ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη εφαρμογής για την αναζήτηση περιεχομένου σε έντυπα συγγράμματα»

Του φοιτητή: **Σολαχίδη Μιχάλη** **Αρ. Μητρώου: 144386**	**Επιβλέπων:** **Κιοσκερίδης Ιορδάνης** **Βαθμίδα: Καθηγητής**

Ημερομηνία 24/01/2025

Τίτλος Π.Ε. Ανάπτυξη εφαρμογής για την αναζήτηση περιεχομένου σε έντυπα συγγράμματα
Κωδικός Π.Ε. 23358
Ονοματεπώνυμο φοιτητή Σολαχίδης Μιχάλης
Ονοματεπώνυμο εισηγητή Κιοσκερίδης Ιορδάνης
Ημερομηνία ανάληψης Π.Ε. 07/01/2024
Ημερομηνία περάτωσης Π.Ε. 24/01/2025

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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

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

Πρόλογος

Ονομάζομαι Σολαχίδης Μιχάλης και επέλεξα αυτή την πτυχιακή διότι μου κέντρισε το ενδιαφέρον, καθώς την βρίσκω ιδιαίτερα χρήσιμη για κάποιον που διαβάζει βιβλία και χρειάζεται ένα εργαλείο για να βελτιώσει την αναζήτηση εντός του φυσικού βιβλίου.

Ακόμη, πρόκειται για μια αρκετά πολύπλοκη υλοποίηση, καθώς για να ολοκληρώσω την πτυχιακή μου, χρειάστηκε να αποκτήσω υψηλού επιπέδου γνώσεις από android front-end development.

Σκοπός μου ήταν να ολοκληρώσω μια εφαρμογή η οποία να είναι σύγχρονη. Αυτό επιτεύχθηκε με τη χρήση των πιο πρόσφατων τεχνολογιών.

Τελειώνοντας, για την επίτευξη της πτυχιακής μου, θα χρειαστώ μια βάση δεδομένων η οποία θα είναι η Firestore. Πρόκειται για μια NoSQL database, πράγμα που σημαίνει ότι δεν θα χρειαστεί μια άλλη γλώσσα προγραμματισμού προκειμένου να αντλήσω τα δεδομένα που θα χρειαστώ και έτσι θα επικεντρωθώ περισσότερο στο Front-end και την διαχείριση των δεδομένων μου.

Η επιλογή του συγκεκριμένου θέματος θα με βοηθήσει και στην αναζήτηση εργασίας στο μέλλον, καθώς θέλω να ασχοληθώ ενδελεχώς με το front-end Android development.

Περίληψη

Η παρούσα πτυχιακή εργασία έχει σαν στόχο την υλοποίηση μιας android εφαρμογής η οποία θα παρέχει την ανάλογη βοήθεια στον χρήστη ώστε να αναζητά το περιεχόμενο ενός φυσικού βιβλίου με έξυπνο τρόπο.

Αυτό θα επιτευχθεί με απλό τρόπο καθώς πρόκειται για μια εύχρηστη εφαρμογή. Ο χρήστης θα σκανάρει ένα QR code που θα υπάρχει στο βιβλίο και στη συνέχεια θα μπορεί να ξεκινήσει την αναζήτηση που επιθυμεί. Η καλύτερη πλατφόρμα για την υλοποίηση μιας τέτοιας εφαρμογής είναι το κινητό τηλέφωνο, διότι μπορεί να το μεταφέρει ο χρήστης οπουδήποτε. Η αρχιτεκτονική που χρησιμοποιήθηκε στην εφαρμογή αυτή είναι το MVVM και clean architecture.

Επιπρόσθετα, έγινε χρήση της Firestore, βάση δεδομένων της google, η οποία είναι μια NoSQL βάση δεδομένων που κάνει χρήση συλλογών και εγγράφων για να αποθηκεύει τα δεδομένα. Όσον αφορά την εισαγωγή του βιβλίου στην Firestore, γίνεται με τη χρήση ενός python script που εισάγει τα δεδομένα σε μια συλλογή και χωρίζει το περιεχόμενο του βιβλίου σε εγγραφές. Τα δεδομένα αυτά θα πρέπει να έχουν μια συγκεκριμένη μορφή.

Τελειώνοντας, μέσα από την πτυχιακή γίνεται φανερό ότι πρόκειται για μια αρκετά εύχρηστη εφαρμογή αρκεί να γίνει η εισαγωγή των δεδομένων στη σωστή τους μορφή.

Λέξεις-κλειδιά: android εφαρμογή, MVVM, clean architecture, Firestore

«App Development for in content search in printed materials»

«Michalis Solachidis»

Abstract

The aim of this thesis is the implementation of an android application which will provide the corresponding help to the user to search the content of a physical book in a helpful way. This will be achieved in a simple way as it is a useful application. The user will scan a QR code which will be in the book and after this the user will be able to start the search. The best platform for the implementation of this app is a mobile phone because the user could use it everywhere. The architecture that has been used in this app is MVVM and Clean Architecture. In addition, Firestore has been used, which is a NoSQL database that uses collections and documents to store data. As for the import of the book into Firebase, it is done by using a python script that imports the data into a collection and splits the content of the book into records. This data should have a specific format. In conclusion, through this thesis it becomes clear that this is a fairly easy to use the application as long as the data is entered in the correct format.

Keywords: android app, MVVM, clean architecture, Firestore

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους γονείς μου, Δαμιανό και Μαρίνα, την κοπέλα μου, Βάλια για την στήριξη όλα αυτά τα χρόνια. Τον καθηγητή μαθηματικών Παλαιολόγο Παύλο που με βοήθησε να περάσω τις εισαγωγικές εξετάσεις και τον καθηγητή Κιοσκερίδη Ιορδάνη για την εξαιρετική συνεργασία και την ιδέα του θέματος.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

MVVM: Model -View -View Model

QR: quick - response

SQL: structured query language

IDE: Integrated Development Environment

SDK: Software Development Kit

UI: User Interface

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στην εποχή μας, η ανάπτυξη κινητών εφαρμογών για smartphones έχει υποστεί μεγάλη άνοδο, καθώς την υποστηρίζει η ευρεία υιοθέτηση και δημοφιλία μεταξύ των τελικών χρηστών [1]. Αυτή η ανερχόμενη ανάπτυξη έχει στρέψει το ενδιαφερον των εταιρειών παροχης λογισμικου να αναζητουν ολο και περισσοτερους προγραμματιστες κινητων εφαρμογων. Αυτό το αυξημένο ενδιαφέρον έχει κατακτήσει την προσοχή των προγραμματιστών κινητών εφαρμογών τα τελευταία χρόνια [2].

Επί του παρόντος, εκατομμύρια κινητές εφαρμογές κατεβάζονται και χρησιμοποιούνται από χρήστες σε ολόκληρο τον κόσμο. Υπάρχουν Διάφορα καταστήματα εφαρμογών για κινητά, όπως το Google Play Store, το Apple App Store, Windows Phone Store, Huawei AppGallery store, παρέχουν τόσο δωρεάν όσο και επί πληρωμή κινητές εφαρμογές στους χρήστες [3].

Υπάρχουν τρείς τρόποι δημιουργιας κινητων εφαρμογων: τοπικές (native), βασισμένες στον ιστό και υβριδικές κινητές εφαρμογές. Εστιάζουμε στη σύγκριση των εργαλείων ανάπτυξης για native και hybrid κινητές εφαρμογές, αποτυπώνοντας τα χαρακτηριστικά, τα πλεονεκτήματα και τους περιορισμούς της κάθε προσέγγισης. Η ανάλυση επικεντρώνεται στην απόδοση, την εμπειρία χρήστη και την οικονομική αποδοτικότητα κατά τη διαδικασία ανάπτυξης και συντήρησης των εφαρμογών[4].

Η ανάπτυξη "τοπικών" εφαρμογών δινει την δυνατοτητα στους προγραμματιστές να αξιοποιήσουν πλήρως τις δυνατότητες της συσκευής, και ετσι, υπαρχουν καλυτερα αποτελεσματα και μεγαλυτερη αποδοση και ποιοτητα εφαρμογων μεσω τον application stores καθε πλατφορμας. Ωστόσο, για να γίνουν σωστά τοπικές εφαρμογες αυτο απαιτει γνωσεις του προγραμματιστη πανω σε διεπαφές προγραμματισμού εφαρμογών (Application Programming Interface—API) και τα Σετ Εργαλείων Ανάπτυξης (Software Development Kit - SDK). Η δεύτερη προσέγγιση αναφέρεται στις κινητές εφαρμογές ιστού, οι οποίες λειτουργούν σαν διακομιστές ιστού και είναι προσβάσιμες μέσω κινητών περιηγητών. Αυτές οι εφαρμογές προσφέρουν μεγάλη φορητότητα σε διάφορες κινητές πλατφόρμες και μειώνουν το κόστος και τον χρόνο ανάπτυξης, καθώς επιτρέπουν τη διασυνοριακή χρήση. Η τρίτη προσέγγιση αναφέρεται στις υβριδικές κινητές εφαρμογές, οι οποίες εγκαθίστανται στις συσκευές και λειτουργούν ως "τοπικές" κινητές εφαρμογές ιστού, ενσωματώνοντας τον έλεγχο περιηγητή της πλατφόρμας. Αυτές οι εφαρμογές επιτρέπουν πρόσβαση σε χαρακτηριστικά υλικού της συσκευής και διατίθενται για λήψη μέσω του αντίστοιχου καταστήματος διανομής εφαρμογών της πλατφόρμας.

