμο, το username και βασικά πεδία προσωπικών στοιχείων.

Στην ProfileScreen υπάρχει πλαϊνό μενού πλοήγησης με τις επιλογές Profile, My Orders, My Teachers, My Subscriptions και My Calendar. Στο επάνω μέρος υπάρχει top app bar με κουμπί μενού και πρόσθετο μενού για Settings και Logout. Με τον τρόπο αυτό, η εφαρμογή προσφέρει συνεπή πρόσβαση στις βασικές λειτουργίες χωρίς να γεμίζει την κύρια οθόνη με πολλά κουμπιά.

Η ProfileScreen επιτελεί διπλό ρόλο. Από τη μία παρουσιάζει δεδομένα χρήστη. Από την άλλη λειτουργεί ως κόμβος πλοήγησης προς τις υπόλοιπες ενότητες. Αυτή η σχεδίαση ταιριάζει σε εφαρμογή όπου η ταυτοποίηση προηγείται όλων των λειτουργιών.

5.6.Αποσύνδεση

Η αποσύνδεση υλοποιείται με δύο τρόπους, ανάλογα με την οθόνη. Στην ProfileActivity υπάρχει logoutUser που ανακτά το token από SharedPreferences και καλεί το AuthRepository.logout. Αν η απάντηση είναι επιτυχής, το token διαγράφεται και ο χρήστης επιστρέφει στη LoginActivity με flags FLAG_ACTIVITY_NEW_TASK και FLAG_ACTIVITY_CLEAR_TASK. Σε άλλες οθόνες, όπως OrdersActivity, CalendarActivity και SettingsActivity, υλοποιείται τοπική αποσύνδεση με διαγραφή του token και καθαρισμό του back stack.

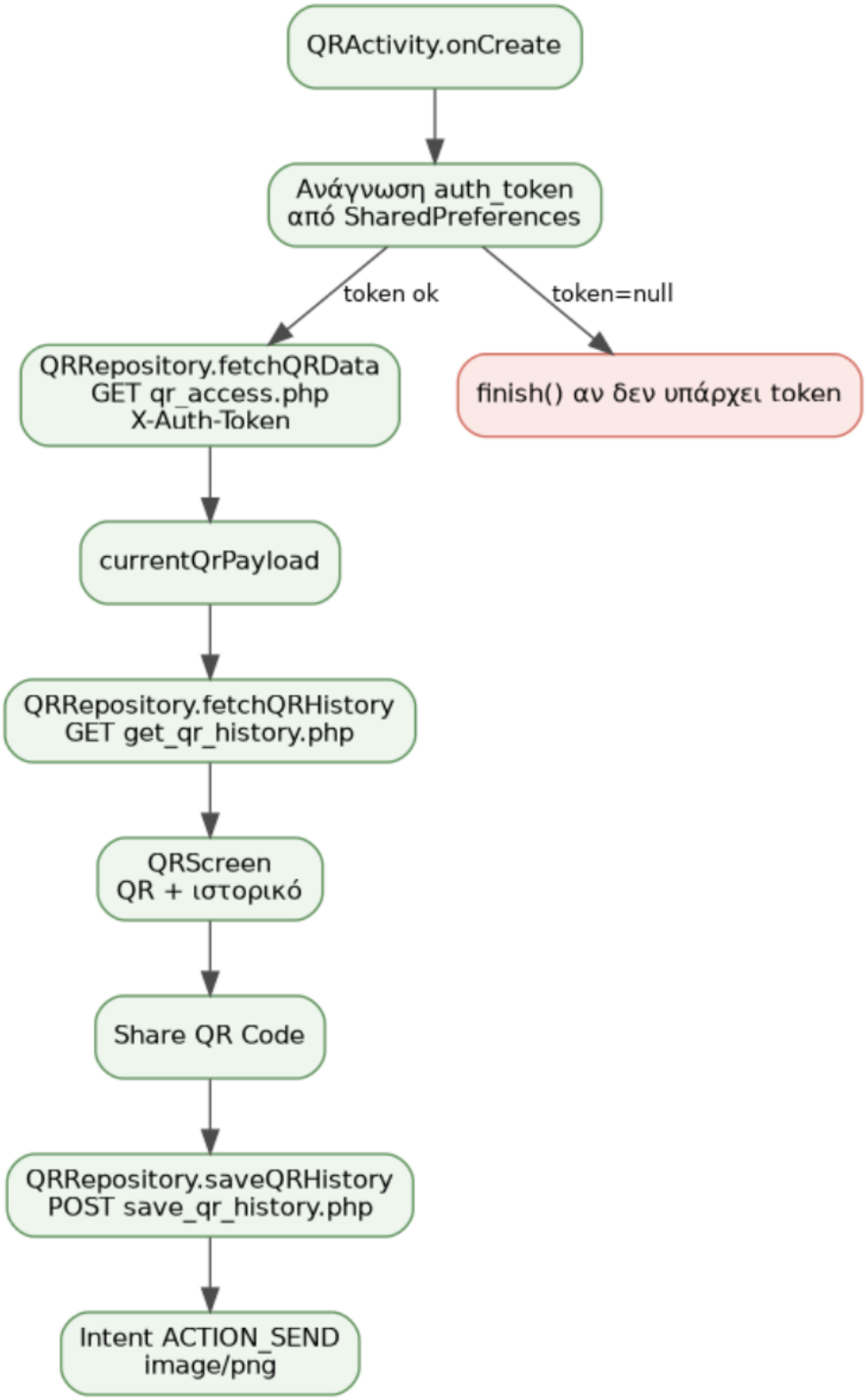
Η διαγραφή του token είναι απαραίτητη για την προστασία της συνεδρίας. Επιπλέον, τα flags καθαρισμού back stack αποτρέπουν την επιστροφή με το κουμπί Back σε προστατευμένη οθόνη μετά την αποσύνδεση.

5.7.QR πρόσβαση και ιστορικό

Η QRActivity αποτελεί ξεχωριστή ενότητα. Κατά την εκκίνηση αναζητά το token από SharedPreferences. Αν δεν υπάρχει token, η Activity τερματίζει. Αν υπάρχει, καλεί fetchQRData, η οποία με τη σειρά της χρησιμοποιεί το QRRepository για να καλέσει το qr_access.php endpoint. Το αποτέλεσμα αποθηκεύεται ως currentQrPayload και ακολουθεί φόρτωση του ιστορικού QR.

Το QRRepository.fetchQRHistory καλεί το get_qr_history.php endpoint και μετατρέπει το JSON array σε λίστα QRHistoryItem. Κάθε εγγραφή περιλαμβάνει id, scan_type, device_info και scanned_at. Η οθόνη QR εμφανίζεται αφού ολοκληρωθεί η φόρτωση του payload και του ιστορικού ή αφού αποτύχει η φόρτωση ιστορικού, ώστε ο χρήστης να μη μένει σε κενή κατάσταση.

Η εφαρμογή παρέχει επίσης δυνατότητα share του QR. Όταν ο χρήστης κάνει share, η QRActivity καλεί saveQRHistory ώστε να καταγραφεί η ενέργεια και στη συνέχεια δημιουργεί ACTION_SEND intent με image/png. Με αυτόν τον τρόπο το QR μπορεί να διαμοιραστεί μέσω εφαρμογών που υποστηρίζουν κοινή χρήση εικόνας.



Σχήμα 5.3: Ροή QR λειτουργίας.

5.8. Παραγγελίες

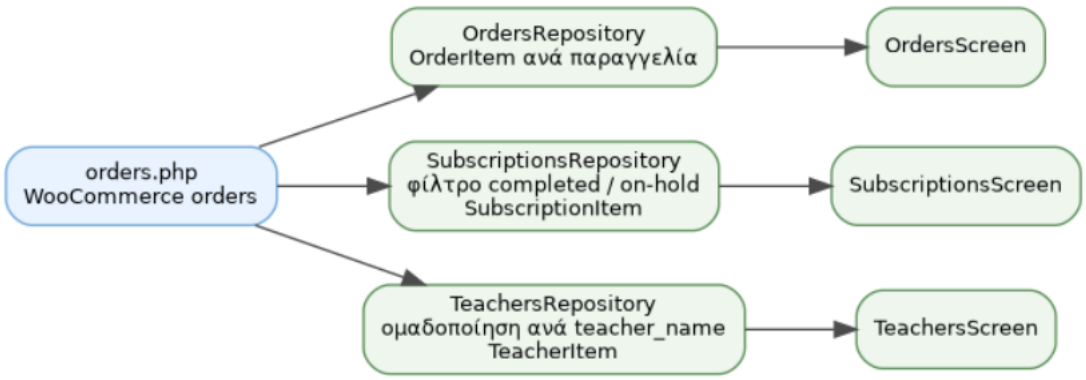
Η OrdersActivity εμφανίζει τις παραγγελίες του συνδεδεμένου χρήστη. Αρχικά ελέγχει την ύπαρξη token. Αν το token λείπει, εμφανίζεται μήνυμα 'Please login again' και η Activity κλείνει. Αν υπάρχει token, καλείται το OrdersRepository.loadOrders. Το repository στέλνει GET αίτημα στο orders.php endpoint με Authorization Bearer token.

Η απάντηση περιέχει λίστα orders. Για κάθε παραγγελία διαβάζονται στοιχεία όπως order_id, date, status, total, currency και αντικείμενο lesson. Από το lesson λαμβάνονται product_id, title, image και price. Τα δεδομένα μετατρέπονται σε OrderItem και δίνονται στην OrdersScreen για εμφάνιση. Η οθόνη επιτρέπει επίσης μετάβαση σε SubscriptionDetailActivity όταν ο χρήστης επιλέξει συγκεκριμένο μάθημα/συνδρομή.

5.9.Συνδρομές και μαθήματα

Η SubscriptionsRepository χρησιμοποιεί επίσης το orders.php endpoint, αλλά εφαρμόζει φίλτρο στις παραγγελίες. Συγκεκριμένα, λαμβάνονται υπόψη οι καταστάσεις completed και on-hold. Για κάθε σχετική παραγγελία δημιουργείται SubscriptionItem που περιλαμβάνει orderId, productId, title, teacherName, schedule, status, purchaseDate, price, currency, image, startDate, endDate, startTime και endTime.

Με τον τρόπο αυτό, η ενότητα 'My Subscriptions' δεν εμφανίζει όλες τις παραγγελίες, αλλά τις παραγγελίες που αντιστοιχούν σε ενεργά ή σχετικά μαθήματα. Η πληροφορία αυτή βοηθά τον χρήστη να βλέπει ποια μαθήματα έχει αγοράσει ή παρακολουθεί, ποιος είναι ο καθηγητής και ποιο είναι το πρόγραμμα.



Σχήμα 5.4: Ροή παραγωγής δεδομένων για Orders, Subscriptions και Teachers.

5.10. Λεπτομέρειες μαθήματος

Όταν ο χρήστης επιλέγει ένα μάθημα από παραγγελία, καθηγητή ή συνδρομή, η εφαρμογή ανοίγει SubscriptionDetailActivity και περνάει το product_id μέσω Intent. Το endpoint lesson_details.php?product_id=... επιστρέφει αναλυτικά στοιχεία για το μάθημα. Η λειτουργία αυτή

απομονώνει την πλήρη πληροφορία μαθήματος από τις λίστες, ώστε οι λίστες να παραμένουν ελαφριές και οι λεπτομέρειες να φορτώνονται μόνο όταν χρειάζεται.

5.11. Καθηγητές

Η TeachersActivity και το TeachersRepository δημιουργούν προβολή καθηγητών με βάση τα μαθήματα που προκύπτουν από τις παραγγελίες του χρήστη. Το repository καλεί το orders.php endpoint και εξετάζει μόνο παραγγελίες με κατάσταση completed ή on-hold. Από κάθε lesson αντικείμενο αναζητά teacher_name, teacher_details και στοιχεία μαθήματος.

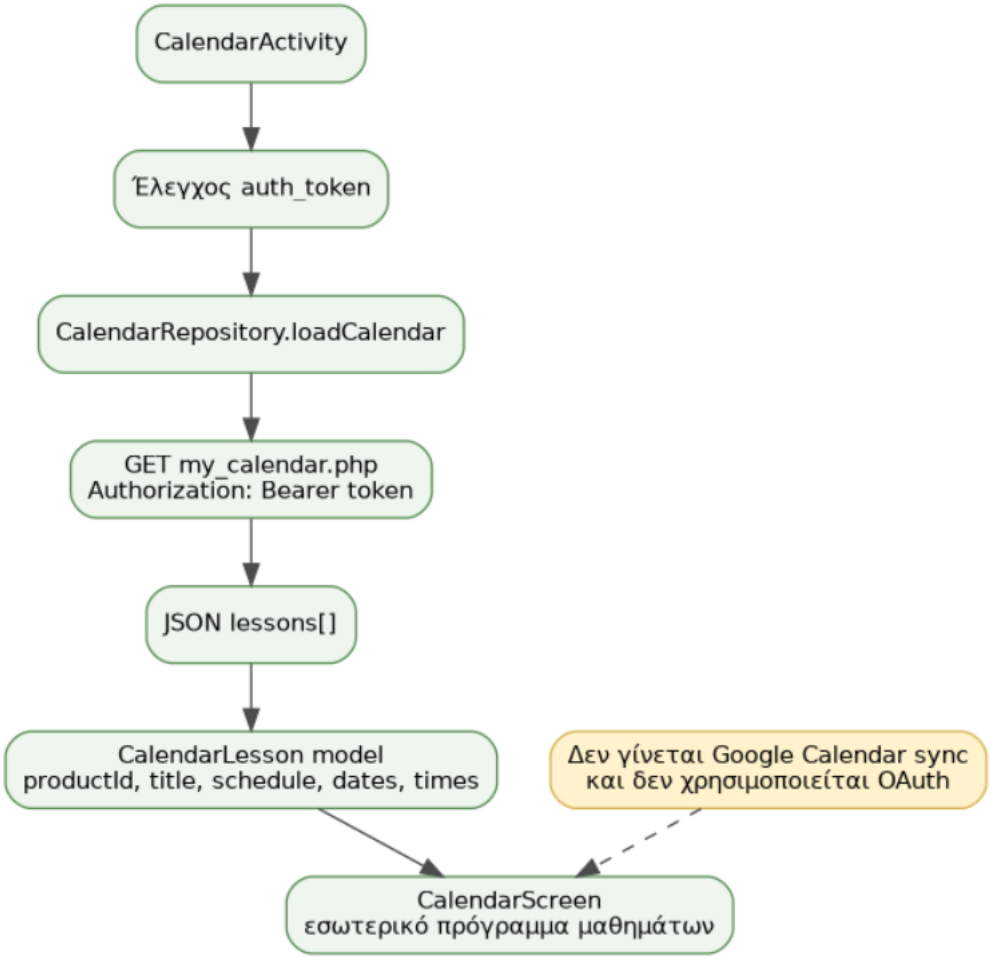
Η λογική ομαδοποιεί τα μαθήματα ανά καθηγητή. Χρησιμοποιείται map όπου το κλειδί είναι το όνομα καθηγητή και η τιμή είναι λίστα μαθημάτων. Παράλληλα αποθηκεύονται TeacherDetails, όπως specialty, short_bio, bio και icon, όταν υπάρχουν. Το αποτέλεσμα ταξινομείται αλφαβητικά και εμφανίζεται στην TeachersScreen. Με αυτόν τον τρόπο, ο χρήστης βλέπει συγκεντρωτικά τους καθηγητές που σχετίζονται με τα μαθήματά του.

