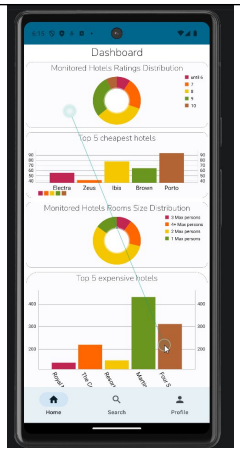
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σύστημα Παρακολούθησης Τιμών Ξενοδοχείων — PriceMonitoring



Φοιτητής: Βασίλειος Σιαμάτρας
……..……
Αρ. Μητρώου: 185273……..

Επιβλέπων
Ονοματεπώνυμο: Μιχάλης Σαλαμπάσης
Βαθμίδα: Καθηγητής

Ημερομηνία ΜΑΙΟΣ 2026

Τίτλος Δ.Ε.: Σύστημα Παρακολούθησης Τιμών Ξενοδοχείων — PriceMonitoring
Κωδικός Δ.Ε. 26274
Ονοματεπώνυμο φοιτητή: Βασίλειος Σιαμάτρας
Ονοματεπώνυμο εισηγητή: Μιχάλης Σαλαμπάσης
Ημερομηνία ανάληψης Δ.Ε. 28-05-2026
Ημερομηνία περάτωσης Δ.Ε. 28-05-2026

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.

«Αφιέρωση»

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος. Αντικείμενό της είναι η σχεδίαση και υλοποίηση εφαρμογής Android για την παρακολούθηση τιμών ξενοδοχείων, ως πελάτη ενός ευρύτερου συστήματος πελάτη-διακομιστή. Η εφαρμογή επικοινωνεί με διακομιστή Spring Boot μέσω διεπαφής REST και παρουσιάζει λίστες παρακολούθησης, ιστορικά τιμών και ειδοποιήσεις.

Το συνολικό σύστημα προέκυψε από τρεις συνεργαζόμενες εργασίες: τον διακομιστή, τον διαδικτυακό πελάτη και την εφαρμογή Android. Το κείμενο που ακολουθεί εστιάζει στην εφαρμογή Android. Ο διακομιστής και ο διαδικτυακός πελάτης τεκμηριώνονται στις αντίστοιχες συνεργαζόμενες εργασίες και αντιμετωπίζονται εδώ ως δεδομένη διεπαφή.

Περίληψη

Η εργασία αναλύει την υλοποίηση εφαρμογής Android, αναπτυγμένης με Jetpack Compose, ως πελάτη συστήματος παρακολούθησης τιμών ξενοδοχείων. Το συνολικό σύστημα προέκυψε από τρεις συνεργαζόμενες εργασίες: την υπηρεσία διακομιστή σε Spring Boot, πελάτη διαδικτυακής εφαρμογής (web) και την παρούσα εφαρμογή Android. Διακομιστής και web πελάτης αναπτύχθηκαν σε ξεχωριστές εργασίες των συνεργατών και αντιμετωπίζονται εδώ εκτός αντικειμένου· ο διακομιστής εμφανίζεται ως δεδομένη διεπαφή REST API, με συνοπτική αναφορά στα δεδομένα, στις διαθέσιμες λειτουργίες και στα λειτουργικά της όρια.

Η εφαρμογή ακολουθεί την αρχιτεκτονική MVVM, με αποθετήρια, Hilt, Navigation Compose και επικοινωνία μέσω Retrofit/OkHttp με διακριτικά JWT. Υποστηρίζει λίστες παρακολούθησης, αναζήτηση μέσω URL, γραφήματα ιστορικού και τοπικές ειδοποιήσεις μέσω WorkManager. Για τον έλεγχο της λειτουργίας της, η στοίβα MySQL, RabbitMQ και διακομιστή αναπαράγεται τοπικά με Docker. Η χρήση αντιγράφου βάσης μέσω mysqldump επιτρέπει την επανάληψη των δοκιμών χωρίς ταύτιση της τοπικής ροής με παραγωγική εγκατάσταση.

Λέξεις-κλειδιά: Android, Spring Boot, REST API, JWT, MVVM, Jetpack Compose, Docker, παρακολούθηση τιμών

Abstract

This thesis analyzes the implementation of an Android application, built with Jetpack Compose, as the client of a hotel price-monitoring system. The complete system arose from three collaborating projects: a Spring Boot server, a web application client, and the present Android application. The server and the web client were developed as separate projects by the collaborators and are treated here as out of scope; the server appears as a given REST API interface, with a brief reference to its data, available operations, and operational limits.

The application follows the MVVM architecture, with repositories, Hilt, Navigation Compose, and communication through Retrofit/OkHttp using JWT tokens. It supports monitor lists, URL-based search, price-history charts, and local notifications via WorkManager. To test its operation, the MySQL, RabbitMQ, and server stack is reproduced locally with Docker. Using a database dump via mysqldump allows the tests to be repeated without coupling the local flow to a production installation.

Keywords: Android, Spring Boot, REST API, JWT, MVVM, Jetpack Compose, Docker, price monitoring

Ευχαριστίες

Ευχαριστίες οφείλονται στον επιβλέποντα καθηγητή Μιχάλη Σαλαμπάση για την καθοδήγηση και τις παρατηρήσεις του κατά την εκπόνηση της εργασίας. Ευχαριστίες αποδίδονται επίσης στους συνεργάτες των δύο συνδεδεμένων εργασιών, για τη συνεργασία στη διαμόρφωση της κοινής διεπαφής REST, καθώς και στην οικογένεια για τη στήριξη καθ' όλη τη διάρκεια των σπουδών.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Στο σώμα της εργασίας οι τεχνικοί όροι που διατηρούνται σε λατινική γραφή τυπογραφούνται με ... για ευανάγνωστη απόδοση. Όπου επιλέγεται ελληνική απόδοση, αυτή συνοδεύεται στην πρώτη εμφάνιση από τον αγγλικό όρο σε παρένθεση. Ο παρακάτω πίνακας συγκεντρώνει τους βασικούς όρους που παραμένουν σε λατινική γραφή, μαζί με σύντομη ελληνική εξήγηση. Ο πίνακας δεν αποσκοπεί σε εξαντλητική καταγραφή όλων των εμπορικών ονομασιών προϊόντων και εργαλείων. Τα ονόματα προϊόντων και βιβλιοθηκών, όπως Android και Spring Boot, παραμένουν αμετάφραστα όπου αποτελούν επίσημη ονομασία.

Συνοπτικό λεξιλόγιο όρων που παραμένουν σε λατινική γραφή στο κείμενο. (συνέχεια)

Όρος (αγγλικά)	**Ελληνική απόδοση / σημείωση**
Όρος (αγγλικά)	**Ελληνική απόδοση / σημείωση**
Συνέχεια στην επόμενη σελίδα	
API (Application Programming Interface)	Διεπαφή προγραμματισμού εφαρμογών· συμβόλαιο κλήσεων μεταξύ πελάτη και διακομιστή.
backend	Διακομιστική πλευρά· λογισμικό και επιχειρησιακή λογική εκτός συσκευής χρήστη. Συνώνυμα στο κείμενο: *διακομιστής*, *υπηρεσία διακομιστή*.
client–server	Αρχιτεκτονική πελάτη—διακομιστή.
composable	Συνάρτηση του Jetpack Compose που περιγράφει τμήμα της διεπαφής χρήστη.
coroutine	Ασύγχρονη ροή εκτέλεσης Kotlin (συνεργατική μονάδα προγραμματισμού).
CRUD	Δημιουργία, ανάγνωση, ενημέρωση, διαγραφή εγγραφών (Create, Read, Update, Delete).
DTO (Data Transfer Object)	Αντικείμενο μεταφοράς δεδομένων μεταξύ επιπέδων ή διαδικτύου.
FCM (Firebase Cloud Messaging)	Υπηρεσία απομακρυσμένων ειδοποιήσεων της Google.
endpoint	Τελικό σημείο διεπαφής REST: συγκεκριμένη διαδρομή URL που υλοποιεί μία λειτουργία.
Gson	Βιβλιοθήκη σειριοποίησης και αποσειριοποίησης JSON για Java/Kotlin.
Hilt	Εργαλείο ενσωμάτωσης εξαρτήσεων για Android, βασισμένο στο Dagger.
HTTP / HTTPS	Πρωτόκολλα μεταφοράς υπερκειμένου (με ασφάλεια διαβίβασης ή χωρίς).
Jetpack Compose	Δηλωτικό πλαίσιο ανάπτυξης διεπαφών για Android.
JWT (JSON Web Token)	Διακριτικό ασφαλείας αυτο-περιεχόμενου τύπου με βάση JSON.
MVVM (Model–View–ViewModel)	Πρότυπο αρχιτεκτονικής· διαχωρισμός μοντέλου, παρουσίασης και λογικής παρουσίασης.
Navigation Compose	Βιβλιοθήκη πλοήγησης του Jetpack Compose.
OkHttp	Βιβλιοθήκη πελάτη HTTP για κλήσεις δικτύου.
Repository (αποθετήριο)	Στρώμα αφαίρεσης πρόσβασης σε δεδομένα· επίπεδο μεταξύ ViewModel και πηγών δεδομένων, όπως δίκτυο και τοπική αποθήκευση.
REST	Αρχιτεκτονικό στυλ μεταφοράς κατάστασης αναπαραστατικού· πόροι και μέθοδοι HTTP.
Retrofit	Βιβλιοθήκη ορισμού και εκτέλεσης κλήσεων προς διεπαφές REST.

Όρος (αγγλικά)	**Ελληνική απόδοση / σημείωση**
Room	Βιβλιοθήκη τοπικής βάσης δεδομένων για Android, βασισμένη σε SQLite.
scraper	Υπηρεσία ή στοιχείο αυτοματοποιημένης ανάκτησης δεδομένων από ιστότοπους (στον τόμο: Node.js σε συνεργασία με το Spring Boot).
singleton (σχέδιο)	Μοναδική υπόσταση αντικειμένου σε όλη την εφαρμογή.
StateFlow	Ροή κατάστασης της Kotlin, κατάλληλη για παρατήρηση μεταβολών από τη διεπαφή.
token (διακριτικό)	Συμβολοσειρά ασφαλείας ή επαλήθευσης· π.χ. JWT, OTP ή διακριτικό επαλήθευσης email.
UI (User Interface)	Διεπαφή χρήστη· οπτική αλληλεπίδραση.
ViewModel	Στρώμα παρουσίασης Android· επιβιώνει αλλαγών διαμόρφωσης και τροφοδοτεί τη διεπαφή χρήστη.
Vico	Βιβλιοθήκη σχεδίασης διαγραμμάτων για Compose.
WorkManager	Android API για προγραμματισμένες εργασίες που επιβιώνουν επανεκκινήσεων.

Κεφάλαιο 1ο: Εισαγωγή

1.1 Περιγραφή του θέματος

Η εργασία παρουσιάζει την ανάπτυξη ενός συστήματος παρακολούθησης τιμών ξενοδοχείων με τρία υποσυστήματα: εφαρμογή διακομιστή (backend) σε Spring Boot, πελάτη διαδικτυακής εφαρμογής (web) και πελάτη για κινητές συσκευές Android. Οι χρήστες παρακολουθούν τιμές δωματίων σε τακτά χρονικά διαστήματα, λαμβάνουν ειδοποιήσεις για μεταβολές και διαχειρίζονται λίστες παρακολούθησης. Οι τεχνικοί όροι που διατηρούνται σε λατινική γραφή συγκεντρώνονται στο Λεξιλόγιο μετά τη βιβλιογραφία (Πίνακας 8.1).

Η διαφορετική τιμολογιακή πολιτική πλατφορμών κρατήσεων μεταβάλλει συχνά τις τιμές δωματίων (ζήτηση, εποχή, παράγοντες αγοράς). Η χειροκίνητη παρακολούθηση αυτών των μεταβολών είναι χρονοβόρα. Το σύστημα που εξετάζεται αυτοματοποιεί τη συλλογή και την προβολή, συγκεντρώνοντας την παρακολούθηση πολλαπλών ξενοδοχείων και δωματίων.

1.2 Κινητρολογία και πεδίο εφαρμογής

Η ανάπτυξη συνδέει την ανάγκη πρακτικής εφαρμογής με τεχνολογίες ανάπτυξης λογισμικού και με απαιτήσεις της τουριστικής αγοράς. Το πεδίο εφαρμογής ενσωματώνει αυτοματοποιημένη συλλογή τιμών από διαδικτυακές πηγές, ασφαλή διαχείριση χρηστών με σύγχρονους μηχανισμούς ταυτοποίησης και διεπαφή αλληλεπίδρασης μέσω κινητών συσκευών.

Η αρχιτεκτονική πελάτη-διακομιστή χωρίζει την επεξεργασία δεδομένων στον διακομιστή από την παρουσίαση στους πελάτες. Η διάκριση βοηθάει τη συντήρηση και επιτρέπει πρόσθετους πελάτες, όπως έκδοση για iOS, χωρίς αλλαγή στην πλευρά διακομιστή.

1.3 Οργάνωση εργασίας και όρια ευθύνης

Το σύστημα αναπτύχθηκε ως τρεις διακριτές συνεργαζόμενες εργασίες. Η υλοποίηση διακομιστή (Spring Boot, αποθήκευση δεδομένων, ασφάλεια διακομιστή, συντονισμός με υπηρεσίες scraper όπου εφαρμόζεται) και ο πελάτης διαδικτυακής εφαρμογής (web) ανήκουν σε ξεχωριστές εργασίες των συνεργατών και δεν αναλύονται εδώ. Το παρόν κείμενο αντιμετωπίζει τον διακομιστή ως δεδομένη διεπαφή REST API, χωρίς ανάλυση της εσωτερικής υλοποίησης, και εστιάζει στην ανάλυση και υλοποίηση της εφαρμογής Android, στη διαμόρφωση της διεπαφής, στη διαχείριση κατάστασης και στην τοπική αναπαραγωγή της στοίβας μέσω Docker για δοκιμές, χωρίς ταύτιση με παραγωγική εγκατάσταση σε συστάδα διακομιστών, όπως Docker Swarm. Όπου η ανάλυση χρειάζεται αναφορά στον διαδικτυακό πελάτη, αυτή παραμένει σε επίπεδο δηλωμένης ύπαρξης, ώστε τα όρια του αντικειμένου της παρούσας εργασίας να μένουν σαφή.

1.4 Στόχοι της εργασίας

Οι στόχοι περιλαμβάνουν ασφαλή ταυτοποίηση, αναζήτηση και καταγραφή ξενοδοχείων μέσω ομοιόμορφου εντοπιστή πόρων (URL) σε προσωπικές λίστες, ειδοποιήσεις για μεταβολές τιμών και οπτική παρουσίαση ιστορικού. Κεντρικό αντικείμενο είναι η υλοποίηση της εφαρμογής Android πάνω από τη διαθέσιμη διεπαφή API.

1.5 Δομή της εργασίας

Η έκθεση ακολουθεί κοινή για τεχνικές πτυχιακές διάταξη: θεωρητικό υπόβαθρο, απαιτήσεις και σχεδιασμός, περιγραφή του διακομιστή ως πλαισίου, υλοποίηση της εφαρμογής Android ως κύριο κορμό, έλεγχος και αποτελέσματα, συμπεράσματα με περιορισμούς και, τέλος, βιβλιογραφία με λεξιλόγιο. Ο κατάλογος περιεχομένων αντιστοιχίζει κάθε θέμα στο κεφάλαιό του.

Κεφάλαιο 2ο: Θεωρητικό Υπόβαθρο και Τεχνολογίες

2.1 Αρχιτεκτονική πελάτη-διακομιστή και REST

Στην αρχιτεκτονική πελάτη-διακομιστή ο πελάτης ζητά υπηρεσίες και ο διακομιστής τις παρέχει, γεγονός που επιτρέπει την ανεξάρτητη εξέλιξη των δύο μερών. Το αρχιτεκτονικό στυλ Representational State Transfer (REST) συστηματοποιεί τις υπηρεσίες μέσω HTTP, με απουσία κατάστασης αιτήματος, ομοιόμορφη διεπαφή και κατάλληλη χρήση μεθόδων και κωδικών κατάστασης (Fielding 2000). Στο παρόν σύστημα, η πλευρά διακομιστή (backend) εκθέτει πόρους για χρήστες, ξενοδοχεία, λίστες και τιμές με σημασιολογία GET/POST/PUT/DELETE.

2.2 Ταυτοποίηση με διακριτικά JWT

Η ταυτοποίηση επιβεβαιώνει την ταυτότητα του χρήστη, ενώ η εξουσιοδότηση καθορίζει τις επιτρεπόμενες ενέργειες. Τα διακριτικά JSON Web Token (JWT) μεταφέρουν υπογεγραμμένες δηλώσεις (claims), όπως αναγνωριστικό χρήστη και ρόλο (Internet Engineering Task Force (IETF) 2015). Το πλαίσιο OAuth 2.0 για εξουσιοδότηση μέσω διακομιστή περιγράφεται στο (Internet Engineering Task Force (IETF) 2012). Το Spring Security επαληθεύει την υπογραφή και τη λήξη, ενώ οι κωδικοί αποθηκεύονται με BCrypt (VMware, Inc., n.d.-b). Η κρυπτογραφημένη μεταφορά σε συνδυασμό με JWT καλύπτει τυπικές απαιτήσεις προστασίας σε επίπεδο εφαρμογής.

2.3 Αρχιτεκτονική MVVM, Clean Architecture και Jetpack Compose

Η αρχιτεκτονική Model–View–ViewModel (MVVM) διαχωρίζει τα δεδομένα, την παρουσίαση και τη λογική παρουσίασης (Fowler 2002). Στο Android, το ViewModel επιβιώνει αλλαγών διαμόρφωσης και τροφοδοτεί τη διεπαφή χρήστη μέσω ροών κατάστασης (Google LLC, n.d.-a). Η Clean Architecture ορίζει ότι οι εξαρτήσεις δείχνουν προς πιο αφηρημένα επίπεδα, ενώ το στρώμα αποθετηρίων απομονώνει την πρόσβαση στα δεδομένα από το ViewModel (Martin 2017). Το Jetpack Compose εισάγει δηλωτική περιγραφή της διεπαφής. Η ανασύνθεση ενημερώνει μόνο τα τμήματα που επηρεάζονται από μεταβολές της κατάστασης (Google LLC, n.d.-c). Η πλοήγηση με Navigation Compose συνδυάζεται συχνά με ορισμούς διαδρομών που διασφαλίζουν ασφάλεια τύπων, όπως οι sealed class οθονών.

2.4 Ενσωμάτωση εξαρτήσεων και ασύγχρονη εκτέλεση

Η ενσωμάτωση εξαρτήσεων (Dependency Injection) παραδίδει τις εξαρτήσεις εξωτερικά, διευκολύνοντας τις δοκιμές και την αντικατάσταση υλοποιήσεων (Gamma et al. 1994). Το Dagger Hilt παρέχει προκαθορισμένους συνδυασμούς επισημάνσεων (annotations) για Android, όπως Application, ViewModel και modules (Google LLC, n.d.-b). Οι ασύγχρονες λειτουργίες δικτύου και αποθήκευσης εκτελούνται εκτός του κύριου νήματος της διεπαφής μέσω Kotlin Coroutines, ώστε να αποφεύγεται ο αποκλεισμός της.

2.5 Περιέκτες λογισμικού με Docker

Οι περιέκτες λογισμικού (containers) ομαδοποιούν εφαρμογή και ρυθμίσεις σε μεταφέρσιμη μονάδα εκτέλεσης στο ίδιο λειτουργικό σύστημα υποδοχής (host), με μικρότερο κόστος από πλήρεις εικονικές μηχανές (Docker Inc., n.d.). Το Docker Compose περιγράφει υπηρεσίες, δίκτυα και τόμους σε ένα αρχείο. Για το παρόν σύστημα, η τοπική αναπαραγωγή με περιέκτες βοηθάει τις δοκιμές του

διακομιστή, της MySQL και του RabbitMQ, χωρίς ταύτιση με συγκεκριμένη παραγωγική εγκατάσταση, όπως το Docker Swarm.

Κεφάλαιο 3ο: Ανάλυση απαιτήσεων και σχεδιασμός

3.1 Λειτουργικές απαιτήσεις

Ο στόχος παρακολούθησης τιμών ξενοδοχείων, με λίστες, ειδοποιήσεις και οπτική παρουσίαση ιστορικού, καθορίζει τις λειτουργικές απαιτήσεις. Η αντιστοίχιση απαίτησης, ενδεικτικού τελικού σημείου διεπαφής και κύριας οθόνης της εφαρμογής Android συνοψίζεται στον Πίνακα 3.1.

Πίνακας 3.1: *Αντιστοίχιση λειτουργικών απαιτήσεων με διεπαφές API και κύριες οθόνες.*

Απαίτηση	**Ενδεικτικά τελικά σημεία διεπαφής**	**Οθόνη (Android)**
Εγγραφή/επαλήθευση email	/customer/email/verification, /customer/register	Signup, OTP
Σύνδεση/έκδοση διακριτικού	/customer/login	Login
Αναζήτηση ξενοδοχείου μέσω URL	/hotel/info, /hotel/{id}	Search
Διαχείριση λιστών (CRUD)	/monitor_list, /monitor_list/{id}	Home, Λίστες
Ιστορικό τιμών	/prices/{monitorListID}/{roomID}	Λεπτομέρεια, διάγραμμα
Τοπικές ειδοποιήσεις	WorkManager, κανάλι ειδοποιήσεων	Προφίλ

Απαιτείται ασφαλής διαχείριση λογαριασμών, αναζήτηση και αποθήκευση ξενοδοχείων και δωματίων, διαχείριση πολλαπλών λιστών ανά χρήστη, παρουσίαση μεταβολών τιμών στο χρόνο και ειδοποίηση του χρήστη όταν ικανοποιούνται κανόνες μεταβολής που ορίζει η εφαρμογή. Οι προδιαγραφές της πλατφόρμας και της διεπαφής χρήστη ευθυγραμμίζονται με την τεκμηρίωση Android (Google LLC, n.d.-a).

3.2 Μη λειτουργικές απαιτήσεις

Οι επικοινωνίες πρέπει να προστατεύονται με κρυπτογράφηση μεταφοράς, οι κωδικοί να αποθηκεύονται με κατακερματισμό και τα διακριτικά JWT να έχουν περιορισμένη διάρκεια. Η διεπαφή ακολουθεί το Material Design (Google LLC, n.d.-d), παρέχει ανατροφοδότηση φόρτωσης και χειρίζεται σφάλματα δικτύου με κατανοητά μηνύματα. Η εφαρμογή στοχεύει Android API 26 και νεότερες εκδόσεις (Google LLC, n.d.-a).

3.3 Μοντέλο δεδομένων και αρχιτεκτονική συστήματος

Το μοντέλο δεδομένων περιλαμβάνει χρήστες, ξενοδοχεία με μοναδικό URL, δωμάτια και χαρακτηριστικά, λίστες παρακολούθησης και εγγραφές τιμών ανά προβολή δωματίου (RoomView/RoomViewPrice). Η αρχιτεκτονική είναι τριεπίπεδη πελάτη-διακομιστή: παρουσίαση στην εφαρμογή Android, επιχειρησιακή λογική και αποθήκευση δεδομένων στον διακομιστή, με χρήση MySQL και RabbitMQ. Η εφαρμογή Android ακολουθεί MVVM, στρώμα αποθετηρίων και Hilt, όπως αναπτύσσεται στο Κεφάλαιο 5, σε συμφωνία με τις πρακτικές διαχωρισμού πελάτη και υπηρεσιών σε κατανεμημένα συστήματα (Newman 2021).

Κεφάλαιο 4ο: Διακομιστής και διεπαφές API

Τα ακόλουθα αφορούν όσα παρέχει η υπηρεσία διακομιστή στον πελάτη Android και ποια δεδομένα διακινούνται. Η λεπτομερής υλοποίηση της πλευράς διακομιστή (backend), που βασίζεται σε Spring Boot, JPA, μηχανισμούς ασφάλειας και ρυθμίσεις παραγωγής, ανήκει σε ξεχωριστή συνεργαζόμενη εργασία. Το ίδιο ισχύει για τον διαδικτυακό πελάτη (web) που καταναλώνει την ίδια διεπαφή και αναπτύχθηκε από άλλον συνεργάτη και δεν αναλύεται εδώ. Ο διακομιστής θεωρείται δεδομένη διεπαφή API και η παρούσα ανάλυση εστιάζει στα συμβόλαια αιτημάτων και αποκρίσεων και στα όρια της συνεργασίας πελάτη-διακομιστή για την πλευρά του Android.