Η ανάπτυξη κινητών εφαρμογών έχει διαδραματίσει καθοριστικό ρόλο στην εξέλιξη του οικοσυστήματος του Android. Η πορεία του Android συνδέεται στενά με την εξέλιξη της ανάπτυξης κινητών εφαρμογών. Αυτό αποτελεί αντικατοπτριστικο του πώς η ευρεία διάδοση και δημοφιλία των smartphones οδήγησε στην καινοτομία και στις μεθόδους ανάπτυξης εφαρμογών. Εκατομμύρια κινητές εφαρμογές, ανεξαρτήτως μεθόδου ανάπτυξής, διατίθενται στο Play Store και αυτό καταδεικνύει την ευρεία αποδοχή και χρήση τους από τους χρήστες παγκοσμίως.

1.2 Android και κινητα

Τα κινητά τηλέφωνα είναι πλέον απαραίτητα εργαλεία στην καθημερινή ζωή, βαθιά ενσωματωμένα στην παγκόσμια επικοινωνία, τα οικονομικά συστήματα και τις κοινωνικές δομές. Ο πολλαπλασιασμός τους αντικατοπτρίζει την ταχεία εξέλιξη της τεχνολογίας, με τα smartphones να αποτελούν την πλειοψηφία των κινητών συσκευών. Το 2023, πάνω από το 83% των ενηλίκων στις ανεπτυγμένες οικονομίες κατείχαν smartphones, ενώ η υιοθέτηση συνεχίζει να αυξάνεται στις αναπτυσσόμενες χώρες, λόγω της διευρυνόμενης προσιτής τιμής τους και της ουσιαστικής λειτουργικότητας τους. Η κυριαρχία του Android και του iOS στην αγορά λειτουργικών συστημάτων αναδεικνύει τον δυοπωλιακό χαρακτήρα της τεχνολογίας αυτής. Το Android κατέχει σημαντικό μερίδιο παγκοσμίως λόγω της αρχιτεκτονικής ανοικτού κώδικα και της προσβασιμότητας του σε ποικίλο υλικό, εξυπηρετώντας ένα ευρύ δημογραφικό φάσμα. Αντίθετα, το iOS καταλαμβάνει ένα μικρότερο αλλά σταθερό μερίδιο, δίνοντας έμφαση στην ασφάλεια, την προστασία της ιδιωτικής ζωής και την εμπειρία του χρήστη, χαρακτηριστικά που συχνά προτιμώνται σε οικονομικά εύπορες περιοχές. Ενώ η υιοθέτηση του Android είναι αξιοσημείωτη στις αναδυόμενες αγορές, όπου οι οικονομικοί περιορισμοί καθιστούν την οικονομική προσιτότητα ζωτικής σημασίας, το iOS ευδοκιμεί στις ανεπτυγμένες οικονομίες, όπου οι πελάτες δίνουν προτεραιότητα στην ενσωμάτωση της συσκευής και στις επιδόσεις[5].

Ο κοινωνικός και οικονομικός αντίκτυπος των smartphones υπερβαίνει την προσωπική επικοινωνία. Χρησιμεύουν ως πλατφόρμες για την εκπαίδευση, την πρόσβαση στην υγειονομική περίθαλψη και την οικονομική ένταξη, ιδίως σε περιοχές όπου οι υποδομές είναι περιορισμένες. Οι έρευνες αναδεικνύουν επίσης τον ρόλο τους στην αναδιαμόρφωση της καταναλωτικής συμπεριφοράς, με τις εφαρμογές να επιτρέπουν την άμεση πρόσβαση σε αγαθά και υπηρεσίες, αυξάνοντας την εξάρτηση από την κινητή τεχνολογία. Αυτή η ευρέως διαδεδομένη εξάρτηση υπογραμμίζει τη διττή φύση των smartphones: ενώ εκδημοκρατίζουν την πρόσβαση στις πληροφορίες, συμβάλλουν επίσης σε προκλήσεις όπως το ψηφιακό χάσμα και οι ανησυχίες για την προστασία της ιδιωτικής ζωής[6]. Η επιστημονική ανάλυση της χρήσης των κινητών τηλεφώνων αποκαλύπτει πρότυπα που επηρεάζονται από κοινωνικοοικονομικούς και πολιτιστικούς παράγοντες, αντανακλώντας ανισότητες στην πρόσβαση και τη χρήση της τεχνολογίας. Μελέτες με κριτές, όπως αυτές στο PLOS ONE και στο IEEE Xplore, συζητούν εκτενώς αυτές τις τάσεις, προσφέροντας πληροφορίες σχετικά με τις συμπεριφορικές και οικονομικές επιπτώσεις της ανάπτυξης της κινητής τεχνολογίας.

1.3 Ο ρόλος της ανάγνωσης βιβλίων

Τα βιβλία αποτελούν εδώ και πολύ καιρό ακρογωνιαίο λίθο του ανθρώπινου πολιτισμού, χρησιμεύοντας ως δοχεία γνώσης, ψυχαγωγίας και αυτογνωσίας. Στη σημερινή ταχέως εξελισσόμενη ψηφιακή εποχή, η ανάγνωση βιβλίων παραμένει μια ζωτική δραστηριότητα που επηρεάζει διάφορες πτυχές της καθημερινής ζωής, προσφέροντας βαθιά οφέλη και θέτοντας μοναδικές προκλήσεις.

Οι λόγοι για τους οποίους οι άνθρωποι διαβάζουν βιβλία είναι τόσο διαφορετικοί όσο και το περιεχόμενό τους. Για πολλούς, η ανάγνωση παρέχει μια μορφή απόδρασης, μεταφέροντάς τους σε διαφορετικούς κόσμους, εποχές και εμπειρίες. Η μυθοπλασία, για παράδειγμα, ενισχύει την ενσυναίσθηση, επιτρέποντας στους αναγνώστες να κατανοήσουν προοπτικές πολύ διαφορετικές από τις δικές τους. Τα μη μυθοπλαστικά βιβλία, από την άλλη πλευρά, χρησιμεύουν ως εργαλεία για προσωπική και επαγγελματική ανάπτυξη, προσφέροντας γνώσεις σε τομείς που κυμαίνονται από την επιστήμη και την ιστορία μέχρι την αυτοβοήθεια και τη διαχείριση. Τα βιβλία διαδραματίζουν επίσης καθοριστικό ρόλο στην εκπαίδευση και την πνευματική ανάπτυξη. Ενισχύουν την κριτική σκέψη παρουσιάζοντας σύνθετες αφηγήσεις και επιχειρήματα που ενθαρρύνουν τον προβληματισμό και την ανάλυση.

Για τον ελεύθερο χρόνο, η ανάγνωση μπορεί να αποτελέσει μια χαλαρωτική δραστηριότητα, μειώνοντας το άγχος και προάγοντας την ψυχική ευεξία. Μια μελέτη που δημοσιεύθηκε στο περιοδικό Social Science & Medicine δείχνει ότι τα άτομα που διαβάζουν τακτικά έχουν χαμηλότερα επίπεδα στρες και αυξημένη αίσθηση ικανοποίησης από τη ζωή.

Οφέλη της ανάγνωσης βιβλίων

- Γνωστική ενίσχυση: Η ανάγνωση βελτιώνει το λεξιλόγιο, την κατανόηση και τη μνήμη. Μελέτες δείχνουν ότι οι τακτικοί αναγνώστες τείνουν να έχουν καλύτερες επιδόσεις σε γνωστικά τεστ, ανεξαρτήτως ηλικίας.
- Συναισθηματική ανάπτυξη: Εκθέτοντας τους αναγνώστες σε διαφορετικές εμπειρίες και χαρακτήρες, τα βιβλία ενισχύουν τη συναισθηματική νοημοσύνη και την ενσυναίσθηση.
- Μείωση του άγχους: Η εμβάθυνση σε ένα βιβλίο έχει αποδειχθεί ότι μειώνει τα επίπεδα κορτιζόλης, κάτι που μοιάζει με τον διαλογισμό.
- Διανοητική διέγερση: Το διάβασμα κρατά τον εγκέφαλο ενεργό και απασχολημένο, καθυστερώντας ενδεχομένως τη γνωστική παρακμή στην τρίτη ηλικία.
- Βελτίωση της εστίασης και της συγκέντρωσης: Σε αντίθεση με την αποσπασματική φύση της κατανάλωσης ψηφιακού περιεχομένου, τα βιβλία ενθαρρύνουν τη διαρκή προσοχή και τη βαθιά εστίαση.

Παρά τα οφέλη της, η ανάγνωση βιβλίων αντιμετωπίζει τον ανταγωνισμό των ψηφιακών μέσων, τα οποία προσφέρουν ταχύτερο και πιο διαδραστικό περιεχόμενο. Οι επικριτές υποστηρίζουν ότι η ανάγνωση μεγάλης διάρκειας απαιτεί περισσότερο χρόνο και συγκέντρωση, γεγονός που την καθιστά λιγότερο ελκυστική σε μια εποχή που κυριαρχεί η άμεση ικανοποίηση. Επιπλέον, η υπερβολική ανάγνωση χωρίς ισορροπία μπορεί να οδηγήσει σε κοινωνική απομόνωση ή παραμέληση των σωματικών δραστηριοτήτων.

Τα βιβλία εμπλουτίζουν τη ζωή, αλλά απαιτούν προσεκτική ενασχόληση για να μεγιστοποιήσουν τα δυνητικά οφέλη τους. Η επίτευξη ισορροπίας μεταξύ της ανάγνωσης και άλλων δραστηριοτήτων διασφαλίζει ότι τα άτομα μπορούν να αξιοποιήσουν τα πλεονεκτήματα των βιβλίων, παραμένοντας παράλληλα συντονισμένα με τις απαιτήσεις της σύγχρονης ζωής.