5.12. Εσωτερικό ημερολόγιο μαθημάτων

Η CalendarActivity εμφανίζει το εσωτερικό ημερολόγιο μαθημάτων της εφαρμογής. Η λειτουργία αυτή δεν αποτελεί ενσωμάτωση με Google Calendar. Δεν δημιουργούνται συμβάντα στο ημερολόγιο της Google και δεν γίνεται συγχρονισμός OAuth. Αντίθετα, η εφαρμογή καλεί το my_calendar.php endpoint και λαμβάνει πληροφορίες μαθημάτων που σχετίζονται με τον χρήστη.

Το CalendarRepository.loadCalendar ανακτά πίνακα lessons και μετατρέπει κάθε εγγραφή σε CalendarLesson. Τα πεδία περιλαμβάνουν productId, title, schedule, startDate, endDate, startTime, endTime, orderId και status. Με βάση αυτά τα δεδομένα η οθόνη μπορεί να παρουσιάσει πρόγραμμα μαθημάτων, ημερομηνίες και ώρες, καθώς και σχετική κατάσταση παραγγελίας ή συμμετοχής.

Η ύπαρξη εσωτερικού ημερολογίου είναι χρήσιμη διότι συγκεντρώνει στο κινητό το πρόγραμμα του χρήστη χωρίς να απαιτείται εξωτερική υπηρεσία. Παράλληλα, αφήνει ανοιχτή τη δυνατότητα μελλοντικής επέκτασης για εξαγωγή σε εξωτερικά ημερολόγια, χωρίς αυτό να θεωρείται μέρος της παρούσας υλοποίησης.



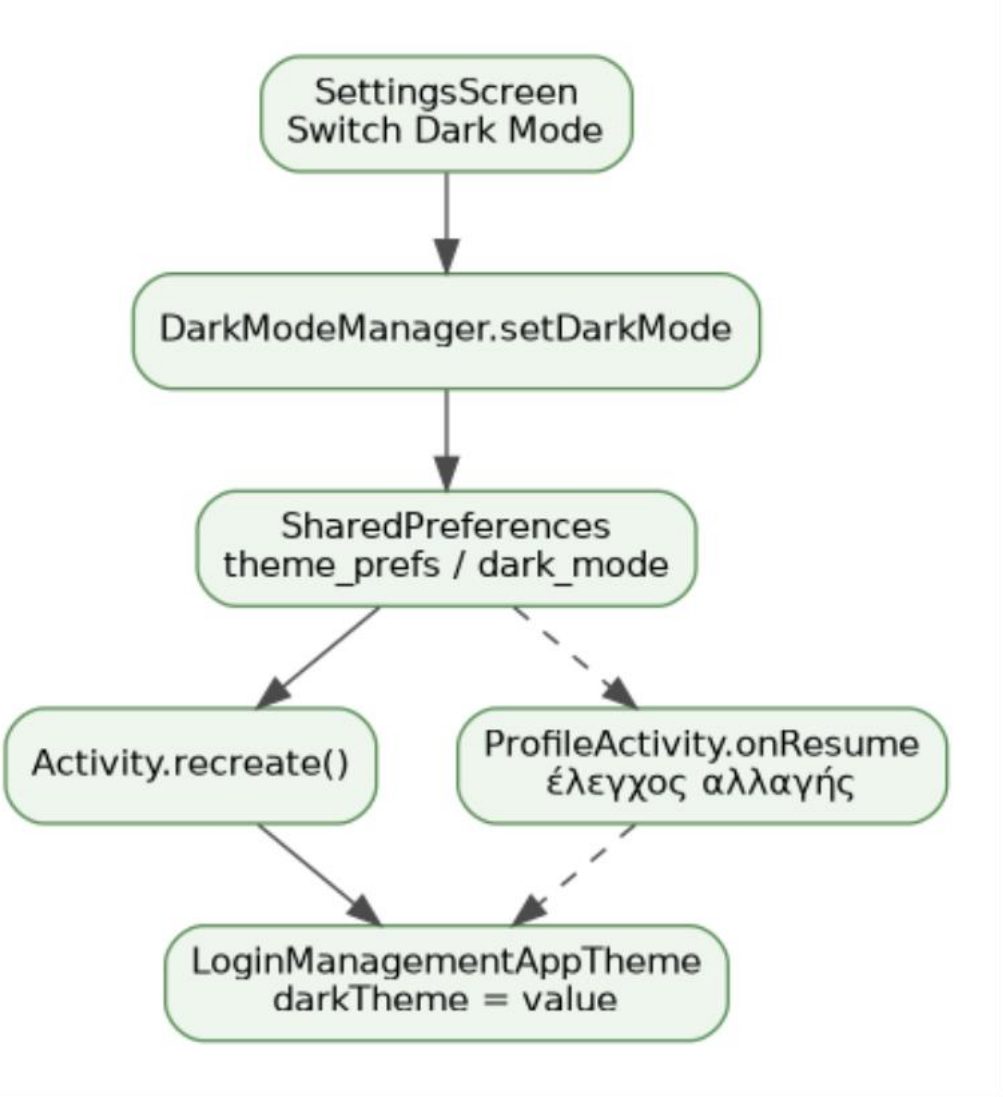
Σχήμα 5.5: Ροή εσωτερικού ημερολογίου μαθημάτων.

5.13. Ρυθμίσεις και Dark Mode

Η SettingsActivity εμφανίζει την SettingsScreen μέσα στο ίδιο theme της εφαρμογής. Η οθόνη περιλαμβάνει ενότητες Appearance, Account και App. Στην ενότητα Appearance υπάρχει επιλογή Dark Mode. Η επιλογή αυτή χρησιμοποιεί το DarkModeManager για αποθήκευση της προτίμησης σε SharedPreferences με όνομα theme_prefs και κλειδί dark_mode.

Όταν ο χρήστης αλλάζει τη ρύθμιση, η εφαρμογή αποθηκεύει τη νέα τιμή και καλεί recreate στην Activity ώστε να εφαρμοστεί άμεσα το νέο theme. Η ProfileActivity επιπλέον ελέγχει στο onResume αν έχει αλλάξει η τιμή dark mode και αναδημιουργεί την οθόνη εφόσον χρειάζεται. Έτσι, η αλλαγή εμφάνισης διατηρείται και μετά την επιστροφή από τις ρυθμίσεις.

Στην ενότητα Account εμφανίζεται επιλογή Change Password, η οποία προς το παρόν εμφανίζει μήνυμα ότι η λειτουργία θα προστεθεί σε επόμενη φάση. Επομένως, δεν παρουσιάζεται ως ολοκληρωμένη αλλαγή κωδικού, αλλά ως σημείο μελλοντικής επέκτασης. Στην ενότητα App υπάρχουν About, Version και Help, με AlertDialog για πληροφορίες και βασική καθοδήγηση.



Σχήμα 5.6: Ροή Dark Mode.

5.14. Θέμα εφαρμογής και επαναχρησιμοποίηση UI

Η εφαρμογή χρησιμοποιεί κοινό LoginManagementAppTheme, το οποίο εφαρμόζεται στις οθόνες μέσω setContent. Κάθε Activity διαβάζει την τρέχουσα τιμή dark mode και περνάει την παράμετρο darkTheme στο theme. Με αυτόν τον τρόπο διατηρείται ενιαία εμφάνιση σε ολόκληρη την εφαρμογή. Τα κύρια UI στοιχεία, όπως cards, buttons και navigation items, ακολουθούν το MaterialTheme.colorScheme ώστε να προσαρμόζονται αυτόματα σε light και dark περιβάλλον.

Η επαναχρησιμοποίηση UI φαίνεται ιδιαίτερα στο drawer navigation. Οι βασικές ενότητες εμφανίζονται με κοινή λογική και κοινά εικονίδια σε περισσότερες οθόνες. Αυτό μειώνει τη γνωστική επιβάρυνση του χρήστη, επειδή η πλοήγηση λειτουργεί με παρόμοιο τρόπο ανεξάρτητα από την τρέχουσα οθόνη.

5.15. Χειρισμός κατάστασης και ασύγχρονη εκτέλεση

Αν και η εφαρμογή δεν χρησιμοποιεί ViewModel σε όλες τις ενότητες, οργανώνει την κατάσταση με απλό και κατανοητό τρόπο. Οι Activity κλάσεις ελέγχουν το token, καλούν repository με callbacks και

στη συνέχεια εμφανίζουν την αντίστοιχη οθόνη. Στις Composable συναρτήσεις χρησιμοποιούνται remember και mutableStateOf για τοπικές καταστάσεις, όπως τα πεδία σύνδεσης, το αν είναι ανοιχτό το drawer ή αν εμφανίζεται dialog.

Οι δικτυακές κλήσεις εκτελούνται μέσα σε Thread ώστε να μη μπλοκάρεται η διεπαφή. Τα callbacks επιστρέφουν αποτελέσματα στην Activity και κάθε αλλαγή UI εκτελείται με runOnUiThread. Η προσέγγιση αυτή είναι απλή, κατανοητή και κατάλληλη για το μέγεθος της πτυχιακής εφαρμογής.

5.16. Συνολική αποτίμηση υλοποίησης

Η υλοποίηση επιτυγχάνει τον στόχο της δημιουργίας Android client που επικοινωνεί με πραγματικό API και παρουσιάζει εξατομικευμένα δεδομένα χρήστη. Οι λειτουργίες είναι οργανωμένες γύρω από την ταυτοποίηση και την ασφαλή ανάκτηση δεδομένων. Η εφαρμογή δεν επιχειρεί να αντικαταστήσει το WordPress/WooCommerce backend, αλλά να λειτουργήσει ως φιλικό κινητό περιβάλλον πρόσβασης στις υπηρεσίες του.

Σημαντικό στοιχείο της υλοποίησης είναι ότι η ίδια βάση δεδομένων παραγγελιών χρησιμοποιείται με διαφορετικούς τρόπους. Οι παραγγελίες εμφανίζονται ως οικονομικές/αγοραστικές εγγραφές, οι συνδρομές ως ενεργά μαθήματα, και οι καθηγητές ως ομαδοποίηση μαθημάτων ανά εκπαιδευτή. Αυτό δείχνει πώς ένας Android client μπορεί να μετασχηματίζει απαντήσεις API σε διαφορετικές εμπειρίες χρήστη.

6. Έλεγχος και Αποτελέσματα

6.1. Στρατηγική ελέγχου

Ο έλεγχος της εφαρμογής πραγματοποιήθηκε με συνδυασμό χειροκίνητων δοκιμών και παρακολούθησης των απαντήσεων API μέσω logs. Επειδή η εφαρμογή βασίζεται σε πραγματικές HTTP κλήσεις, η ορθή λειτουργία κάθε οθόνης ελέγχεται τόσο από την πλευρά του UI όσο και από την πλευρά της απάντησης του endpoint. Σε κάθε σενάριο εξετάζεται αν υπάρχει token, αν η κλήση επιστρέφει success και αν η οθόνη παρουσιάζει σωστά τα δεδομένα.

Οι δοκιμές πραγματοποιήθηκαν σε περιβάλλον Android με ενεργό API endpoint. Για κάθε προστατευμένη οθόνη ελέγχθηκε η συμπεριφορά με έγκυρο token και η συμπεριφορά χωρίς token. Επιπλέον, ελέγχθηκε ότι η αποσύνδεση καθαρίζει την τοπική συνεδρία και επιστρέφει τον χρήστη στη LoginActivity.

ID	Σενάριο	Βήματα	Αναμενόμενο αποτέλεσμα	Κατάσταση

TC-01	Επιτυχής σύνδεση	Εισαγωγή έγκυρων username/password και πάτημα Login	Αποθήκευση token και μετάβαση σε Profile	Επιτυχία
TC-02	Αποτυχία σύνδεσης	Εισαγωγή λάθος στοιχείων	Εμφάνιση Toast με μήνυμα σφάλματος	Επιτυχία
TC-03	Φόρτωση προφίλ	Επιτυχής login και fetch profile	Εμφάνιση username, email, first/last name και εικόνας	Επιτυχία
TC-04	Άνοιγμα QR	Πάτημα Show QR Code	Εμφάνιση QR payload και ιστορικού	Επιτυχία
TC-05	Share QR	Πάτημα share στην QR οθόνη	Καταγραφή ιστορικού και εμφάνιση share chooser	Επιτυχία
TC-06	Παραγγελίες	Άνοιγμα My Orders	Εμφάνιση λίστας παραγγελιών χρήστη	Επιτυχία
TC-07	Συνδρομές	Άνοιγμα My Subscriptions	Εμφάνιση μαθημάτων από completed/on-hold orders	Επιτυχία
TC-08	Καθηγητές	Άνοιγμα My Teachers	Ομαδοποίηση μαθημάτων ανά καθηγητή	Επιτυχία
TC-09	Εσωτερικό ημερολόγιο	Άνοιγμα My Calendar	Εμφάνιση προγράμματος μαθημάτων από API	Επιτυχία

TC-10	Dark Mode	Ενεργοποίηση/απενεργοποίηση dark mode	Άμεση αλλαγή theme και διατήρηση προτίμησης	Επιτυχία
TC-11	Logout	Πάτημα Logout	Διαγραφή token και επιστροφή σε LoginActivity	Επιτυχία
TC-12	Χωρίς token	Άνοιγμα προστατευμένης οθόνης χωρίς token	Μήνυμα Please login again και κλείσιμο οθόνης	Επιτυχία

Πίνακας 6.1: Ενδεικτικά σενάρια ελέγχου.

6.2. Έλεγχος ανά λειτουργία

6.2.1. Σύνδεση και προφίλ

Η σύνδεση ελέγχεται αρχικά με έγκυρα στοιχεία χρήστη. Το αναμενόμενο αποτέλεσμα είναι η λήψη token, η αποθήκευσή του στις SharedPreferences και η κλήση profile.php. Η επιτυχής φόρτωση προφίλ επιβεβαιώνεται οπτικά από την εμφάνιση πραγματικών στοιχείων χρήστη στην ProfileScreen. Με λάθος στοιχεία, η εφαρμογή δεν πρέπει να προχωρά σε ProfileActivity και πρέπει να εμφανίζει μήνυμα σφάλματος.

6.2.2. QR λειτουργία

Η QR λειτουργία ελέγχεται με συνδεδεμένο χρήστη. Η εφαρμογή πρέπει να ανακτά QR payload από qr_access.php, να εμφανίζει QR κωδικό και να φορτώνει το ιστορικό. Κατά το share, πρέπει να γίνεται κλήση save_qr_history.php και το σύστημα Android πρέπει να εμφανίζει επιλογές κοινής χρήσης εικόνας.