4.1 Ρόλος της υπηρεσίας διακομιστή και επίπεδα

Η υπηρεσία διακομιστή υλοποιεί REST API πάνω από Spring Boot, με επίπεδα ελεγκτή (controller), υπηρεσίας (service) και αποθετηρίου (repository) και με αποθήκευση δεδομένων σε MySQL (VMware, Inc., n.d.-a). Ο πελάτης Android καλεί τελικά σημεία διεπαφής μέσω HTTPS/JSON, στέλνει JWT σε προστατευμένα αιτήματα και λαμβάνει αντικείμενα μεταφοράς δεδομένων (DTO) συμβατά με τα μοντέλα που χαρτογραφούνται στον Gson του πελάτη. Η ασφάλεια διακομιστή και οι ρυθμίσεις εξουσιοδότησης παραπέμπουν στην τεκμηρίωση Spring Security (VMware, Inc., n.d.-b).

4.2 Κύρια τελικά σημεία διεπαφής και ροές

Η ταυτοποίηση καλύπτει εγγραφή με επαλήθευση διεύθυνσης email, σύνδεση, επαναφορά κωδικού και ανανέωση διαπιστευτηρίων. Οι λίστες παρακολούθησης δημιουργούνται, ενημερώνονται και διαγράφονται με αιτήματα που φέρουν το JWT του χρήστη. Η αναζήτηση ξενοδοχείου βασίζεται σε URL πλατφόρμας κρατήσεων. Ο διακομιστής συντονίζει την ανάκτηση δομημένων στοιχείων μέσω ξεχωριστής υπηρεσίας συλλογής δεδομένων (scraper) σε περιβάλλον Node.js, που, σε περιβάλλοντα χωρίς πλήρη πρόσβαση στον ιστό, μπορεί να χρησιμοποιεί ελεγχόμενα δοκιμαστικά δεδομένα. Το ιστορικό τιμών επιστρέφεται σελιδοποιημένα για κάθε συνδυασμό λίστας και δωματίου.

4.3 Ασφάλεια και θέματα λειτουργίας

Η πρόσβαση σε προστατευμένους πόρους απαιτεί έγκυρο JWT, ενώ οι ρόλοι χρηστών περιορίζουν λειτουργίες και όρια χρήσης, όπως τον αριθμό λιστών και δωματίων. Ισχύει περιορισμός ρυθμού αιτημάτων σε ευαίσθητα τελικά σημεία διεπαφής, όπως η αποστολή email επαλήθευσης. Σε προφίλ ανάπτυξης η αποστολή email μπορεί να αντικαθίσταται από καταγραφή του διακριτικού, ώστε η ροή εγγραφής να παραμένει αναπαραγώγιμη χωρίς εξωτερικό πάροχο αλληλογραφίας (VMware, Inc., n.d.-b).

4.4 Τοπική αναπαραγωγή με Docker

Η στοίβα MySQL, RabbitMQ και διακομιστή περιγράφεται στο αρχείο docker-compose.backend.yml, ενώ παρέχεται και έτοιμη εικόνα μέσω docker-compose.dockerhub.yml. Οι μεταβλητές περιβάλλοντος καθορίζουν προφίλ Spring, συμβόλαια σύνδεσης JDBC και ρυθμίσεις AMQP. Η πλήρης αναπαραγωγή βάσης δεδομένων με περιεχόμενο συμβατό προς την παραγωγή βασίζεται σε αρχείο εξαγωγής (mysqldump), που μπορεί να εισαχθεί κατά την πρώτη αρχικοποίηση κενού τόμου δεδομένων ή σε ενεργό container, όπως περιγράφεται στην τεκμηρίωση του διακομιστή. Η παραγωγική ανάπτυξη, για παράδειγμα σε Docker Swarm, είναι ξεχωριστό πλαίσιο από την τοπική ροή ανάπτυξης (Docker Inc., n.d.).

Κεφάλαιο 5ο: Υλοποίηση εφαρμογής Android

5.1 Αρχιτεκτονική MVVM και Clean Architecture

Η εφαρμογή Android υλοποιήθηκε με την αρχιτεκτονική Model–View–ViewModel (MVVM) σε συνδυασμό με αρχές Clean Architecture (Martin 2017; Fowler 2002). Η προσέγγιση αυτή διαχωρίζει τη διεπαφή χρήστη από την επιχειρηματική λογική και επιτρέπει την ανεξάρτητη δοκιμή των επιπέδων.

Το επίπεδο Model περιλαμβάνει τις κλάσεις δεδομένων που αναπαριστούν τις οντότητες της εφαρμογής (User, MonitorList, Hotel, Room) και τη διεπαφή υπηρεσίας API για επικοινωνία με την πλευρά διακομιστή. Οι κλάσεις δεδομένων ορίζονται ως κλάσεις δεδομένων της Kotlin με επισημάνσεις του Gson για σειριοποίηση.

Το επίπεδο παρουσίασης (View) αποτελείται από συναρτήσεις του Jetpack Compose. Κάθε οθόνη δέχεται ως παράμετρο το αντίστοιχο ViewModel. Η διεπαφή χρήστη παρακολουθεί την κατάσταση μέσω του StateFlow και ενημερώνεται αυτόματα όταν αλλάζουν τα δεδομένα.

Το επίπεδο ViewModel φέρει την κύρια λογική παρουσίασης. Κάθε ViewModel μετατρέπει τις προθέσεις του χρήστη σε ενέργειες και προμηθεύει τα απαραίτητα δεδομένα στη διεπαφή. Τα ViewModel επιβιώνουν από αλλαγές διαμόρφωσης, όπως η περιστροφή της συσκευής, διασφαλίζοντας ότι τα δεδομένα δεν χάνονται.

Το στρώμα αποθετηρίων προσφέρει αφαίρεση για την πρόσβαση στα δεδομένα. Τα UserRepository και MonitorRepository χειρίζονται τις κλήσεις προς τη διεπαφή API και την τοπική αποθήκευση μέσω SessionManager. Η οργάνωση αυτή επιτρέπει την αντικατάσταση της πηγής δεδομένων, όπως με προσθήκη προσωρινής μνήμης (cache), χωρίς τροποποίηση των ViewModel.

5.2 Jetpack Compose και δομή οθονών

5.2.1 Συναρτήσεις Compose

Η διεπαφή χρήστη υλοποιήθηκε αποκλειστικά με Jetpack Compose, ένα δηλωτικό εργαλείο ανάπτυξης διεπαφών της Google (Google LLC, n.d.-c). Κάθε οθόνη ορίζεται ως συνάρτηση composable που περιγράφει τη διεπαφή με βάση την τρέχουσα κατάσταση. Η ανασύνθεση (recomposition) εξασφαλίζει ότι ενημερώνονται μόνο τα τμήματα που επηρεάζονται από μεταβολές.

Οι βασικές οθόνες της εφαρμογής είναι οι SplashScreen (εκκίνηση με έλεγχο διακριτικού), LoginScreen (σύνδεση), SignupScreen (εγγραφή), OTPScreen (επαλήθευση email) και DashboardScreen (κύρια οθόνη με τρεις καρτέλες).

5.2.2 Πλοήγηση

Η πλοήγηση υλοποιήθηκε με Navigation Compose. Οι διαδρομές ορίζονται σε sealed class Screen, ώστε να διασφαλίζεται ασφάλεια τύπων. Ο NavHost διαχειρίζεται την εμφάνιση των οθονών και τη στοίβα πλοήγησης.

Οι διαδρομές ομαδοποιούνται σε sealed class με σταθερές για Splash, Login, Signup, OTP και Dashboard. Ο NavHost ορίζεται στη ρίζα της εφαρμογής και χειρίζεται τις μεταβάσεις μεταξύ

οθονών. Οι παράμετροι μεταβιβάζονται μέσω ViewModel δεσμευμένου στο NavHost, ώστε να διατηρείται η κατάσταση σε όλη τη ροή ταυτοποίησης.

5.2.3 Material Design 3

Η εφαρμογή ακολουθεί το Material Design 3 με προσαρμοσμένη χρωματική παλέτα (Google LLC, n.d.-d). Το θέμα ορίζεται στο Color.kt με βασικά, δευτερεύοντα και τριτεύοντα χρώματα. Η τυπογραφία χρησιμοποιεί τη γραμματοσειρά Nunito. Τα βασικά στοιχεία της διεπαφής, όπως κουμπιά, πεδία κειμένου και κάρτες, ακολουθούν τις προδιαγραφές του Material Design 3.

5.3 Λειτουργίες ανά ροή οθόνης

Οι κύριες ροές της εφαρμογής εξετάζονται παρακάτω σε αντιστοιχία με τα αντίστοιχα ViewModel, αποθετήρια και τελικά σημεία διεπαφής.

5.3.1 Εκκίνηση και οθόνη Splash

Η SplashScreen αναλαμβάνει την αρχική διακλάδωση ώστε να μην απαιτείται επανεισαγωγή διαπιστευτηρίων σε έγκυρη συνεδρία. Διαβάζει το JWT από το SessionManager και, αν υπάρχει, η πλοήγηση οδηγεί στο DashboardScreen. Διαφορετικά πηγαίνει στο LoginScreen.

Πιο αναλυτικά, ο πίνακας ελέγχου παρουσιάζεται μόνο μετά την επιτυχή ανάκληση διακριτικού. Το πεδίο user_token του SessionManager αποθηκεύεται στο SharedPreferences, με αναζήτηση σε σταθερό χρόνο κατά την εκκίνηση. Η οθόνη φιλοξενεί ένα κεντραρισμένο Lottie γραφικό και εκτελεί την ανάγνωση εντός LaunchedEffect, ώστε το νήμα της διεπαφής να μην εμπλακεί σε I/O. Όταν η διακλάδωση ολοκληρωθεί, καλείται navController.navigate(...)popUpTo(Splash){ inclusive = true}, αφαιρώντας την οθόνη από τη στοίβα πλοήγησης ώστε ένα πάτημα του πλήκτρου επιστροφής να μην επαναφέρει την αναμονή. Αν το διακριτικό υπάρχει αλλά έχει λήξει, η πρώτη κλήση προς το παρασκήνιο επιστρέφει 401 και το ViewModel εκτελεί αυτόματη αποσύνδεση με κλήση clearSession(), επιστρέφοντας τον χρήστη στην οθόνη σύνδεσης.

5.3.2 Σύνδεση (στόχος και ροή δεδομένων)

Η σύνδεση εκδίδει JWT και το αποθηκεύει για μελλοντικά αιτήματα. Η LoginScreen συνδέεται με το LoginViewModel, ενώ το UserRepository αποστέλλει POST /customer/login. Σε επιτυχία ενημερώνεται το SessionManager και η εφαρμογή μεταβαίνει στον πίνακα ελέγχου.

Το ορατό μέρος της σύνδεσης οργανώνεται σε ConstraintLayout με τρία σύνολα στοιχείων: τη μάσκα φωτογραφίας στο φόντο, την κάθετη στοίβα πεδίων email και κωδικού, και τον σύνδεσμο μετάβασης στην εγγραφή. Τα πεδία βασίζονται σε OutlinedTextField με προσαρμοσμένα χρώματα ώστε να διαβάζονται πάνω από το σκούρο φόντο. Η εμφάνιση κωδικού (password reveal) είναι προαιρετική και αλλάζει το VisualTransformation ανάλογα με την επιθυμητή ορατότητα. Το ViewModel δημοσιεύει loginEvent: SharedFlow<UserUiEvents> τον οποίο η οθόνη συγκεντρώνει μέσω collectLatest. Στην εμφάνιση UserUiEvents.LoginSuccess καλείται το προβλεπόμενο onLoginClick(). Σε αποτυχία εμφανίζονται διακριτά μηνύματα ανά κωδικό HTTP: 401 για λανθασμένα διαπιστευτήρια, 429 για ρυθμό υπερβολής, 503 για διακοπή του διακομιστή και 408 για ενδεχόμενη απουσία δικτύου.

Με τον τρόπο αυτό ο χρήστης κατανοεί άμεσα αν πρόκειται για δικό του λάθος ή για βραχύχρονο πρόβλημα υποδομής.

5.3.3 Ροή εγγραφής και OTP

Η ροή εγγραφής δημιουργεί λογαριασμό με επαλήθευση email σε δύο βήματα. Η SignupScreen ολοκληρώνει το πρώτο (POST /customer/email/verification) και η OTPScreen το δεύτερο (POST /customer/register). Τα αντίστοιχα ViewModel εκθέτουν κατάσταση φόρτωσης και σφάλματος, ώστε τα πεδία και τα κουμπιά να παραμένουν συγχρονισμένα με την ασύγχρονη απόκριση του διακομιστή.

Στην οθόνη εγγραφής τοπική κατάσταση User συγκεντρώνει τα πέντε πεδία της φόρμας (email, όνομα, κωδικός, επανάληψη κωδικού, τηλέφωνο). Το requestEmailVerification επικυρώνει τοπικά: απουσία πεδίου, μη έγκυρη μορφή email, μη ταυτιζόμενοι κωδικοί, έλλειψη ψηφίων ή γραμμάτων στον κωδικό. Σε επιτυχή τοπική επικύρωση το ViewModel αποθηκεύει τον pendingRegistrationUser σε ιδιωτική μεταβλητή και υποβάλλει το αίτημα. Η οθόνη OTPScreen φιλοξενεί OutlinedTextField 140 dp σε ύψος για επικόλληση του διακριτικού επαλήθευσης που στάλθηκε στο ηλεκτρονικό ταχυδρομείο. Από το πάτημα Create account εκπέμπεται UserIntent.OTP(token) και το ViewModel αποθηκεύει το διακριτικό σε pendingEmailJwt, εκπέμποντας UserUiEvents.EmailOtpVerified ώστε ο πλοηγός να συνεχίσει στο επόμενο βήμα, την επαλήθευση τηλεφώνου.

5.3.4 Πίνακας ελέγχου (τρεις καρτέλες)

Ο πίνακας ελέγχου συγκεντρώνει τη διαχείριση λιστών, την αναζήτηση ξενοδοχείου και τις ρυθμίσεις λογαριασμού. Το DashboardViewModel τροφοδοτεί την καρτέλα Home (MonitorListTab), την Search (SearchHotelTab) και την Profile (ProfileTab). Τα δεδομένα προέρχονται από τα MonitorRepository και UserRepository, για παράδειγμα από τα GET monitor_list, POST /hotel/info και από τα τελικά σημεία διεπαφής της συνδρομής.

Η επιλογή καρτέλας διατηρείται ως τοπική κατάσταση στο Scaffold, ενώ η χειρονομία ανανέωσης επανενεργοποιεί τις κλήσεις προς το αποθετήριο. Στο κάτω μέρος εμφανίζεται κάθετη περιοχή πλοήγησης με τρία NavigationBarItem και αντίστοιχα εικονίδια Home, Search και Person. Το επιλεγμένο στοιχείο παρακολουθείται μέσω currentBackStackEntryAsState και ο πλοηγός αλλάζει καρτέλα με popUpTo(graph.startDestinationId) { launchSingleTop = true } ώστε να μη συσσωρεύονται διπλές καταχωρίσεις στη στοίβα. Η ίδια σύσταση δικτύου εξυπηρετεί και τις νέες οθόνες My subscription, Event preferences και Notification delivery που ξεκινούν από την καρτέλα Προφίλ και επιστρέφουν σε αυτήν με το πλήκτρο επιστροφής.

Σε κάθε καρτέλα οι αρχικές κλήσεις τρέχουν εντός LaunchedEffect(Unit), ώστε ο χρήστης να βλέπει αμέσως δεδομένα από προηγούμενη συνεδρία (αν υπάρχουν στο cache του StateFlow) ενώ το νέο αίτημα εκτελείται στο παρασκήνιο.

5.3.5 Λεπτομέρεια λίστας και δωματίων

Η οθόνη λεπτομερειών προβάλλει δωμάτια, τιμές και διαθέσιμες ενέργειες διαχείρισης. Συνδέεται με ViewModel που φορτώνει τη λίστα και τα δωμάτια μέσω MonitorRepository. Η ροή κατάστασης (StateFlow) ενημερώνει τη LazyColumn και τους σχετικούς διαλόγους.

Πάνω από τη λίστα δωματίων εμφανίζεται κάρτα-επικεφαλίδα με όνομα λίστας, αριθμό δωματίων, εύρος ημερών παρακολούθησης και συγκεντρωτικά Min, Avg, Max τιμών σε ζώνη Surface. Κάθε δωμάτιο παρουσιάζεται ως κάρτα με όνομα, σύνδεσμο ξενοδοχείου σε μπλε χρώμα, εικονίδιο ιστορικού τιμών στα δεξιά και τρεις διακριτές κάψουλες στατιστικών. Το πάτημα στο εικονίδιο ιστορικού καλεί onHistoryClick(room) που ανοίγει AlertDialog με το γράφημα και ραδιοεπιλογές διαστημάτων (όλα, 0 ημέρες, 7, 14, 21). Το πάτημα στον σύνδεσμο ξενοδοχείου ανοίγει τη γενική HotelScreen.

Το ViewModel χρησιμοποιεί ξεχωριστά StateFlow για το MonitorListResponse και για τις προβλέψεις τιμών, ώστε η ανανέωση μιας πτυχής (π.χ. νέα τιμή πρόβλεψης μετά από slider) να μην επανασχεδιάζει ολόκληρη τη LazyColumn. Η οθόνη παρακολουθεί επίσης διαμοιραζόμενες ροές DashboardEvents ώστε σε σφάλμα δικτύου να εμφανίζεται Snackbar με ευδιάκριτο κουμπί επανάληψης χωρίς να καθαρίζει το ορατό περιεχόμενο.

5.3.6 Ιστορικό τιμών και Vico

Για την οπτική σύγκριση της χρονοσειράς τιμών, το ViewModel ζητά σελιδοποιημένα ή φιλτραρισμένα δεδομένα ιστορικού από τη διεπαφή API. Η βιβλιοθήκη Vico σχεδιάζει το διάγραμμα από τη λίστα σημείων που έχει ήδη μετατραπεί σε κατάλληλη δομή για το Compose.

Το γράφημα ζωγραφίζει δύο διακριτές σειρές: τη σειρά παρατηρούμενων τιμών με μπλε γραμμή και κύκλους και τη σειρά πρόβλεψης με κόκκινη διακεκομμένη γραμμή. Το πλάτος του οριζόντιου άξονα προσαρμόζεται αυτόματα στις διαθέσιμες ημερομηνίες. Οι ετικέτες περιστρέφονται κατά 45 μοίρες ώστε να χωράνε σε στενή οθόνη. Η ζώνη τίτλου εμφανίζει συγκεντρωτικά Min, Avg, Max σε τρία πλαίσια ίδιου ύψους, ώστε η αναγνωσιμότητα να μην αλλάζει ανάλογα με το πλήθος ψηφίων. Το LiveData του ViewModel ανανεώνει το GraphPrice μέσω collectAsState, οπότε η αλλαγή διαστήματος στις ραδιοεπιλογές εκπέμπει νέο DashboardIntent.OnDistanceSelected και η οθόνη ξαναζωγραφίζει αυτόματα.

5.3.7 Προφίλ, συνδρομή και ειδοποιήσεις

Η καρτέλα Profile καλύπτει τη διαχείριση πλάνου, τις προτιμήσεις ειδοποιήσεων και την έξοδο από την εφαρμογή. Στην κορυφή εμφανίζεται κάρτα συγκεντρωτικών στοιχείων με αριθμό ενεργών λιστών και ηλεκτρονική διεύθυνση του συνδεδεμένου χρήστη. Στο κέντρο η ζώνη συνδρομής δείχνει το τρέχον πλάνο και κουμπί αναβάθμισης, ενώ ένα κουμπί Send test notification ενεργοποιεί δοκιμαστική ώθηση μέσω του διακομιστή.

Από κάτω τοποθετείται καρτέλα τριών νέων εγγραφών πλοήγησης: “My subscription”, Event preferences και Notification delivery. Κάθε εγγραφή είναι Row με ετικέτα 16 sp και βέλος δεξιά, διαχωριζόμενες με λεπτή γραμμή. Η εφαρμογή ζητά την άδεια POST_NOTIFICATIONS στο Android 13 και νεότερες εκδόσεις, με ξεχωριστό διάλογο όταν η άδεια απορρίπτεται οριστικά, ώστε ο χρήστης να μπορεί να ανοίξει τις ρυθμίσεις συστήματος.

5.3.8 Προγραμματισμένες εργασίες (WorkManager)

Για τοπική υπενθύμιση χωρίς απομακρυσμένη ώθηση, το ReminderWorker δημιουργεί ειδοποίηση, ενώ το OneTimeWorkRequest με καθυστέρηση αποθηκεύεται από το σύστημα και εκτελείται ακόμη και μετά το κλείσιμο της εφαρμογής, εντός των περιορισμών εκτέλεσης του Android.

Το ίδιο ReminderWorker συνυπάρχει με την FCM ώθηση: όταν η συσκευή είναι εκτός δικτύου, η τοπική υπενθύμιση εξυπηρετεί ως fallback, ενώ όταν η συσκευή ανακτά συνδεσιμότητα η ώθηση FCM αναλαμβάνει την πρωτεύουσα παράδοση. Η σημασιολογία αποκλεισμού διπλότυπων στηρίζεται στο πεδίο notification_id που εκπέμπει ο διακομιστής: αν η ίδια ταυτότητα έχει ήδη παρουσιαστεί στο NotificationManagerCompat, η πλατφόρμα αντικαθιστά την προηγούμενη παρουσίαση, χωρίς διπλή ηχητική ενημέρωση.

5.3.9 Φόρτωση, κενές καταστάσεις και σφάλματα

Για ανθεκτική συμπεριφορά σε αποτυχία δικτύου, τα ViewModel εκθέτουν σημαίες φόρτωσης και μετατρέπουν τις εξαιρέσεις σε μηνύματα χρήστη, συμπεριλαμβανομένων των κωδικών 401 και 404, χωρίς εμφάνιση τεχνικών λεπτομερειών στη διεπαφή.

Συγκεκριμένα, το ContentWithProgress τοποθετεί ένα CircularProgressIndicator πάνω σε ημιδιαφανές φόντο που απορροφά τα συμβάντα αφής, αποτρέποντας τις διπλές υποβολές κατά τη διάρκεια αιτημάτων. Σε κενή απάντηση (π.χ. καμία λίστα παρακολούθησης) η οθόνη MonitorListScreen εναλλάσσει περιεχόμενο: εμφανίζει κατάσταση κενής λίστας με τίτλο “No monitor lists yet” και προτροπή “Go to Search tab to add your first one”. Αυτή η συμπεριφορά καθοδηγεί τον νέο χρήστη χωρίς αντικειμενική αποτυχία, διατηρώντας το LazyColumn ζωντανό για άμεση εμφάνιση δεδομένων όταν εισαχθούν.

5.4 Ροή ταυτοποίησης

5.4.1 Σύνδεση

Η οθόνη σύνδεσης (LoginScreen) παρέχει δύο πεδία εισαγωγής, email και κωδικό, καθώς και κουμπί σύνδεσης. Το ViewModel ελέγχει την εγκυρότητα των δεδομένων και αποστέλλει αίτημα POST /customer/login μέσω του UserRepository.

Σε επιτυχία, το διακριτικό JWT αποθηκεύεται στο SessionManager (SharedPreferences) και η εφαρμογή πλοηγείται στον πίνακα ελέγχου. Σε αποτυχία εμφανίζεται μήνυμα σφάλματος κατάλληλο για τον τύπο της αποτυχίας. Το ViewModel χειρίζεται εξαιρέσεις όπως HttpException, UnknownHostException και SocketTimeoutException.

5.4.2 Ροή εγγραφής

Η διαδικασία εγγραφής αποτελείται από δύο βήματα. Στο πρώτο βήμα (SignupScreen), ο χρήστης εισάγει email, όνομα, κωδικό και επιβεβαίωση κωδικού. Η αποστολή email επαλήθευσης αντιστοιχεί στο αίτημα POST /customer/email/verification.