Τα βιβλία, είτε σε έντυπη είτε σε ψηφιακή μορφή, παραμένουν ένα βασικό μέσο για τη γνώση, την έμπνευση και την προσωπική ανάπτυξη. Γεφυρώνουν το χάσμα μεταξύ παρελθόντος και παρόντος, συνδέοντας τους ανθρώπους με ιδέες που ξεπερνούν τον χρόνο.

1.4 Κίνητρο Έρευνας

Η απόφαση για την εκπόνηση αυτής της πτυχιακής καθοδηγήθηκε από έναν συνδυασμό προσωπικών και ακαδημαϊκών φιλοδοξιών. Μεταξύ των πρωταρχικών κινήτρων μου ήταν η επιθυμία να αναλάβω ένα δύσκολο και ουσιαστικό έργο, ένα έργο που θα ξεπερνούσε τα όρια των ικανοτήτων μου, ενώ παράλληλα θα αντιμετώπιζα ένα πρακτικό ζήτημα.

Η ιδέα της δημιουργίας μιας ψηφιακής λύσης για τη βελτίωση του τρόπου με τον οποίο οι άνθρωποι αλληλεπιδρούν με τα βιβλία με ενθουσίασε βαθιά, καθώς συνδύαζε το πάθος μου για την τεχνολογία με την εμπειρία μου ως φοιτητής που ασχολείται συχνά με ακαδημαϊκά κείμενα.

Μία από τις επαναλαμβανόμενες απογοητεύσεις που αντιμετώπιζα κατά τη διάρκεια των σπουδών μου ήταν η δυσκολία αποτελεσματικής πλοήγησης σε φυσικά και ψηφιακά βιβλία για τον εντοπισμό συγκεκριμένων πληροφοριών. Τα ακαδημαϊκά κείμενα είναι συχνά πυκνά και η χειροκίνητη αναζήτηση σε σελίδες για την εύρεση μιας βασικής έννοιας ή ενός αποσπάσματος μπορεί να είναι χρονοβόρα και να διαταράσσει τη μαθησιακή διαδικασία. Αυτή η πρόκληση με ενέπνευσε να αναπτύξω ένα σύστημα που θα απλοποιούσε τη διαδικασία, κάνοντας τα βιβλία πιο προσιτά και διαδραστικά. Η ιδέα να δοθεί η δυνατότητα στους χρήστες να πραγματοποιούν γρήγορες αναζητήσεις μέσα στα κείμενα ήταν μια πρακτική λύση σε ένα πρόβλημα που είχα αντιμετωπίσει προσωπικά και ταίριαζε απόλυτα με τις απαιτήσεις της πτυχιακής. Είδα επίσης αυτή τη πτυχιακή ως μια ευκαιρία να δημιουργήσω ένα έργο που να εξισορροπεί την απλότητα και τη χρησιμότητα.

Στόχος μου ήταν να σχεδιάσω ένα σύστημα που, ενώ θα ήταν τεχνολογικά εξελιγμένο, αλλά να είναι φιλικό προς τον χρήστη. Αυτή η φιλοδοξία απαιτούσε προσεκτικό σχεδιασμό και εκτέλεση, καθώς απαιτούσε βαθιά κατανόηση των αρχών ανάπτυξης λογισμικού, του σχεδιασμού διεπαφής χρήστη (UI) και της διαχείρισης βάσεων δεδομένων. Η πρόκληση της εφαρμογής τεχνολογιών όπως το Kotlin, το Jetpack Compose και το Firestore με παρακίνησε περαιτέρω, καθώς η γνώση αυτών των εργαλείων θα ενίσχυε σημαντικά την τεχνική μου εξειδίκευση και θα με προετοίμαζε για το μέλλον.

Τέλος, αυτή η πτυχιακή εργασία αποτελούσε μια ευκαιρία να συνεισφέρω ουσιαστικά στον τομέα της ψηφιακής μάθησης και της ανάκτησης πληροφοριών. Τα βιβλία είναι διαχρονικοί πόροι για την εκπαίδευση και την προσωπική ανάπτυξη και η βελτίωση του τρόπου με τον οποίο οι άνθρωποι αλληλεπιδρούν μαζί τους μπορεί να έχει διαρκή αντίκτυπο. Γεφυρώνοντας το χάσμα μεταξύ των παραδοσιακών μεθόδων ανάγνωσης και της σύγχρονης τεχνολογίας, ελπίζω να δημιουργήσω ένα εργαλείο όχι μόνο καινοτόμο αλλά και ευρέως ωφέλιμο για τους άλλους.

Αυτός ο συνδυασμός προσωπικών εμπειριών, διανοητικής περιέργειας και επιθυμίας για την ανάπτυξη πρακτικών λύσεων τροφοδότησε τον ενθουσιασμό μου για αυτή τη διατριβή, καθιστώντας την μια ανταμείβοντας και εμπλουτίζοντας την προσπάθεια.

1.5 Δομή Πτυχιακής

Στο 2ο κεφάλαιο πραγματοποιείται έρευνα σχετικά με τις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση αυτής της εφαρμογής, κάνοντας μια μελέτη στις κατευθυντήριες γραμμές προκειμένου να δημιουργηθεί σωστά η εφαρμογή. Αυτό σημαίνει πως δόθηκε μεγάλη σημασία στις οδηγίες που αφορούν την εμφάνιση μιας εφαρμογής σύμφωνα με το Material Design και τη λειτουργικότητα μιας εφαρμογής.

Στο 3ο κεφάλαιο ακολουθεί η βάση δεδομένων της εφαρμογής που βασίστηκε στη Firebase - Firestore και Firebase Storage, πώς αυτή δημιουργήθηκε για να μπορέσει να στηρίξει τη δημιουργία της εφαρμογής.

Στο 4ο κεφάλαιο γίνεται ανάλυση του κομματιού του UI της εφαρμογής καθώς και παραδείγματα από κώδικα, διάγραμμα ροής, τη λειτουργικότητα, για το flow της, και τα βήματα που χρειάζεται για να λειτουργήσει ορθά.

Τέλος, είναι το 5ο κεφάλαιο, που είναι τα συμπεράσματα και οι μελλοντικές βελτιώσεις της εφαρμογής.

1.6 Επίλογος

Σε αυτό το κεφάλαιο έγινε μια εισαγωγή στο θέμα της πτυχιακής, τις τεχνολογίες, βάση την Firebase - Firestore ως back-end, που ειναι μια NoSQL βάση δεδομένων. Η επιλογή αυτής της τεχνολογίας επέτρεψε τη δυναμική και αποτελεσματική διαχείριση δεδομένων. Ταυτόχρονα, εργαλεία όπως η Kotlin και το Jetpack Compose ενίσχυσαν τη χρηστικότητα και τη σταθερότητα του συστήματος, δημιουργώντας μια εφαρμογή που ανταποκρίνεται στις απαιτήσεις των χρηστών Βελτιστοποιώντας τη ομαλή λειτουργία της εφαρμογής.

Η ανάλυση των βημάτων υλοποίησης έδειξε τη σημασία του σωστού σχεδιασμού, της προνοητικότητας στις τεχνικές επιλογές και της ευελιξίας στη δομή της εφαρμογής. Κάθε τεχνολογική απόφαση βασίστηκε όχι μόνο στην κάλυψη των παρόντων απαιτήσεων, αλλά και στην εξασφάλιση της μελλοντικής επεκτασιμότητας καθώς είναι γραμμένο με σύγχρονες τεχνικές. Ιδιαίτερα η σύνδεση μεταξύ του front-end και του back-end υλοποιήθηκε με τρόπο που διασφαλίζει τη σταθερότητα του συστήματος και διευκολύνει τη μελλοντική συντήρηση και βελτίωση.

Παράλληλα, το κεφάλαιο ανέδειξε την πολυπλοκότητα της ανάπτυξης μιας εφαρμογής που επιδιώκει να συνδυάσει σύγχρονες δυνατότητες, όπως η χρήση αποθηκευτικών χώρων στο cloud μέσω του Firebase Storage. Αυτές οι δυνατότητες εμπλουτίζουν την εμπειρία του χρήστη, καθιστώντας την εφαρμογή πιο διαδραστική και πολύπλευρη.

Εν κατακλείδι, το κεφάλαιο αυτό αποτελεί την εισαγωγή για την κατανόηση των τεχνολογικών επιλογών και της μεθοδολογίας που εφαρμόστηκαν στην ανάπτυξη της εφαρμογής. Ο σχεδιασμός και η υλοποίηση βασίστηκαν σε πρακτικές που διασφαλίζουν την ανθεκτικότητα, τη λειτουργικότητα και τη δυνατότητα μελλοντικής εξέλιξης. Έτσι, η εφαρμογή όχι μόνο ανταποκρίνεται στις σημερινές απαιτήσεις, αλλά είναι επίσης έτοιμη να προσαρμοστεί σε ένα συνεχώς εξελισσόμενο τεχνολογικό περιβάλλον, προσφέροντας διαρκή αξία στους χρήστες της.