6.2.3. Παραγγελίες, συνδρομές και καθηγητές

Οι οθόνες Orders, Subscriptions και Teachers βασίζονται στο orders.php. Ελέγχεται ότι η ίδια API απάντηση μετατρέπεται κατάλληλα ανά οθόνη: στις παραγγελίες εμφανίζονται όλες οι σχετικές αγορές, στις συνδρομές φιλτράρονται συγκεκριμένες καταστάσεις και στους καθηγητές γίνεται ομαδοποίηση ανά teacher_name.

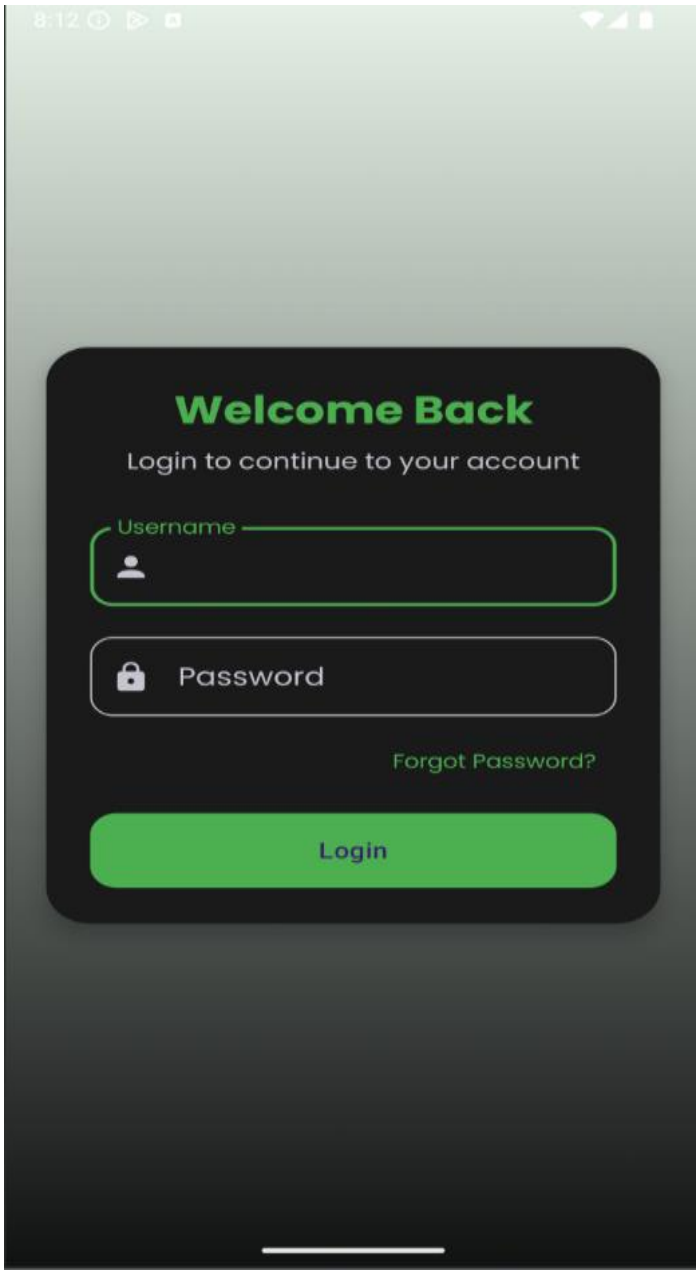
6.2.4. Εσωτερικό ημερολόγιο

Η οθόνη My Calendar ελέγχεται με μαθήματα που διαθέτουν στοιχεία schedule, start_date, end_date, start_time και end_time. Το αναμενόμενο αποτέλεσμα είναι η εμφάνιση του προγράμματος στην εφαρμογή. Η δοκιμή σημειώνει ρητά ότι δεν υπάρχει εξωτερικός συγχρονισμός με Google Calendar.

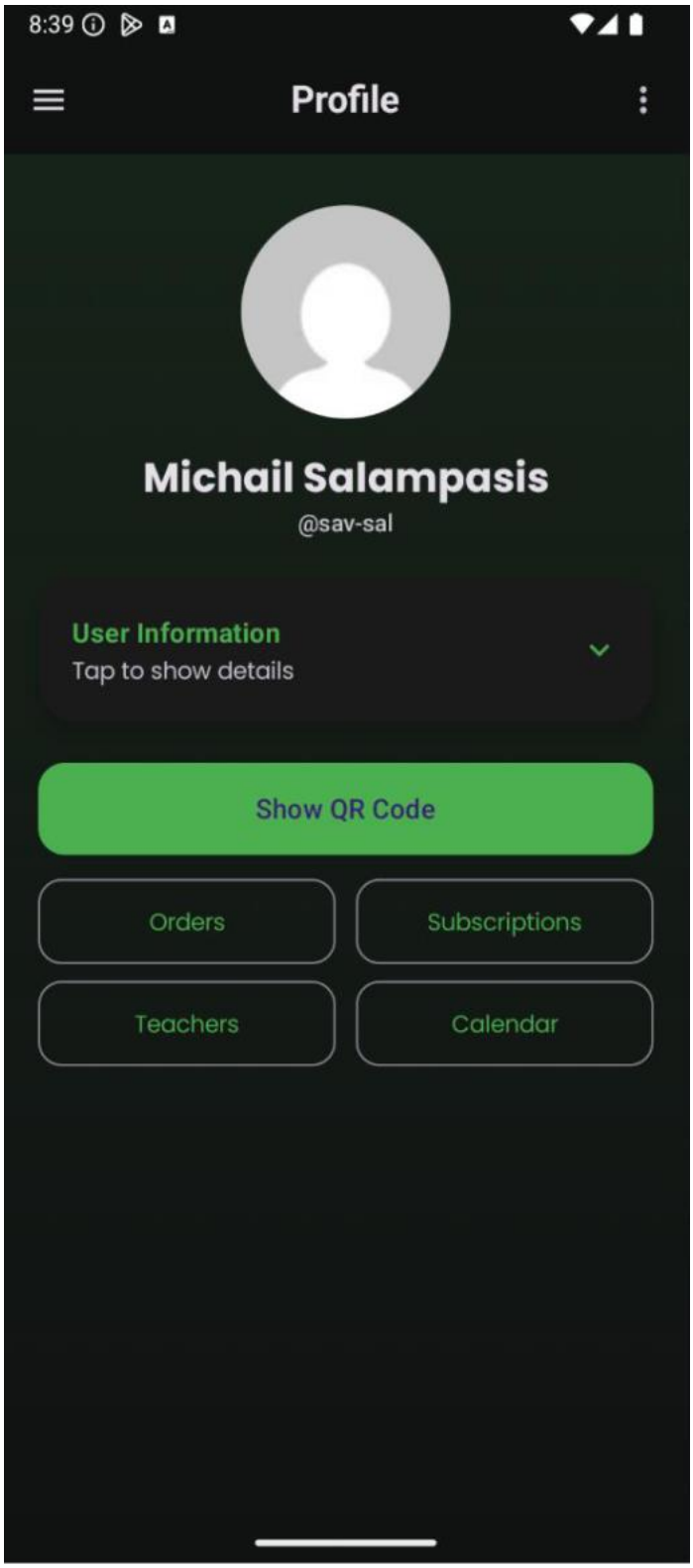
6.2.5. Ρυθμίσεις και dark mode

Η ενεργοποίηση του dark mode ελέγχεται από την SettingsScreen. Η αλλαγή πρέπει να αποθηκεύεται στο DarkModeManager και να εφαρμόζεται μετά το recreate της Activity. Κατά την επιστροφή στην ProfileActivity, η onResume λογική πρέπει να ανιχνεύει την αλλαγή και να ανανεώνει την εμφάνιση.

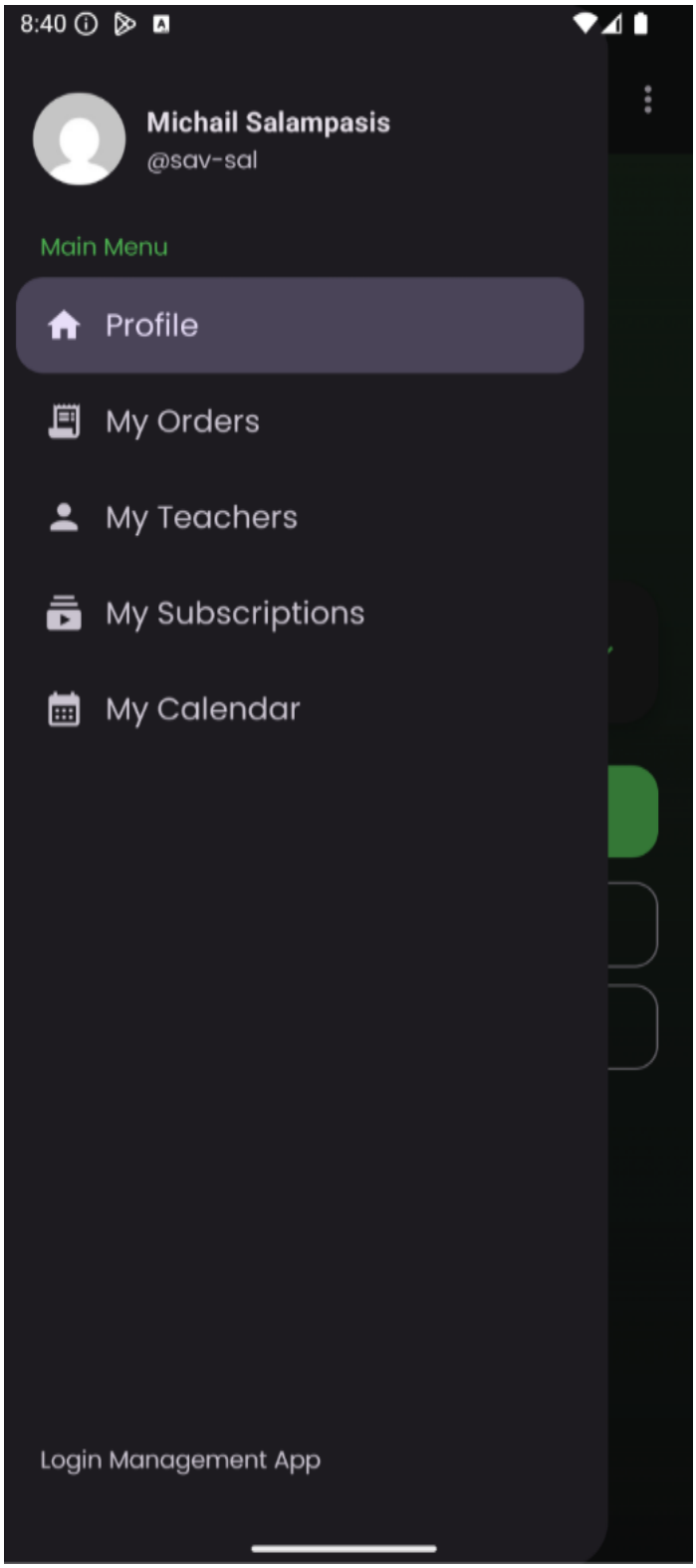
6.3. Καταγραφές οθόνης



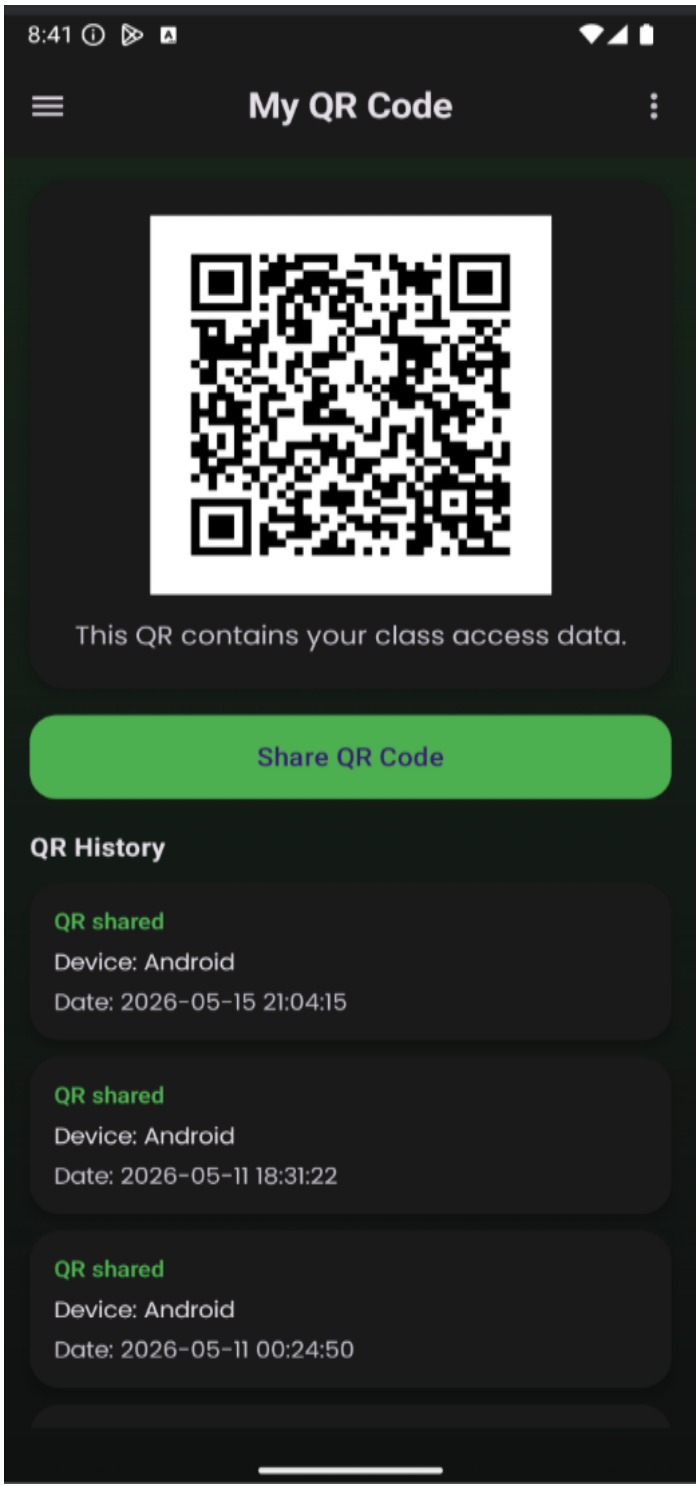
Σχήμα 6.1: Οθόνη σύνδεσης με πραγματικό UI.