Ο διακομιστής αποστέλλει email με διακριτικό επαλήθευσης JWT. Σε επιτυχία, η εφαρμογή πλοηγείται στην OTPScreen, όπου ο χρήστης εισάγει το διακριτικό. Το ViewModel αποστέλλει POST

/customer/register με τα διαπιστευτήρια και το διακριτικό επαλήθευσης. Σε επιτυχία, ο χρήστης μεταφέρεται στην LoginScreen.

5.4.3 Διαχείριση διακριτικών

Το SessionManager χειρίζεται την αποθήκευση και ανάκτηση του JWT. Χρησιμοποιεί SharedPreferences για επίμονη αποθήκευση. Ο AuthInterceptor προσθέτει αυτόματα το διακριτικό σε κάθε αίτημα HTTP, εκτός από τα τελικά σημεία ταυτοποίησης. Ο ResponseInterceptor επεξεργάζεται τις αποκρίσεις και ελέγχει για σφάλματα.

Η ίδια κλάση προσφέρει επίσης τη μέθοδο clearSession() που καλείται σε λήξη ή ανάκληση διακριτικού. Η εκκαθάριση δεν περιορίζεται στο διαγράφειν το πεδίο user_token, αλλά επανεκκινεί Intent-ικά την εφαρμογή με Intent.makeRestartActivityTask, ώστε όλη η ιεραρχία της σύνθεσης Compose να ξεκινήσει καθαρή, χωρίς εκκρεμή Flow ή υπολειπόμενες αναφορές στον προηγούμενο χρήστη. Συνδυαστικά με την υποστήριξη απρόσκοπτης διακλάδωσης στο SplashScreen, η εμπειρία αποσύνδεσης γίνεται διαφανής.

Πέραν του διακριτικού, το SessionManager αποθηκεύει την προτίμηση εμφάνισης ειδοποιήσεων (user_notifications). Η προτίμηση καθορίζει αν οι εγγενείς υπενθυμίσεις του ReminderWorker ενεργοποιούνται, ανεξάρτητα από τις απομακρυσμένες FCM ειδοποιήσεις. Με τον τρόπο αυτό η χρήστρια μπορεί να σιγάσει τις τοπικές υπενθυμίσεις διατηρώντας ενεργές τις απομακρυσμένες (ή το αντίστροφο).

5.5 Πίνακας ελέγχου και λίστες παρακολούθησης

5.5.1 Δομή πίνακα ελέγχου

Το DashboardScreen υλοποιείται με Scaffold, που περιλαμβάνει κάτω γραμμή πλοήγησης και περιοχή περιεχομένου. Οι τρεις καρτέλες είναι οι Home (MonitorListTab), Search (SearchHotelTab) και Profile (ProfileTab).

Η κατάσταση της επιλεγμένης καρτέλας διατηρείται τοπικά. Κάθε καρτέλα είναι συνάρτηση composable που δέχεται το DashboardViewModel, που διαχειρίζεται τα δεδομένα και για τις τρεις ενότητες.

5.5.2 Καρτέλα λιστών

Η καρτέλα Home εμφανίζει τις λίστες παρακολούθησης του χρήστη με LazyColumn. Κάθε στοιχείο αποδίδεται ως Card με όνομα λίστας, αριθμό δωματίων και βασικά στατιστικά. Υποστηρίζεται χειρονομία ολίσθησης για διαγραφή, ενώ η χειρονομία ανανέωσης επιτρέπει την επαναφόρτωση των δεδομένων.

Τα διαγράμματα υλοποιούνται με τη βιβλιοθήκη Vico (compose-m3). Εμφανίζονται διαγράμματα πίτας για την κατανομή δωματίων και διαγράμματα ράβδων για τη σύγκριση τιμών.

5.5.3 Καρτέλα αναζήτησης

Η καρτέλα Search δίνει πεδίο εισαγωγής URL για αναζήτηση ξενοδοχείου. Το ViewModel αποστέλλει POST /hotel/info και ενημερώνει τη διεπαφή με τα αποτελέσματα. Τα δωμάτια εμφανίζονται σε λίστα με τα χαρακτηριστικά τους. Παρατεταμένο πάτημα σε δωμάτιο εμφανίζει διάλογο για προσθήκη σε λίστα παρακολούθησης.

5.5.4 Καρτέλα προφίλ

Η καρτέλα Profile εμφανίζει στοιχεία χρήστη και προσφέρει επιλογές αναβάθμισης συνδρομής (Basic/Premium), ελέγχου ειδοποιήσεων, δοκιμαστικής ειδοποίησης και αποσύνδεσης. Η αναβάθμιση συνδρομής αποστέλλει POST /subscription/{type}.

Στη νέα έκδοση η οθόνη ανανεώνεται με τρεις πρόσθετες εγγραφές πλοήγησης ([fig:profile-new-entries]). Ο χρήστης πατά “My subscription” για να ανοίξει την οθόνη διαχείρισης συνδρομής, Event preferences για να επιλέξει σε ποια συμβάντα θέλει ειδοποίηση και Notification delivery για να επιλέξει push ή ηλεκτρονικό ταχυδρομείο ως δίαυλο παράδοσης. Η οθόνη γίνεται έτσι κεντρικός κόμβος όλων των ρυθμίσεων χρήστη, χωρίς να υπερφορτώνεται οπτικά.

Στο κάτω μέρος υπάρχει More Section με γρήγορες ενέργειες: ενεργοποίηση/απενεργοποίηση ειδοποιήσεων, σύνδεσμος About με στατικά στοιχεία επικοινωνίας και έκδοσης, σύνδεσμος Share App που εκπέμπει Intent.ACTION_SEND με συντομευμένο σύνδεσμο Play Store και κουμπί Logout που καλεί SessionManager.clearSession().

5.6 Επικοινωνία HTTP και ενσωμάτωση API

Επιπλέον, η ζώνη ScrapperConfiguration κατασκευάζει WebClient με τη βάση διεύθυνσης http://scraper:3001, την οποία ο διακομιστής χρησιμοποιεί για να φτάσει στην υπηρεσία αναζήτησης που εκτελείται ως ξεχωριστό container στο ίδιο Docker Compose έργο. Η ίδια WebClient ομάδα χρησιμοποιείται για όλες τις κλήσεις προς τον scraper, ώστε ο συντονισμός χρόνων αναμονής και η επικύρωση σφαλμάτων να συντηρούνται σε ένα σημείο. Σε επίπεδο δίκτυο, η αναζήτηση DNS ολοκληρώνεται από το ενσωματωμένο embedded DNS του Docker: το όνομα scraper αναλύεται στη διεύθυνση εσωτερικού δικτύου του αντίστοιχου container.

5.6.1 Διαμόρφωση Retrofit

Η διεπαφή υπηρεσίας API υλοποιήθηκε με Retrofit 2.9.0 (Square, Inc., n.d.-b). Η MonitorApiService δηλώνει συναρτήσεις αναστολής (suspend functions) για κάθε τελικό σημείο διεπαφής. Τα αντικείμενα μεταφοράς δεδομένων (DTO) ορίζονται ως κλάσεις δεδομένων με επισημάνσεις @SerializedName για αντιστοίχιση με πεδία JSON.

```
@POST("customer/login")
suspend fun loginUser(@Body userLogin: UserLoginRequest):
Response<UserLoginResponse>

@GET("monitor_list")
suspend fun getAllMonitorLists(): Response<MonitorListsResponse>
```

Το Retrofit διαμορφώνεται με GsonConverterFactory για σειριοποίηση και ScalarsConverterFactory για απλές αποκρίσεις κειμένου.

5.6.2 Διαμόρφωση OkHttp

Ο OkHttpClient διαμορφώνεται με διαμεσολαβητές (interceptors): τον AuthInterceptor, που προσθέτει την επικεφαλίδα Authorization, τον ResponseInterceptor, που επεξεργάζεται τις αποκρίσεις, και τον HttpLoggingInterceptor, που καταγράφει την κυκλοφορία HTTP για σκοπούς αποσφαλμάτωσης (Square, Inc., n.d.-a). Τα χρονικά όρια ορίζονται σε 30 seconds για σύνδεση, ανάγνωση και εγγραφή.

Η βασική διεύθυνση URL εισάγεται στο BuildConfig.BASE_URL από το build.gradle.kts. Οι τιμές για τους τύπους κατασκευής debug και release λαμβάνονται από το gradle.properties (BACKEND_URL_DEBUG, BACKEND_URL_RELEASE), με προεπιλογή http://10.0.2.2:8085/ για τον προσομοιωτή. Σε φυσική συσκευή στο ίδιο τοπικό δίκτυο, η τιμή debug αντικαθίσταται από τη διεύθυνση IP του υπολογιστή που εκτελεί την εφαρμογή διακομιστή.

5.6.3 Χειρισμός σφαλμάτων

Το ViewModel χειρίζεται σφάλματα μέσω try-catch ή του τελεστή .catch του Flow. Οι HttpException μετατρέπονται σε κατανοητά μηνύματα για τον χρήστη. Οι κωδικοί 401 (Unauthorized) και 404 (Not Found) αντιμετωπίζονται με κατάλληλη ανατροφοδότηση, ενώ η διεπαφή εμφανίζει ενδείξεις φόρτωσης κατά την αναμονή αποκρίσεων.

5.7 Ενσωμάτωση εξαρτήσεων με Hilt

5.7.1 Αρθρώματα Hilt

Το ApiModule (@Module @InstallIn(SingletonComponent::class)) προσφέρει εξαρτήσεις εμβέλειας εφαρμογής (singleton), όπως OkHttpClient, Retrofit και MonitorApiService (Google LLC, n.d.-b). Ο AuthInterceptor και ο ResponseInterceptor ενσωματώνονται στον OkHttpClient.

Το ApiModule καλύπτει επίσης το Retrofit με BuildConfig.BASE_URL, μετατροπείς Gson/Scalars και τον διαμορφωμένο OkHttpClient. Το SharedPreferencesModule δίνει SessionManager με @ApplicationContext για πρόσβαση στις SharedPreferences.

5.7.2 Ενσωμάτωση ViewModel

Τα ViewModel επισημαίνονται με @HiltViewModel και λαμβάνουν εξαρτήσεις μέσω @Inject constructor. Η πρακτική αυτή επιτρέπει την αυτόματη παροχή των αποθετηρίων και του SessionManager.

5.8 Ειδοποιήσεις με WorkManager

5.8.1 ReminderWorker

Το ReminderWorker επεκτείνει την κλάση Worker και στη doWork() κατασκευάζει ειδοποίηση με NotificationCompat και εκκρεμή πρόθεση ενεργοποίησης (pending intent) προς την εφαρμογή. Η ειδοποίηση εμφανίζεται μέσω NotificationManagerCompat.

5.8.2 Προγραμματισμός με WorkManager

Το DashboardViewModel προγραμματίζει ειδοποιήσεις με WorkManager (Google LLC, n.d.-e). Ο OneTimeWorkRequestBuilder δημιουργεί αίτημα εργασίας με καθυστέρηση (setInitialDelay). Η εργασία αποθηκεύεται και εκτελείται ακόμη και αν η εφαρμογή κλείσει.

5.8.3 Κανάλι ειδοποιήσεων

Το κανάλι ειδοποιήσεων δημιουργείται κατά την εκκίνηση της εφαρμογής για Android O και νεότερες εκδόσεις. Το επίπεδο προτεραιότητας HIGH επιτρέπει την άμεση προβολή της ειδοποίησης. Η άδεια POST_NOTIFICATIONS ζητείται δυναμικά σε Android 13 και νεότερες εκδόσεις.

5.9 Διαμόρφωση κατασκευής

5.9.1 Gradle

Το build.gradle.kts του αρθρώματος εφαρμογής δηλώνει εξαρτήσεις για Jetpack Compose, Hilt, Retrofit, WorkManager και τη βιβλιοθήκη Vico για διαγράμματα συμβατά με Material Design 3. Η συμβατότητα Java ορίζεται στην έκδοση 17.

5.9.2 Τύποι build και διεύθυνση API

Οι τύποι κατασκευής debug και release ορίζουν διακριτό BuildConfig.BASE_URL μέσω ιδιοτήτων έργου, ώστε η έκδοση ανάπτυξης να στρέφεται στον τοπικό διακομιστή και η έκδοση παραγωγής στο απομακρυσμένο API, χωρίς αλλαγή κώδικα οθονών.

5.10 Επαλήθευση κινητού με mock SMS

Η διαδικασία εγγραφής επεκτείνεται ώστε ο υπό δημιουργία λογαριασμός να επιβεβαιώνει το κινητό τηλέφωνο, παράλληλα με την επαλήθευση του ηλεκτρονικού ταχυδρομείου. Η ροή σπάει σε δύο διαδοχικά βήματα: το πρώτο εκδίδει διακριτικό επαλήθευσης ηλεκτρονικής διεύθυνσης, το δεύτερο εκδίδει ξεχωριστό διακριτικό επαλήθευσης τηλεφωνικού αριθμού. Ο διακομιστής συνδυάζει τα δύο διακριτικά κατά την οριστική καταχώριση του χρήστη.

5.10.1 Αρχιτεκτονική Mock SMS

Στο πακέτο service.sms του διακομιστή ορίζεται διεπαφή SmsService και υλοποίηση MockSmsService σημασμένη με @Profile("!default"), που αντί για αποστολή πραγματικού SMS καταγράφει στα αρχεία καταγραφής τον εξαψήφιο κωδικό μίας χρήσης. Η σύμβαση καταγραφής έχει τη μορφή Imitating SMS send to {phone}: OTP={code}, ώστε ο αναπτυσσόμενος να βρίσκει

εύκολα τον κωδικό από τα docker compose logs. Ένας πραγματικός πάροχος (π.χ. Twilio, Vonage) εισάγεται αντικαθιστώντας μόνο την συγκεκριμένη υλοποίηση χωρίς αλλαγές σε ανώτερα στρώματα.

Η οντότητα PhoneVerification αποθηκεύει ηλεκτρονική διεύθυνση, τηλέφωνο, αποτύπωμα κωδικού BCrypt, χρόνο λήξης, αριθμό αποτυχημένων προσπαθειών και χρόνο ανάλωσης. Ο μέγιστος αριθμός προσπαθειών εξάντλησης ορίζεται σε πέντε και ο χρόνος ζωής σε δέκα λεπτά. Ο κωδικός δεν αποθηκεύεται σε καθαρή μορφή. Η σύγκριση γίνεται μέσω PasswordEncoder.matches, αναπαράγοντας το πρότυπο σύγκρισης συνθηματικών.

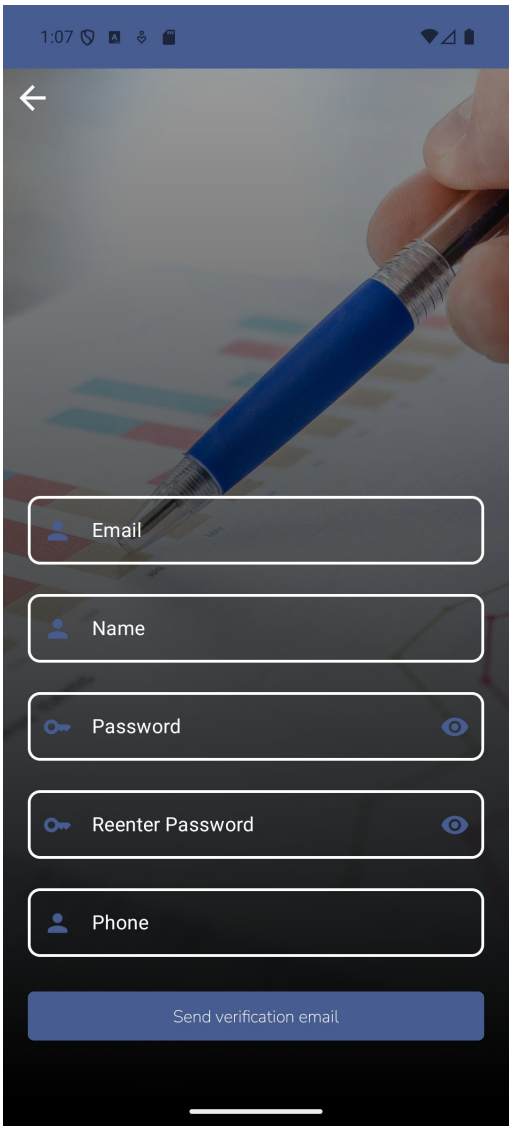
5.10.2 Έκδοση διακριτικού επαλήθευσης τηλεφώνου

Το JWTGenerator επεκτείνεται με μέθοδο generatePhoneVerificationJWT(email, phone) που εκδίδει υπογεγραμμένο διακριτικό HS512, με υποκείμενο την ηλεκτρονική διεύθυνση και πρόσθετη απαίτηση scope=phone_verification. Το διακριτικό περιλαμβάνει επιπλέον το πεδίο phone, ώστε ο διακομιστής να ανακτά τον επαληθευμένο αριθμό κατά την εγγραφή χωρίς νέα κλήση. Η διάρκεια ζωής ταυτίζεται με τον χρόνο ζωής του διακριτικού επαλήθευσης ηλεκτρονικού ταχυδρομείου, διατηρώντας ένα ενιαίο πρότυπο.

5.10.3 Διεπαφή χρήστη επαλήθευσης κινητού

Η οθόνη εγγραφής (SignupScreen) εμπλουτίζεται με πεδίο τηλεφωνικού αριθμού. Όταν ο χρήστης ολοκληρώσει την επαλήθευση ηλεκτρονικής διεύθυνσης, ο πλοηγός μεταβαίνει στο PhoneOtpScreen. Η οθόνη αυτή ζητά αυτόματα την αποστολή κωδικού μέσω UserIntent.RequestPhoneOtp, εμφανίζει ένα αριθμητικό πεδίο εισόδου έξι ψηφίων με αυτόματο φιλτράρισμα μη-ψηφιακών χαρακτήρων και προσφέρει κουμπί επανάληψης αποστολής. Η οριστική επιβεβαίωση εκπέμπει ConfirmPhoneOtp, που καλεί τον διακομιστή και, σε επιτυχία, ολοκληρώνει την εγγραφή POST /customer/register συνδυάζοντας το διακριτικό ηλεκτρονικής διεύθυνσης με το νεοεκδοθέν διακριτικό τηλεφώνου.

Στο LoginViewModel προστίθενται οι κατάσταση pendingRegistrationUser και pendingEmailJwt, οι οποίες διατηρούν τον ημιτελή χρήστη και το επαληθευμένο διακριτικό ηλεκτρονικής διεύθυνσης μεταξύ των τριών διαδοχικών οθονών. Σε αποτυχία οπουδήποτε εκπέμπεται UserUiEvents.Failure με μήνυμα ανάλογο της απάντησης διακομιστή και η ροή επανέρχεται στην οθόνη εγγραφής χωρίς απώλεια πεδίων.



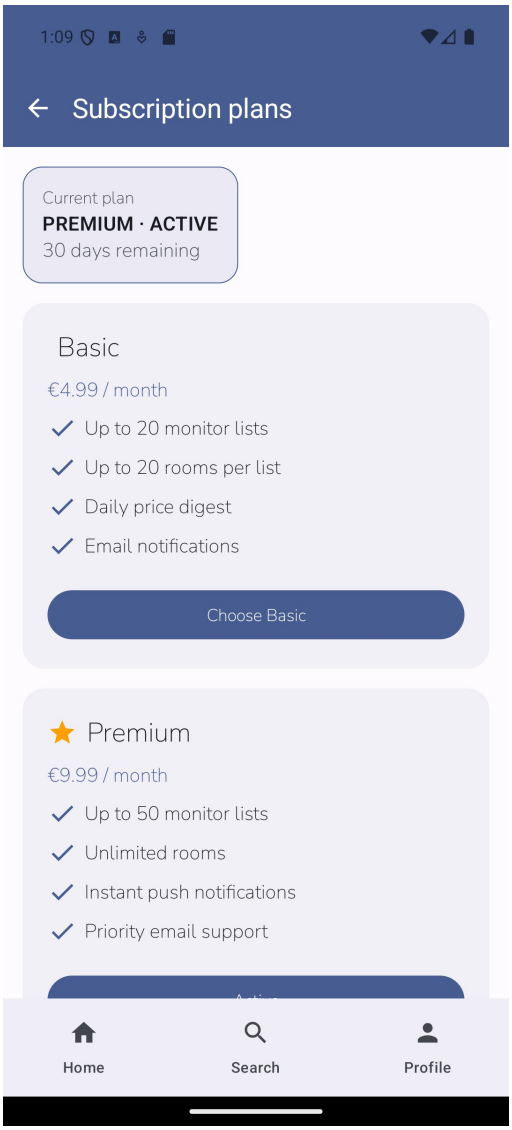
Σχήμα 5.1: *Οθόνη εγγραφής με νέο πεδίο τηλεφώνου.*

5.11 Διαχείριση μηνιαίας συνδρομής

Το μοντέλο συνδρομής αντικαθιστά την προηγούμενη ενιαία αναβάθμιση ρόλου με πραγματική γραμμή χρόνου: κάθε συνδρομή έχει ημερομηνία έναρξης, ημερομηνία λήξης, κατάσταση και προαιρετική αυτόματη ανανέωση. Η οντότητα Subscription συσχετίζει χρήστη με πλάνο (BASIC ή PREMIUM) και κατάσταση (ACTIVE, EXPIRED, CANCELLED). Η μηνιαία λήξη δίνει τον σαφή κανόνα ότι, εφόσον δεν ανανεωθεί, ο χρήστης επιστρέφει στον δωρεάν ρόλο USER με τα αντίστοιχα όρια.

5.11.1 Διεπαφή επιλογής πλάνου

Η οθόνη SubscriptionPlansScreen εμφανίζει τα διαθέσιμα πλάνα ως αυτοτελείς κάρτες με τα οφέλη κωδικοποιημένα σε λίστα με εικονίδια ολοκλήρωσης. Η ενεργή συνδρομή του χρήστη εμφανίζεται σε ξεχωριστή ζώνη στην κορυφή, με την ένδειξη υπολειπόμενων ημερών. Η ίδια ζώνη ενημερώνεται όταν η οθόνη ανακτά εκ νέου την κατάσταση από το /subscription/current, ώστε αμέσως μετά την αγορά να αντικατοπτρίζει τη νέα λήξη.

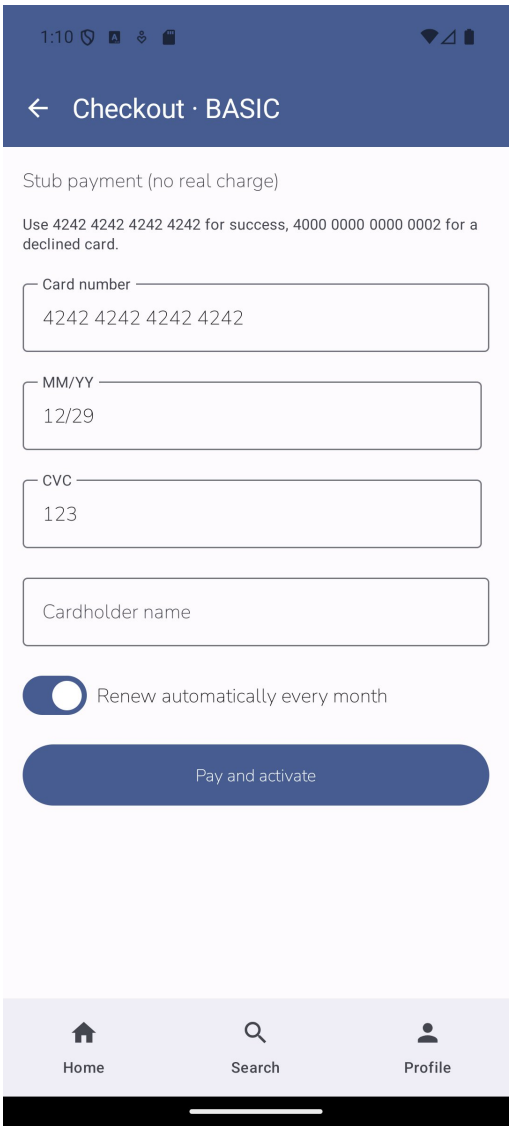


Σχήμα 5.2: *Οθόνη επιλογής πλάνου με κάρτες Basic και Premium.*

5.11.2 Stub payment και ροή αγοράς

Η οθόνη CheckoutScreen ζητά αριθμό κάρτας, ημερομηνία λήξης, CVC και ονοματεπώνυμο κατόχου, σε συμφωνία με το πρότυπο των πραγματικών παρόχων. Η επικύρωση κάρτας γίνεται και στις δύο πλευρές: ο πελάτης ελέγχει μήκος και αριθμητικό περιεχόμενο, ο διακομιστής εφαρμόζει αλγόριθμο Luhn για να απορρίψει τυπικά λανθασμένους αριθμούς. Για επίδειξη, ο διακομιστής αποδέχεται την κάρτα 4242 4242 4242 4242 και απορρίπτει την 4000 0000 0000 0002 επιστρέφοντας HTTP 402. Η οθόνη μετατρέπει το HTTP 402 σε ευανάγνωστο μήνυμα Payment declined.