Κεφάλαιο 2ο: Τεχνολογίες Εφαρμογής

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα δούμε τις τεχνολογίες, τις βιβλιοθήκες, τα εργαλεία καθώς και την αρχιτεκτονική που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

2.2 Android

Το Android είναι ένα λειτουργικό σύστημα ανοικτού κώδικα που έχει σχεδιαστεί κυρίως για κινητές συσκευές όπως smartphones και tablets. Το Android παρέχει μια ευέλικτη, προσαρμόσιμη και ισχυρή πλατφόρμα που υποστηρίζει ένα ευρύ φάσμα εφαρμογών και υλικού. Υπό τη διαχείριση της Google, το Android είναι το πιο ευρέως χρησιμοποιούμενο λειτουργικό σύστημα για κινητά παγκοσμίως, τροφοδοτώντας δισεκατομμύρια συσκευές[7].

Το Android αναπτύχθηκε αρχικά από την Android Inc., μια εταιρεία που ιδρύθηκε από τους Andy Rubin, Rich Miner, Nick Sears και Chris White το 2003. Αρχικά είχε ως στόχο τη δημιουργία ενός προηγμένου λειτουργικού συστήματος για ψηφιακές φωτογραφικές μηχανές, αλλά αργότερα η ομάδα μετατόπισε την προσοχή της στα κινητά τηλέφωνα λόγω των αυξανόμενων δυνατοτήτων της αγοράς.

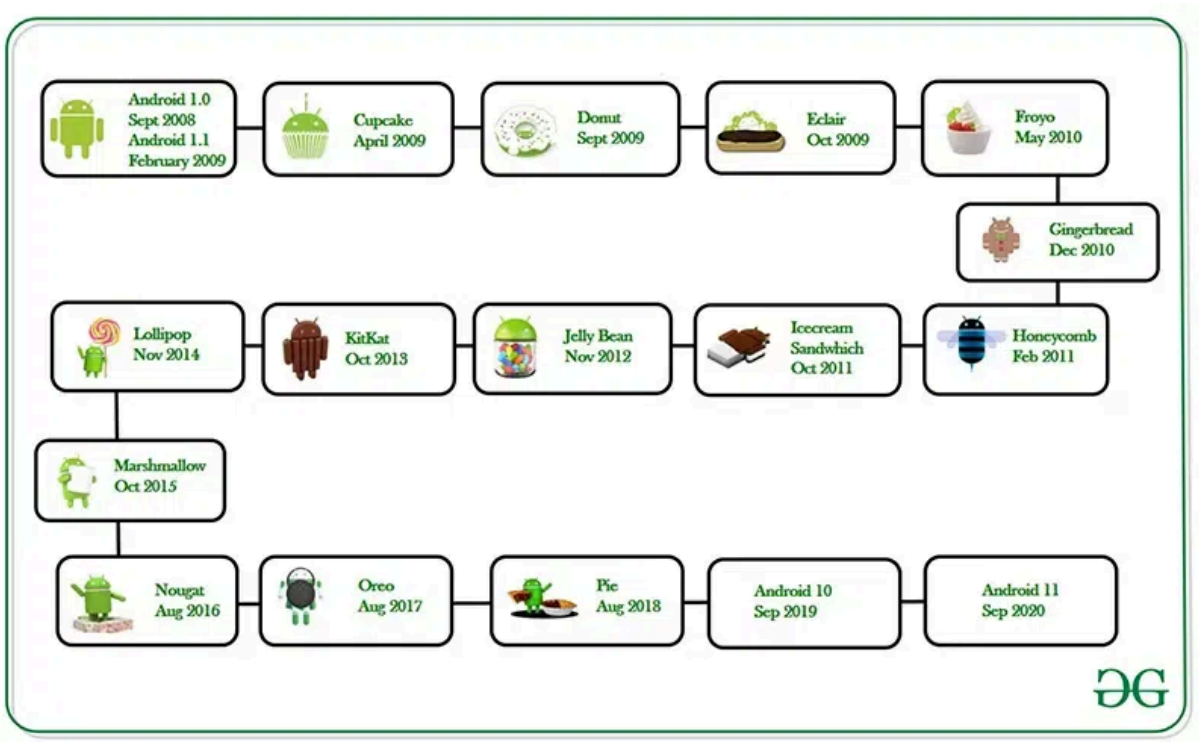
Το 2005, η Google εξαγόρασε την Android Inc. και άρχισε να αναπτύσσει την πλατφόρμα για ευρύτερη εμπορική χρήση. Η πρώτη επίσημη έκδοση, το Android 1.0, κυκλοφόρησε τον Σεπτέμβριο του 2008, με το HTC Dream (T-Mobile G1) να είναι η πρώτη εμπορικά διαθέσιμη συσκευή Android.

Οι πρώτες εκδόσεις του Android προσέφεραν βασικές λειτουργίες, όπως ένα πρόγραμμα περιήγησης στο διαδίκτυο, υποστήριξη κάμερας και πρόσβαση σε εφαρμογές μέσω του Android Market. [8]

Από το ντεμπούτο του, το Android έχει υποστεί συνεχή ανάπτυξη, με σημαντικές ενημερώσεις εκδόσεων που φέρνουν σημαντικές βελτιώσεις στις επιδόσεις, την ασφάλεια και τα χαρακτηριστικά. Βασικά ορόσημα στην εξέλιξη του Android είναι τα εξής:

- Android 2.2 (Froyo): Εισήγαγε τις ειδοποιήσεις push και το USB tethering.
- Android 4.0 (Ice Cream Sandwich): Ενοποίησε το UI των smartphone και των tablet.
- Android 5.0 (Lollipop): Εισήγαγε το Material Design και την υποστήριξη 64-bit.
- Android 10: Σηματοδότησε τη μετάβαση στην πλοήγηση με χειρονομίες και τη σκοτεινή λειτουργία.
- Android 13: Βελτίωσε το απόρρητο, την ασφάλεια και την εξατομίκευση.
- Android 15: Εμφανίστηκε το Σεπτέμβριο του 2024 και είναι η τελευταία σταθερή έκδοση λογισμικού για συσκευές android.

Σήμερα, το Android τροφοδοτεί ένα ποικίλο οικοσύστημα συσκευών, από smartphones και wearables μέχρι τηλεοράσεις και συσκευές IoT, αποδεικνύοντας την επεκτασιμότητα και την ευελιξία του. [9]



Σχήμα 2.1 Android History

2.2.1 Android Studio IDE

Το Android Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για εφαρμογές Android, το οποίο αναπτύχθηκε από την Google. Ανακοινώθηκε για πρώτη φορά στο Google I/O 2013 και αντικατέστησε το Eclipse ως το κύριο IDE για την ανάπτυξη Android εφαρμογών. Βασισμένο στο IntelliJ IDEA της JetBrains, το Android Studio παρέχει εργαλεία ειδικά προσαρμοσμένα για το σχεδιασμό, την κωδικοποίηση, την αποσφαλμάτωση και τη δοκιμή εφαρμογών Android. Υποστηρίζει γλώσσες προγραμματισμού όπως Java, Kotlin και C++, καθιστώντας το ευέλικτο για πολλούς προγραμματιστές.

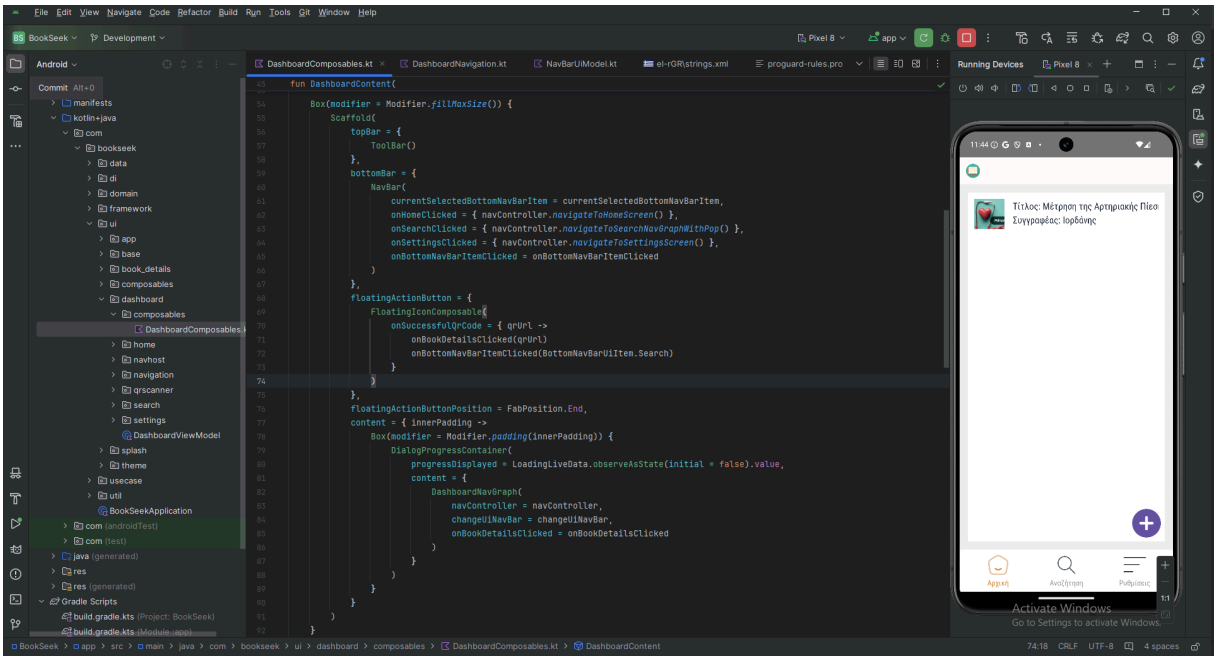
Με την εκτεταμένη γκάμα χαρακτηριστικών του και την ενσωμάτωση με τα Android SDKs και τα εργαλεία, το Android Studio έχει γίνει το κύριο εργαλείο για την ανάπτυξη εφαρμογών Android.