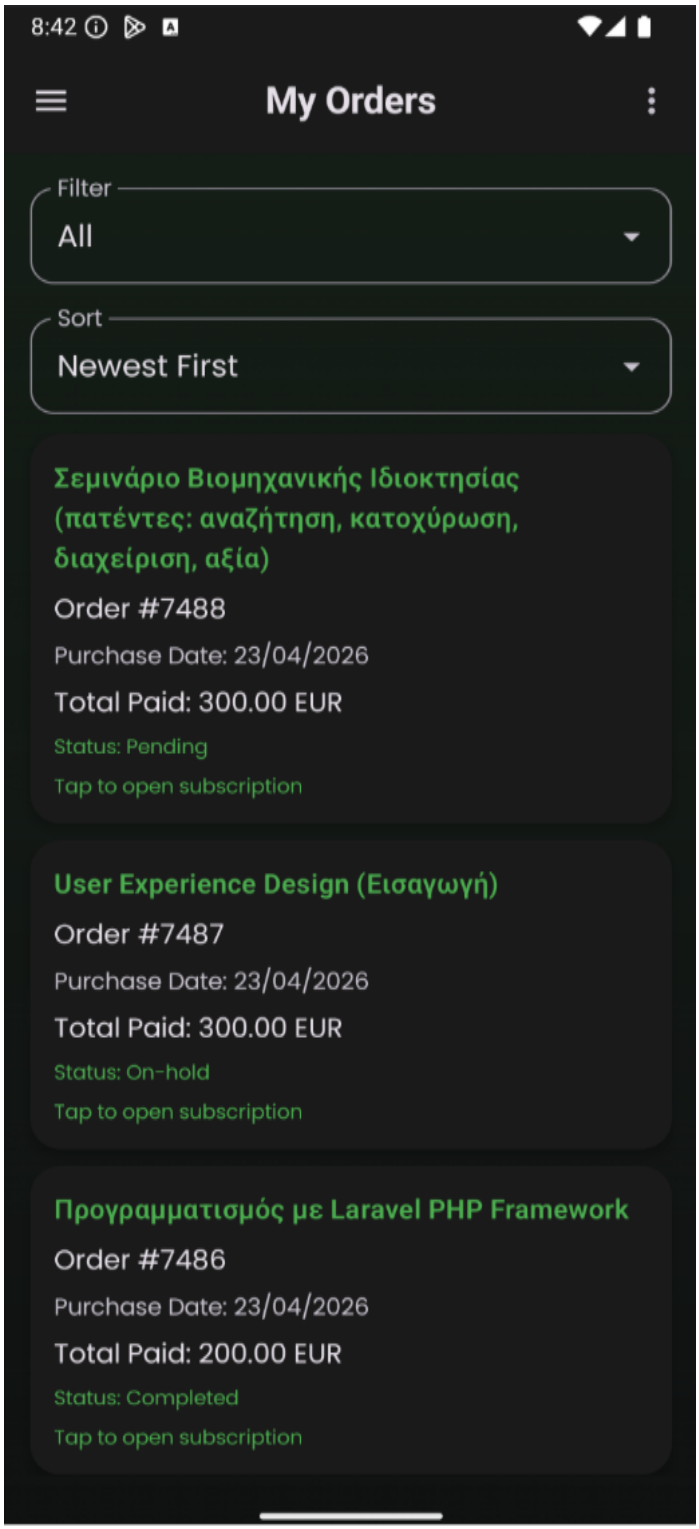
Σχήμα 6.2: Οθόνη προφίλ μετά από επιτυχή σύνδεση.



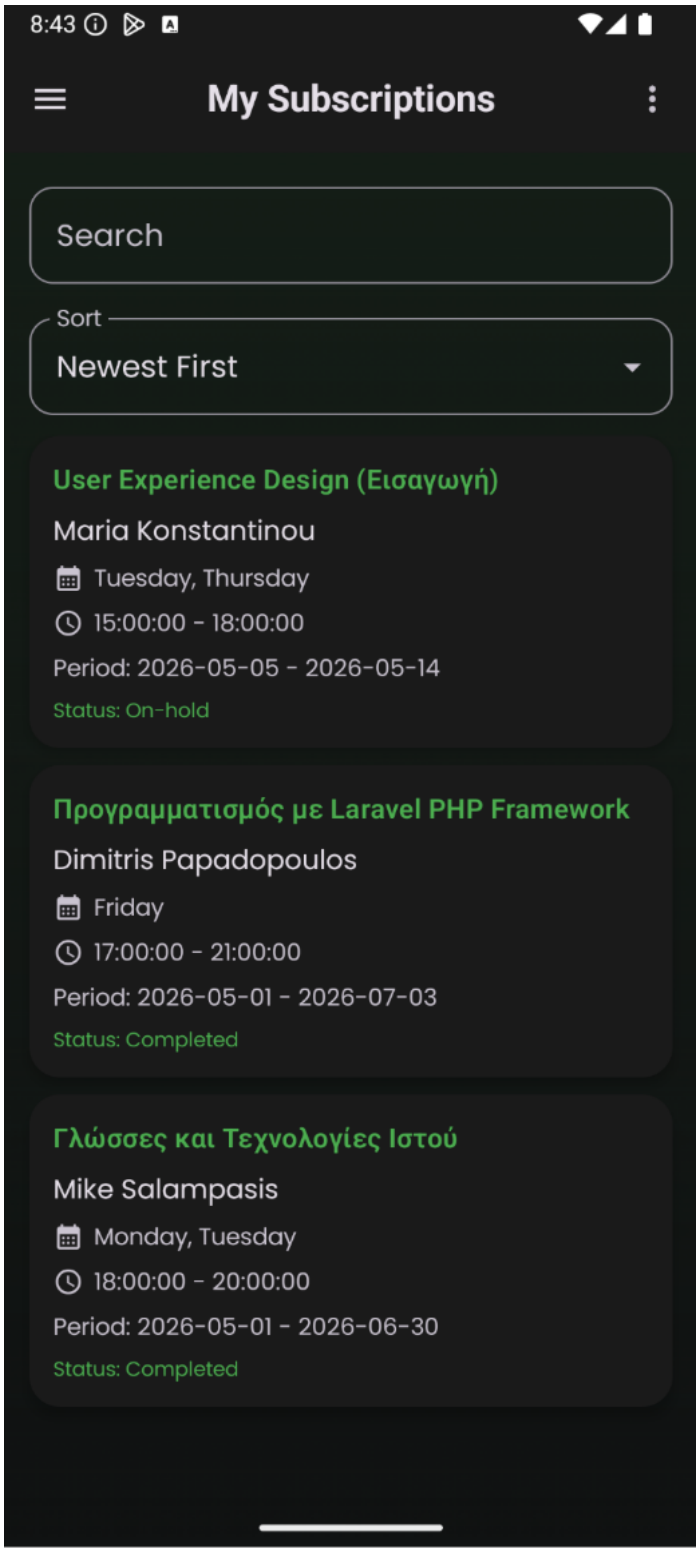
Σχήμα 6.3: Πλαϊνό μενού πλοήγησης.



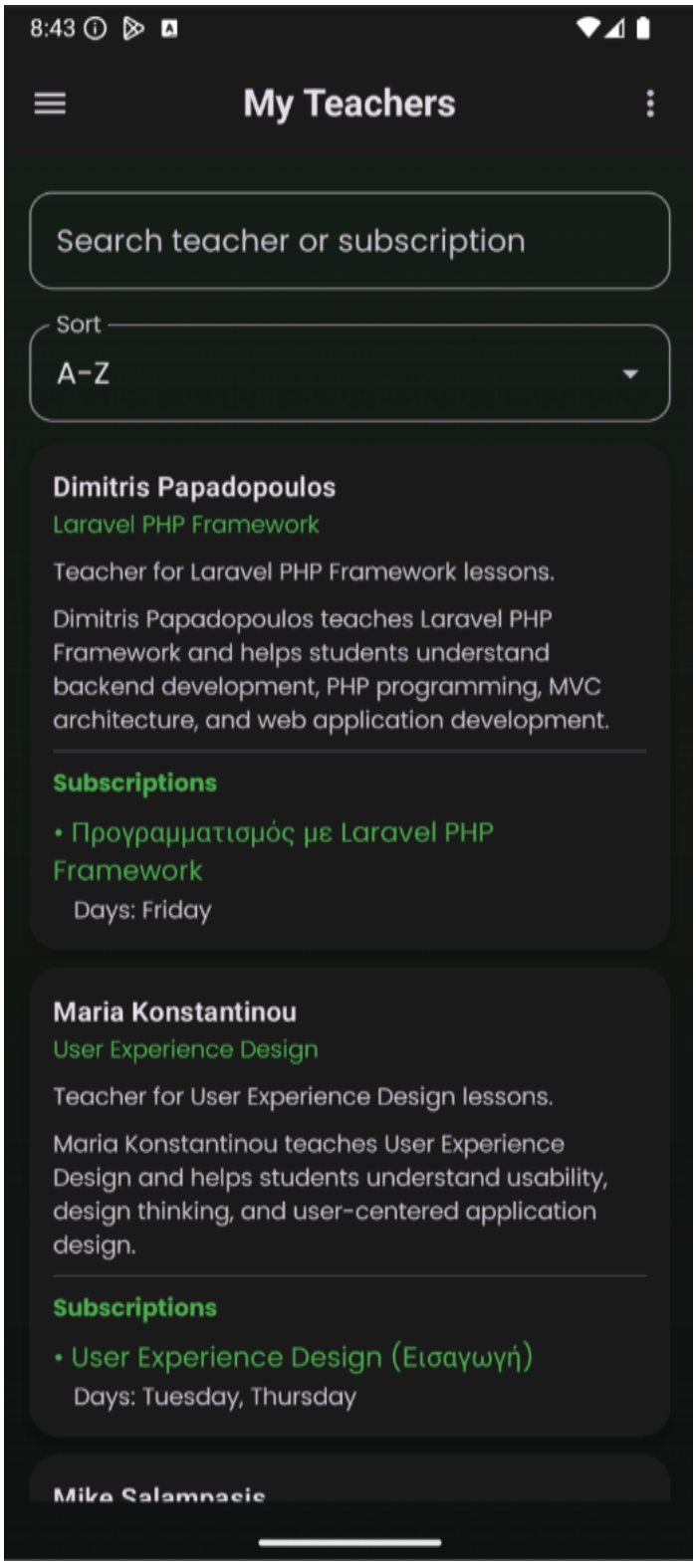
Σχήμα 6.4: Οθόνη QR με ιστορικό.



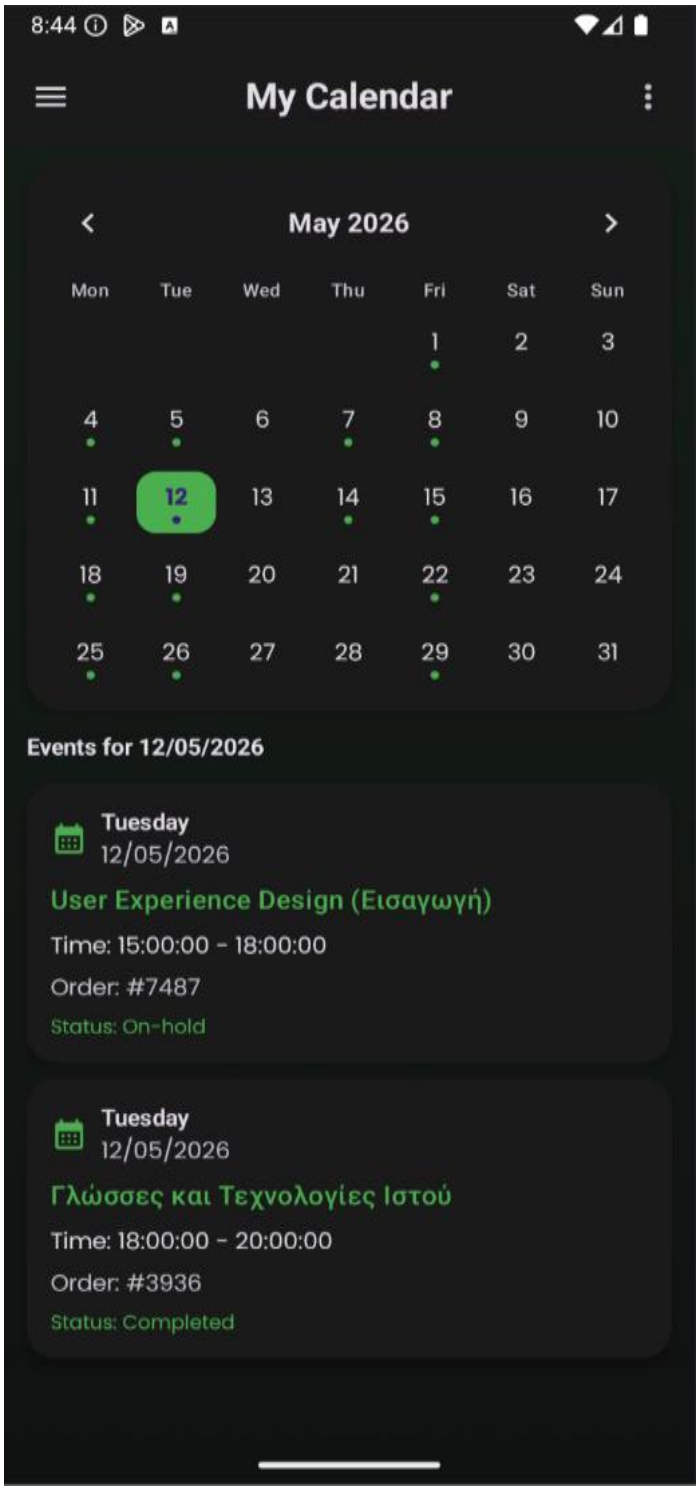
Σχήμα 6.5: Οθόνη παραγγελιών.



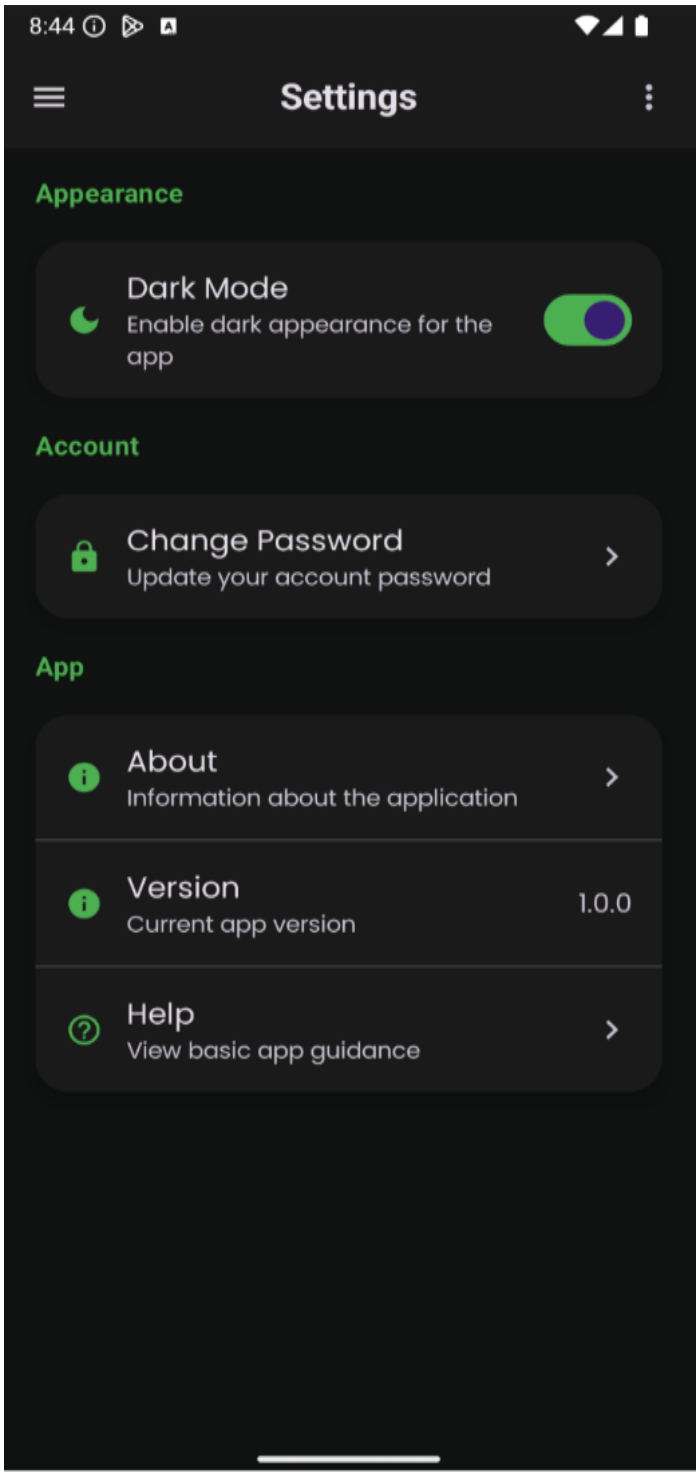
Σχήμα 6.6: Οθόνη συνδρομών/μαθημάτων.