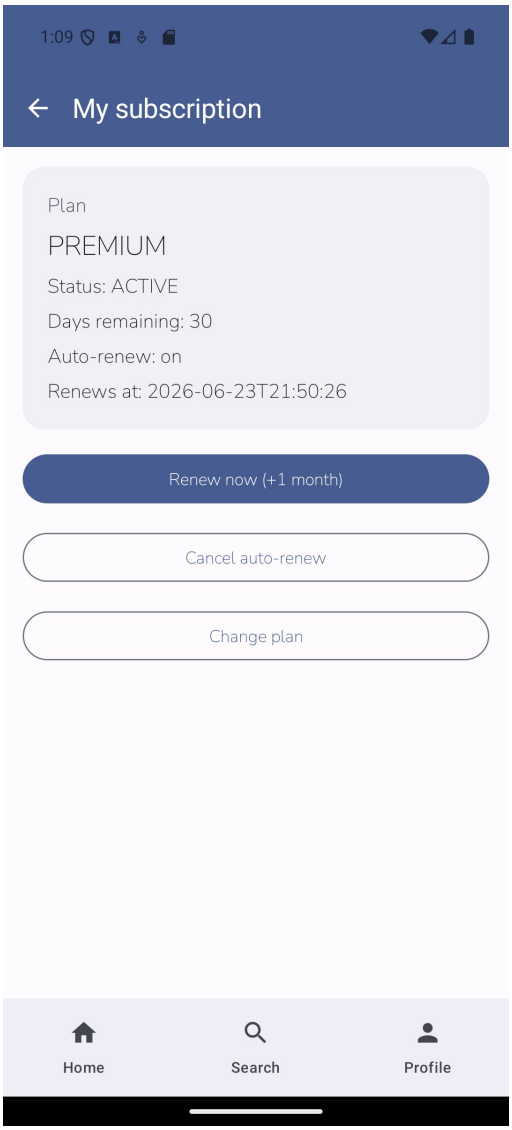
Σε επιτυχία, το SubscriptionService καταχωρεί νέα γραμμή στο payment_transaction με κατάσταση SUCCEEDED και τέσσερα τελευταία ψηφία της κάρτας, και την ίδια στιγμή δημιουργεί νέα γραμμή στο subscription με starts_at = now και expires_at = starts_at + 1 μήνας. Η προηγούμενη ενεργή συνδρομή (αν υπάρχει) σημαίνεται ως EXPIRED με ώρα ακύρωσης το τρέχον χρονικό σημείο. Το διακριτικό απάντησης ενημερώνεται έτσι ώστε η εφαρμογή να γνωρίζει αμέσως τους νέους περιορισμούς πλάνου χωρίς νέα σύνδεση.



Σχήμα 5.3: *Οθόνη stub payment με τυποποιημένα πεδία κάρτας.*

5.11.3 Διαχείριση και ανανέωση

Η οθόνη SubscriptionManagementScreen δείχνει την τρέχουσα συνδρομή με πλάνο, κατάσταση, υπολειπόμενες ημέρες, ένδειξη αυτόματης ανανέωσης και ημερομηνία λήξης. Τρεις πράξεις παρέχονται: άμεση ανανέωση (προσθέτει έναν επιπλέον μήνα στο expires_at), διακοπή αυτόματης ανανέωσης (διατηρεί την πρόσβαση έως τη λήξη) και αλλαγή πλάνου (μεταβαίνει στο SubscriptionPlansScreen). Καθημερινό προγραμματισμένο έργο (SubscriptionExpiryScheduler) σαρώνει στις 01:00 τις ενεργές συνδρομές με expires_at < now, τις σημαίνει ως EXPIRED και επιστρέφει τον χρήστη σε USER, εφόσον δεν υπάρχει άλλη ενεργή συνδρομή.



Σχήμα 5.4: *Οθόνη διαχείρισης συνδρομής με ανανέωση, ακύρωση και αλλαγή πλάνου.*

5.12 Προτιμήσεις ειδοποιήσεων

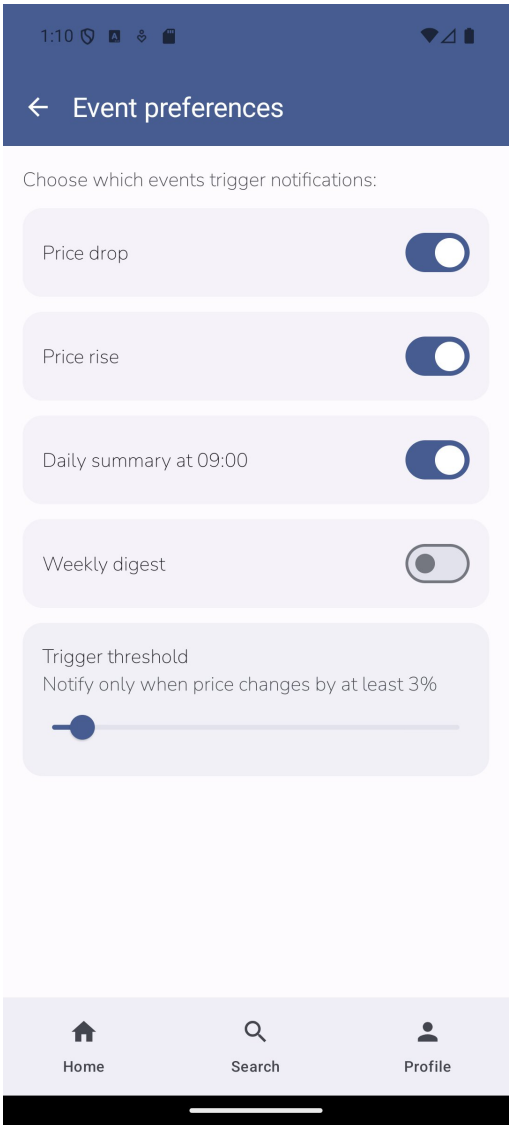
Δύο διακριτές προτιμήσεις παραμετροποιούνται από τον χρήστη: το *τι* και το *πώς*. Το πρώτο αναφέρεται στους τύπους συμβάντων που ο χρήστης δέχεται. Το δεύτερο αφορά τους διαύλους που χρησιμοποιεί η πλατφόρμα για την παράδοση.

5.12.1 Προτιμήσεις συμβάντων

Η οθόνη EventPreferencesScreen προσφέρει διακόπτες (switches) για τέσσερα συμβάντα: μείωση τιμής, αύξηση τιμής, ημερήσια σύνοψη στις 09:00 και εβδομαδιαία ανασκόπηση. Ένα ξεχωριστό πεδίο με συρόμενο επιλογέα (slider) ορίζει το κατώφλι ποσοστιαίας μεταβολής που ενεργοποιεί ειδοποίηση. Η οντότητα user_notification_preferences αποθηκεύει τις προτιμήσεις με κλειδί ταυτόσημο με τον χρήστη, ώστε να μη χρειάζεται ξεχωριστό αναγνωριστικό.

Ο διακομιστής διαβάζει τις προτιμήσεις στον NotificationDispatcher: για κάθε συμβάν προς αποστολή ελέγχει αν το αντίστοιχο πεδίο είναι ενεργοποιημένο και, σε αρνητική απάντηση, ακυρώνει

την παράδοση. Με τη συμπεριφορά αυτή ο χρήστης αποκλείει συγκεκριμένες κατηγορίες ειδοποιήσεων χωρίς να επηρεάζει την παρακολούθηση τιμών στο παρασκήνιο.

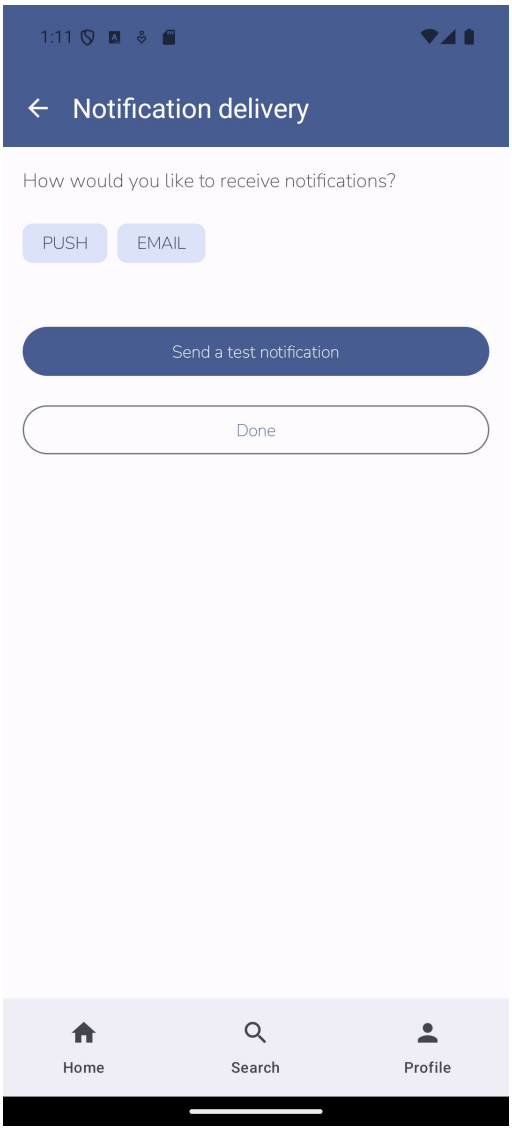


Σχήμα 5.5: *Οθόνη προτιμήσεων συμβάντων με διακόπτες και κατώφλι ποσοστιαίας μεταβολής.*

5.12.2 Προτιμήσεις διαύλου παράδοσης

Η οθόνη NotificationDeliveryScreen ορίζει αν ο χρήστης λαμβάνει ειδοποιήσεις μέσω push, ηλεκτρονικού ταχυδρομείου ή και των δύο, με χρήση δύο φίλτρων (filter chips). Η αλλαγή προτίμησης αποθηκεύεται αυτόματα μέσω PUT /notifications/preferences χωρίς ανάγκη κουμπιού αποθήκευσης, περιορίζοντας τις περιττές αλληλεπιδράσεις.

Στο ίδιο σημείο εκτίθεται κουμπί Send a test notification που καλεί POST /notifications/test. Ο διακομιστής διασπείρει αμέσως ειδοποίηση χρησιμοποιώντας το ίδιο μονοπάτι κανονικής παράδοσης, ώστε η αλληλεπίδραση να αποδεικνύει ολόκληρη την αλυσίδα από την προτίμηση χρήστη ως τη συσκευή.



Σχήμα 5.6: *Οθόνη προτιμήσεων διαύλου παράδοσης και αποστολή δοκιμαστικής ειδοποίησης.*

5.13 Ενσωμάτωση Firebase Cloud Messaging

Η εφαρμογή ενσωματώνει στρώμα αναγγελίας διακριτικού συσκευής στον διακομιστή ώστε οι ειδοποιήσεις να φτάνουν στη συσκευή ακόμη και όταν η εφαρμογή είναι κλειστή. Η ενσωμάτωση δομείται σε δύο επίπεδα: στον διακομιστή με τη βιβλιοθήκη firebase-admin, και στη συσκευή με FcmTokenManager που καταχωρεί διακριτικό μετά τη σύνδεση.

5.13.1 Διακομιστής με υπό συνθήκη FCM

Το αρχείο FirebaseConfiguration σημαίνεται με @ConditionalOnProperty(name="firebase.enabled", havingValue="true"), ώστε σε περιβάλλον ανάπτυξης χωρίς πιστοποιητικά Firebase ο διακομιστής να μη γνωρίζει το FCM και να χρησιμοποιεί την εναλλακτική υλοποίηση LoggingPushNotificationService. Η εναλλακτική καταγράφει στα αρχεία καταγραφής τις ωθούμενες ειδοποιήσεις που θα παραδίδονταν, ώστε ολόκληρος ο μηχανισμός παρακολούθησης και διασποράς να μπορεί να δοκιμαστεί χωρίς πρόσβαση σε Firebase project.

Όταν είναι ενεργοποιημένο, το FcmPushNotificationService ομαδοποιεί διακριτικά συσκευής σε MulticastMessage και καλεί FirebaseMessaging.getInstance().sendMulticast(message), κατασκευάζοντας προαιρετική επικεφαλίδα και σώμα μαζί με χάρτη δεδομένων εφαρμογής. Η ίδια προγραμματιστική διεπαφή PushNotificationService χρησιμοποιείται και από τις δύο υλοποιήσεις, καθιστώντας τη μετάβαση εντελώς διαφανή για τα ανώτερα στρώματα.

5.13.2 Πελάτης με FcmTokenManager

Ο FcmTokenManager εκτελείται μετά την επιτυχή σύνδεση και αναζητά διακριτικό σε τοπικό αποθετήριο. Αν δεν υπάρχει, παράγει συνθετικό αναγνωριστικό synthetic-{UUID}. Στη συνέχεια καλεί POST /notifications/devices με το διακριτικό, το αναγνωριστικό συσκευής (πρώτοι 120 χαρακτήρες του Build.FINGERPRINT) και την πλατφόρμα. Η συνθετική συμπεριφορά καλύπτει το περιβάλλον ανάπτυξης χωρίς ρυθμίσεις Firebase, διατηρώντας λειτουργικό τον υπόλοιπο μηχανισμό παράδοσης. Σε παραγωγή αντικαθίσταται από FirebaseMessaging.getInstance().token.await(), χωρίς αλλαγή στους καλούντες.

Η οντότητα fcm_token αποθηκεύει μοναδικό διακριτικό ανά συσκευή και διατηρεί χρόνο τελευταίας προσπέλασης. Η εγγραφή ενημερώνεται σε κάθε νέα σύνδεση μέσω upsert, ώστε η ίδια συσκευή να μη δημιουργεί διπλότυπες εγγραφές, ακόμη και αν αλλάξει ο αντίστοιχος χρήστης.

5.13.3 Ανίχνευση μεταβολών τιμών

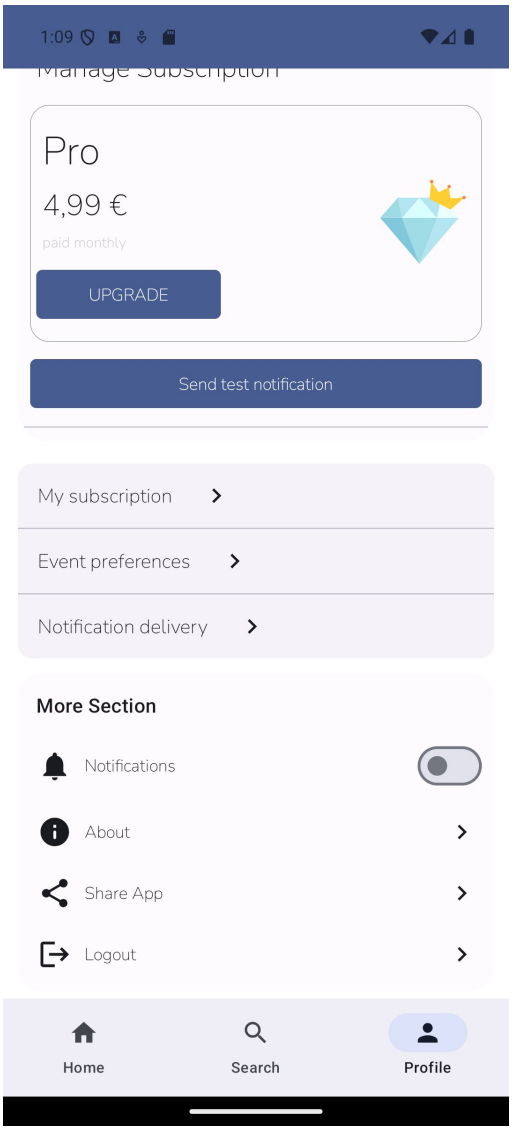
Ο PriceChangeDetector είναι προγραμματισμένο έργο (@Scheduled(cron="0 */15 * * * *")) που σαρώνει τις ενεργές λίστες παρακολούθησης. Για κάθε δωμάτιο ανακτά τις δύο πιο πρόσφατες τιμές. Αν η ποσοστιαία διαφορά υπερβαίνει το κατώφλι του χρήστη (πεδίο threshold_percent των προτιμήσεων), εκπέμπει συμβάν είτε PRICE_DROP είτε PRICE_RISE. Ο NotificationDispatcher συνεκτιμά τις προτιμήσεις και παραδίδει την ειδοποίηση στους επιλεγμένους διαύλους.

Παράλληλα, ένα δεύτερο προγραμματισμένο έργο (@Scheduled(cron="0 0 9 * * *")) εκπέμπει συμβάν DAILY_SUMMARY για κάθε χρήστη με ενεργοποιημένη την ημερήσια σύνοψη, αναφέροντας τις παρακολουθούμενες λίστες και δωμάτια. Η χρονοδρομολόγηση ενεργοποιείται από την ανώτατη κλάση PriceMonitoringBackendApplication με την σήμανση @EnableScheduling.

5.14 Επικαιροποιήσεις στην οθόνη Προφίλ

Η καρτέλα Profile εμπλουτίζεται με τρεις νέες εγγραφές πλοήγησης: My subscription, Event preferences και Notification delivery. Οι εγγραφές δομούνται σε κοινό Surface με RoundedCornerShape 12 dp και Divider μεταξύ τους. Κάθε σειρά εκτείνεται σε ολόκληρο το διαθέσιμο πλάτος, δείχνει ετικέτα 16 sp αριστερά και βέλος δεξιά, ακολουθώντας το πρότυπο Material 3 για λίστες πλοήγησης.

Η μετάβαση ορίζεται στον HomeNavHost: η οθόνη ProfileTabScreen δέχεται τρία onXxxClick lambda, τα οποία ο γονέας αποδίδει σε navController.navigate. Με τον τρόπο αυτό η οθόνη παραμένει αδιάφορη ως προς τη δομή του πλοηγού και δοκιμάζεται μεμονωμένα.



Σχήμα 5.7: *Καρτέλα προφίλ με νέες εγγραφές προς συνδρομή και προτιμήσεις.*

5.15 Δομή πακέτων και αρθρωμάτων

Η εφαρμογή οργανώνεται με κριτήριο τα χαρακτηριστικά (feature modules), εντός ενός μοναδικού αρθρώματος Gradle για λόγους ευκολίας ανάπτυξης. Στο εσωτερικό του βασικού πακέτου com.pricemon.pricemonitoring συναντώνται οι ακόλουθες ανώτερες ομάδες:

- features.splash: η οθόνη εκκίνησης και ο μηχανισμός διακλάδωσης μετά την ανάγνωση διακριτικού.
- features.login: σύνδεση, εγγραφή, OTP email και σχετικά ViewModel και repository.
- features.verifyEmail: η OTPScreen και τα βοηθητικά συστατικά εισαγωγής διακριτικού.
- features.verifyPhone: η νέα οθόνη επαλήθευσης κινητού (PhoneOtpScreen) με αυτόματη αποστολή κωδικού και επανάληψη.
- features.dashboard: το DashboardScreen, οι τρεις καρτέλες και το DashboardViewModel.

- features.details: η οθόνη λεπτομερειών λίστας και ο διάλογος ιστορικού τιμών.
- features.hotel: η οθόνη ξενοδοχείου.
- features.subscription: οι οθόνες πλάνων, checkout και διαχείρισης, μαζί με το SubscriptionViewModel.
- features.preferences: οι οθόνες προτιμήσεων συμβάντων και διαύλου παράδοσης, με κοινό PreferencesViewModel.
- features.fcm: ο FcmTokenManager και οι βοηθητικές κλάσεις αναγγελίας διακριτικού.
- base: κοινά ApiService, SessionManager, components και ui.
- data: αντικείμενα μεταφοράς δεδομένων (requests/responses) με σήμανση Gson ή kotlinx serialization.
- domain.repositories: UserRepository, SubscriptionRepository, NotificationsRepository.
- ui.theme: παλέτα χρωμάτων, τυπογραφία και διαχείριση θέματος.

Η διαίρεση αυτή διασφαλίζει ότι κάθε νέα δυνατότητα προστίθεται ως αυτοτελής φάκελος με τις απαραίτητες Composable, ViewModel και Hilt ενότητες, ελαχιστοποιώντας τις παρενέργειες σε άλλα τμήματα. Παράλληλα τα κοινά base και ui.theme καλύπτουν τις οπτικές και δικτυακές ανάγκες όλων των οθονών, αποφεύγοντας τη διπλασιασμένη αναπαραγωγή κώδικα.

5.16 Σχέδιο πλοήγησης και deep links

Το αρχείο Screen.kt δηλώνει sealed class με σταθερά αντικείμενα ένα ανά οθόνη. Η επιλογή sealed class αντί απλών συμβολοσειρών επιτρέπει στη μεταγλώττιση να εντοπίζει απουσία when κλάδων κατά την επέκταση, αποτρέποντας σιωπηλές παραλείψεις. Κάθε αντικείμενο φέρει route: String και προαιρετική λίστα navArguments, ενώ οι οθόνες με παραμέτρους παρέχουν createRoute(...): String ώστε ο καλών να μη συνθέτει χειροκίνητα τις παραμέτρους.

Δύο διακριτά NavHost συντονίζουν την πλοήγηση. Το ένα (MonitorAppNavHost) χειρίζεται την αρχική ροή χωρίς ταυτοποίηση: Splash, Login, Signup, OTP, PhoneOtp και τέλος Dashboard. Το δεύτερο (HomeNavHost) ζει εντός του Dashboard και ομαδοποιεί τις τρεις κάτω καρτέλες με τις δευτερεύουσες οθόνες (λεπτομέρεια λίστας, ξενοδοχείο, οθόνες συνδρομής και προτιμήσεων).

Η σταθερά Screen.Splash.route χρησιμοποιείται από όλους τους κλάδους που πρέπει να επιστρέψουν στην εκκίνηση μετά από επιτυχή αποσύνδεση, αποτρέποντας την περιπλάνηση σε στοίβες πλοήγησης που δεν γνωρίζουν αν ο χρήστης είναι ταυτοποιημένος. Όπου είναι σκόπιμη η άμεση μετάβαση χωρίς πιθανότητα επιστροφής (π.χ. από OTP προς PhoneOtp), χρησιμοποιείται popUpTo(...){inclusive = true}.

5.17 Σύστημα θέματος και προσβασιμότητα

Το θέμα ορίζεται στο πακέτο ui.theme. Η βάση είναι MaterialTheme με δυναμική παλέτα: σε συσκευές Android 12 και νεότερες ο dynamicLightColorScheme εξάγει τόνους από την ταπετσαρία, ώστε η εφαρμογή να εναρμονίζεται με την υπόλοιπη συσκευή. Σε παλαιότερες εκδόσεις χρησιμοποιείται κατασκευασμένη παλέτα με τιμές που έχουν επαληθευτεί για ικανοποιητική αντίθεση WCAG AA ανάμεσα σε φόντο και κείμενο.

Η τυπογραφία εκτείνεται στα προκαθορισμένα στυλ display, headline, title, body και label. Όλες οι οθόνες χρησιμοποιούν αυτά τα στυλ αντί ευκαιριακών TextStyle, ώστε αλλαγές μεγέθους ή γραμματοσειράς να αντικατοπτρίζονται κεντρικά. Τα εικονίδια χρησιμοποιούν Icons.Filled και Icons.AutoMirrored.Filled για στοιχεία που πρέπει να αναστραφούν σε γλώσσες δεξιά-προς-αριστερά.