Το Android Studio θεωρείται ευρέως ως το καλύτερο εργαλείο για την ανάπτυξη Android λόγω του:

- Εύκολη χρήση: Η φιλική προς το χρήστη διεπαφή απλοποιεί τη διαδικασία ανάπτυξης, είναι φιλικό στους νέους χρήστες.
- Επίσημη υποστήριξη: Ως το επίσημο IDE, λαμβάνει έγκαιρες ενημερώσεις και υποστηρίζει τα πιο πρόσφατα χαρακτηριστικά και βιβλιοθήκες του Android.
- Επεκτασιμότητα: Μπορούν να προστεθούν πρόσθετα και επεκτάσεις για την προσαρμογή του περιβάλλοντος σε συγκεκριμένες ανάγκες.
- Υποστήριξη πολλαπλών πλατφορμών: Τρέχει σε Windows, macOS και Linux, εξασφαλίζοντας συμβατότητα για τους προγραμματιστές σε όλα τα λειτουργικά συστήματα.

Καποια απο τα βασικά χαρακτηριστικά του Android Studio είναι

- Έξυπνος επεξεργαστής κώδικα: Ενσωματωμένα χαρακτηριστικά όπως η αυτόματη συμπλήρωση, η επισήμανση συντακτικού και τα εργαλεία αναδιαμόρφωσης απλοποιούν την κωδικοποίηση.
- Επεξεργαστής διάταξης: Προσφέρει ένα οπτικό περιβάλλον drag-and-drop για το σχεδιασμό διατάξεων εφαρμογών, συμπεριλαμβανομένης της υποστήριξης των ConstraintLayout και Compose.
- Σύστημα κατασκευής Gradle: Αυτοματοποιεί τη διαδικασία κατασκευής με υποστήριξη για έργα πολλαπλών μονάδων και μεγάλης κλίμακας.
- Εξομοιωτής συσκευών: Επιτρέπει στους προγραμματιστές να δοκιμάζουν εφαρμογές σε εικονικές συσκευές με διάφορα μεγέθη οθόνης και διαμορφώσεις.
- Εργαλεία προφιλοσοφίας: Επιτρέπει την παρακολούθηση της χρήσης της CPU, της μνήμης και του δικτύου για βελτιστοποίηση της απόδοσης.
- Ενσωμάτωση ελέγχου εκδόσεων: Ενσωματώνεται απρόσκοπτα με το Git και άλλα συστήματα ελέγχου εκδόσεων για αποτελεσματική ομαδική συνεργασία.
- Υποστήριξη για Kotlin: Η εγγενής υποστήριξη της γλώσσας Kotlin καθιστά τις σύγχρονες πρακτικές ανάπτυξης ευκολότερες και πιο αποτελεσματικές.



Σχήμα 2.2: Android IDE - Android Studio

2.2.2 Android SDK

Το Android Software Development Kit (SDK) είναι μια συλλογή εργαλείων, βιβλιοθηκών, API και τεκμηρίωσης που επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές για το λειτουργικό σύστημα Android. Λειτουργεί ως η κύρια γέφυρα μεταξύ του κώδικα μιας εφαρμογής και της πλατφόρμας Android, παρέχοντας τους απαραίτητους πόρους για τον αποτελεσματικό σχεδιασμό, τη

δοκιμή και την αποσφαλμάτωση εφαρμογών Android. Το Android SDK διανέμεται από την Google και ενημερώνεται τακτικά για την υποστήριξη νέων χαρακτηριστικών, API και εκδόσεων Android.

Βασικά συστατικά του Android SDK είναι η Βιβλιοθήκες API που παρέχουν προκατασκευασμένες κλάσεις και μεθόδους για την ανάπτυξη Android. Περιλαμβάνει λειτουργικότητα για στοιχεία UI, αποθήκευση δεδομένων, αισθητήρες, δικτύωση και άλλα. Τα Εργαλεία κατασκευής είναι απαραίτητα για τη μεταγλώττιση, τη συσκευασία και την ανάπτυξη εφαρμογών. Περιλαμβάνει εργαλεία όπως το AAPT2 (Android Asset Packaging Tool)[19], το ART (Android runtime) και το DEX (Dalvik executable)[20]. Ένα άλλο χαρακτηριστικό είναι ο Εξομοιωτής Android που είναι μια εικονική συσκευή Android που επιτρέπει στους προγραμματιστές να δοκιμάζουν τις εφαρμογές τους χωρίς να χρειάζονται φυσική συσκευή.

Τα Εργαλεία πλατφόρμας Περιλαμβάνουν βοηθητικά προγράμματα όπως το adb (Android Debug Bridge) για την αποσφαλμάτωση και την εγκατάσταση εφαρμογών και το fastboot για τη διαχείριση της συσκευής.

Τα Εργαλεία προγραμματιστών Περιλαμβάνουν Εργαλεία για τη δημιουργία προφίλ και την παρακολούθηση της απόδοσης των εφαρμογών, όπως το traceview και το hierarchy viewer, η τεκμηρίωση και δείγματα περιλαμβάνουν λεπτομερείς οδηγούς, αναφορές API και έργα-δείγματα για να βοηθήσει τους προγραμματιστές να κατανοήσουν και να εφαρμόσουν αποτελεσματικά τα χαρακτηριστικά.

Το Android SDK απλοποιεί την ανάπτυξη εφαρμογών προσφέροντας έτοιμα προς χρήση εργαλεία και πόρους προσαρμοσμένους στη μοναδική αρχιτεκτονική του Android. Τα βασικά πλεονεκτήματα περιλαμβάνουν:

- Συμβατότητα πλατφόρμας: Εξασφαλίζει την απρόσκοπτη λειτουργία των εφαρμογών σε όλες τις συσκευές και τις εκδόσεις του Android.
- Προσαρμοστικότητα: Επιλογή συγκεκριμένων στοιχείων του SDK για τον εξορθολογισμό της διαδικασίας ανάπτυξης.
- Δοκιμές (testing): Με τον εξομοιωτή και τις βιβλιοθήκες δοκιμών, γίνεται να προσομοιωστούν διάφορα σενάρια και να διασφαλίσουν τη σταθερότητα της εφαρμογής.
- Υποστήριξη πολλαπλών API: Περιλαμβάνει API για υπηρεσίες τοποθεσίας, πρόσβαση στην κάμερα, ειδοποιήσεις και άλλα.
- Συμβατότητα προς τα πίσω: Επιτρέπει στις εφαρμογές να στοχεύουν σε παλαιότερες εκδόσεις Android χρησιμοποιώντας βιβλιοθήκες συμβατότητας.
- Τακτικές ενημερώσεις: Υποστηρίζει τα πιο πρόσφατα χαρακτηριστικά του Android και τις βελτιώσεις ασφαλείας.
- Ενσωμάτωση με το Android Studio: Πλήρως ενσωματωμένο στο Android Studio IDE, απλοποιώντας τη διαδικασία εγκατάστασης και χρήσης.

2.2.3 Android Gradle

Κατά την ανάπτυξη Android εφαρμογών η διαδικασία δημιουργίας παίζει καθοριστικό ρόλο στη μετατροπή του ακατέργαστου κώδικα, των πόρων και των εξαρτήσεων σε ένα λειτουργικό πακέτο εφαρμογής (APK)[21]. Στο επίκεντρο αυτής της διαδικασίας βρίσκεται το Gradle [22], ένα ευέλικτο εργαλείο αυτοματοποίησης κατασκευής που έχει γίνει το πρότυπο για την ανάπτυξη Android. Με τα

ισχυρά χαρακτηριστικά του και την ενσωμάτωσή του στο Android Studio, το Gradle όχι μόνο απλοποιεί την πολυπλοκότητα της σύγχρονης ανάπτυξης εφαρμογών, αλλά εξασφαλίζει επίσης αποτελεσματικότητα και επεκτασιμότητα.

Το Gradle είναι ένα εργαλείο κατασκευής ανοιχτού κώδικα που έχει σχεδιαστεί για να διαχειρίζεται εργασίες αυτοματοποίησης έργων. Εισήχθη στο οικοσύστημα του Android για να παρέχει στους προγραμματιστές ένα ισχυρό και ευέλικτο σύστημα κατασκευής ικανό να διαχειρίζεται μεγάλα και πολύπλοκα έργα. Σε αντίθεση με τα παραδοσιακά εργαλεία, το Gradle χρησιμοποιεί μια Domain-Specific Language (DSL) βασισμένη στην Groovy ή την Kotlin για τον ορισμό σεναρίων κατασκευής. Αυτή η ευελιξία επιτρέπει στους προγραμματιστές να προσαρμόζουν με ευκολία τις διαμορφώσεις κατασκευής τους, ικανοποιώντας ένα ευρύ φάσμα απαιτήσεων εφαρμογών.