Σχήμα 6.7: Οθόνη καθηγητών.



Σχήμα 6.8: Εσωτερικό ημερολόγιο μαθημάτων.



Σχήμα 6.9: Οθόνη ρυθμίσεων.

6.4. Αποτελέσματα ελέγχου

Τα αποτελέσματα των δοκιμών επιβεβαιώνουν ότι όλες οι βασικές λειτουργίες της εφαρμογής εκτελούνται σύμφωνα με τις απαιτήσεις. Οι προστατευμένες οθόνες απαιτούν token, τα δεδομένα εμφανίζονται μόνο μετά από επιτυχή API απάντηση και η αποσύνδεση καθαρίζει την τοπική συνεδρία. Η εφαρμογή διατηρεί συνεπή εμπειρία χρήστη σε light και dark εμφάνιση, ενώ οι οθόνες που βασίζονται στο orders.php αξιοποιούν την ίδια απάντηση API με διαφορετική λογική παρουσίασης.

7. Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1.Σύνοψη υλοποίησης

Η πτυχιακή εργασία οδήγησε στην υλοποίηση μιας λειτουργικής εφαρμογής Android που συνδέεται με προσαρμοσμένο WordPress/PHP API και εμφανίζει εξατομικευμένα δεδομένα χρήστη. Η εφαρμογή καλύπτει σύνδεση, προφίλ, QR πρόσβαση, ιστορικό QR, παραγγελίες, συνδρομές/μαθήματα, καθηγητές, εσωτερικό ημερολόγιο και ρυθμίσεις εμφάνισης. Η υλοποίηση βασίστηκε σε Kotlin, Jetpack Compose και Material 3, με απλή αλλά ξεκάθαρη αρχιτεκτονική διαχωρισμού UI και δικτυακής λογικής.

7.2. Τεχνικά οφέλη

Το σημαντικότερο τεχνικό όφελος είναι ότι η εφαρμογή λειτουργεί ως πραγματικός client υπάρχοντος συστήματος και όχι ως στατική επίδειξη. Τα δεδομένα λαμβάνονται από endpoints, το token αποθηκεύεται τοπικά και οι οθόνες παρουσιάζουν πληροφορίες που προέρχονται από το backend. Επιπλέον, η οργάνωση σε repositories καθιστά πιο εύκολη τη μελλοντική αλλαγή της δικτυακής λογικής, για παράδειγμα αντικατάσταση του HttpURLConnection με Retrofit.

Η υλοποίηση δείχνει επίσης πώς το ίδιο endpoint μπορεί να αξιοποιηθεί με πολλαπλούς τρόπους. Το orders.php χρησιμοποιείται τόσο για παραγγελίες όσο και για συνδρομές και καθηγητές, με διαφορετική επεξεργασία στον client. Αυτό αναδεικνύει την αξία του repository layer ως σημείου μετασχηματισμού δεδομένων.

7.3.Περιορισμοί

Η εφαρμογή έχει ορισμένους περιορισμούς που πρέπει να καταγραφούν με σαφήνεια. Πρώτον, η επικοινωνία γίνεται μέσω HTTP στο περιβάλλον δοκιμών και όχι υποχρεωτικά HTTPS. Δεύτερον, το token αποθηκεύεται σε απλές SharedPreferences. Τρίτον, η εφαρμογή δεν διαθέτει offline βάση δεδομένων και βασίζεται στη διαθεσιμότητα του API. Τέταρτον, η αλλαγή κωδικού εμφανίζεται ως μελλοντική δυνατότητα και όχι ως ολοκληρωμένη λειτουργία. Πέμπτον, το ημερολόγιο είναι εσωτερική προβολή μαθημάτων και όχι ενσωμάτωση Google Calendar.

7.4.Μελλοντικές επεκτάσεις

Μελλοντικά θα μπορούσαν να υλοποιηθούν επιπλέον λειτουργίες, χωρίς όμως να θεωρούνται μέρος της παρούσας εργασίας. Ενδεικτικά, θα μπορούσε να προστεθεί κρυπτογραφημένη αποθήκευση token, πλήρης αλλαγή κωδικού μέσα από την εφαρμογή, push notifications, offline cache με Room, βελτιωμένη ασφάλεια QR με υπογεγραμμένα προσωρινά tokens και πιθανή εξαγωγή προγράμματος σε εξωτερικό ημερολόγιο. Οι επεκτάσεις αυτές θα πρέπει να σχεδιαστούν ως νέα στάδια και να τεκμηριωθούν ξεχωριστά.

7.5.Κατακλείδα

Συνολικά, το έργο επιτυγχάνει τον στόχο μιας σύγχρονης Android εφαρμογής που συνδέεται με πραγματικό backend, προσφέρει οργανωμένη διεπαφή και παρουσιάζει χρήσιμες πληροφορίες στον τελικό χρήστη. Η εργασία δείχνει πώς μπορεί να δημιουργηθεί ένας mobile client πάνω από υπάρχον WordPress/WooCommerce σύστημα, αξιοποιώντας Kotlin, Compose και custom REST/PHP endpoints.

8. Παραπομπές και Πρόσβαση σε Τεκμηρίωση

8.1.Πρόσβαση στον πηγαίο κώδικα

Ο πηγαίος κώδικας της εφαρμογής βρίσκεται στο GitHub repository: https://github.com/kostasstrap/LoginManagementApp. Το repository αποτελεί την κύρια τεκμηρίωση της υλοποίησης, καθώς περιέχει τις Activity κλάσεις, τις Composable οθόνες, τα repositories, τα models, τα utilities και τα αρχεία διαμόρφωσης Gradle.

8.2.Οδηγίες αναπαραγωγής

Για την αναπαραγωγή της εφαρμογής απαιτείται Android Studio, ενεργό Android SDK, πρόσβαση στο repository και διαθέσιμο API endpoint. Ο χρήστης ανοίγει το project στο Android Studio, συγχρονίζει το Gradle, ελέγχει την τιμή BASE_URL στην κλάση ApiClient και εκτελεί την εφαρμογή σε emulator ή φυσική συσκευή. Αν το API βρίσκεται σε διαφορετική διεύθυνση, η BASE_URL πρέπει να ενημερωθεί ανάλογα.

8.3. Βιβλιογραφία

[1] Google LLC, “Android Developers Documentation,” Android Developers. [Online]. Available: https://developer.android.com/docs. [Accessed: May 28, 2026].

[2] Google LLC, “Jetpack Compose Documentation,” Android Developers. [Online]. Available: https://developer.android.com/develop/ui/compose. [Accessed: May 28, 2026].

[3] Google LLC, “Material Design 3,” Material Design. [Online]. Available: https://m3.material.io/. [Accessed: May 28, 2026].

[4] JetBrains, “Kotlin Documentation,” JetBrains. [Online]. Available: https://kotlinlang.org/docs/home.html. [Accessed: May 28, 2026].

[5] WordPress.org, “WordPress Developer Resources,” WordPress Developer Resources. [Online]. Available: https://developer.wordpress.org/. [Accessed: May 28, 2026].

[6] WooCommerce, “WooCommerce REST API Documentation,” WooCommerce Documentation. [Online]. Available: https://woocommerce.github.io/woocommerce-rest-api-docs/. [Accessed: May 28, 2026].

[7] ZXing Project, “ZXing: Barcode Image Processing Library,” GitHub repository. [Online]. Available: https://github.com/zxing/zxing. [Accessed: May 28, 2026].

[8] JourneyApps, “ZXing Android Embedded,” GitHub repository. [Online]. Available: https://github.com/journeyapps/zxing-android-embedded. [Accessed: May 28, 2026].

[9] Coil Contributors, “Coil: Image Loading for Android and Compose,” Coil Documentation. [Online]. Available: https://coil-kt.github.io/coil/. [Accessed: May 28, 2026].

[10] K. Straplakis, “LoginManagementApp,” GitHub repository, 2026. [Online]. Available: https://github.com/kostasstrap/LoginManagementApp. [Accessed: May 28, 2026].

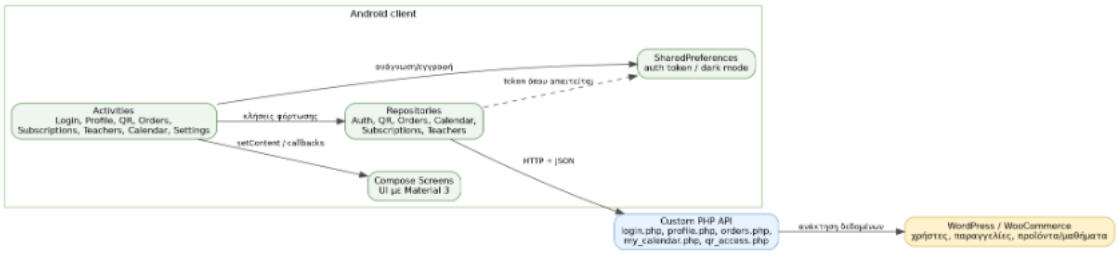
8.4.Λεξιλόγιο τεχνικών όρων

Όρος	**Ελληνική εξήγηση**
Activity	Βασικό συστατικό Android που φιλοξενεί οθόνη ή ροή οθόνης.
Composable	Συνάρτηση Jetpack Compose που περιγράφει τμήμα διεπαφής χρήστη.
Repository	Κλάση που απομονώνει την πρόσβαση σε δεδομένα ή API.
API	Διεπαφή προγραμματισμού εφαρμογών για επικοινωνία μεταξύ συστημάτων.
Endpoint	Συγκεκριμένη διεύθυνση/διαδρομή API που εξυπηρετεί μία λειτουργία.
Token	Διακριτικό πρόσβασης που αποδεικνύει ότι ο χρήστης έχει συνδεθεί.
SharedPreferences	Μηχανισμός τοπικής αποθήκευσης απλών τιμών στο Android.
QR Code	Δισδιάστατος κώδικας που μπορεί να περιέχει δεδομένα πρόσβασης ή αναγνώρισης.
WooCommerce	Σύστημα ηλεκτρονικού εμπορίου πάνω σε WordPress.
Dark Mode	Σκοτεινή εμφάνιση διεπαφής με διαφορετική χρωματική παλέτα.
JSON	Μορφή ανταλλαγής δεδομένων που χρησιμοποιείται στις αποκρίσεις του API.
HttpURLConnection	Κλάση Java/Kotlin για δημιουργία HTTP συνδέσεων χωρίς πρόσθετη βιβλιοθήκη.

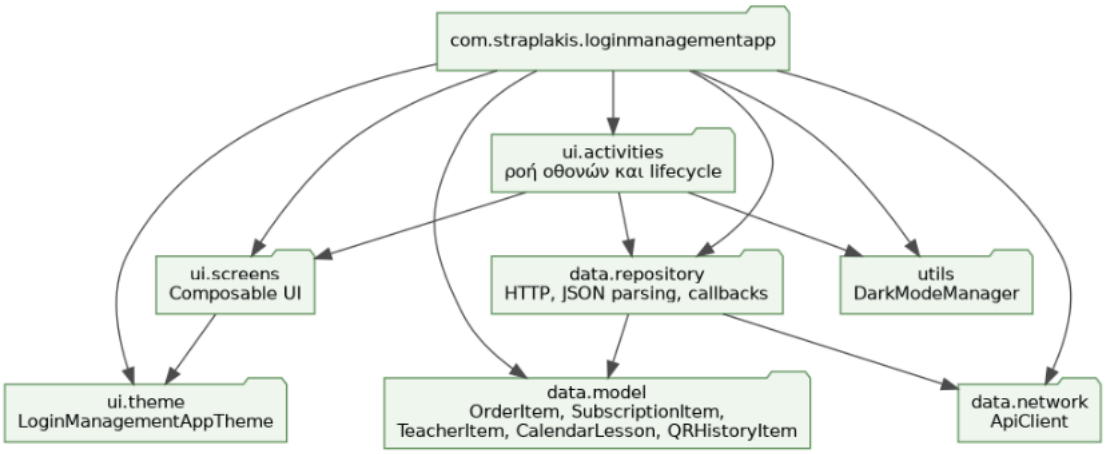
Πίνακας 8.1: Βασικό λεξιλόγιο τεχνικών όρων.

Παράρτημα Α: Διαγράμματα Συστήματος

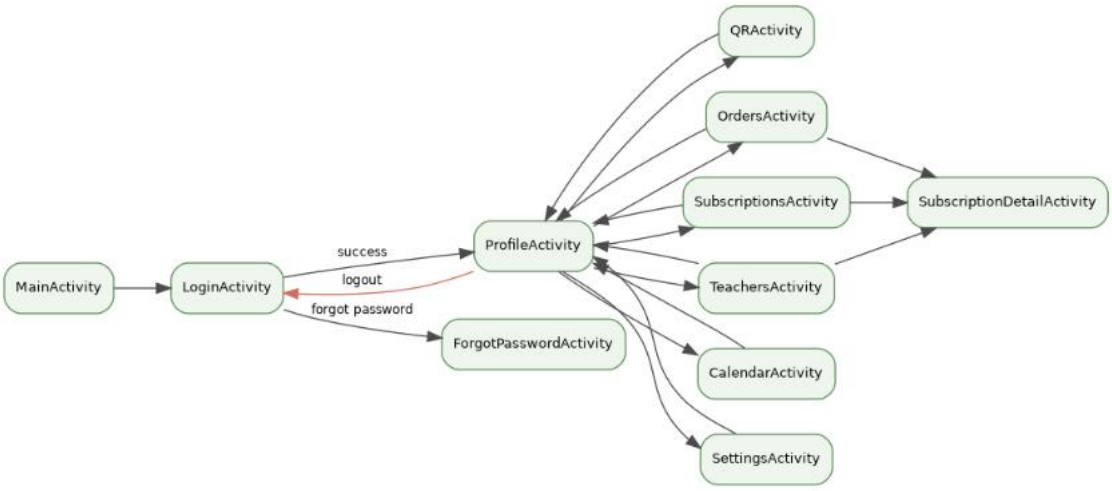
Στο παράρτημα συγκεντρώνονται τα βασικά διαγράμματα που περιγράφουν τη δομή και τις ροές της εφαρμογής. Τα διαγράμματα δεν εισάγουν νέα λειτουργικότητα, αλλά αποτυπώνουν γραφικά όσα παρουσιάστηκαν στα προηγούμενα κεφάλαια.