Για προσβασιμότητα, κάθε σημαντικό συστατικό συνοδεύεται από contentDescription ή semantics εφόσον δεν φέρει ορατή ετικέτα. Τα κουμπιά έχουν ελάχιστο μέγεθος αφής 48 dp, ενώ τα πεδία εισαγωγής χρησιμοποιούν keyboardOptions με κατάλληλο KeyboardType, π.χ. Number για τον OTP κωδικό κινητού. Στο OutlinedTextField της κάρτας υιοθετείται KeyboardType.NumberPassword, ώστε ο CVC να μην παρουσιάζεται σε καθαρή μορφή.

5.18 Ελεγκτές κατάστασης και StateFlow

Όλα τα ViewModel εκθέτουν δημόσια StateFlow ή SharedFlow κατάστασης. Η επιλογή τύπου ροής εξαρτάται από τη σημασιολογία:

- StateFlow: για κατάσταση που πρέπει να ξαναεμφανίζεται σε νέους συλλέκτες, π.χ. current: SubscriptionCurrentResponse ή state: PreferencesUiState. Η οθόνη συγκεντρώνει με collectAsState() και ο πρώτος συλλέκτης λαμβάνει την τελευταία γνωστή τιμή.
- SharedFlow: για συμβάντα μίας χρήσης, π.χ. events: SharedFlow<SubscriptionEvent>. Η οθόνη συγκεντρώνει με collectLatest και χειρίζεται καθέκαστο μήνυμα χωρίς το γεγονός να αναπαράγεται σε επανασύνθεση.

Η ξεκάθαρη αυτή διάκριση αποτρέπει το πρόβλημα “διπλό Toast” που εμφανίζεται όταν LiveData κρατά τελευταίο γεγονός. Επιπλέον, ο σχεδιασμός ViewModel → repository → ApiService απομονώνει τον δικτυακό κώδικα από την παρουσίαση. Ο repository ενθυλακώνει την επιλογή του πηγαίου σημείου (π.χ. απομακρυσμένο ή τοπικό cache) και επιστρέφει Response<...>. Το ViewModel αναλαμβάνει την μετατροπή σε δηλωτική κατάσταση οθόνης, χωρίς να εκθέτει στις οθόνες ωμά αντικείμενα δικτύου.

5.19 Αντιμετώπιση βραχύχρονων σφαλμάτων

Η εφαρμογή πρέπει να ανταποκρίνεται σε τρεις βασικές κατηγορίες βραχύχρονων σφαλμάτων: αποτυχία DNS, υπερχείλιση χρονοδιαγράμματος TCP και έγκυρη απάντηση HTTP με κωδικό αποτυχίας.

Στο πρώτο επίπεδο, οι catch τμήματα του Flow καταγράφουν τη συγκεκριμένη εξαίρεση και εκπέμπουν Failure με ευανάγνωστο μήνυμα. Στο δεύτερο, ο πελάτης OkHttp έχει ρυθμισμένο connectTimeout = 10s, readTimeout = 30s και writeTimeout = 30s. Η υπέρβαση αυτών εκπέμπει SocketTimeoutException, την οποία οι ViewModel μεταφράζουν σε Server timeout (no response).

Στο τρίτο επίπεδο, ο LoginViewModel εξετάζει τον κωδικό απάντησης: 400 ως Bad request, 401 ως Unauthorized, 403 ως Forbidden, 404 ως Not found, 409 ως This user already exists, 429 ως Too many requests from this network. Try again later, 500 ως Server error και 503 ως Service unavailable. Το ίδιο πρότυπο εφαρμόζεται και στα νέα ViewModel (SubscriptionViewModel και PreferencesViewModel) μέσω της κοινής συνάρτησης parseErrorMessage, ώστε ο χρήστης να αναγνωρίζει εύκολα την κατηγορία αποτυχίας.

Επιπρόσθετα, ο επικείμενος Authorization ένθετος (interceptor) διαβάζει το user_token του SessionManager κατά κάθε κλήση και εφαρμόζει header("Authorization", "Bearer $token"). Έτσι η οθόνη ή το ViewModel δεν χρειάζεται να γνωρίζει το διακριτικό, περιορίζοντας τη διασπορά μυστικών στα αρχεία πηγής. Όταν ο ένθετος ανιχνεύσει αναβαθμισμένο διακριτικό μέσα στην απάντηση (π.χ. μετά από επιτυχή αγορά συνδρομής), ενημερώνει αυτόματα τον SessionManager, ώστε οι επόμενες κλήσεις να φέρουν τους ενημερωμένους περιορισμούς πλάνου.

5.20 Αυτόματη καταχώριση συσκευής

Στο πεδίο εφαρμογών για κινητά, η αναγνώριση συσκευής συντονίζεται με την παρακολούθηση χρήστη: για να φτάσει μια απομακρυσμένη ώθηση στη σωστή συσκευή, ο διακομιστής γνωρίζει το διακριτικό FCM κάθε συσκευής. Η εφαρμογή χρησιμοποιεί τον FcmTokenManager ως ενιαίο σημείο εισόδου.

Σχεδιαστικά, ο FcmTokenManager έχει σήμανση @Singleton ώστε να επιζεί όλη τη διάρκεια της εφαρμογής. Το registerAfterLogin() καλείται από το Compose DashboardScreen εντός LaunchedEffect(Unit), τυλιγμένο σε ξεχωριστό ViewModel (FcmRegistrationViewModel) ώστε να συνεργάζεται σωστά με το Hilt. Η κλάση ξεκινάει CoroutineScope με SupervisorJob ώστε αποτυχία σε μία καταχώριση να μη διακόπτει μελλοντικές προσπάθειες.

Η λογική παραγωγής διακριτικού είναι διαμορφούμενη: σε παραγωγή ζητά το πραγματικό διακριτικό μέσω FirebaseMessaging.getInstance().token.await()· σε ανάπτυξη παράγει συνθετικό αναγνωριστικό synthetic-{UUID} που αποθηκεύεται σε ιδιωτικό SharedPreferences και επαναχρησιμοποιείται σε επόμενες εκκινήσεις. Ο διακομιστής δεν διακρίνει μεταξύ συνθετικού και πραγματικού διακριτικού, ώστε η ροή αναγγελίας να ασκείται μέχρι τέλους ακόμη και χωρίς ρύθμιση Firebase.

Η οντότητα fcm_token χρησιμοποιεί token ως μοναδικό κλειδί. Όταν ένας χρήστης συνδέεται σε νέα συσκευή με νέο διακριτικό, δημιουργείται νέα γραμμή· όταν ξανασυνδέεται με υπάρχουσα συσκευή, η εγγραφή ενημερώνεται με νέο last_seen_at. Η αποσύνδεση καλεί DELETE /notifications/devices/{token} ώστε να μην αποστέλλονται ειδοποιήσεις σε συσκευή που εγκατέλειψε ο χρήστης.

5.21 Πρότυπα Compose και ανάκληση συνθέσεων

Η εφαρμογή υιοθετεί τέσσερα πρότυπα Compose που επαναλαμβάνονται στα μεγαλύτερα σημεία διεπαφής. Πρώτο, η *ανύψωση κατάστασης* (state hoisting) τοποθετεί τη μεταβλητή κατάσταση στον υψηλότερο απαραίτητο συντελεστή, εκθέτοντας προς τα κάτω ένα ζεύγος ανάγνωσης και πραγματεύσης. Στην CheckoutScreen για παράδειγμα τα πεδία cardNumber, expiry, cvc ζουν εντός remember{ mutableStateOf(...) } στο επίπεδο της οθόνης, ενώ καθέκαστο OutlinedTextField δέχεται value και onValueChange, χωρίς να γνωρίζει την υπόλοιπη φόρμα.

```
var cardNumber by remember { mutableStateOf("4242 4242 4242 4242") }
OutlinedTextField(
   value = cardNumber,
   onValueChange = { cardNumber = it },
   label = { Text("Card number") },
   keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
)
```

Δεύτερο, ο *έλεγχος επανασύνθεσης* μέσω derivedStateOf περιορίζει τα παράγωγα κόμβων σύνθεσης. Στις λίστες με στατιστικά, παράγωγη τιμή *ελάχιστη τιμή* υπολογίζεται μόνο όταν αλλάζει το σύνολο δωματίων, όχι σε κάθε LazyColumn επανασύνθεση.

Τρίτο, το *LaunchedEffect-driven side effect* αναλαμβάνει τη σύνδεση μεταξύ ViewModel συμβάντων και ορατών ενεργειών. Κάθε οθόνη που πρέπει να αντιδρά σε νέα συμβάντα συγκεντρώνει με collectLatest, ώστε ένα νέο Loading γεγονός να ακυρώνει την επεξεργασία του προηγούμενου.

```
LaunchedEffect(Unit) {
   viewModel.events.collectLatest { event ->
      when (event) {
         SubscriptionEvent.Loading -> loading = true
         SubscriptionEvent.CheckoutSuccess -> {
            context.showToast("Welcome to $planId")
            onDone()
         }
         is SubscriptionEvent.Failure -> context.showToast(event.msg)
         else -> {}
      }
   }
}
```

Τέταρτο, οι *slot σύνθεσες* επιτρέπουν στις γονικές οθόνες να αποφασίζουν τι παρουσιάζεται. Η ContentWithProgress δέχεται την τρέχουσα κατάσταση φόρτωσης και προβάλλει CircularProgressIndicator σε κεντρικό Box πάνω από το παιδί που της εμπιστεύεται η γονική. Το ίδιο πρότυπο εφαρμόζεται και στις Snackbar ζώνες, χωρίς να αναγκάζει την υπόλοιπη οθόνη να γνωρίζει το ScaffoldState.

5.22 Διαχείριση εξαρτήσεων με Hilt

Το Hilt σχηματίζει τέσσερις βασικές ενότητες (Modules): NetworkModule που εκθέτει OkHttpClient, Retrofit και MonitorApiService· DispatcherModule που εκθέτει Dispatchers.IO

ως CoroutineDispatcher· StorageModule που δίνει SessionManager και SharedPreferences. Τέλος, ένα έμμεσο σύνολο από ViewModel εργοστάσια που δημιουργούνται αυτόματα μέσω της @HiltViewModel σήμανσης.

Με την προσθήκη των νέων χαρακτηριστικών εισήλθαν νέα προφίλ δίκτυου: SubscriptionRepository, NotificationsRepository και FcmTokenManager. Όλα δηλώνονται με @Inject constructor, ώστε ο γράφος εξαρτήσεων του Hilt να τα συντηρεί χωρίς πρόσθετη ρύθμιση. Όπου χρειάζεται κοινή στιγμιότυπος (π.χ. ο FcmTokenManager κρατά τον SupervisorJob), η σήμανση @Singleton ενεργοποιείται. Όπου χρειάζεται νέα στιγμιότυπος ανά οθόνη (π.χ. ένα SubscriptionViewModel ανά πλοηγηση), χρησιμοποιείται @HiltViewModel χωρίς πρόσθετη σήμανση.

Η σύνδεση με τη σύνθεση Compose γίνεται μέσω hiltViewModel() εντός των οθονών, ώστε το ViewModel να συνδέεται με την αντίστοιχη καταχώριση NavBackStackEntry. Όταν η οθόνη αποσυνδέεται από τη στοίβα (π.χ. με popBackStack), το ViewModel και οι εκκρεμείς Job του εκκαθαρίζονται αυτόματα, χωρίς ρητή διαχείριση.

5.23 Διασύνδεση με Manifest και Activity

Το AndroidManifest.xml δηλώνει την άδεια android.permission.INTERNET και την άδεια χρόνου εκτέλεσης android.permission.POST_NOTIFICATIONS (απαραίτητη από Android 13 και νεότερες εκδόσεις). Η android:name ορίζεται σε .MonitorApplication, μια κλάση που σημαίνεται με @HiltAndroidApp ώστε να αρχικοποιείται ο γράφος εξαρτήσεων πριν ξεκινήσει η MainActivity.

Η MainActivity είναι ComponentActivity που καλεί setContent με την κορυφαία PriceMonitorApp σύνθεση. Η μοναδική ευθύνη της είναι η σύνθεση του ViewTree και η εφαρμογή του PriceMonitoringTheme, χωρίς ίδιο επιχειρησιακό περιεχόμενο. Η σήμανση @AndroidEntryPoint επιτρέπει την έγχυση Hilt σε επόμενα Activity/Service, αν και η εφαρμογή ακολουθεί μονοακτιβιτιακό πρότυπο.

Για την παράδοση push ειδοποιήσεων υπό κανονικές συνθήκες παραγωγής, στο AndroidManifest.xml προστίθεται καταχώριση ξεχωριστού Service που επεκτείνει FirebaseMessagingService. Η καταχώριση φέρει <intent-filter> με <action android:name="com.google.firebase.MESSAGING_EVENT" />, ώστε το σύστημα Android να δρομολογεί εισερχόμενες ειδοποιήσεις στην αντίστοιχη υπηρεσία. Σε περιβάλλον ανάπτυξης χωρίς ρυθμίσεις Firebase, η υπηρεσία αυτή απουσιάζει και τη θέση της παίρνει η συνθετική καταχώριση διακριτικού του FcmTokenManager, ώστε ο κορμός παράδοσης να μπορεί να δοκιμαστεί.

5.24 Στρατηγική δοκιμών νέων χαρακτηριστικών

Τα νέα χαρακτηριστικά απαιτούν τόσο χειροκίνητες όσο και αυτόματες δοκιμές. Η ομάδα δοκιμών οργανώνεται σε τρία επίπεδα.

Στο πρώτο επίπεδο, τα *ενδιάμεσα μονοδιάστατα τεστ* ελέγχουν τη λογική των ViewModel χωρίς εξάρτηση από διακομιστή. Το SubscriptionViewModel δοκιμάζεται με ψεύτικο

SubscriptionRepository που επιστρέφει σταθερές απαντήσεις: ένα τεστ ελέγχει ότι σε επιτυχία checkout αποθηκεύεται το νέο διακριτικό και ενημερώνεται η ροή current, ένα δεύτερο ότι σε HTTP 402 εκπέμπεται Failure με κατάλληλο μήνυμα.

Στο δεύτερο επίπεδο, τα *τεστ ενσωμάτωσης* κατεβάζουν τοπικά Spring Boot και χτυπούν τα νέα τελικά σημεία διεπαφής με MockMvc και Testcontainers. Παράδειγμα τεστ: αίτημα POST /phone/verification/request ακολουθούμενο από POST /phone/verification/confirm με τον σωστό κωδικό παράγει JWT. Ένας δοκιμαστικός χρήστης μπορεί να εκτελέσει POST /customer/register συνδυάζοντας email JWT και phone JWT, και η εγγραφή να έχει σωστά συμπληρωμένα τα πεδία phone και phone_verified_at.

Στο τρίτο επίπεδο, τα *χειροκίνητα σενάρια* εξετάζουν τη ροή από άκρο σε άκρο σε προσομοιωτή. Το σενάριο "Επιτυχής αγορά Premium" περιλαμβάνει: εγγραφή νέου χρήστη με επαλήθευση κινητού, σύνδεση, μετάβαση στις προτιμήσεις, ενεργοποίηση event_price_drop, μετάβαση στις συνδρομές, επιλογή Premium, εισαγωγή κάρτας 4242 4242 4242 4242 με αυτόματη ανανέωση και επιβεβαίωση ότι η οθόνη My subscription εμφανίζει ACTIVE και υπολειπόμενες 30 ημέρες. Το αντίθετο σενάριο εκτελείται με την κάρτα 4000 0000 0000 0002: το HTTP 402 μεταφράζεται σε Payment declined και η συνδρομή παραμένει USER.

5.25 Παρατηρητικότητα και αρχεία καταγραφής

Η εφαρμογή και ο διακομιστής συντονίζονται μέσω συμβατικών αρχείων καταγραφής, ώστε η εξέλιξη ενός σεναρίου από τη συσκευή έως το MySQL να μπορεί να επαληθευτεί.

Στη συσκευή, ο Log.d("AuthFlow", …) καταγράφει εκπομπές συνεδρίας: Login using BASE_URL=…, Login response status=… και ισοδύναμα για εγγραφή. Παρόμοιοι λογότυποι (Sub, Prefs, Fcm) καλύπτουν τις νέες ροές. Ο φάκελος logcat φιλτράρεται από την εντολή adb logcat -s AuthFlow Sub Prefs Fcm για στοχευμένη παρακολούθηση.

Στον διακομιστή, το logback-spring.xml προωθεί καταγραφές σε stdout, οπότε το docker compose logs backend αναπαριστά την ίδια στοίβα γεγονότων. Σε σενάριο αγοράς, ο διακομιστής καταγράφει Stub payment OK με αναγνωριστικό αναφοράς, Expired subscription {id} σε αυτόματη λήξη και Imitating push send σε διασπορά μέσω LoggingPushNotificationService. Σε σενάριο επαλήθευσης κινητού καταγράφει Imitating SMS send to {phone}: OTP={code}, που είναι το μόνο εργαλείο ανάκτησης του κωδικού σε ανάπτυξη χωρίς πραγματικό πάροχο SMS.

Η συνθήκη αυτή σχηματίζει αναθεωρήσιμο πλάνο: σε αποτυχία ροής, ο χρήστης ή ο προγραμματιστής συσχετίζει το χρονοσήμαντρο της συσκευής με αυτό του διακομιστή και αναγνωρίζει σε ποιο στρώμα συνέβη η απόρριψη ή η εξαίρεση. Σε παραγωγή, η ίδια προσέγγιση ενσωματώνεται σε centralized log aggregation (π.χ. Loki/Grafana) χωρίς αλλαγή κώδικα.

5.26 Σχεσιακό σχήμα νέων χαρακτηριστικών

Οι νέες δυνατότητες απαιτούν επεκτάσεις του σχήματος της βάσης δεδομένων. Όλες οι αλλαγές γίνονται μέσω Liquibase σε διακριτά changeSet αρχεία, ώστε η μετάβαση να είναι εμπρός μόνο (forward-only) και ελέγξιμη με DATABASECHANGELOG.

Πίνακας users: προστίθενται οι στήλες phone VARCHAR(20) και phone_verified_at DATETIME, και οι δύο επιτρέπουν NULL ώστε υπάρχοντες χρήστες να διατηρούν λογαριασμό χωρίς υποχρεωτική επαναλήψη επαλήθευσης κινητού. Για νέους χρήστες, το πεδίο phone_verified_at συμπληρώνεται κατά την οριστική εγγραφή.

Πίνακας phone_verification: αποθηκεύει μεταβατικά στοιχεία επαλήθευσης κινητού. Στήλες: email VARCHAR(254), phone VARCHAR(20), otp_hash VARCHAR(72), attempts INT DEFAULT 0, expires_at DATETIME, consumed_at DATETIME NULL, created_at DATETIME DEFAULT NOW(). Συμπληρωματικός δείκτης σε email επιταχύνει την αναζήτηση της πιο πρόσφατης ενεργής εγγραφής για συγκεκριμένο χρήστη.

Πίνακας subscription: user_id BIGINT FK users.id, plan VARCHAR(20), status VARCHAR(20), starts_at DATETIME, expires_at DATETIME, auto_renew BOOLEAN DEFAULT FALSE, cancelled_at DATETIME NULL, created_at DATETIME DEFAULT NOW(). Συνθετικός δείκτης (user_id, status) επιταχύνει την αναζήτηση ενεργής συνδρομής.

Πίνακας payment_transaction: user_id BIGINT, subscription_id BIGINT NULL FK subscription.id, amount DECIMAL(10,2), currency VARCHAR(3) DEFAULT 'EUR', status VARCHAR(20), provider VARCHAR(20) DEFAULT 'STUB', provider_ref VARCHAR(64) NULL, card_last4 VARCHAR(4) NULL, created_at DATETIME DEFAULT NOW(). Οι συναλλαγές αποτυχίας αποθηκεύονται με status='DECLINED' και χωρίς subscription_id, ώστε ο διακομιστής να εξάγει στατιστικά αποτυχίας.

Πίνακας user_notification_preferences: user_id BIGINT PK FK users.id, channels VARCHAR(40) DEFAULT 'PUSH,EMAIL', event_price_drop BOOLEAN DEFAULT TRUE, event_price_rise BOOLEAN DEFAULT FALSE, event_daily_summary BOOLEAN DEFAULT FALSE, event_weekly_digest BOOLEAN DEFAULT FALSE, threshold_percent INT DEFAULT 5, updated_at DATETIME DEFAULT NOW(). Η αποθήκευση των διαύλων ως απλή συμβολοσειρά με χωριστικό κόμμα βοηθάει την επέκταση (προσθήκη SMS ως μελλοντικού καναλιού) χωρίς αναδρομικές μεταβάσεις σχήματος.

Πίνακας fcm_token: user_id BIGINT FK users.id, token VARCHAR(512) UNIQUE, device_id VARCHAR(128) NULL, platform VARCHAR(20) DEFAULT 'ANDROID', created_at, last_seen_at. Ο περιορισμός μοναδικότητας στο token εμποδίζει διπλότυπα, ενώ ο δείκτης σε user_id καθιστά αποδοτική την ανάκτηση όλων των διακριτικών ενός χρήστη κατά τη διάρκεια διασποράς ειδοποίησης.

Στο επίπεδο εφαρμογής, οι αντίστοιχες οντότητες JPA εκθέτουν απλά var πεδία αντί val, επιτρέποντας στο Hibernate τη φόρτωση και επαναεγγραφή χωρίς διπλασιασμένα πρότυπα κατασκευής. Όπου

χρησιμοποιείται Enum, η αντίστοιχη @Enumerated(EnumType.STRING) σήμανση εξασφαλίζει ότι οι αλλαγές σειράς δηλώσεων enum δεν ακυρώνουν ιστορικά δεδομένα.

5.27 Συμπεριφορά εκτός σύνδεσης

Η εφαρμογή δεν αποθηκεύει εκτενώς δεδομένα τοπικά, αλλά οι μηχανισμοί StateFlow κρατούν την τελευταία γνωστή τιμή στη μνήμη όσο ζει το ViewModel. Όταν η συσκευή χάνει συνδεσιμότητα, η οθόνη δείχνει την προηγούμενη κατάσταση μαζί με ένα Snackbar με μήνυμα No internet connection.

Για τα δεδομένα συνδρομής, η ροή είναι ιδιαίτερα ευαίσθητη: σε απουσία δικτύου, η οθόνη My subscription εξακολουθεί να δείχνει την κατάσταση που ανέκτησε το προηγούμενο επιτυχές αίτημα. Όταν η συσκευή ξαναβρίσκει δίκτυο, το LaunchedEffect(Unit) της οθόνης ξανατρέχει refresh(), ενημερώνοντας τα δεδομένα. Με τον τρόπο αυτό αποφεύγεται η εμφάνιση κενής ή ψευδώς EXPIRED κατάστασης σε προσωρινό δικτυακό σφάλμα.

Οι ειδοποιήσεις διαχειρίζονται διπλά: το FCM εξασφαλίζει παράδοση όταν η συσκευή έχει σύνδεση, ενώ το ReminderWorker παράγει υπενθυμίσεις από δεδομένα που έχουν ήδη αποθηκευτεί τοπικά. Σε περίπτωση μακροχρόνιας αποσύνδεσης, ο διακομιστής δεν εκπέμπει αναδρομικά τις ληφθείσες ειδοποιήσεις: το FCM κρατά μόνο περιορισμένο αριθμό ειδοποιήσεων στην ουρά (collapse_key), οπότε ο χρήστης λαμβάνει την πιο πρόσφατη.

Η εφαρμογή φροντίζει ώστε σε επιστροφή στο δίκτυο ο FcmTokenManager να ξαναεγγραφεί το διακριτικό. Αν το διακριτικό άλλαξε όσο ήταν εκτός σύνδεσης (π.χ. μετά από επαναφορά συσκευής), η FirebaseMessagingService καλεί onNewToken(), που προωθεί το νέο διακριτικό στον FcmTokenManager. Σε ανάπτυξη, το συνθετικό διακριτικό παραμένει σταθερό από την πρώτη παραγωγή.

5.28 Συμμόρφωση με κατευθυντήριες Android

Η εφαρμογή στοχεύει compileSdk = 34 και minSdk = 26. Η εκλογή minSdk καλύπτει την ευρεία βάση συσκευών χωρίς την παράπλευρη πολυπλοκότητα παλαιότερων εκδόσεων Android.