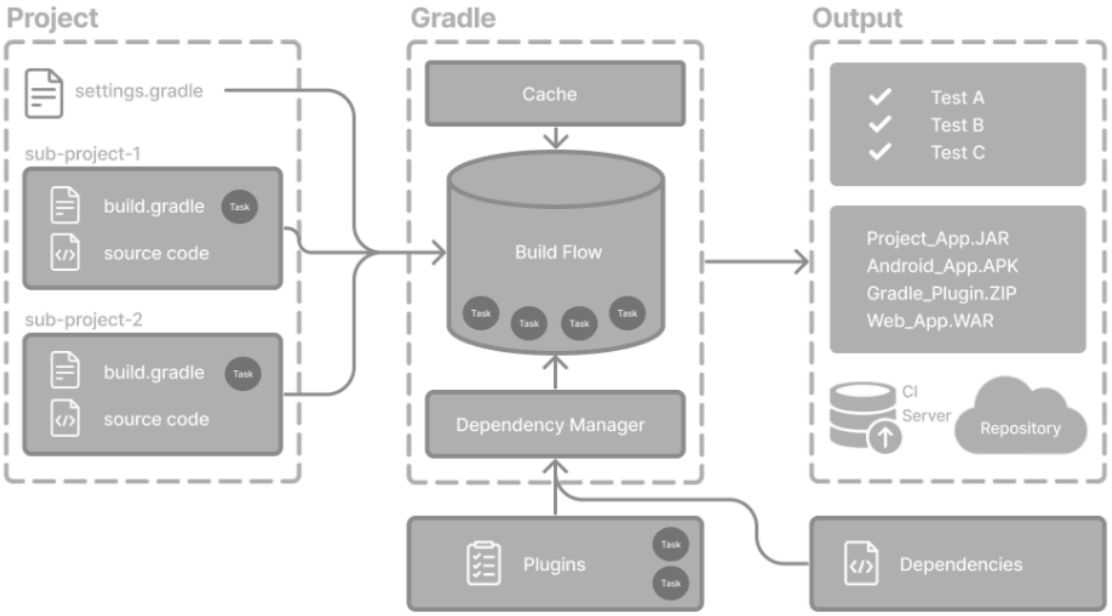
Το Gradle απαρτίζεται απο τα παρακάτω χαρακτηριστικά:

- Διαχείριση εξαρτήσεων: Αυτοματοποιεί τη διαδικασία επίλυσης και ενσωμάτωσης βιβλιοθηκών από αποθετήρια όπως το Maven και το JCenter. Καθορίζοντας εξαρτήσεις στο αρχείο build.gradle [22], έτσι, γίνεται να διασφαλιστεί ότι το έργο χρησιμοποιεί τις σωστές εκδόσεις των απαιτούμενων βιβλιοθηκών, μειώνοντας τα πιθανά προβλήματα συμβατότητας.
- Παραλλαγές κατασκευής: Απλοποιεί τη δημιουργία πολλαπλών εκδόσεων μιας εφαρμογής, όπως δωρεάν και premium builds ή διαμορφώσεις debug και release. Αυτό το χαρακτηριστικό επιτρέπει στους προγραμματιστές να διατηρούν μια ενιαία βάση κώδικα, προσαρμόζοντας παράλληλα τις κατασκευές σε διαφορετικά κοινά ή περιβάλλοντα.
- Πολλαπλά builds: Για τη βελτιστοποίηση της αποδοτικότητας, το Gradle εντοπίζει τις αλλαγές στο έργο και ανακατασκευάζει μόνο τα επηρεαζόμενα στοιχεία. Αυτό μειώνει σημαντικά τους χρόνους build, ειδικά σε έργα μεγάλης κλίμακας.
- Υποστήριξη πολλαπλών μοντέλων: Το Gradle υποστηρίζει την αρθρωματοποίηση, επιτρέποντας στους προγραμματιστές να χωρίσουν ένα έργο σε μικρότερα, διαχειρίσιμα στοιχεία. Κάθε ενότητα μπορεί να κατασκευαστεί και να δοκιμαστεί ανεξάρτητα, προωθώντας την επαναχρησιμοποίηση και τη συντηρησιμότητα του κώδικα.
- Επεκτασιμότητα μέσω plugins: Το οικοσύστημα πρόσθετων του Gradle ενισχύει τις δυνατότητές του. Το Android Gradle Plugin (AGP), για παράδειγμα, παρέχει εργαλεία ειδικά προσαρμοσμένα στην πλατφόρμα Android, και το APK ανήκει στα plugins.

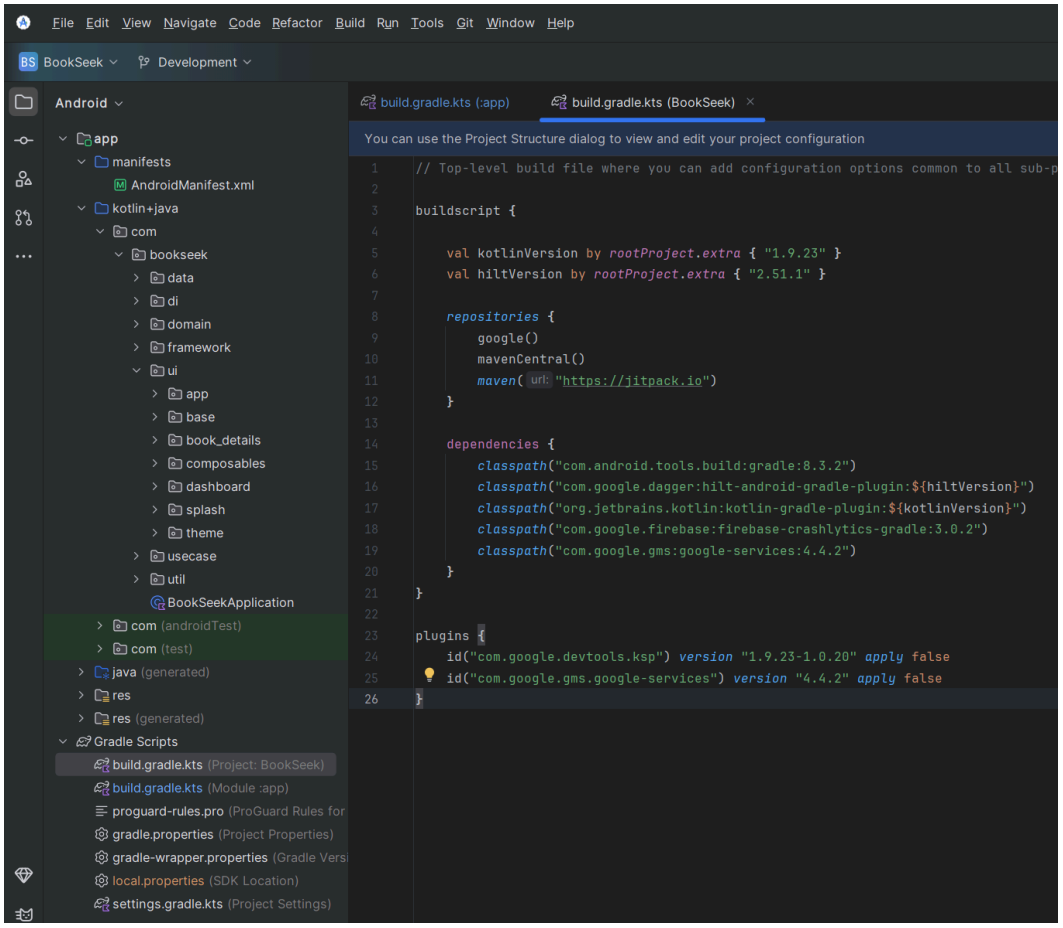
Ενώ το Gradle είναι ένα απαραίτητο εργαλείο υπάρχουν και προκλήσεις. Μία από τις πιο κοινές ανησυχίες μεταξύ των προγραμματιστών είναι η πιθανότητα για μεγάλους χρόνους build, ιδιαίτερα σε μεγάλα έργα με πολλές εξαρτήσεις.

Ωστόσο, έχουν εισαχθεί λύσεις όπως το Gradle Profiler, η προσωρινή αποθήκευση build και η παράλληλη εκτέλεση για τον μετριασμό αυτών των προβλημάτων.