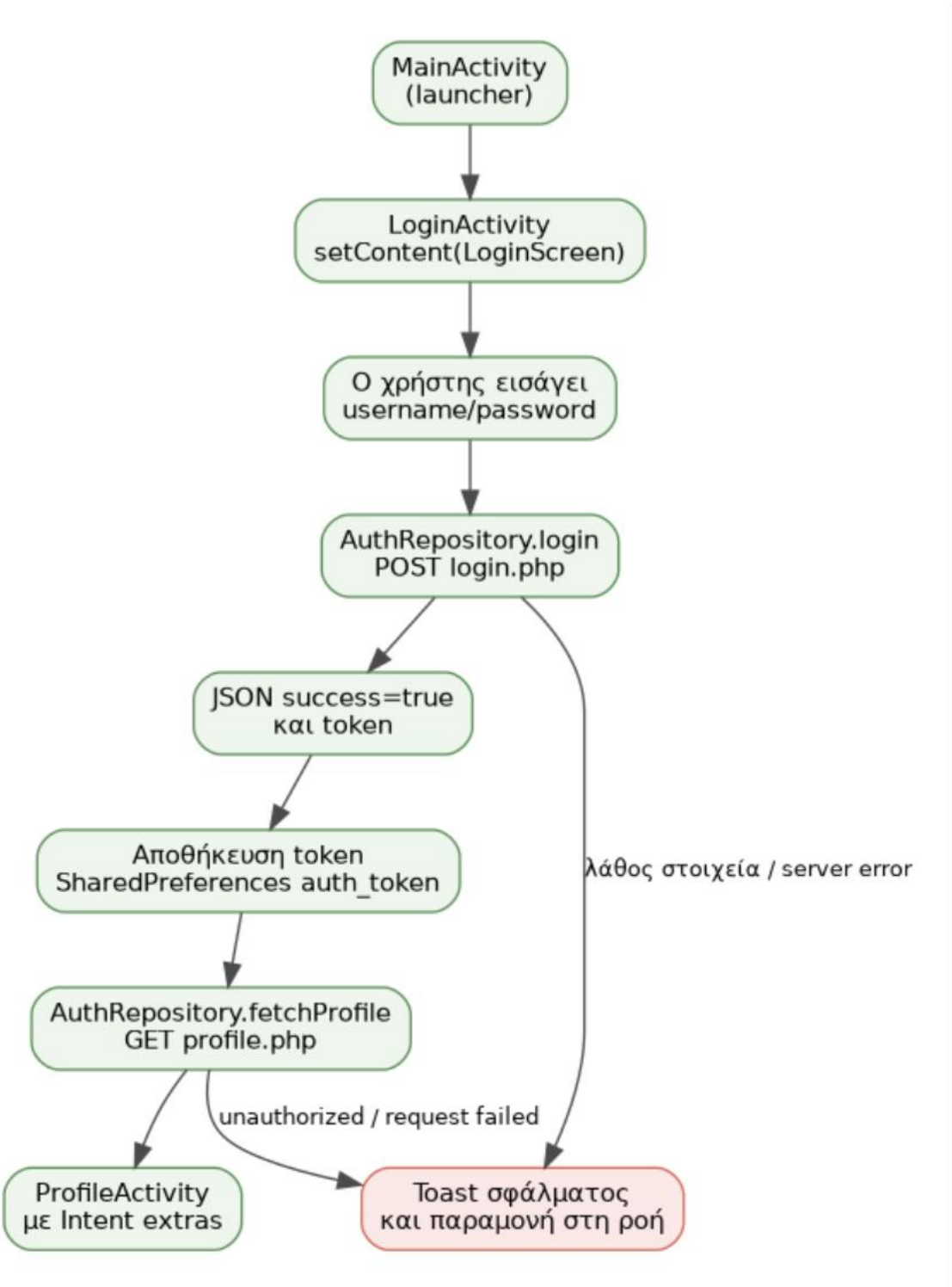
Σχήμα Α.1: Αρχιτεκτονική συστήματος Android client - PHP API - WordPress/WooCommerce.



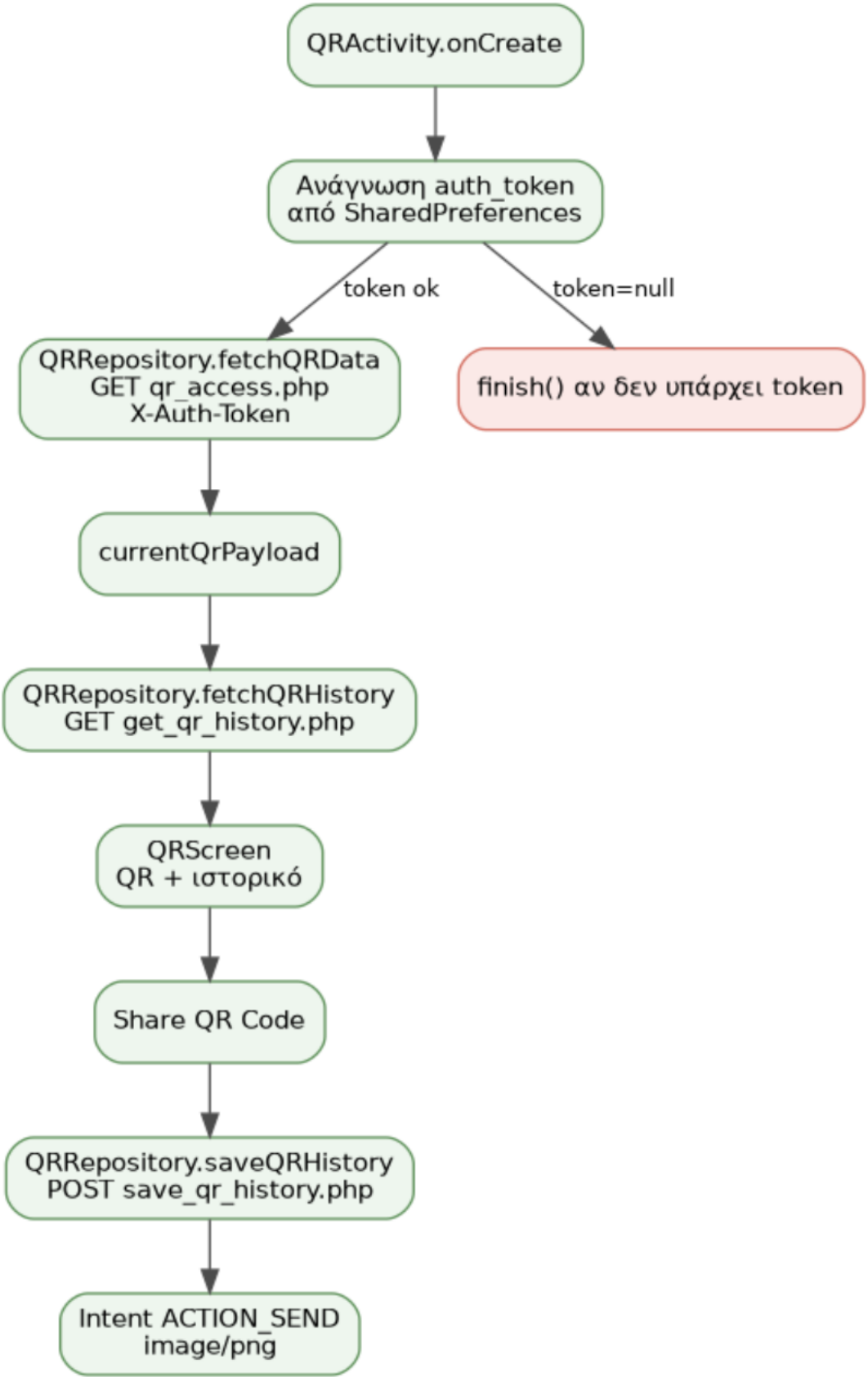
Σχήμα Α.2: Δομή πακέτων του Android project.



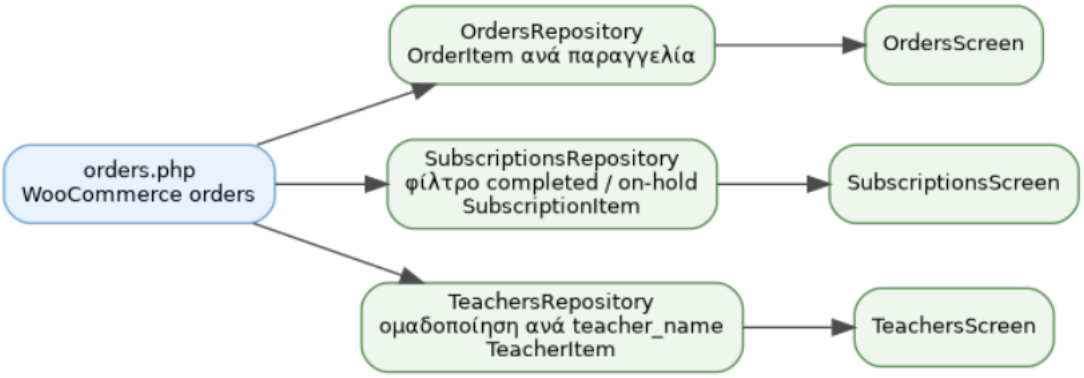
Σχήμα Α.3: Πλοήγηση Activity κλάσεων.



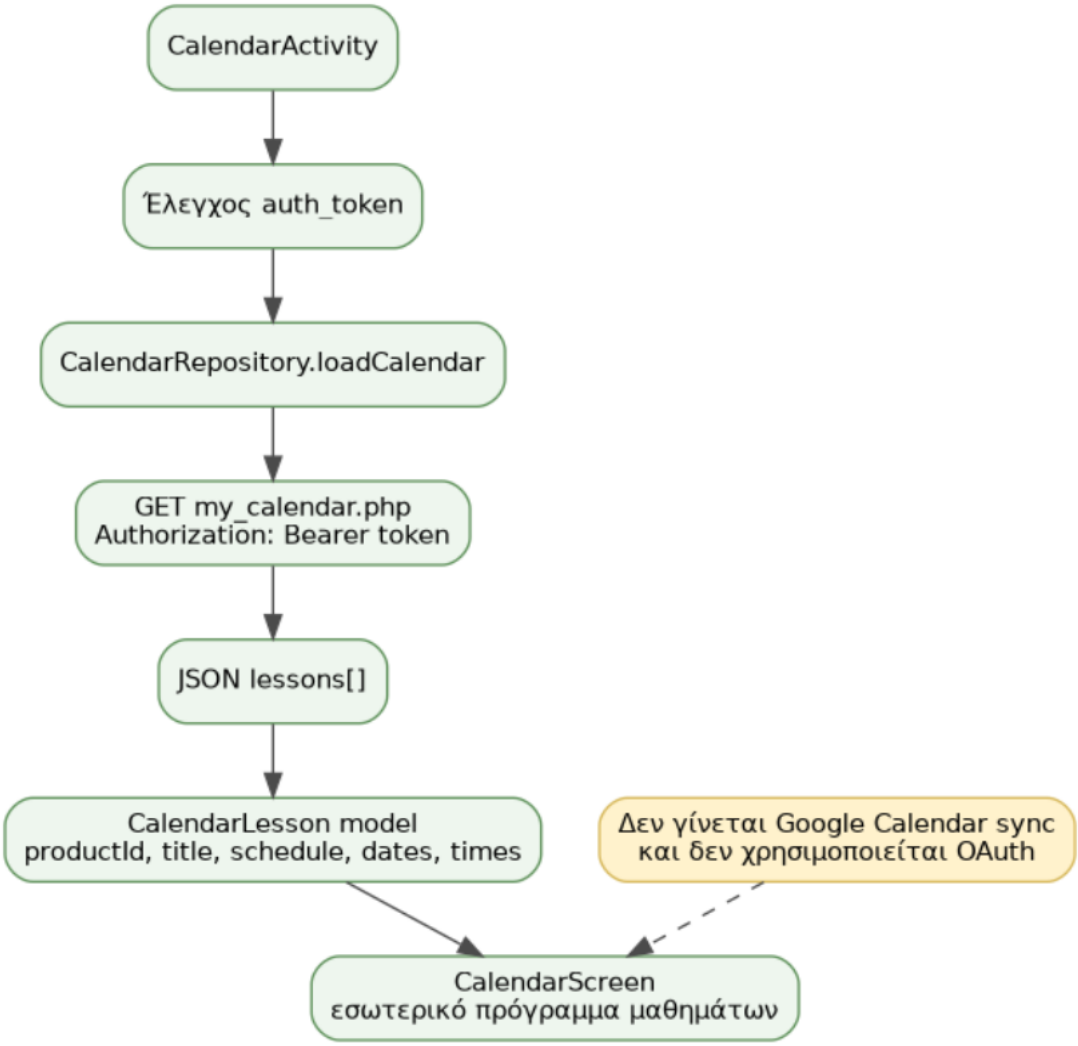
Σχήμα Α.4: Ροή σύνδεσης και ανάκτησης προφίλ.