Οι κατευθυντήριες Material 3 ακολουθούνται με αναπρόσδιόρισα χρώματα και τυπογραφία στο ui.theme. Όπου εμφανίζεται στοιχείο επιλογής, χρησιμοποιείται FilterChip (στις προτιμήσεις διαύλου) ή NavigationBarItem (στην κάτω γραμμή). Η ελάχιστη εξάρτηση από XML layouts κρατά τη συνολική εφαρμογή ευέλικτη και ευκολότερα τροποποιήσιμη.

Για την εμπειρία επαναφοράς, η ζώνη android:dataExtractionRules και android:fullBackupContent στο AndroidManifest ορίζουν αρχεία XML που εξαιρούν τα ευαίσθητα SharedPreferences του SessionManager από τη συνηθισμένη αντιγραφή ασφαλείας του Google. Με τον τρόπο αυτό, η μετάβαση σε νέα συσκευή απαιτεί επανασύνδεση, αλλά τα ευαίσθητα διακριτικά δεν αναπαράγονται σε Google Drive.

Σε επίπεδο διαμόρφωσης πλοήγησης, η networkSecurityConfig ορίζει cleartext επικοινωνία μόνο για τις δύο γνωστές διευθύνσεις ανάπτυξης (10.0.2.2 και η διεύθυνση LAN), ώστε η εκτέλεση εκτός

παραγωγής να μη ζητά πιστοποιητικά TLS. Σε παραγωγή, η εφαρμογή υποχρεούται σε https:// με έγκυρο πιστοποιητικό.

5.29 Σύνοψη αλλαγών

Η υλοποίηση που περιγράφηκε καλύπτει τα ακόλουθα:

- Η οργάνωση της εφαρμογής Android σε πακέτα χαρακτηριστικών και η αρχιτεκτονική MVVM με ViewModel–Repository–ApiService.
- Η πλοήγηση μεταξύ οθονών μέσω NavHost/HomeNavHost και η διαχείριση σύνθεσης Composable.
- Η ροή ταυτοποίησης σε δύο διαδοχικά διακριτικά (email + κινητό) με νέα PhoneOtpScreen και επεκταμένη SignupScreen.
- Η διαχείριση μηνιαίας συνδρομής με πλάνα, stub payment, οθόνη διαχείρισης και χρονοδρομολογημένη αυτόματη λήξη.
- Οι προτιμήσεις ειδοποιήσεων ανά συμβάν και δίαυλο, με κατώφλι ποσοστιαίας μεταβολής για την ενεργοποίηση κάθε ωθούμενης ειδοποίησης.
- Η ενσωμάτωση Firebase Cloud Messaging με υπό συνθήκη εναλλαγή σε αρχείο καταγραφής, ώστε η ροή να ασκείται και χωρίς Firebase project σε ανάπτυξη.
- Οι επεκτάσεις της οθόνης Προφίλ με τρεις νέες εγγραφές πλοήγησης που οδηγούν σε διαχείριση συνδρομής και προτιμήσεις.
- Η συμπεριφορά εκτός σύνδεσης, οι κατευθυντήριες Material 3, η ασφάλεια αντιγραφών και η συμμόρφωση με τις πιο πρόσφατες απαιτήσεις του Android.

Όλες οι αλλαγές παραμένουν συμβατές προς τα πίσω με υπάρχοντες λογαριασμούς και επιτρέπουν στον διαχειριστή του συστήματος να ενεργοποιήσει σταδιακά τις προηγμένες δυνατότητες (π.χ. πραγματικό FCM project, πραγματικός πάροχος SMS, πραγματική gateway πληρωμών) χωρίς αλλαγή κώδικα στα ανώτερα στρώματα.

5.30 Ενημερωμένη ροή εγγραφής βήμα προς βήμα

Η ενημερωμένη ροή εγγραφής συντονίζει τρεις διακριτές οθόνες και δύο εκδόσεις διακριτικού. Παρουσιάζεται εδώ σε βήματα ώστε να αναδειχθεί η αλληλεπίδραση μεταξύ LoginViewModel, διακομιστή και τοπικής κατάστασης.

1. Ο χρήστης ανοίγει την SignupScreen. Η οθόνη παρουσιάζει πέντε πεδία: ηλεκτρονική διεύθυνση, όνομα, κωδικός, επανάληψη κωδικού, αριθμός τηλεφώνου. Όλα τα πεδία αποθηκεύονται σε User (data class) μέσω remember.
2. Πάτημα του κουμπιού Send verification email εκπέμπει UserIntent.Register(user). Το LoginViewModel επικυρώνει τοπικά και, σε επιτυχία, καλεί POST /customer/email/verification με το πεδίο email. Ο pendingRegistrationUser αποθηκεύεται σε ιδιωτική μεταβλητή.

3. Ο διακομιστής επιστρέφει HTTP 200 και στέλνει ηλεκτρονικό μήνυμα με JWT επαλήθευσης. Σε ανάπτυξη, το NoOpEmailService καταγράφει το διακριτικό στα αρχεία καταγραφής, ώστε ο προγραμματιστής να το αντιγράψει.
4. Η εφαρμογή πλοηγείται στο OTPScreen. Ο χρήστης επικολλά το διακριτικό και πατά Create account. Το LoginViewModel αποθηκεύει το διακριτικό σε pendingEmailJwt και εκπέμπει UserUiEvents.EmailOtpVerified.
5. Η εφαρμογή πλοηγείται στο PhoneOtpScreen. Το LaunchedEffect(Unit) εκπέμπει αμέσως UserIntent.RequestPhoneOtp, που καλεί POST /phone/verification/request με τα πεδία email και phone. Ο διακομιστής παράγει εξαψήφιο κωδικό, αποθηκεύει το BCrypt αποτύπωμα στο phone_verification και καλεί τον MockSmsService ή τον πραγματικό πάροχο σε παραγωγή.
6. Ο χρήστης εισάγει τον κωδικό. Το πάτημα "Verify and create account" εκπέμπει UserIntent.ConfirmPhoneOtp(code), που καλεί POST /phone/verification/confirm. Σε επιτυχία επιστρέφεται το phoneJwt.
7. Το LoginViewModel καλεί την οριστική POST /customer/register αποστέλλοντας jwt (από email), phoneJwt, όνομα, κωδικό και επανάληψη. Ο διακομιστής επιβεβαιώνει και τα δύο διακριτικά, αποθηκεύει τον νέο χρήστη με συμπληρωμένα τα πεδία phone και phone_verified_at και επιστρέφει HTTP 200.
8. Η οθόνη εκπέμπει UserUiEvents.RegisterSuccess, ο πλοηγός γυρίζει στην οθόνη σύνδεσης με popUpTo(Screen.Signup) { inclusive = true }. Ο χρήστης μπορεί τώρα να συνδεθεί με τα νέα διαπιστευτήρια.

Κάθε βήμα της ροής χειρίζεται διακριτή αποτυχία. Σε αποτυχία επικύρωσης τοπικής φόρμας εμφανίζεται Toast και η ροή δεν προχωρά. Σε αποτυχία HTTP 429 (πολλά αιτήματα) εμφανίζεται σχετικό μήνυμα και η οθόνη ενθαρρύνει αναμονή. Σε αποτυχία 400 (π.χ. λάθος μορφή τηλεφώνου, ο διακομιστής εφαρμόζει Pattern "^\+?[1-9][0-9]{6,14}$") εμφανίζεται μήνυμα με τις απαιτήσεις. Σε αποτυχία διακριτικού (π.χ. λήξη πριν την αναβάθμιση σε phone_verification_confirm) ο χρήστης ξεκινά τη ροή από την αρχή.

5.31 Σύνοψη επιλογών σχεδιασμού

Οι κύριες σχεδιαστικές αποφάσεις που έγιναν κατά την υλοποίηση συνοψίζονται ως εξής.

Πρώτον, η εφαρμογή χωρίζει αυστηρά τη ροή χρήστη χωρίς ταυτοποίηση από τη ροή ταυτοποιημένου χρήστη, ώστε ο γράφος πλοήγησης να μη μπορεί να ξεγλιστρά σε μη επιτρεπτή κατάσταση. Δεύτερον, όλα τα νέα χαρακτηριστικά παραμένουν συμβατά με υπάρχοντες λογαριασμούς: χρήστες χωρίς τηλέφωνο μπορούν να συνδεθούν, χωρίς συνδρομή μπορούν να χρησιμοποιήσουν τη βασική λειτουργικότητα, χωρίς καταχωρημένο FCM διακριτικό μπορούν να λειτουργήσουν με τοπικές ειδοποιήσεις. Τρίτον, το έργο διατηρεί διπλά μονοπάτια για δοκιμή (mock SMS, stub payment, logging push) ώστε η ανάπτυξη να μη χρειάζεται προπληρωμένες υπηρεσίες.

Σε επίπεδο διεπαφής, το θέμα ακολουθεί Material 3 χωρίς παρεκκλίσεις, μειώνοντας το γνωστικό βάρος. Σε επίπεδο διακομιστή, οι νέες προγραμματισμένες εργασίες (PriceChangeDetector,

SubscriptionExpiryScheduler) ζουν σε διακριτές κλάσεις με @Scheduled σήμανση, επιτρέποντας άμεση διαφοροποίηση χρόνου ή απενεργοποίηση χωρίς αλλαγή λογικής. Σε επίπεδο βάσης δεδομένων, οι νέοι πίνακες σχεδιάζονται με κανονικοποίηση και κατάλληλους δείκτες, διατηρώντας την αποδοτικότητα ακόμη και σε χιλιάδες χρήστες.

Η σχέση NotificationDispatcher → PushNotificationService αξίζει ιδιαίτερη αναφορά: το LoggingPushNotificationService και το FcmPushNotificationService είναι δύο εναλλάξιμες υλοποιήσεις της ίδιας διεπαφής, ορίζοντας έναν σαφή μηχανισμό strategy pattern. Ο κωδικός επιλέγει την αντίστοιχη υλοποίηση με @ConditionalOnProperty, μια διακηρυκτική προσέγγιση που επιτρέπει στον διαχειριστή του συστήματος να ενεργοποιήσει το πραγματικό FCM με αλλαγή μιας μόνο γραμμής ρύθμισης.

Συνολικά, η εφαρμογή Android εξυπηρετεί τέσσερις διακριτές απαιτήσεις χρήστη: επαλήθευση κινητού στο ξεκίνημα, διαχείριση μηνιαίας συνδρομής με δυνατότητα αυτοεξυπηρέτησης, παραμετροποίηση συμβάντων ειδοποίησης και επιλογή διαύλου παράδοσης. Όλες οι απαιτήσεις υλοποιήθηκαν με συνεπή προσέγγιση MVVM, StateFlow ροές και δηλωτικά Composable με Material 3 εμφάνιση.

Κεφάλαιο 6ο: Έλεγχος και αποτελέσματα

6.1 Δοκιμές διακομιστή και συμβατότητα API

Η υλοποίηση του διακομιστή, με δοκιμές μονάδας και ολοκλήρωσης σε JUnit 5, Mockito, MockMvc και πραγματική βάση δεδομένων όπου απαιτείται, ανήκει στη συνεργαζόμενη εργασία του διακομιστή. Οι δοκιμές του διαδικτυακού πελάτη (web) εμπίπτουν αντίστοιχα στη δική του εργασία. Τα αποτελέσματα και των δύο δεν αναπαράγονται εδώ. Πριν από την ενσωμάτωση με την εφαρμογή Android, επαληθεύθηκε ότι το δημόσιο συμβόλαιο REST ανταποκρίνεται στις προσδοκίες του πελάτη, ως προς τους κωδικούς κατάστασης, τη μορφή JSON και την ταυτοποίηση με JWT (Internet Engineering Task Force (IETF) 2015).

6.2 Δοκιμές εφαρμογής Android

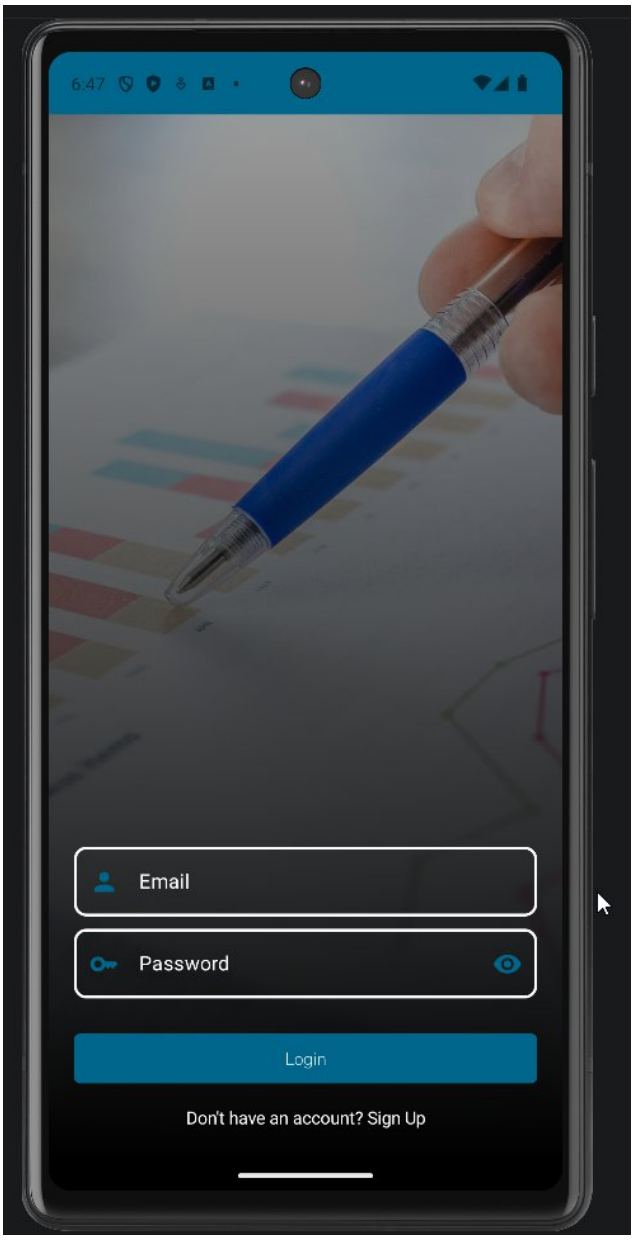
Η εφαρμογή ελέγχθηκε με δοκιμές λογικής των ViewModel, με δοκιμαστικά αποθετήρια και coroutines, καθώς και με ενσωματωμένες δοκιμές διεπαφής που εκτελούνται σε συσκευή ή προσομοιωτή όπου κρίθηκε αναγκαίο. Η κύρια επαλήθευση πραγματοποιήθηκε με χειροκίνητα σενάρια που καλύπτουν ροές σύνδεσης, αναζήτησης, διαχείρισης λιστών, προβολής ιστορικού τιμών και προγραμματισμένης τοπικής ειδοποίησης. Ο Πίνακας 6.1 συγκεντρώνει ενδεικτικά σενάρια, ενώ τα αντίστοιχα στιγμιότυπα διεπαφής παρουσιάζονται στην ενότητα 6.3.

Πίνακας 6.1: *Ενδεικτικά σενάρια και αναμενόμενα αποτελέσματα.*

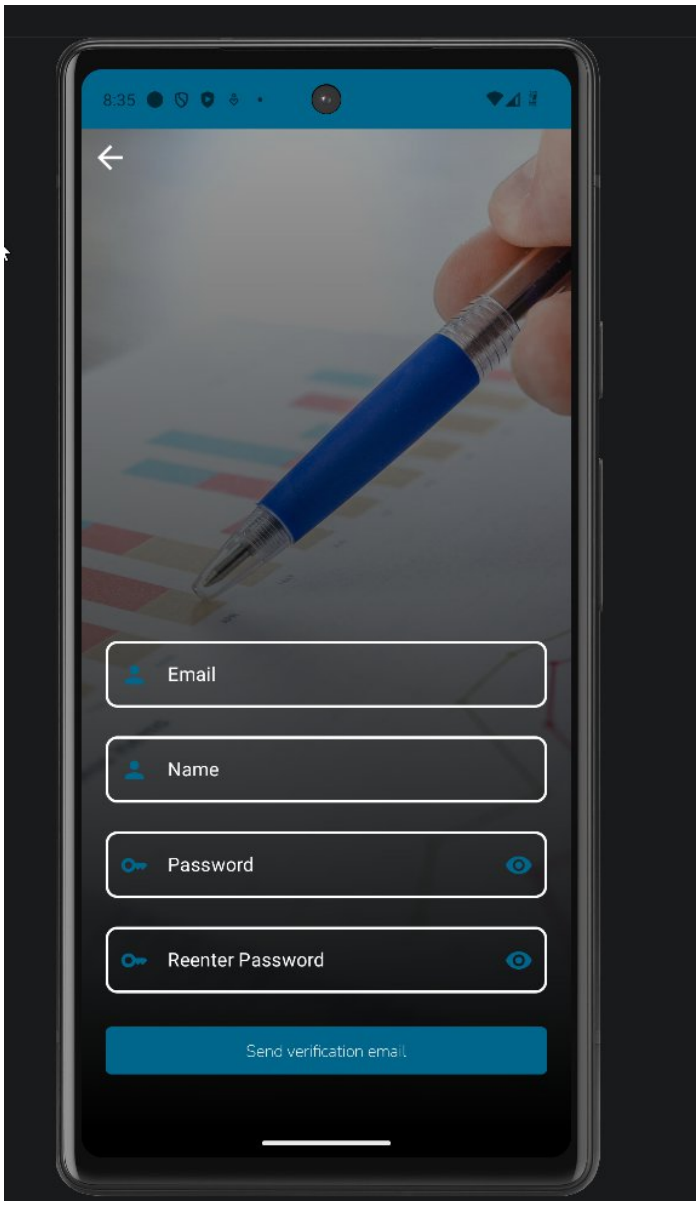
Σενάριο	**Αναμενόμενο αποτέλεσμα**
Εγγραφή με επαλήθευση email	Ολοκλήρωση ροής χωρίς σφάλματα επικύρωσης, αποθήκευση χρήστη.
Σύνδεση	Επιτυχής έκδοση JWT, αποθήκευση στο SessionManager, πρόσβαση σε προστατευμένα τελικά σημεία διεπαφής.
Αναζήτηση μέσω URL	Εμφάνιση ξενοδοχείου και δωματίων στη διεπαφή αναζήτησης.
Δημιουργία λίστας	Δημιουργία και εμφάνιση στην καρτέλα αρχικής οθόνης, πλοήγηση σε λεπτομέρειες.
Γράφημα τιμών	Ορθή σειρά σημείων από τη διεπαφή API, απόδοση γραφήματος Vico.
Τοπική ειδοποίηση	Εκτέλεση εργασίας WorkManager, εμφάνιση στο κανάλι ειδοποιήσεων.

6.3 Καταγραφές οθόνης

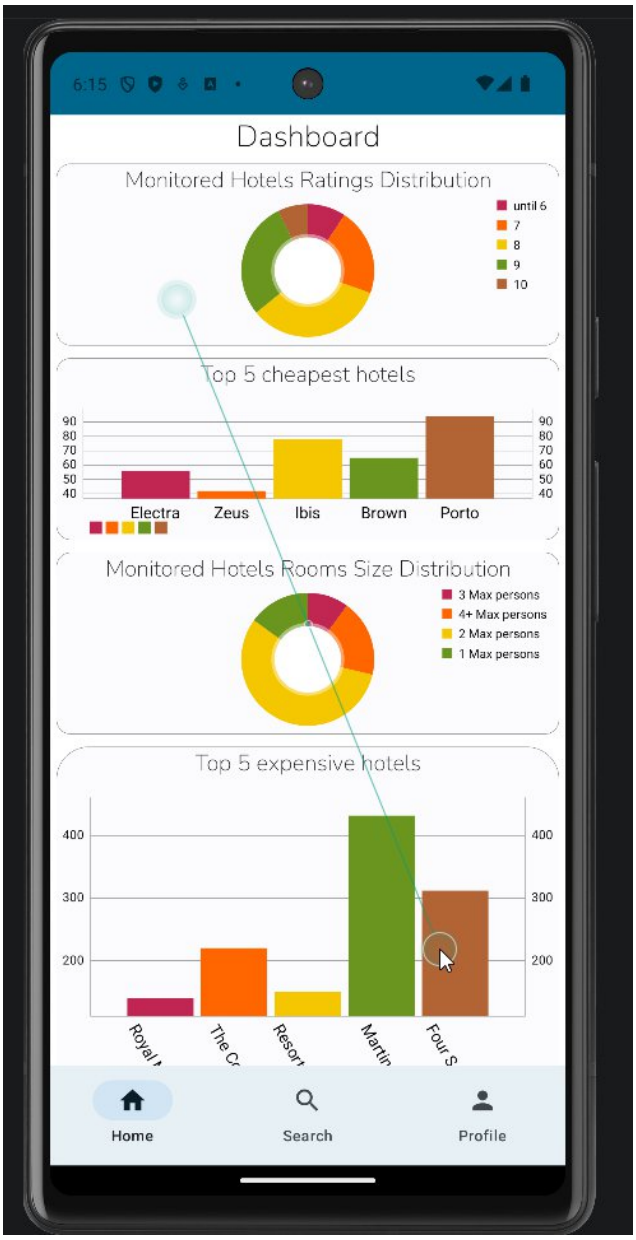
Τα παρακάτω σχήματα προέρχονται από προσομοιωτή ή φυσική συσκευή και επαληθεύουν οπτικά τις ροές του Πίνακα 6.1. Τα αρχεία βρίσκονται στον κατάλογο thesis/screenshots/ με τα ονόματα που ορίζει το README.txt.