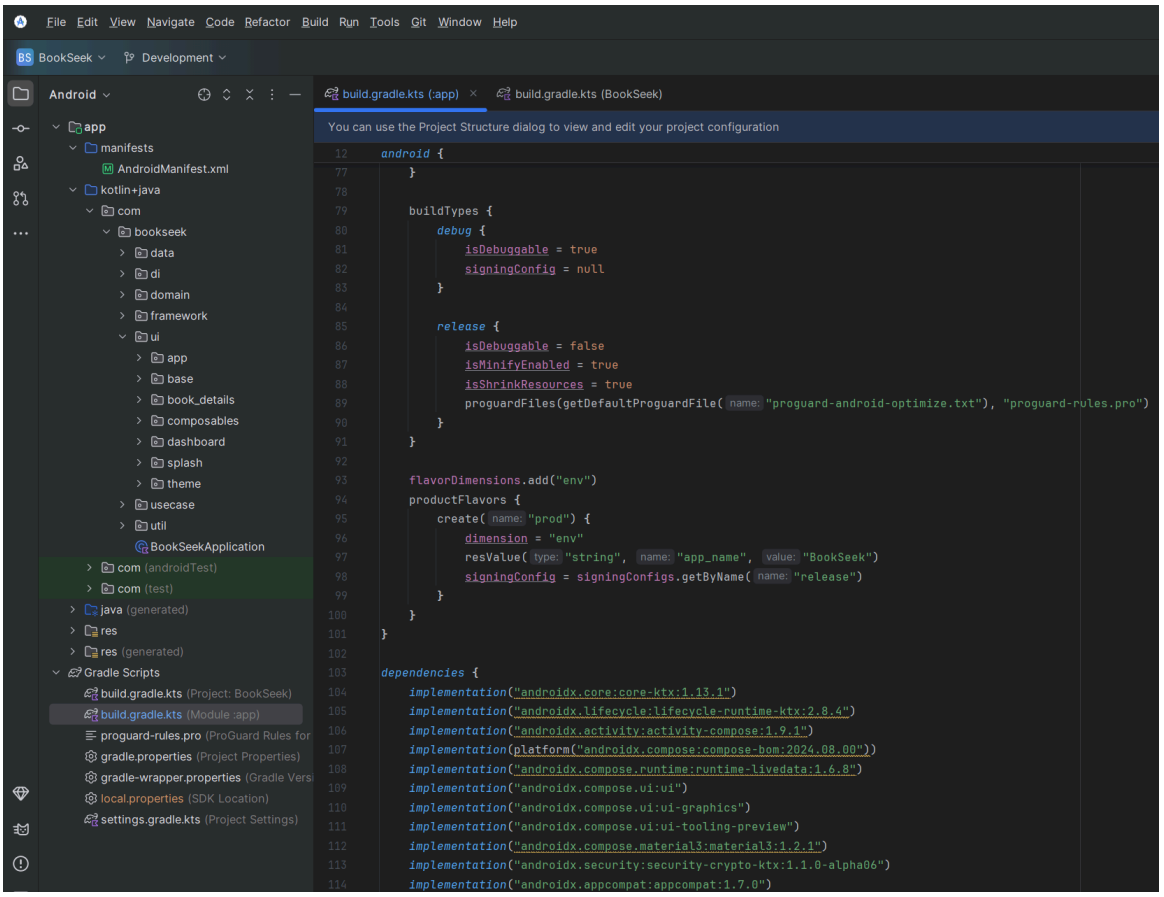
Πρόσφατες έρευνες, αναδεικνύουν αποτελεσματικές στρατηγικές για τη βελτιστοποίηση των Gradle build, όπως ο εξορθολογισμός των ρυθμίσεων εξαρτήσεων και η αξιοποίηση των προηγμένων εργαλείων προφίλ του Gradle. Η βαθιά ενσωμάτωση του Gradle με το Android Studio αυξάνει περαιτέρω τη χρησιμότητά του. Μέσω του Android Studio, γίνεται η διαχείριση του Gradle, να ρυθμίζονται σενάρια κατασκευής και να παρακολουθούνται τα αποτελέσματα της κατασκευής χωρίς να εγκαταλειφθεί το περιβάλλον ανάπτυξης. Αυτή η ενσωμάτωση μειώνει τα έξοδα και επιτρέπει μια πιο εύκολη εμπειρία ανάπτυξης [23].



Σχημα 2.3: Δομή Gradle



Σχήμα 2.3.1: Android studio Gradle σε επίπεδο project



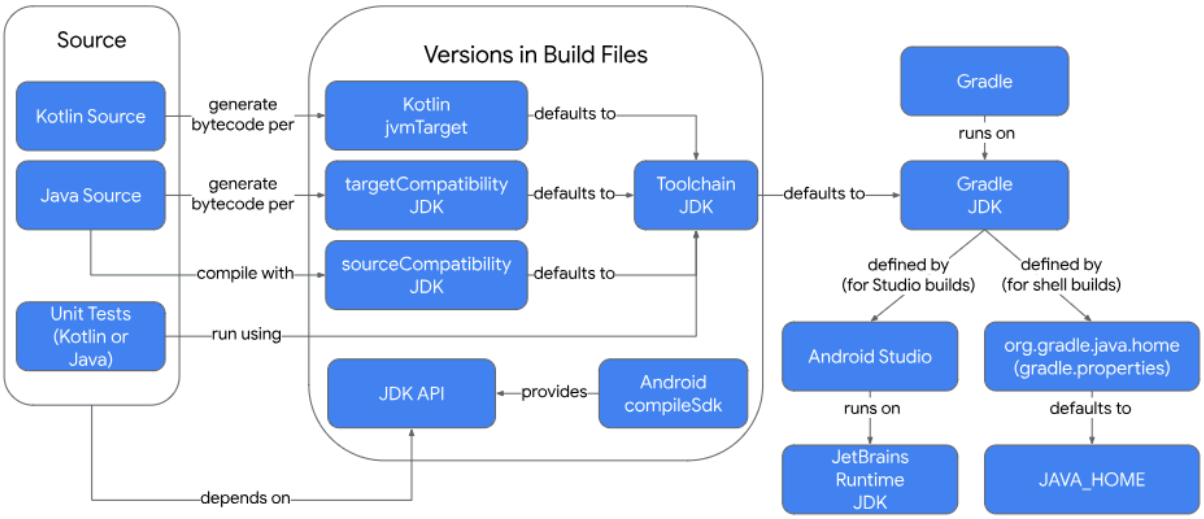
Σχήμα 2.3.2: Android studio Gradle σε επίπεδο Module

2.2.4 Kotlin

Η Kotlin είναι μια στατικά τυποποιημένη γλώσσα προγραμματισμού που εκτελείται στην εικονική μηχανή της Java (Java Virtual Machine) που ειναι μερος του Java Development Kit [24]. Σχεδιασμένη από την JetBrains, φημίζεται για το συνοπτικό συντακτικό της, τη διαλειτουργικότητα με τη Java και τα χαρακτηριστικά ασφαλείας της.

Η Kotlin αναγνωρίστηκε επίσημα ως προτιμώμενη γλώσσα για την ανάπτυξη Android από την Google το 2017 και έκτοτε έχει γίνει ακρογωνιαίος λίθος των σύγχρονων εφαρμογών Android. Η ανάπτυξη της Kotlin ξεκίνησε το 2010 από την JetBrains, τους δημιουργούς του IntelliJ IDEA. Κυκλοφόρησε επίσημα το 2016, με στόχο την αντιμετώπιση κοινών προβλημάτων που αντιμετωπίζουν οι προγραμματιστές της Java, όπως η πολυλογία και η ασφάλεια null. Η έγκριση της Google το 2017 και η ανακήρυξή της ως επίσημης γλώσσας ανάπτυξης Android το 2019 εδραίωσαν τη σημασία της στο οικοσύστημα του Android.

Η Kotlin είναι ανοικτού κώδικα υπό την άδεια Apache 2.0, και η ανάπτυξη που καθοδηγείται από την κοινότητά της είναι αξιοσημείωτη.



2.4 Android JDK

Μερικά από τα χαρακτηριστικά της Kotlin είναι:

- Η Kotlin έχει σχεδιαστεί για να μειώνει τον κώδικα που χρησιμοποιείται από πολλούς, καθιστώντας τα προγράμματα ευκολότερα στη συγγραφή και την ανάγνωση. Για παράδειγμα: Στην Java, η δημιουργία ενός απλού POJO (Plain Old Java Object) απαιτεί πολλές γραμμές κώδικα για getters, setters και constructors. Στην Kotlin, αυτό απλοποιείται με τις κλάσεις δεδομένων(data class), οι οποίες δημιουργούν αυτόματα αυτές τις μεθόδους. Η Kotlin χρησιμοποιεί επίσης συνοπτικές εκφράσεις για κοινές λειτουργίες, όπως val για αμετάβλητες μεταβλητές και πρότυπα συμβολοσειρών (“Hello, $name”) αντί για συνένωση. Αυτή η μειωμένη πολυπλοκότητα οδηγεί σε καθαρότερο κώδικα, βελτιώνοντας τη συντηρησιμότητα και μειώνοντας την πιθανότητα σφαλμάτων.
- Ένα από τα πιο απογοητευτικά σφάλματα στη Java είναι η NullPointerException, που συχνά αναφέρεται ως το «λάθος του δισεκατομμυρίου δολαρίων»[25]. Η Kotlin εξαλείφει αυτό το ζήτημα εισάγοντας έναν ισχυρό μηχανισμό ασφάλειας null.
- Οι μεταβλητές είναι εξ ορισμού μη μηδενικές (π.χ. var name: String). Οι nullable τύποι δηλώνονται ρητά χρησιμοποιώντας ένα ? (π.χ. var name: String?), διασφαλίζοντας ότι η δήλωση null είναι μια συνειδητή απόφαση. Ο τελεστής safe-call της Kotlin (π.χ. name?.) και ο τελεστής Elvis (π.χ. name ?: “onoma”) απλοποιούν την εργασία με nullable τύπους, αποτρέποντας τις καταρρεύσεις - crash κατά την εκτέλεση. Αυτό το χαρακτηριστικό εξασφαλίζει ασφαλέστερο κώδικα και μειώνει την ανάγκη για επαναλαμβανόμενους ελέγχους null.
- Διαλειτουργικότητα με Java, η Kotlin είναι πλήρως συμβατή με τη Java, επιτρέποντας τη συνύπαρξη και των δύο γλωσσών στο ίδιο έργο. Αυτό επιτυγχάνεται μέσω: Την απρόσκοπτη κλήση μεθόδων της Java και την πρόσβαση σε βιβλιοθήκες της Java από την Kotlin. Μετατροπή κώδικα Java σε Kotlin χρησιμοποιώντας το ενσωματωμένο εργαλείο μετατροπή στο Android Studio. Σταδιακή υιοθέτηση της Kotlin σε παλαιότερα έργα χωρίς να απαιτείται πλήρης επανεγγραφή. Αυτή η λειτουργικότητα καθιστά την Kotlin ιδανική επιλογή για ομάδες που μεταβαίνουν από τη Java σε Kotlin.