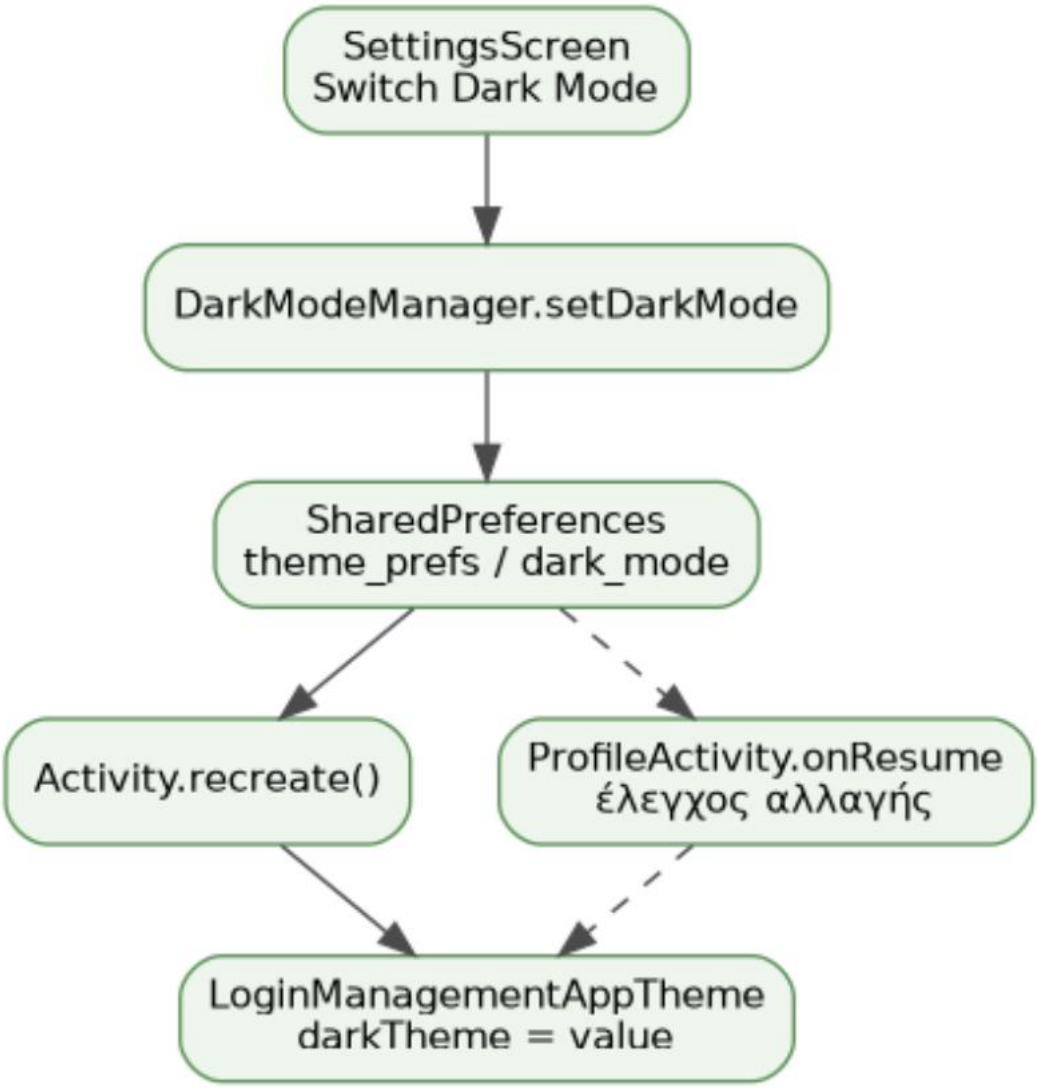
Σχήμα Α.5: Ροή QR λειτουργίας.



Σχήμα Α.6: Μετασχηματισμός δεδομένων orders σε orders, subscriptions και teachers.



Σχήμα Α.7: Ροή εσωτερικού ημερολογίου μαθημάτων.



Σχήμα Α.8: Ροή αποθήκευσης και εφαρμογής Dark Mode.

Παράρτημα Β: Αναλυτική Τεχνική Τεκμηρίωση και Κώδικας

Το παρόν παράρτημα τεκμηριώνει με μεγαλύτερη λεπτομέρεια κρίσιμα σημεία της υλοποίησης. Τα αποσπάσματα κώδικα προέρχονται από τις βασικές κλάσεις του Android project και συνοδεύονται από σύντομη ερμηνεία του ρόλου τους.

Β.1 LoginActivity.handleLogin

Η μέθοδος συντονίζει την είσοδο του χρήστη. Δεν εκτελεί η ίδια HTTP αίτημα, αλλά καλεί το AuthRepository. Σε επιτυχία αποθηκεύει το token και συνεχίζει στην ανάκτηση προφίλ.

```
private fun handleLogin(username: String, password: String) {
  authRepository.login(
    username = username,
    password = password,
    onSuccess = { token ->
      runOnUiThread {
        val sharedPref = getSharedPreferences("auth", MODE_PRIVATE)
        sharedPref.edit().putString("auth_token", token).apply()
        fetchProfileWithToken(token)
      }
    },
    onError = { message ->
      runOnUiThread {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
      }
    }
  )
}
```

Σχήμα Β.1: Απόσπασμα κώδικα σύνδεσης και αποθήκευσης token.

B.2 AuthRepository.login

Το απόσπασμα δείχνει την πλήρη ροή HTTP σύνδεσης: δημιουργία σύνδεσης, ορισμός μεθόδου POST, αποστολή δεδομένων, ανάγνωση απάντησης και ανάλυση JSON.

```
val url = URL(ApiClient.LOGIN_URL)
val connection = url.openConnection() as HttpURLConnection
connection.requestMethod = "POST"
connection.doOutput = true
connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded")

val postData =
  "username=${URLEncoder.encode(username, "UTF-
8")}&password=${URLEncoder.encode(password, "UTF-8")}"
connection.outputStream.use { it.write(postData.toByteArray()) }

val responseCode = connection.responseCode
val responseStream = if (responseCode in 200..299) connection.inputStream else
connection.errorStream
val response = responseStream.bufferedReader().use { it.readText() }
val json = JSONObject(response)
if (json.getBoolean("success")) {
  val token = json.getJSONObject("data").getString("token")
  onSuccess(token)
} else {
  onError(json.getString("message"))
}
```

Σχήμα Β.2: Απόσπασμα κώδικα AuthRepository.login με form-urlencoded POST.

B.3 AuthRepository.fetchProfile

Η ανάκτηση προφίλ είναι προστατευμένη κλήση. Το token στέλνεται ως Authorization Bearer και η απάντηση μετατρέπεται σε επιμέρους πεδία που περνούν στην ProfileActivity.

```
connection.requestMethod = "GET"
connection.setRequestProperty("Authorization", "Bearer $token")
val response = responseStream.bufferedReader().use { it.readText() }
val json = JSONObject(response)
if (json.getBoolean("success")) {
  val user = json.getJSONObject("data").getJSONObject("user")
  onSuccess(
    user.getString("username"),
    user.getString("email"),
    user.getString("firstname"),
    user.getString("lastname"),
    user.getString("role"),
    user.getString("profile_picture")
  )
}
```

Σχήμα Β.3: Απόσπασμα κώδικα ανάκτησης προφίλ με Bearer token.

Β.4 ProfileScreen drawer navigation

Το drawer item δείχνει την αρχή λειτουργίας του πλαϊνού μενού. Η Composable κλείνει το drawer και καλεί callback που παρέχεται από την Activity.

```
NavigationDrawerItem(
  label = { Text("My Orders") },
  icon = { Icon(Icons.Default.ReceiptLong, contentDescription = null) },
  selected = false,
  shape = RoundedCornerShape(16.dp),
  onClick = {
    scope.launch { drawerState.close() }
    onOpenOrders()
  }
)
```

Σχήμα Β.4: Απόσπασμα κώδικα ProfileScreen drawer navigation.

Β.5 QRRepository.fetchQRHistory

Το ιστορικό QR φορτώνεται με X-Auth-Token και μετατρέπεται σε λίστα QRHistoryItem. Η χρήση optString βοηθά ώστε η εφαρμογή να αντέχει ελλιπή πεδία.

```
val url = URL(ApiClient.QR_HISTORY_URL)
val connection = url.openConnection() as HttpURLConnection
connection.requestMethod = "GET"
connection.setRequestProperty("X-Auth-Token", token)
val response = responseStream.bufferedReader().use { it.readText() }
val json = JSONObject(response)
val history = mutableListOf<QRHistoryItem>()
if (json.getBoolean("success")) {
  val array = json.getJSONObject("data").getJSONArray("history")
  for (i in 0 until array.length()) {
    val item = array.getJSONObject(i)
    history.add(QRHistoryItem(
      id = item.getInt("id"),
      scanType = item.optString("scan_type", ""),
```

```
        deviceInfo = item.optString("device_info", ""),
        scannedAt = item.optString("scanned_at", "")
      ))
    }
}
onSuccess(history)
```

Σχήμα Β.5: Απόσπασμα κώδικα QRRepository.fetchQRHistory.

Β.6 TeachersRepository grouping

Η ίδια απάντηση orders μετασχηματίζεται σε λίστα καθηγητών. Το map οργανώνει τα μαθήματα ανά teacherName και αποφεύγει διπλότυπα productId.

```
val teacherLessonsMap = mutableMapOf<String, MutableList<TeacherLessonItem>>()
val teacherDetailsMap = mutableMapOf<String, TeacherDetails>()

for (i in 0 until ordersArray.length()) {
  val orderObj = ordersArray.getJSONObject(i)
  val status = orderObj.getString("status").lowercase()
  if (status in allowedStatuses) {
    val lessonObj = orderObj.getJSONObject("lesson")
    val teacherName = lessonObj.optString("teacher_name", "").trim()
    val lessonTitle = lessonObj.optString("title", "").trim()
    if (teacherName.isNotBlank()) {
      val teacherLessons = teacherLessonsMap.getOrPut(teacherName) { mutableListOf() }
      val productId = lessonObj.optInt("product_id", 0)
      if (teacherLessons.none { it.productId == productId }) {
        teacherLessons.add(TeacherLessonItem(productId, lessonTitle,
lessonObj.optString("schedule", "")))
      }
    }
  }
}
```

Σχήμα Β.6: Απόσπασμα κώδικα TeachersRepository με ομαδοποίηση καθηγητών.

Β.7 CalendarRepository.loadCalendar

Το εσωτερικό ημερολόγιο βασίζεται αποκλειστικά στο custom API και όχι σε Google Calendar. Το repository μετατρέπει lessons[] σε CalendarLesson αντικείμενα.

```
val url = URL(ApiClient.CALENDAR_URL)
val connection = url.openConnection() as HttpURLConnection
connection.requestMethod = "GET"
connection.setRequestProperty("Authorization", "Bearer $token")
val json = JSONObject(response)
val lessonsJson = json.getJSONObject("data").getJSONArray("lessons")
val lessons = mutableListOf<CalendarLesson>()
for (i in 0 until lessonsJson.length()) {
  val item = lessonsJson.getJSONObject(i)
  lessons.add(CalendarLesson(
    productId = item.getInt("product_id"),
    title = item.getString("title"),
    schedule = item.optString("schedule", ""),
```

```
        startDate = item.optString("start_date", ""),
        endDate = item.optString("end_date", ""),
        startTime = item.optString("start_time", ""),
        endTime = item.optString("end_time", ""),
        orderId = item.getInt("order_id"),
        status = item.getString("status")
    ))
}
```

Σχήμα Β.7: Απόσπασμα κώδικα CalendarRepository.loadCalendar.

B.8 DarkModeManager

Το DarkModeManager συγκεντρώνει τη λογική αποθήκευσης και ανάκτησης της προτίμησης εμφάνισης. Η χρήση object το καθιστά απλό singleton utility.

```
object DarkModeManager {
    private const val PREF_NAME = "theme_prefs"
    private const val KEY_DARK_MODE = "dark_mode"

    fun isDarkMode(context: Context): Boolean {
        val prefs = context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE)
        return prefs.getBoolean(KEY_DARK_MODE, false)
    }

    fun setDarkMode(context: Context, enabled: Boolean) {
        val prefs = context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE)
        prefs.edit().putBoolean(KEY_DARK_MODE, enabled).apply()
    }
}
```

Σχήμα Β.8: Απόσπασμα κώδικα DarkModeManager.

B.9 ApiClient

Η κλάση ApiClient λειτουργεί ως κεντρικό σημείο ρύθμισης διευθύνσεων API. Έτσι αποφεύγεται η διάσπαρτη χρήση URL strings σε πολλά αρχεία.

```
object ApiClient {
    private const val BASE_URL = "http://195.251.123.142/s2c/s2c_local_api"
    const val LOGIN_URL = "$BASE_URL/login.php"
    const val PROFILE_URL = "$BASE_URL/profile.php"
    const val LOGOUT_URL = "$BASE_URL/logout.php"
    const val ORDERS_URL = "$BASE_URL/orders.php"
    const val CALENDAR_URL = "$BASE_URL/my_calendar.php"
    const val QR_ACCESS_URL = "$BASE_URL/qr_access.php"
    const val QR_HISTORY_URL = "$BASE_URL/get_qr_history.php"
    const val SAVE_QR_HISTORY_URL = "$BASE_URL/save_qr_history.php"
    const val FORGOT_PASSWORD_URL = "$BASE_URL/forgot_password.php"
    fun lessonDetailsUrl(productId: Int): String {
        return "$BASE_URL/lesson_details.php?product_id=$productId"
    }
}
```

Σχήμα Β.9: Απόσπασμα κώδικα ApiClient με τα βασικά endpoints.

Παράρτημα Γ: Αναλυτική αποτίμηση αρχιτεκτονικής

Γ.1 Διαχωρισμός ευθυνών

Η αρχιτεκτονική της εφαρμογής δεν ακολουθεί πλήρες MVVM σε όλες τις οθόνες, αλλά εφαρμόζει σαφή διαχωρισμό μεταξύ διεπαφής, Activity ροής και repository δικτύου. Οι Composable συναρτήσεις εστιάζουν στην παρουσίαση. Οι Activity κλάσεις αναλαμβάνουν τον έλεγχο token, την έναρξη νέων Activity και τον συντονισμό callbacks. Τα repositories εκτελούν HTTP κλήσεις, χειρίζονται JSON και επιστρέφουν έτοιμα αντικείμενα ή μηνύματα σφάλματος.

Ο διαχωρισμός αυτός είναι επαρκής για το μέγεθος της εφαρμογής και βοηθά την τεκμηρίωση. Ο φοιτητής ή ο συντηρητής μπορεί να εντοπίσει γρήγορα πού βρίσκεται κάθε ευθύνη: UI στο ui.screens, lifecycle/πλοήγηση στο ui.activities, data fetching στο data.repository και βοηθητικές ρυθμίσεις στο utils.

Γ.2 Ροές δεδομένων και μετασχηματισμοί

Η εφαρμογή δεν εμφανίζει απευθείας το JSON. Κάθε repository αναλαμβάνει να μετατρέψει την απάντηση σε μοντέλα που ταιριάζουν με την οθόνη. Για παράδειγμα, το OrdersRepository δημιουργεί OrderItem, το SubscriptionsRepository δημιουργεί SubscriptionItem και το TeachersRepository δημιουργεί TeacherItem. Η ύπαρξη διαφορετικών μοντέλων για διαφορετικές οθόνες διευκολύνει την ανάγνωση και μειώνει την ανάγκη σύνθετης λογικής μέσα στις Composable συναρτήσεις.

Ιδιαίτερο ενδιαφέρον παρουσιάζει το γεγονός ότι το orders.php endpoint υποστηρίζει τρεις ενότητες. Η επιλογή αυτή μειώνει τον αριθμό endpoints, αλλά απαιτεί προσεκτικό client-side parsing. Η εργασία αναδεικνύει αυτόν τον συμβιβασμό και δείχνει πώς η τοπική επεξεργασία μπορεί να δημιουργήσει διαφορετικές προβολές πάνω στα ίδια δεδομένα.

Γ.3 Ανθεκτικότητα σε σφάλματα

Η εφαρμογή ελέγχει αν υπάρχει token πριν από κάθε προστατευμένη οθόνη. Όταν το token λείπει, εμφανίζεται μήνυμα και η οθόνη κλείνει ή επιστρέφει στη LoginActivity. Αυτό αποτρέπει την εμφάνιση άδειων οθονών και περιορίζει περιπτώσεις μη εξουσιοδοτημένης πρόσβασης. Στα repositories, οι εξαιρέσεις καλούνται με try-catch και μετατρέπονται σε callbacks onError.