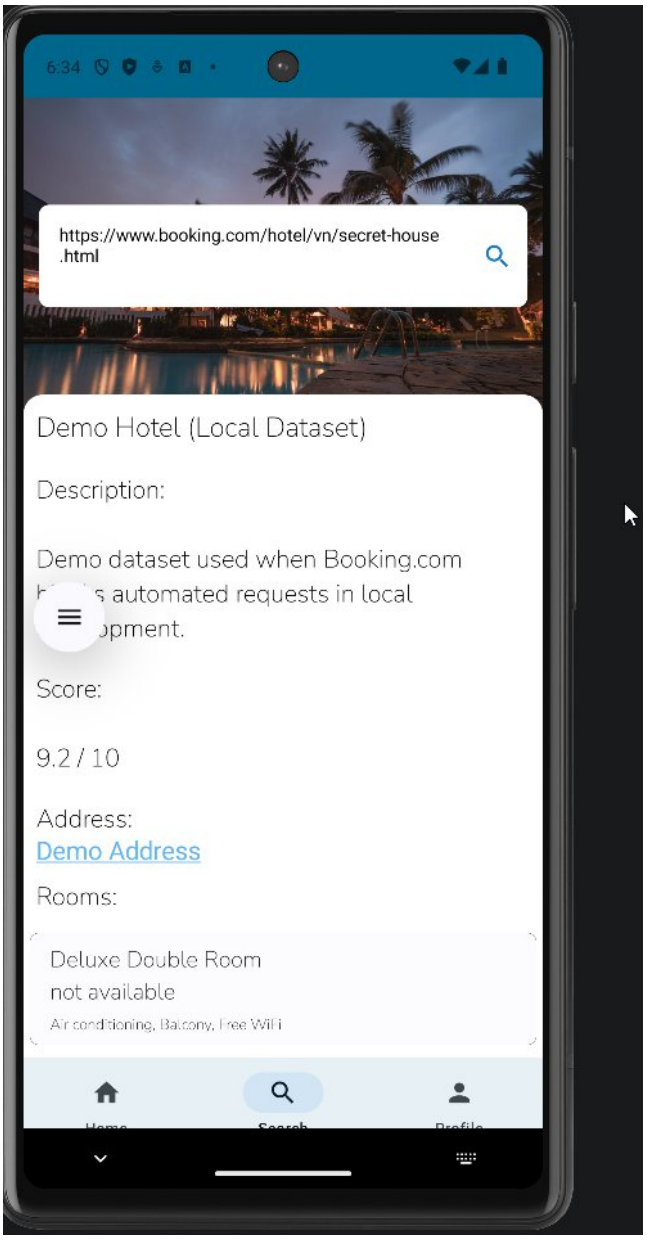
Σχήμα 6.1: *Οθόνη σύνδεσης: πεδία email/κωδικού και μετάβαση προς εγγραφή.*



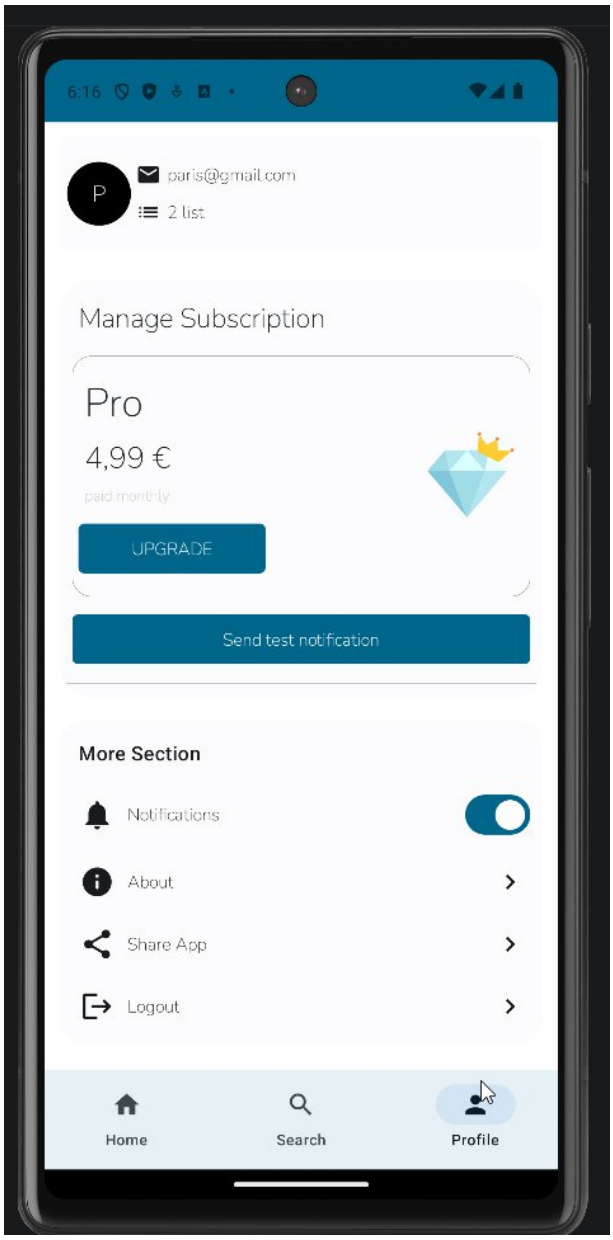
Σχήμα 6.2: *Οθόνη εγγραφής: στοιχεία λογαριασμού και αποστολή email επαλήθευσης.*



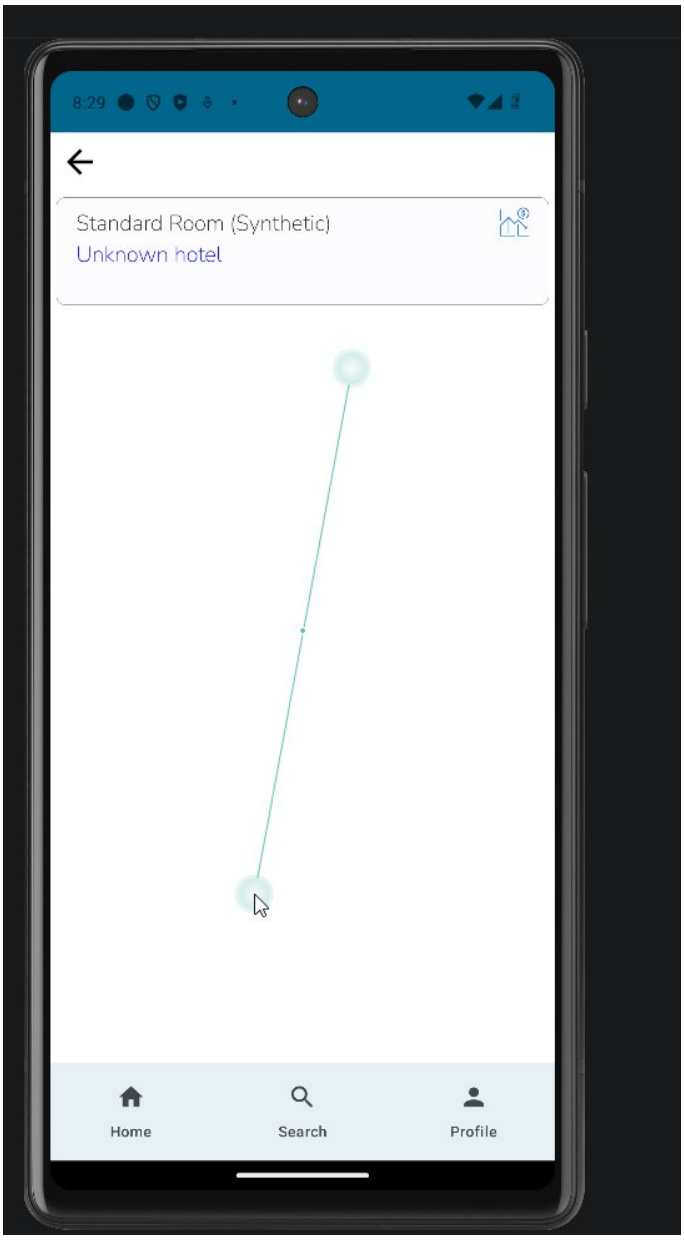
Σχήμα 6.3: *Καρτέλα αρχικής οθόνης: λίστες παρακολούθησης και σύνοψη δωματίων.*



Σχήμα 6.4: *Καρτέλα αναζήτησης: εισαγωγή URL και αποτελέσματα ξενοδοχείου.*

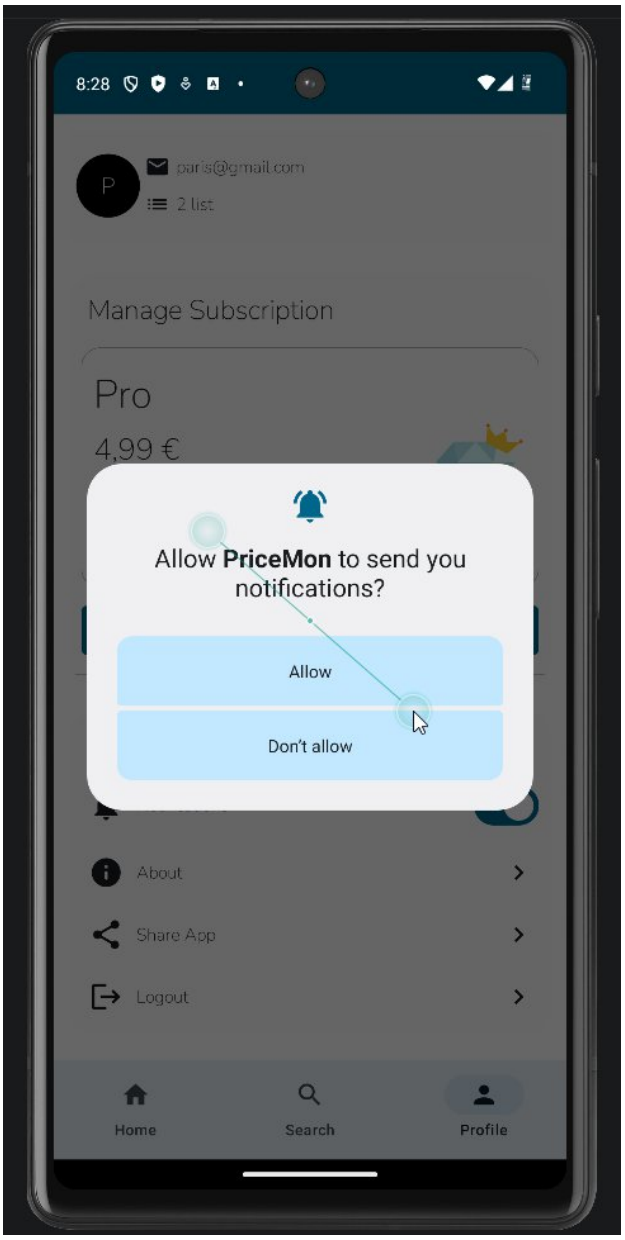


Σχήμα 6.5: *Καρτέλα προφίλ: στοιχεία λογαριασμού, συνδρομή και ρυθμίσεις ειδοποιήσεων.*



Σχήμα 6.6: *Λεπτομέρεια λίστας: δωμάτια, τιμές και ενέργειες διαχείρισης.*

Σχήμα 6.7: *Ιστορικό τιμών: διάγραμμα χρονοσειράς (Vico) έναντι API.*



Σχήμα 6.8: *Τοπική ειδοποίηση μετά από προγραμματισμένη εργασία (WorkManager).*

6.4 Τοπική στοίβα και διεπαφή

Με εκκίνηση MySQL, RabbitMQ και διακομιστή μέσω Docker Compose, καθώς και με ρύθμιση της έκδοσης ανάπτυξης (debug) προς 10.0.2.2:8085 ή προς τη διεύθυνση IP του τοπικού δικτύου σε φυσική συσκευή, επαληθεύθηκε η ροή ελέγχου διαθεσιμότητας, σύνδεσης και CRUD λιστών έναντι αναπαραγόμενης βάσης (Docker Inc., n.d.). Η πρόσβαση στο /hotel/info εξακολουθεί να εξαρτάται από τη διαθεσιμότητα του scraper και από τις πολιτικές της πηγής δεδομένων.

6.5 Δοκιμές νέων χαρακτηριστικών

Τα νέα χαρακτηριστικά (επαλήθευση κινητού, μηνιαία συνδρομή, προτιμήσεις ειδοποιήσεων, FCM αναγγελία) επικυρώθηκαν με μεθοδικά σενάρια που συνδυάζουν χειροκίνητη χρήση της εφαρμογής με άμεσες κλήσεις διεπαφής REST προς τον τοπικό διακομιστή.

6.5.1 Σενάριο εγγραφής με επαλήθευση κινητού

Το σενάριο ξεκινάει με αίτημα POST /phone/verification/request που εμπεριέχει νέα ηλεκτρονική διεύθυνση και αριθμό τηλεφώνου σε μορφή E.164. Ο διακομιστής επιστρέφει HTTP 200 και η καταγραφή Imitating SMS send to +306900111222: OTP=945940 στα αρχεία καταγραφής του backend container παρέχει τον εξαψήφιο κωδικό. Δευτερεύον αίτημα POST /phone/verification/confirm με τον κωδικό επιστρέφει phoneJwt. Παράλληλο αίτημα POST /customer/email/verification επιστρέφει emailJwt μέσω της ίδιας οδού καταγραφής. Το τελικό αίτημα POST /customer/register με και τα δύο διακριτικά δημιουργεί τη γραμμή στο users με συμπληρωμένα τα πεδία phone και phone_verified_at.

Στη συνέχεια η ίδια ροή εκτελείται μέσω του προσομοιωτή. Η SignupScreen συγκεντρώνει τα στοιχεία, η OTPScreen δέχεται τον κωδικό email και η PhoneOtpScreen τον κωδικό κινητού. Σε επιτυχία, ο πλοηγός επιστρέφει στην LoginScreen με μήνυμα Account created.

6.5.2 Σενάριο αγοράς συνδρομής

Συνδεδεμένος χρήστης ζητά GET /subscription/current: η απάντηση επιστρέφει null (δεν υπάρχει συνδρομή). Αίτημα POST /subscription/checkout με "cardNumber":"4242424242424242", "plan":"PREMIUM", "autoRenew":true επιστρέφει HTTP 200 με ανανεωμένο token. Νέο GET /subscription/current επιστρέφει "plan":"PREMIUM", "status":"ACTIVE" και "daysRemaining":30.

Αρνητικό σενάριο: αίτημα με "cardNumber":"4000000000000002" επιστρέφει HTTP 402 και σώμα {"error":"Card declined"}. Η αντίστοιχη γραμμή στο payment_transaction καταχωρεί status='DECLINED' χωρίς αναφορά σε subscription. Η εφαρμογή μεταφράζει τον κωδικό σε Payment declined. Try a different card.

Σενάριο λήξης: για επιδεικτικούς σκοπούς, αλλαγή του expires_at σε χθεσινή ημερομηνία και εκκίνηση του SubscriptionExpiryScheduler (χειροκίνητα μέσω docker exec backend java -cp app.jar ...) ή αναμονή της 01:00 σημαίνει την συνδρομή ως EXPIRED και επαναφέρει τον ρόλο σε USER. Νέο GET /subscription/current επιστρέφει τη συνδρομή με κατάσταση EXPIRED και daysRemaining: 0.

6.5.3 Σενάριο προτιμήσεων και ειδοποίησης

Αίτημα PUT /notifications/preferences με σώμα {"channels":["PUSH","EMAIL"], "eventPriceDrop":true, "eventPriceRise":true, "eventDailySummary":true, "eventWeeklyDigest":false, "thresholdPercent":3} επικυρώνεται με επόμενο GET. Στον πίνακα user_notification_preferences η αντίστοιχη γραμμή ενημερώνεται.

Δοκιμαστική ειδοποίηση εκπέμπεται με POST /notifications/test. Η καταγραφή Imitating push send: tokens=1 title='Test notification' body='If you can read this, push delivery works.'

επιβεβαιώνει ότι ο διακομιστής βρίσκει διακριτικό και διασπείρει την ειδοποίηση. Σε παραγωγή, με ενεργό firebase.enabled=true, η ίδια κλήση παράγει ειδοποίηση FCM στη συσκευή.

Αρνητικό σενάριο: αλλαγή event_price_drop σε false και χειροκίνητη εισαγωγή γραμμής στο prices που υποδηλώνει μείωση. Ο PriceChangeDetector ανιχνεύει τη μεταβολή, αλλά ο NotificationDispatcher βλέπει την απενεργοποιημένη προτίμηση και ακυρώνει την παράδοση. Η συσκευή δεν λαμβάνει ειδοποίηση.

6.5.4 Σενάριο αναγγελίας FCM διακριτικού

Αίτημα POST /notifications/devices με σώμα {"token":"synthetic-abc", "deviceId":"emulator-5554", "platform":"ANDROID"} επιστρέφει HTTP 200 και δημιουργεί γραμμή στο fcm_token. Επανάληψη του ίδιου αιτήματος ενημερώνει last_seen_at χωρίς δημιουργία διπλοτύπου, χάρη στον περιορισμό μοναδικότητας στο πεδίο token. Αίτημα DELETE /notifications/devices/synthetic-abc αφαιρεί τη γραμμή.

Στο επίπεδο της εφαρμογής, η εκκίνηση μετά από επιτυχή σύνδεση καταγράφει στα αρχεία καταγραφής Android Token registered (status=200), εμφανίζοντας ότι ο FcmTokenManager συντόνισε με τον διακομιστή. Σε παραγωγή, η ίδια ροή αντικαθιστά το συνθετικό διακριτικό με το πραγματικό από FirebaseMessaging.getInstance().token.

6.5.5 Σενάριο αλληλεπίδρασης με υπάρχοντα χαρακτηριστικά

Τα νέα χαρακτηριστικά δεν διαταράσσουν τα υπάρχοντα. Συνδεδεμένος χρήστης με role='PREMIUM' μπορεί να δημιουργήσει νέες λίστες παρακολούθησης (POST /monitor_list), να προσθέσει δωμάτια και να εμφανίσει το γράφημα Vico. Η οθόνη λεπτομερειών εμφανίζει τα στατιστικά Min/Avg/ Max χωρίς αλλαγή. Η αναζήτηση ξενοδοχείου μέσω POST /hotel/info εξακολουθεί να εκτελείται κανονικά, ζητώντας από τον scraper container τα δεδομένα δωματίων.

6.6 Σύνοψη

Τα σενάρια που σχετίζονται με την εφαρμογή Android και τη χρήση του API ικανοποιήθηκαν υπό τις προϋποθέσεις διαθέσιμου διακομιστή και έγκυρων δεδομένων. Η πλήρης αυτοματοποίηση της συλλογής τιμών από εξωτερικούς ιστότοπους παραμένει εξαρτημένη από υποδομή scraper που ξεφεύγει από το κύριο αντικείμενο του παρόντος κειμένου.

Κεφάλαιο 7ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

7.1 Σύνοψη

Περιγράφεται σύστημα παρακολούθησης τιμών ξενοδοχείων με διακομιστή Spring Boot, πελάτη διαδικτυακής εφαρμογής (web) και εφαρμογή Android σε Jetpack Compose. Η ανάλυση εστιάζει στον πελάτη Android και στην τήρηση του συμβολαίου REST. Η πλευρά διακομιστή και ο διαδικτυακός πελάτης αναπτύχθηκαν σε ξεχωριστές συνεργαζόμενες εργασίες των συνεργατών και αντιμετωπίζονται εδώ ως δεδομένα στοιχεία. Ο διακομιστής εμφανίζεται ως σταθερή διεπαφή API, ενώ ο web πελάτης αναγνωρίζεται ως συμπληρωματικός καταναλωτής της ίδιας διεπαφής, χωρίς περαιτέρω ανάλυση. Η τοπική αναπαραγωγή με Docker και η εισαγωγή αντιγράφου βάσης επιτρέπουν επαλήθευση της συμπεριφοράς χωρίς ταύτιση με συγκεκριμένη παραγωγική εγκατάσταση.

Η εφαρμογή εφαρμόζει MVVM, αποθετήρια, Hilt και ασύγχρονες ροές (Martin 2017; Google LLC, n.d.-b), ενώ η πλοήγηση και η διάταξη της διεπαφής ακολουθούν το Material Design 3 (Google LLC, n.d.-d, n.d.-c). Η ταυτοποίηση μέσω JWT ενσωματώνεται στον πελάτη με διαμεσολαβητές (interceptors) (Internet Engineering Task Force (IETF) 2015), και τα διαγράμματα ιστορικού τιμών υλοποιούνται με βιβλιοθήκη συμβατή με το Material Design 3.

7.2 Περιορισμοί

Η συλλογή τιμών από εξωτερικούς ιστότοπους εξαρτάται από τη διαθεσιμότητα περιεχομένου, από μηχανισμούς παρεμπόδισης αυτοματοποιημένων αιτημάτων και από τις ρυθμίσεις του scraper. Σε ακατάλληλες συνθήκες ενεργοποιούνται ελεγχόμενα εναλλακτικά μονοπάτια για τοπική επίδειξη. Οι ειδοποιήσεις βασίζονται σε WorkManager και σε τοπικό κανάλι ειδοποιήσεων, όχι σε απομακρυσμένες ειδοποιήσεις. Η εφαρμογή προϋποθέτει σύνδεση δικτύου για τις περισσότερες λειτουργίες, ενώ η επέκταση προς εκτενή τοπική αποθήκευση με Room για λειτουργία χωρίς σύνδεση δεν εξετάστηκε εδώ. Οι ροές πληρωμής συνδρομής, όπου υποστηρίζονται από τη διεπαφή API, δεν συνδέονται με πραγματικό πάροχο χρεώσεων.

7.3 Μελλοντικές κατευθύνσεις

Πιθανές επεκτάσεις του Android πελάτη περιλαμβάνουν σταθερότερη συλλογή δεδομένων από τον διακομιστή με βελτιωμένο scraper και κανονικοποίηση πεδίων, απομακρυσμένες ειδοποιήσεις μέσω FCM με υποστήριξη στον διακομιστή και ενδιάμεση αποθήκευση για χρήση χωρίς συνεχή σύνδεση. Η προσθήκη νέων πελατών, όπως έκδοσης για iOS, είναι συμβατή με το υφιστάμενο API, δεδομένου ότι ο διαδικτυακός πελάτης και η εφαρμογή Android καταναλώνουν ήδη τα ίδια τελικά σημεία. Οι κατευθύνσεις αυτές συνάδουν με τη βιβλιογραφία για μικροϋπηρεσίες και αυτόνομα όρια υπηρεσιών (Newman 2021).

7.4 Κατακλείδα

Προκύπτει λειτουργική εφαρμογή Android με σύγχρονα εργαλεία πλατφόρμας και σαφή διαχωρισμό επιπέδων. Ενδεχόμενες επεκτάσεις μπορούν να προστεθούν χωρίς να ακυρώνεται η υφιστάμενη διαίρεση μεταξύ πελάτη και διακομιστή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Docker Inc. n.d. “Docker Documentation.” Accessed February 28, 2026. https://docs.docker.com/.

Fielding, Roy Thomas. 2000. “Architectural Styles and the Design of Network-Based Software Architectures.” PhD thesis, University of California, Irvine. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Fowler, Martin. 2002. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Google LLC. n.d.-a. “Android Developers Documentation.” Accessed February 28, 2026. https://developer.android.com/docs.

Google LLC. n.d.-b. “Hilt.” Accessed February 28, 2026. https://developer.android.com/training/dependency-injection/hilt-android.

Google LLC. n.d.-c. “Jetpack Compose Documentation.” Accessed February 28, 2026. https://developer.android.com/jetpack/compose/documentation.

Google LLC. n.d.-d. “Material Design 3.” Accessed February 28, 2026. https://m3.material.io/.

Google LLC. n.d.-e. “WorkManager.” Accessed February 28, 2026. https://developer.android.com/topic/libraries/architecture/workmanager.

Internet Engineering Task Force (IETF). 2012. *The OAuth 2.0 Authorization Framework*. RFC No. 6749. RFC Editor. https://www.rfc-editor.org/rfc/rfc6749.

Internet Engineering Task Force (IETF). 2015. *JSON Web Token (JWT)*. RFC No. 7519. RFC Editor. https://www.rfc-editor.org/rfc/rfc7519.

Martin, Robert C. 2017. *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*. Prentice Hall.

Newman, Sam. 2021. *Building Microservices: Designing Fine-Grained Systems*. 2nd ed. O’Reilly Media.

Square, Inc. n.d.-a. “OkHttp.” Accessed February 28, 2026. https://square.github.io/okhttp/.

Square, Inc. n.d.-b. “Retrofit.” Accessed February 28, 2026. https://square.github.io/retrofit/.

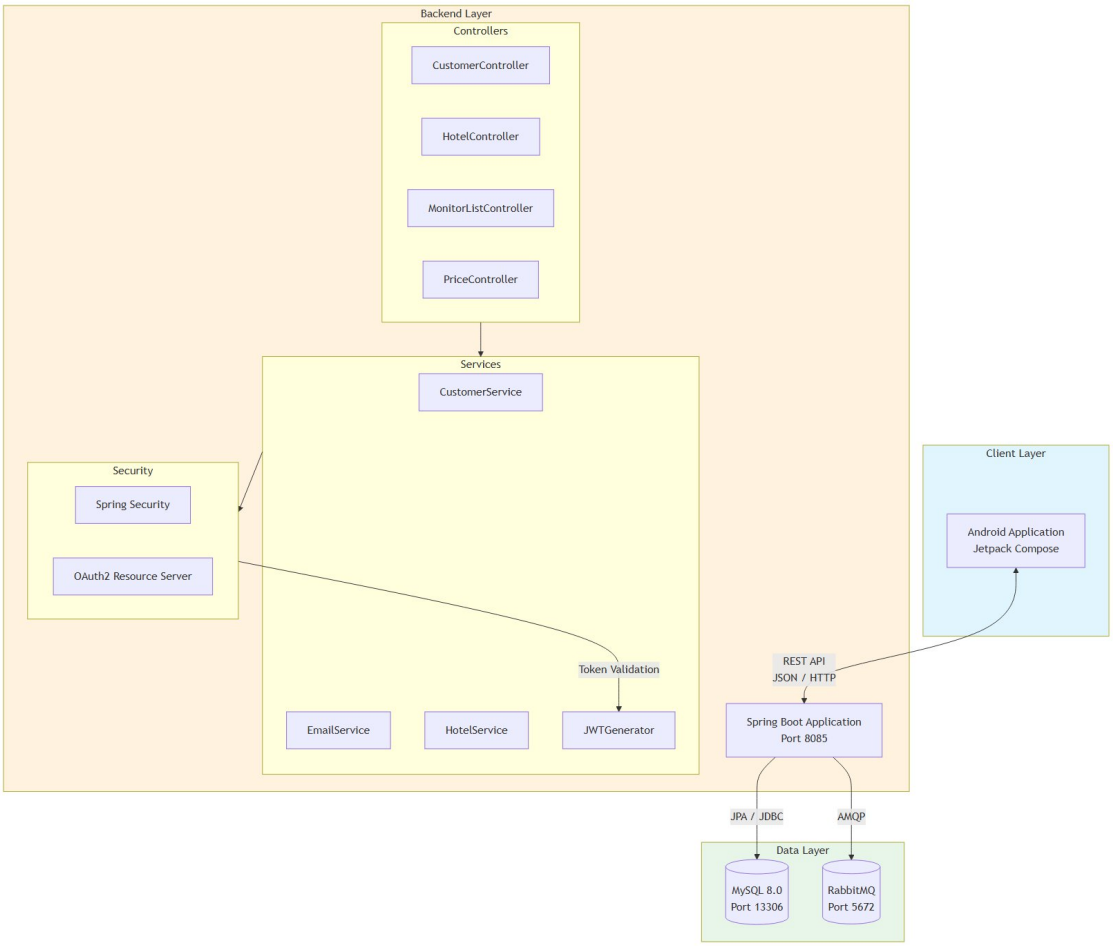
VMware, Inc. n.d.-a. “Spring Boot Reference Documentation.” Accessed February 28, 2026. https://docs.spring.io/spring-boot/docs/current/reference/html/.