Τα μηνύματα Toast χρησιμοποιούνται για άμεση ανατροφοδότηση. Σε μελλοντική έκδοση θα μπορούσαν να αντικατασταθούν από πιο οργανωμένο σύστημα UI state με loading, success και error states. Ωστόσο, για την τρέχουσα πτυχιακή υλοποίηση, η προσέγγιση είναι λειτουργική και κατανοητή.

Γ.4 Δυνατότητες εξέλιξης

Η αρχιτεκτονική επιτρέπει σταδιακή εξέλιξη. Η πρώτη προφανής βελτίωση θα ήταν η εισαγωγή ViewModel και coroutines. Αυτό θα μείωνε την ανάγκη runOnUiThread και θα έκανε την κατάσταση των οθονών πιο lifecycle-aware. Η δεύτερη βελτίωση θα ήταν η αντικατάσταση του HttpURLConnection με Retrofit ή Ktor client, ώστε οι κλήσεις API να δηλώνονται με πιο τυποποιημένο τρόπο.

Άλλη σημαντική εξέλιξη θα ήταν η εισαγωγή EncryptedSharedPreferences για το token και HTTPS ως υποχρεωτική επικοινωνία. Τέλος, μπορεί να προστεθεί τοπικό cache με Room ώστε ο χρήστης να βλέπει προηγούμενα δεδομένα ακόμη και όταν το API δεν είναι διαθέσιμο. Οι βελτιώσεις αυτές δεν ακυρώνουν την τρέχουσα υλοποίηση, αλλά επεκτείνουν τη βάση που έχει δημιουργηθεί.

Παράρτημα Δ: Αναλυτική Περιγραφή Οθονών και Ροών Χρήστη

Το παρόν παράρτημα συμπληρώνει την τεχνική ανάλυση με πιο αναλυτική περιγραφή των βασικών οθονών, των αλληλεπιδράσεων και των αποφάσεων σχεδίασης. Η περιγραφή βασίζεται στις υλοποιημένες οθόνες και στις καταγραφές που παρουσιάζονται στο Κεφάλαιο 6.

Δ.1 Οθόνη σύνδεσης

Η οθόνη σύνδεσης σχεδιάστηκε ώστε να είναι η πρώτη καθαρή επαφή του χρήστη με την εφαρμογή. Τα πεδία username και password εμφανίζονται μέσα σε κάρτα, ώστε ο χρήστης να καταλαβαίνει άμεσα ότι πρόκειται για φόρμα εισόδου. Η επιλογή να χρησιμοποιηθεί Card με στρογγυλεμένες γωνίες και μικρή σκιά δημιουργεί οπτική ιεράρχηση χωρίς να χρειάζονται πολλά διακοσμητικά στοιχεία.

Η λειτουργία Forgot Password βρίσκεται στην ίδια οθόνη, αλλά δεν ανταγωνίζεται το κύριο κουμπί Login. Με αυτόν τον τρόπο η βασική ενέργεια παραμένει σαφής, ενώ η βοηθητική ενέργεια είναι διαθέσιμη όταν χρειάζεται. Από τεχνική πλευρά, η οθόνη δεν περιέχει λογική δικτύου. Η ευθύνη της είναι να συλλέξει τις τιμές και να καλέσει το callback που παρέχει η LoginActivity.

Δ.2 Οθόνη προφίλ

Η οθόνη προφίλ επιλέχθηκε ως κύρια οθόνη μετά τη σύνδεση επειδή συγκεντρώνει την ταυτότητα του χρήστη και λειτουργεί ως κεντρικός κόμβος. Η εφαρμογή εμφανίζει το ονοματεπώνυμο, το username και τα βασικά στοιχεία λογαριασμού. Η εικόνα προφίλ φορτώνεται με AsyncImage, γεγονός που επιτρέπει την απόδοση απομακρυσμένης εικόνας χωρίς χειροκίνητο χειρισμό bitmap.

Η επιλογή Show QR Code τοποθετείται εμφανώς επειδή η QR λειτουργία αποτελεί σημαντική λειτουργία πρόσβασης. Το UI αποφεύγει να παρουσιάσει όλες τις ενότητες ως κουμπιά στο κέντρο της

οθόνης και χρησιμοποιεί drawer navigation. Αυτό κρατά το προφίλ καθαρό και παράλληλα δίνει πρόσβαση σε My Orders, My Teachers, My Subscriptions, My Calendar και Settings.

Δ.3 Πλαϊνό μενού πλοήγησης

Το πλαϊνό μενού εξασφαλίζει συνεπή πλοήγηση μεταξύ των βασικών ενοτήτων. Κάθε επιλογή συνοδεύεται από εικονίδιο, ώστε η αναγνώριση να είναι γρήγορη ακόμη και όταν ο χρήστης δεν διαβάζει ολόκληρη την ετικέτα. Η υλοποίηση με ModalNavigationDrawer ακολουθεί τη λογική του Material Design και είναι κατάλληλη για εφαρμογές με περισσότερες από δύο ή τρεις κύριες ενότητες.

Από τεχνική πλευρά, κάθε item του drawer κλείνει πρώτα το drawer και έπειτα καλεί την αντίστοιχη ενέργεια πλοήγησης. Αυτό αποφεύγει το φαινόμενο να παραμένει ανοιχτό το drawer τη στιγμή που ξεκινά νέα Activity. Η συμπεριφορά αυτή κάνει την πλοήγηση ομαλή και προβλέψιμη.

Δ.4 Οθόνη QR

Η οθόνη QR συνδυάζει τρία διαφορετικά στοιχεία: τον ίδιο τον κώδικα, την επεξήγηση του περιεχομένου και το ιστορικό. Η επεξήγηση κάτω από το QR βοηθά τον χρήστη να καταλάβει ότι ο κώδικας αφορά την πρόσβαση ή τα στοιχεία τάξης και όχι τυχαίο περιεχόμενο. Το ιστορικό εμφανίζεται ως λίστα καρτών, ώστε κάθε παλαιότερη εγγραφή να είναι διακριτή.

Η δυνατότητα Share QR Code αξιοποιεί μηχανισμό του Android και όχι χειροποίητη επιλογή εφαρμογών. Με το ACTION_SEND intent, το λειτουργικό σύστημα εμφανίζει αυτόματα τις διαθέσιμες εφαρμογές αποστολής ή αποθήκευσης. Η καταγραφή στο save_qr_history.php πριν ή κατά τη διαδικασία διαμοιρασμού επιτρέπει στο σύστημα να γνωρίζει ότι ο χρήστης πραγματοποίησε σχετική ενέργεια.

Δ.5 Οθόνη παραγγελιών

Η οθόνη παραγγελιών παρουσιάζει οικονομικές ή αγοραστικές εγγραφές του χρήστη. Η κάθε παραγγελία εμφανίζεται ως κάρτα με τίτλο μαθήματος, αριθμό παραγγελίας, ημερομηνία αγοράς, συνολικό ποσό και κατάσταση. Η χρήση φίλτρων και ταξινόμησης βοηθά στην καλύτερη ανάγνωση όταν ο χρήστης έχει πολλές παραγγελίες.

Η ενότητα αυτή δείχνει καθαρά τη σύνδεση της εφαρμογής με WooCommerce δεδομένα. Η Android εφαρμογή δεν δημιουργεί παραγγελίες, αλλά τις ανακτά και τις παρουσιάζει. Αυτή η απόφαση μειώνει την πολυπλοκότητα της mobile εφαρμογής και αφήνει το υπάρχον σύστημα να παραμένει υπεύθυνο για τις εμπορικές εγγραφές.

Δ.6 Οθόνη συνδρομών/μαθημάτων

Η οθόνη My Subscriptions έχει διαφορετικό στόχο από την οθόνη Orders. Ενώ οι παραγγελίες δείχνουν αγορές, οι συνδρομές δείχνουν μαθήματα που σχετίζονται με τον χρήστη. Για αυτό χρησιμοποιείται

φιλτράρισμα στις καταστάσεις completed και on-hold. Το αποτέλεσμα είναι μία πιο εκπαιδευτική προβολή, όπου ο χρήστης βλέπει τίτλο μαθήματος, καθηγητή, ημέρες, ώρες και περίοδο.

Η υλοποίηση δείχνει σημαντική αρχιτεκτονική επιλογή: η ίδια πηγή δεδομένων μπορεί να εξυπηρετήσει διαφορετικό UI, αρκεί το repository να μετασχηματίσει σωστά την απάντηση. Μελλοντικά, αν το API προσφέρει ξεχωριστό endpoint για subscriptions, η αλλαγή θα γίνει κυρίως στο SubscriptionsRepository.

Δ.7 Οθόνη καθηγητών

Η οθόνη καθηγητών δεν είναι απλή λίστα χρηστών. Δημιουργείται από τα μαθήματα που έχει ο χρήστης και ομαδοποιεί τα σχετικά μαθήματα ανά καθηγητή. Έτσι ο χρήστης βλέπει με ποιους καθηγητές συνδέεται μέσω των μαθημάτων του και μπορεί να μεταβεί στις λεπτομέρειες των αντίστοιχων μαθημάτων.

Η χρήση teacher_details, όταν υπάρχουν στο JSON, εμπλουτίζει την παρουσίαση με ειδικότητα, σύντομο βιογραφικό ή εικονίδιο. Όταν αυτά τα στοιχεία λείπουν, η εφαρμογή δεν αποτυγχάνει, αλλά χρησιμοποιεί κενές τιμές. Αυτός ο χειρισμός κάνει την οθόνη πιο ανθεκτική σε ατελείς API απαντήσεις.

Δ.8 Εσωτερικό ημερολόγιο

Το εσωτερικό ημερολόγιο παρουσιάζει το πρόγραμμα μαθημάτων σε μορφή φιλική για κινητή συσκευή. Τα δεδομένα προέρχονται από my_calendar.php και όχι από υπηρεσία Google. Η διάκριση αυτή είναι σημαντική για την τεκμηρίωση, επειδή το σύστημα δεν απαιτεί OAuth, Google account ή άδειες εξωτερικού ημερολογίου.

Η παρουσίαση προγράμματος μέσα στην εφαρμογή είναι αρκετή για το scope της πτυχιακής. Ο χρήστης μπορεί να δει ημερομηνίες, ώρες και σχετικές εγγραφές μαθημάτων χωρίς να φύγει από την εφαρμογή. Η μελλοντική εξαγωγή σε Google Calendar θα μπορούσε να προστεθεί ως ξεχωριστή λειτουργία, αλλά δεν ανήκει στην τρέχουσα υλοποίηση.

Δ.9 Οθόνη ρυθμίσεων

Η οθόνη ρυθμίσεων συγκεντρώνει τις επιλογές που δεν ανήκουν σε κάποια συγκεκριμένη εκπαιδευτική ροή. Η ενότητα Appearance περιλαμβάνει το Dark Mode, ενώ η ενότητα App περιλαμβάνει πληροφορίες, έκδοση και βοήθεια. Με αυτόν τον τρόπο η εφαρμογή διαχωρίζει τις λειτουργίες περιεχομένου από τις λειτουργίες ρύθμισης.

Το Dark Mode είναι πλήρως λειτουργικό και αποθηκεύεται τοπικά. Η επιλογή Change Password εμφανίζεται ως σημείο διεπαφής, αλλά δεν παρουσιάζεται στην εργασία ως ολοκληρωμένη υλοποίηση. Αυτός ο διαχωρισμός αποτρέπει την υπερβολική παρουσίαση λειτουργιών που δεν έχουν ολοκληρωθεί.

Δ.10 Συνολική εμπειρία χρήστη

Η συνολική εμπειρία χρήστη βασίζεται σε σταθερό μοτίβο: login, profile, drawer, ενότητα, επιστροφή ή logout. Η επανάληψη του ίδιου τρόπου πλοήγησης σε διαφορετικές οθόνες κάνει την εφαρμογή προβλέψιμη. Η χρήση dark θεματολογίας στις καταγραφές δείχνει επίσης ότι η εφαρμογή υποστηρίζει συνεπή εμφάνιση σε σκοτεινό περιβάλλον.

Από άποψη σχεδίασης, η εφαρμογή προτιμά κάρτες, στρογγυλεμένα στοιχεία και σαφείς ετικέτες. Αυτό ταιριάζει σε εκπαιδευτική εφαρμογή όπου ο χρήστης αναζητά γρήγορη πρόσβαση σε οργανωμένη πληροφορία και όχι σύνθετη επεξεργασία δεδομένων.

Παράρτημα Ε: Τελική Χαρτογράφηση Υλοποίησης

Ο παρακάτω πίνακας συγκεντρώνει την αντιστοίχιση ανάμεσα στις λειτουργίες της εφαρμογής, στα αρχεία που τις υλοποιούν και στο είδος δεδομένων που χρησιμοποιούν. Η χαρτογράφηση βοηθά στην τελική αξιολόγηση της εργασίας, επειδή αποδεικνύει ότι κάθε λειτουργία που περιγράφεται συνδέεται με συγκεκριμένο τμήμα κώδικα.

Λειτουργία	**Κύρια αρχεία**	**Δεδομένα**	**Πηγή**
Σύνδεση	LoginActivity, LoginScreen, AuthRepository	username/password, token	login.php
Προφίλ	ProfileActivity, ProfileScreen, AuthRepository	username, email, firstname, lastname, profile_picture	profile.php
Αποσύνδεση	ProfileActivity, SettingsActivity, OrdersActivity κ.ά.	auth_token	logout.php ή τοπική διαγραφή
Forgot Password	ForgotPasswordActivity, AuthRepository	user_input, reset_url	forgot_password.php
QR πρόσβαση	QRActivity, QRScreen, QRRepository	qrPayload, QRHistoryItem	qr_access.php, get_qr_history.php
QR ιστορικό/Share	QRActivity, QRRepository	qr_code, scan_type, device_info	save_qr_history.php

Παραγγελίες	OrdersActivity, OrdersScreen, OrdersRepository	OrderItem	orders.php
Συνδρομές/Μαθήματα	SubscriptionsActivity, SubscriptionsRepository	SubscriptionItem	orders.php
Λεπτομέρειες μαθήματος	SubscriptionDetailActivity	product_id και lesson details	lesson_details.php
Καθηγητές	TeachersActivity, TeachersScreen, TeachersRepository	TeacherItem, TeacherDetails, TeacherLessonItem	orders.php
Εσωτερικό ημερολόγιο	CalendarActivity, CalendarScreen, CalendarRepository	CalendarLesson	my_calendar.php
Ρυθμίσεις/Dark Mode	SettingsActivity, SettingsScreen, DarkModeManager	dark_mode preference	SharedPreferences

Πίνακας Ε.1: Τελική χαρτογράφηση λειτουργιών, αρχείων και πηγών δεδομένων.

Η χαρτογράφηση επιβεβαιώνει ότι η εργασία δεν βασίζεται σε μη υλοποιημένες λειτουργίες. Κάθε ενότητα που παρουσιάζεται έχει αντίστοιχο Activity ή repository και συνδέεται με πραγματική πηγή δεδομένων. Τα μελλοντικά χαρακτηριστικά, όπως εξωτερικός συγχρονισμός ημερολογίου, πλήρης αλλαγή κωδικού ή offline cache, δεν περιλαμβάνονται στον πίνακα ως υλοποιημένα στοιχεία.