VMware, Inc. n.d.-b. “Spring Security Reference.” Accessed February 28, 2026. https://docs.spring.io/spring-security/reference/.

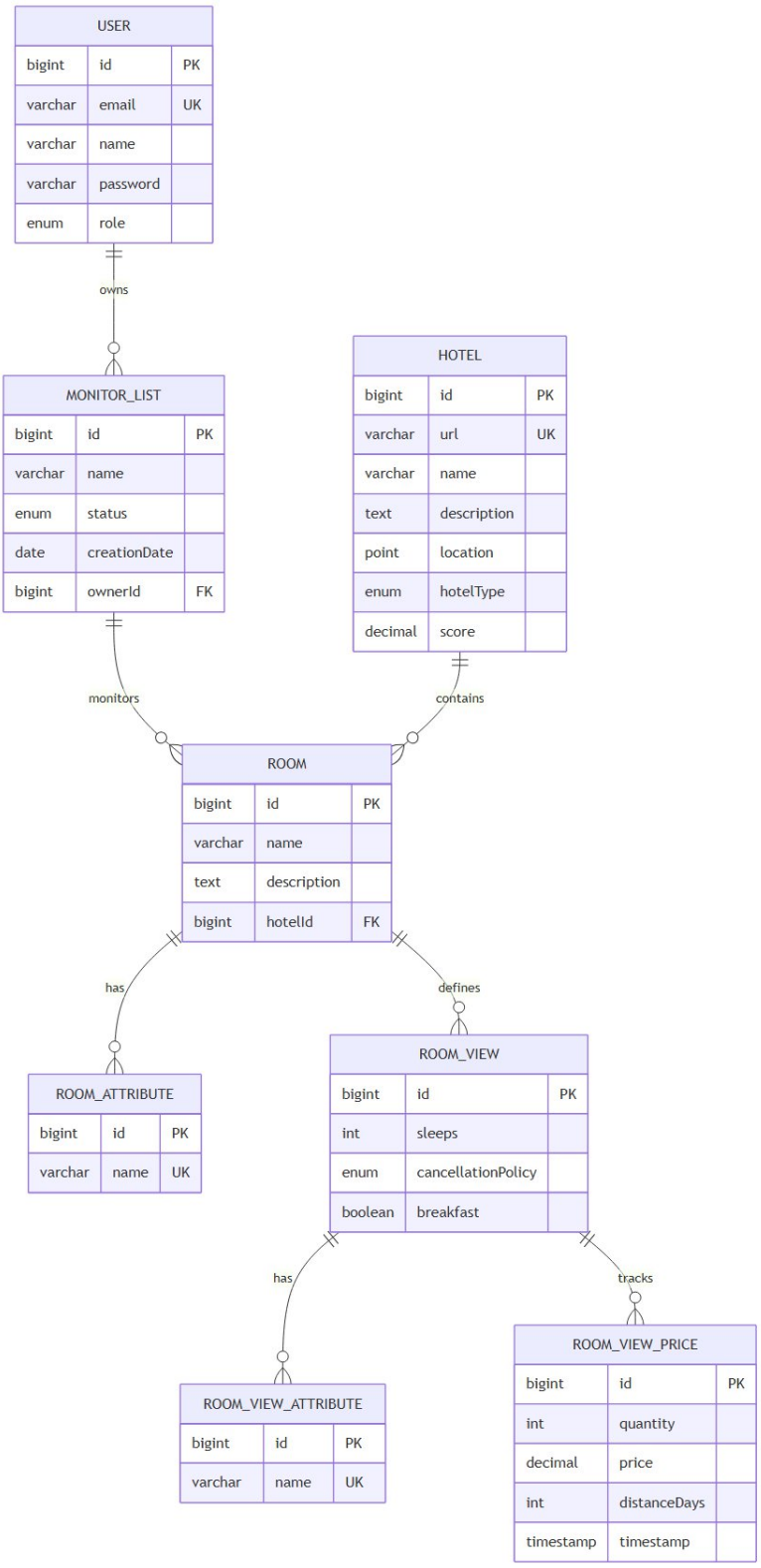
Παράρτημα

Διαγράμματα συστήματος

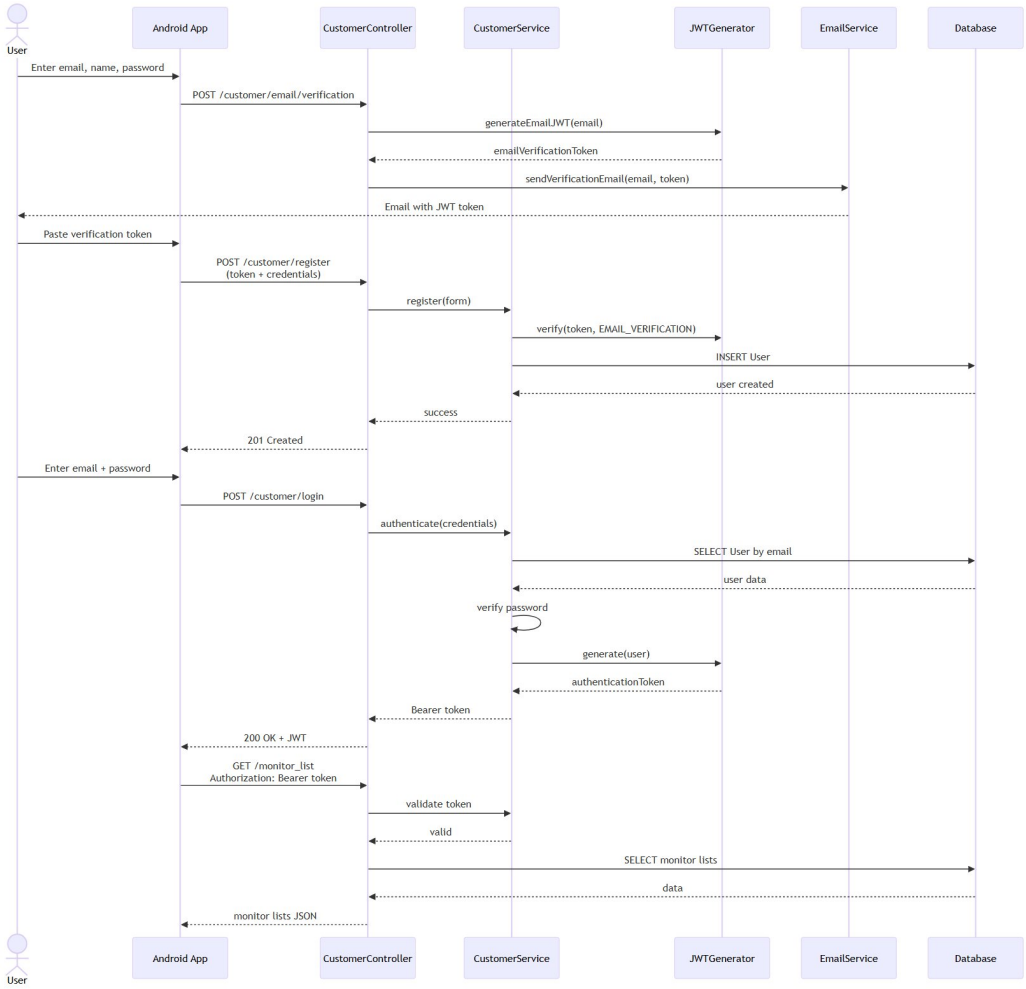
Για συνεπή απόδοση στο PDF ενσωματώνονται ως εικόνες (PNG), με επεξηγηματικές λεζάντες, διαγράμματα που καλύπτουν την αρχιτεκτονική, το σχεσιακό μοντέλο δεδομένων (ERD), τη ροή ταυτοποίησης, την αρχιτεκτονική MVVM, την πλοήγηση, την τοπολογία Docker, τη ροή δεδομένων, την ενσωμάτωση εξαρτήσεων με Hilt και ενδεικτική ροή πελάτη (ViewModel–αποθετήριο–API).



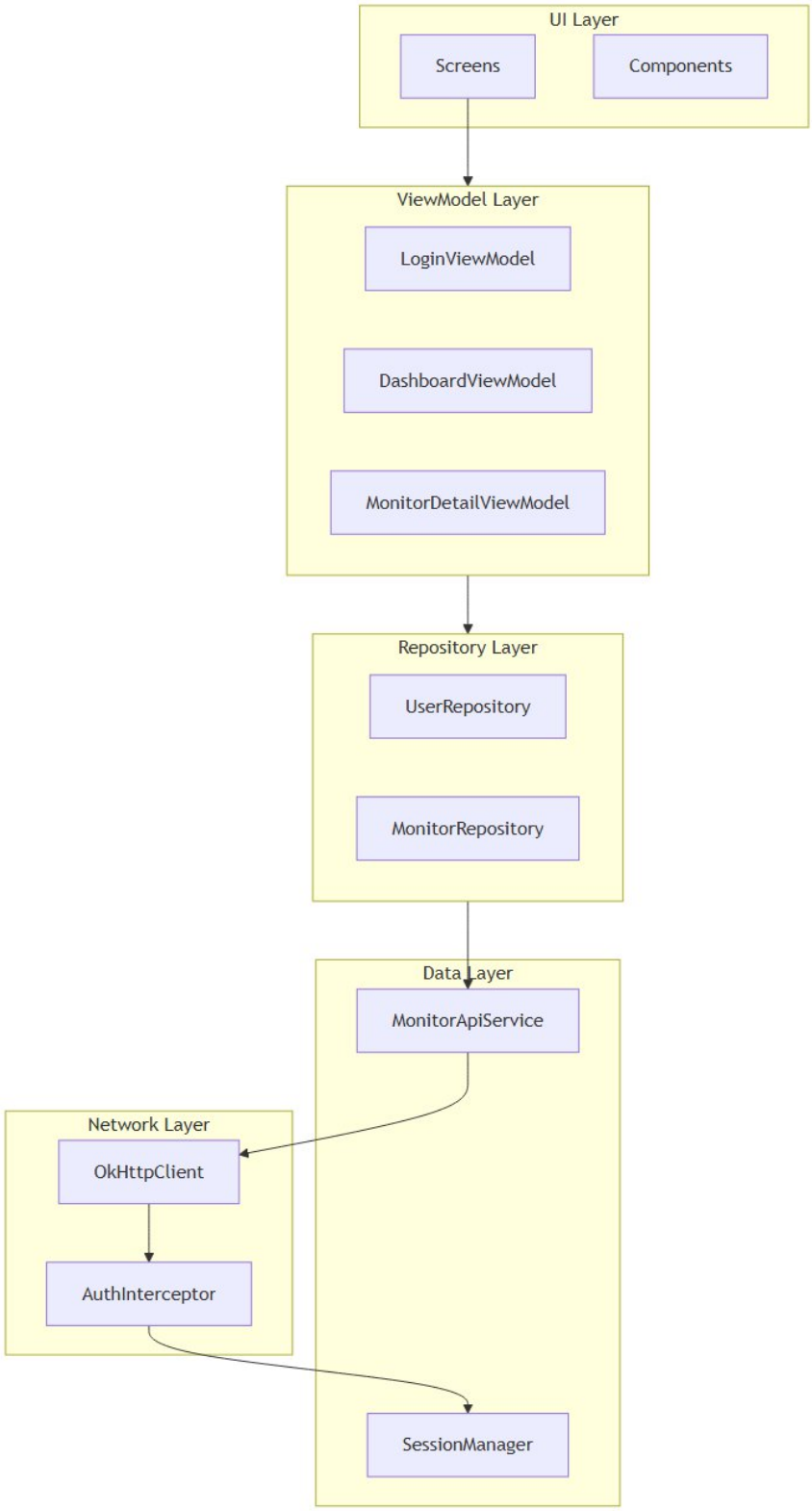
Σχήμα Α.1: *Αρχιτεκτονική συστήματος πελάτη-διακομιστή και βασικά υποσυστήματα.*



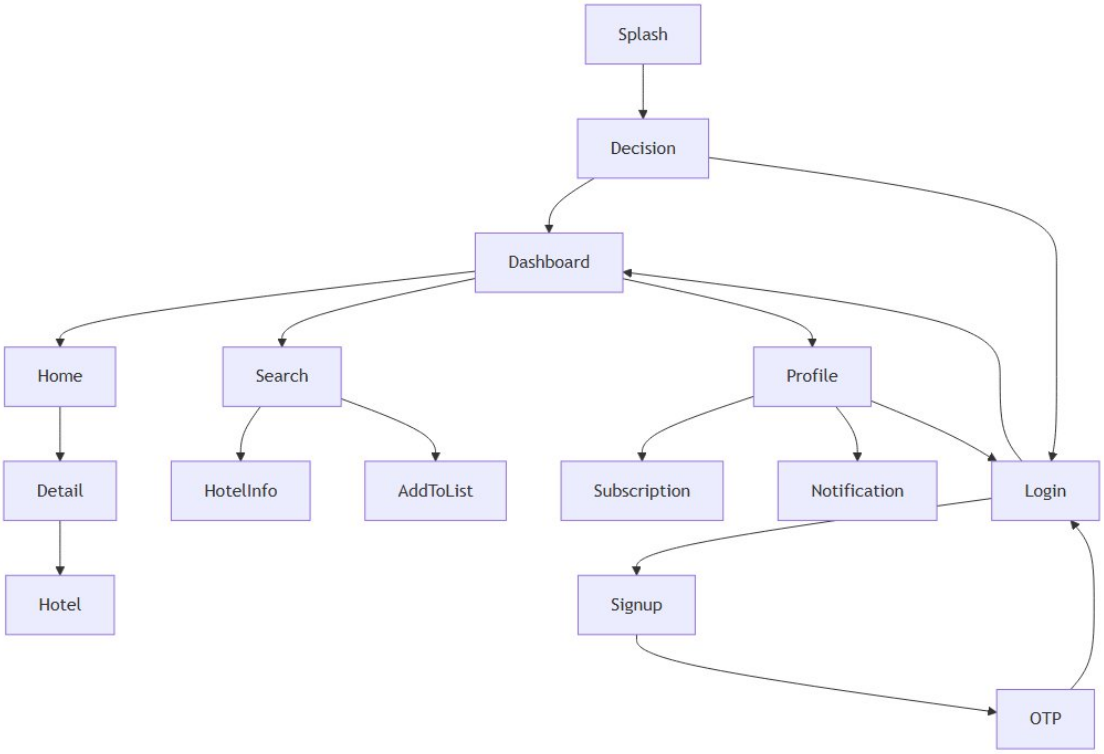
Σχήμα Α.2: *Σχεσιακό μοντέλο δεδομένων (ERD): κύριες οντότητες και συσχετίσεις.*



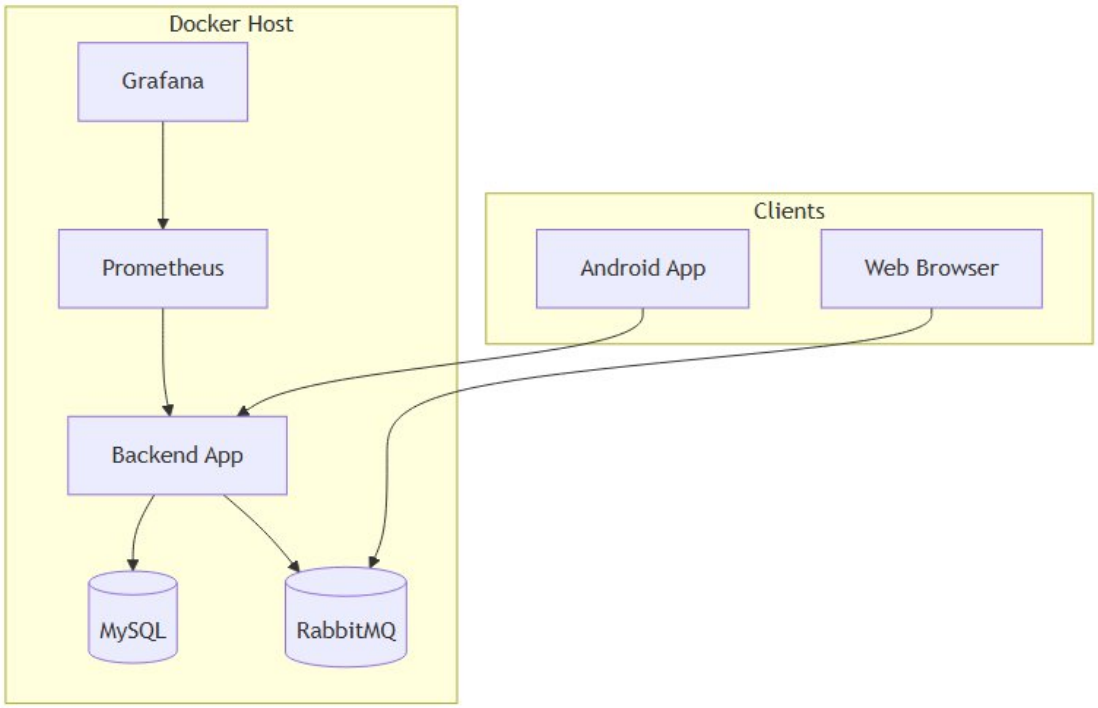
Σχήμα Α.3: *Ροή ταυτοποίησης και εγγραφής (ενδεικτική).*



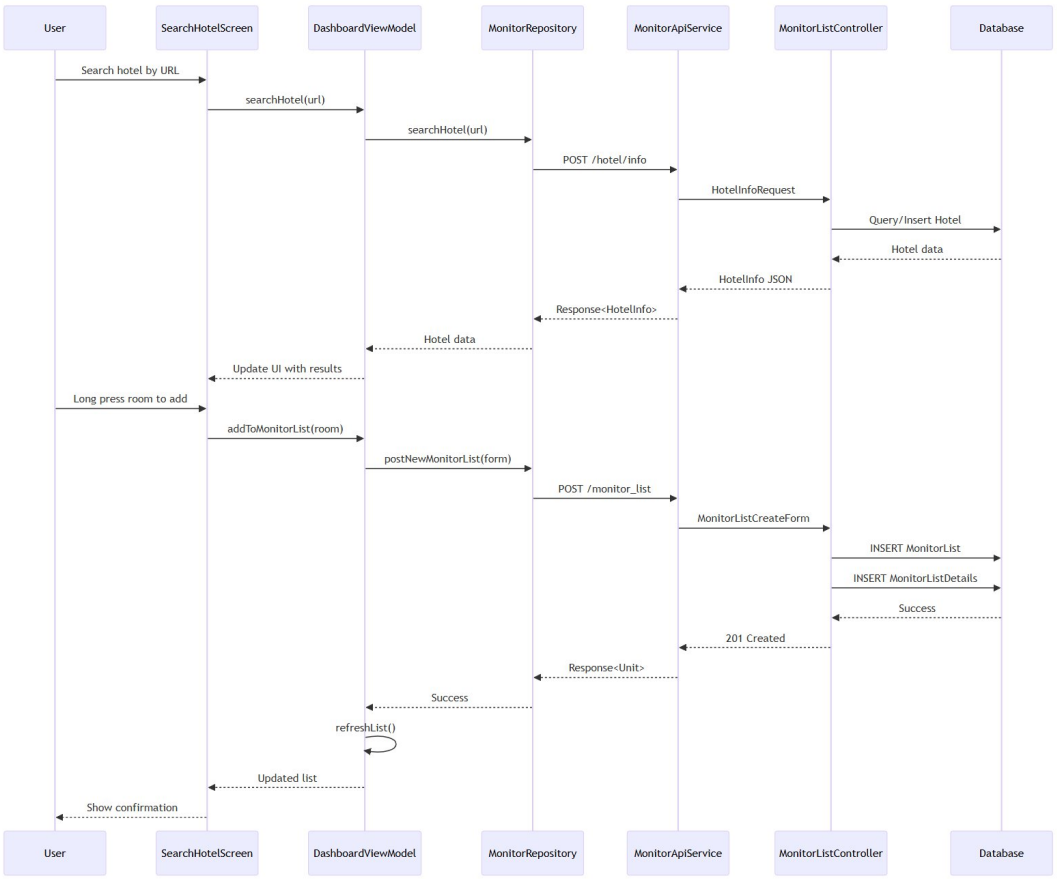
Σχήμα Α.4: *Αρχιτεκτονική Android (MVVM και ροές δεδομένων).*



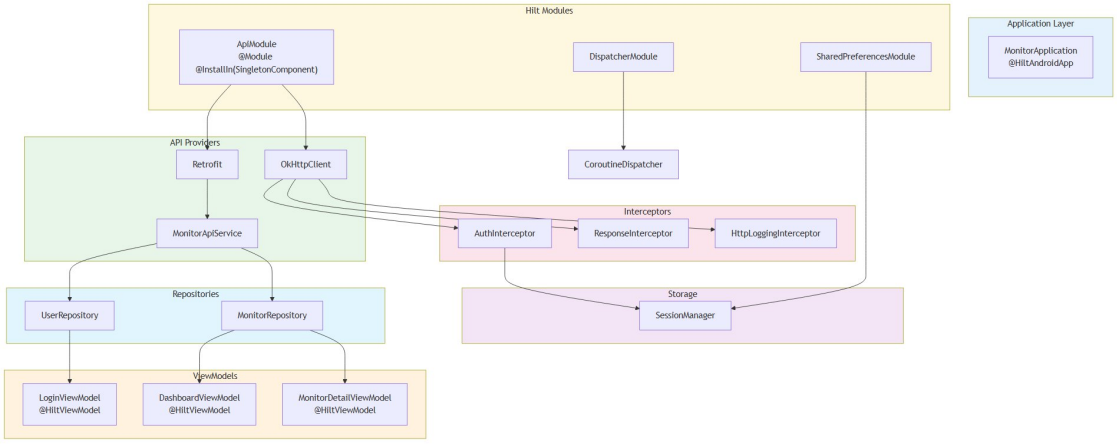
Σχήμα Α.5: *Ροή πλοήγησης (κύριες οθόνες).*



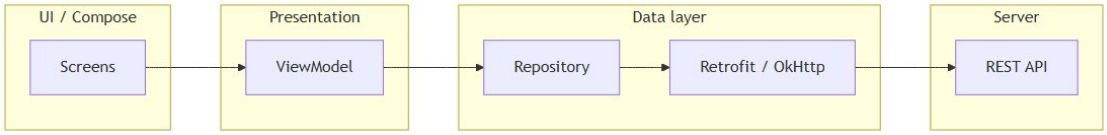
Σχήμα Α.6: *Τοπολογία Docker για τοπική αναπαραγωγή υπηρεσιών (ενδεικτική).*



Σχήμα Α.7: *Ροή δεδομένων μεταξύ διεπαφής, διακομιστή και επίμονης αποθήκευσης.*



Σχήμα Α.8: *Ενσωμάτωση εξαρτήσεων με Hilt και ρόλος των αποθετηρίων.*



Σχήμα Α.9: *Ροή δεδομένων στην εφαρμογή Android (ViewModel → αποθετήριο → Retrofit → API).*

Πρόσβαση σε πηγαίο κώδικα

Ο πηγαίος κώδικας της υπηρεσίας διακομιστή και τα σχετικά αρχεία εκτέλεσης διατίθενται μαζί με το συνοδευτικό υλικό της εργασίας ή σε ξεχωριστό αποθετήριο που παραδίδεται με αυτήν. Ο κώδικας της εφαρμογής Android βρίσκεται στο τοπικό έργο που συνοδεύει την εργασία.