- Χαρακτηριστικά λειτουργικού προγραμματισμού η Kotlin συνδυάζει τον λειτουργικό προγραμματισμό με τον αντικειμενοστραφή προγραμματισμό, παρέχοντας στους προγραμματιστές ισχυρά εργαλεία για καθαρότερο και αποτελεσματικότερο κώδικα:
    1. Lambdas και συναρτήσεις ανώτερης τάξης: Δίνει την δυνατότητα να περνούν συναρτήσεις ως παραμέτρους, καθιστώντας τον κώδικα επαναχρησιμοποιήσιμο και μειώνοντας τον πλεονασμό.
    2. Αμετάβλητο: Ενθαρρύνει τη χρήση αμετάβλητων λιστών (listOf, mapOf) για την αποφυγή ακούσιων αλλαγών κατάστασης, προωθώντας την ασφάλεια.
    3. Συναρτήσεις επέκτασης: Η Kotlin επιτρέπει την προσθήκη νέων λειτουργιών σε υπάρχουσες κλάσεις χωρίς να τις τροποποιεί, βελτιώνοντας την ευελιξία και την αρθρωτότητα.

  Αυτά τα χαρακτηριστικά καθιστούν την Kotlin μια ισχυρή επιλογή για την υλοποίηση πολύπλοκων αλγορίθμων και καθαρών αρχιτεκτονικών προτύπων.

- Η διαχείριση εργασιών στο παρασκήνιο, όπως οι αιτήσεις δικτύου ή οι λειτουργίες βάσης δεδομένων, μπορεί να είναι δυσκολότερη στη Java. Οι coroutines της Kotlin[26] απλοποιούν τον ασύγχρονο προγραμματισμό, οι coroutines επιτρέπουν μη μπλοκαρισμένες, ταυτόχρονες λειτουργίες, μειώνοντας την πολυπλοκότητα του threading[27]. Με απλές λέξεις-κλειδιά όπως launch και async, εκτελούνται εργασίες στο παρασκήνιο και να χειρίζονται τα αποτελέσματά τους με ελάχιστο κώδικα. Σε αντίθεση με τις παραδοσιακές προσεγγίσεις που βασίζονται σε callback, οι coroutines αποφεύγουν τα callback, με αποτέλεσμα πιο ευανάγνωστο και συντηρήσιμο κώδικα. Οι coroutines είναι ιδιαίτερα χρήσιμες στην ανάπτυξη Android, όπου η απόκριση του UI είναι κρίσιμη.
- Smart Casts της Kotlin μειώνουν την ανάγκη για δήλωσει τύπων. Ο μεταγλωττιστής χειρίζεται αυτόματα τους ελέγχους τύπων και τις μετατροπές, όπου αυτό είναι εφικτό: Μετά από έναν έλεγχο τύπου (if (x = String)), η Kotlin μετατρέπει αυτόματα το x σε String εντός του σχετικού πεδίου εφαρμογής, εξαλείφοντας την ανάγκη για χειροκίνητες μετατροπές. Αυτό το χαρακτηριστικό βελτιώνει την αναγνωσιμότητα του κώδικα και μειώνει τα πιθανά σφάλματα κατά τον χρόνο εκτέλεσης.
- Οι προεπιλεγμένες παράμετροι μπορούν να έχουν αρχικές τιμές, έτσι, βελτιώνει την ευελιξία των συναρτήσεων, μειώνοντας την ανάγκη για υπερφορτωμένες μεθόδους. Οι Ονομαστικές παράμετροι μπορούν να μεταβιβάζονται με όνομα, καθιστώντας τον κώδικα πιο αυτο-τεκμηριωμένο και αποφεύγοντας τη σύγχυση της σειράς των παραμέτρων.
- Εργαλεία και κοινοτική υποστήριξη η Kotlin υποστηρίζεται από ένα ισχυρό οικοσύστημα:
    1. Android Studio Integration: συμπεριλαμβανομένης της επισήμανσης σύνταξης, της αποσφαλμάτωσης και της συμπλήρωσης κώδικα.
    2. Βιβλιοθήκες και frame: Η Kotlin συνεργάζεται απρόσκοπτα με δημοφιλείς βιβλιοθήκες Java και προσφέρει τις δικές της προηγμένες βιβλιοθήκες.
    3. Κοινότητα: Η Kotlin διαθέτει μια ζωντανή και υποστηρικτική κοινότητα, με εκτεταμένη τεκμηρίωση, σεμινάρια και ενεργά φόρουμ.

Ο συνδυασμός της Kotlin με το συνοπτικό συντακτικό, τα χαρακτηριστικά ασφαλείας και τα σύγχρονα παραδείγματα προγραμματισμού έχει φέρει επανάσταση στην ανάπτυξη του Android. Η ικανότητά της να αντιμετωπίζει τα κοινά δύσκολα σημεία, ενώ ενσωματώνεται απρόσκοπτα με τα υπάρχοντα οικοσυστήματα Java, την έχει καταστήσει τη γλώσσα επιλογής τόσο για νέα έργα όσο και για παλιά που θέλουν να τροποποιήσουν τον υπάρχων κώδικα. Με την αυξανόμενη υιοθέτησή της και

το εξελισσόμενο οικοσύστημά της, η Kotlin αντιπροσωπεύει το μέλλον του προγραμματισμού για Android [10].

2.3 Github

Το GitHub[28] είναι μια διαδικτυακή πλατφόρμα που έχει σχεδιαστεί για τον εξορθολογισμό του ελέγχου εκδόσεων και την προώθηση της συνεργασίας μεταξύ τον προγραμματιστών σε μια ομάδα ανάπτυξης λογισμικού. Βασισμένο στο Git[29], ένα κατανεμημένο σύστημα ελέγχου εκδόσεων ανοικτού κώδικα, το GitHub ξεκίνησε το 2008 από τους Tom Preston-Werner, Chris Wanstrath, PJ Hyett και Scott Chacon.

Από την ίδρυσή του, έχει φέρει επανάσταση στον τρόπο συνεργασίας των προγραμματιστών, παρέχοντας εργαλεία για τη διαχείριση του κώδικα, την παρακολούθηση των αλλαγών και τον συντονισμό των ομαδικών προσπαθειών. Τα χαρακτηριστικά του το έχουν καταστήσει απαραίτητο εργαλείο για τη σύγχρονη ανάπτυξη λογισμικού, επιτρέποντας στις ομάδες να συνεργάζονται πέρα από αποστάσεις και χρονικές ζώνες.

Στη βάση του, το GitHub βασίζεται στο Git για τον έλεγχο εκδόσεων, επιτρέποντας στους προγραμματιστές να παρακολουθούν και να διαχειρίζονται αποτελεσματικά τις αλλαγές. Με το GitHub, υπάρχει δυνατότητα δημιουργίας κλαδιών, το λεγομενο branch, για να εργάζονται ανεξάρτητα σε συγκεκριμένα χαρακτηριστικά ή διορθώσεις σφαλμάτων και αργότερα να συγχωνεύουν αυτές τις αλλαγές στο κύριο έργο μετά από ενδελεχή έλεγχο. Η πλατφόρμα φιλοξενεί repository, τα οποία λειτουργούν ως κεντρικές τοποθεσίες για τον κώδικα, την τεκμηρίωση και άλλους πόρους που σχετίζονται με ένα έργο. Αυτά τα repository μπορεί να είναι δημόσια, προωθώντας τη συνεργασία ανοικτού κώδικα, ή ιδιωτικά, προστατεύοντας το ιδιόκτητο λογισμικό. Η συνεργασία βρίσκεται στο επίκεντρο της λειτουργικότητας του GitHub.

Τα Pull requests είναι ένα βασικό χαρακτηριστικό που επιτρέπει στους προγραμματιστές να προτείνουν αλλαγές, οι οποίες μπορούν στη συνέχεια να επανεξεταστούν και να συζητηθούν από τα μέλη της ομάδας πριν ενσωματωθούν στην κύρια βάση κώδικα. Αυτό διευκολύνει την αξιολόγηση από ομοτίμους και συμβάλλει στη διατήρηση της ποιότητας και της ακεραιότητας του κώδικα.

Για τον περαιτέρω εξορθολογισμό της συνεργασίας, το GitHub περιλαμβάνει ένα σύστημα παρακολούθησης προβλημάτων που βοηθά τις ομάδες να αναφέρουν, να διαχειρίζονται και να επιλύουν αποτελεσματικά σφάλματα ή αιτήματα. Οι πίνακες έργων, εμπνευσμένοι από τους πίνακες Kanban, επιτρέπουν στις ομάδες να οργανώνουν οπτικά τις εργασίες και να παρακολουθούν την πρόοδο, ενισχύοντας την παραγωγικότητα και τη διαφάνεια. Πέρα από τον έλεγχο εκδόσεων και τη συνεργασία, το GitHub παρέχει εργαλεία αυτοματοποίησης μέσω του GitHub Actions.

Αυτά τα εργαλεία επιτρέπουν στους προγραμματιστές να δημιουργούν, να δοκιμάζουν και να αναπτύσσουν κώδικα αυτόματα, βελτιώνοντας τις διαδικασίες συνεχούς ολοκλήρωσης και παράδοσης. Για τις ανάγκες φιλοξενίας, το GitHub υποστηρίζει την ανάπτυξη στατικών ιστότοπων μέσω του GitHub Pages, επιτρέποντας στους προγραμματιστές να παρουσιάζουν έργα, τεκμηρίωση ή χαρτοφυλάκια απευθείας από τα repository τους.

Το GitHub υπερέχει επίσης στην εμπλοκή της κοινότητας. Η διακλάδωση Repository για την να δημιουργήσουν ανεξάρτητα αντίγραφα, επιτρέποντάς τους να πειραματιστούν χωρίς να επηρεάσουν

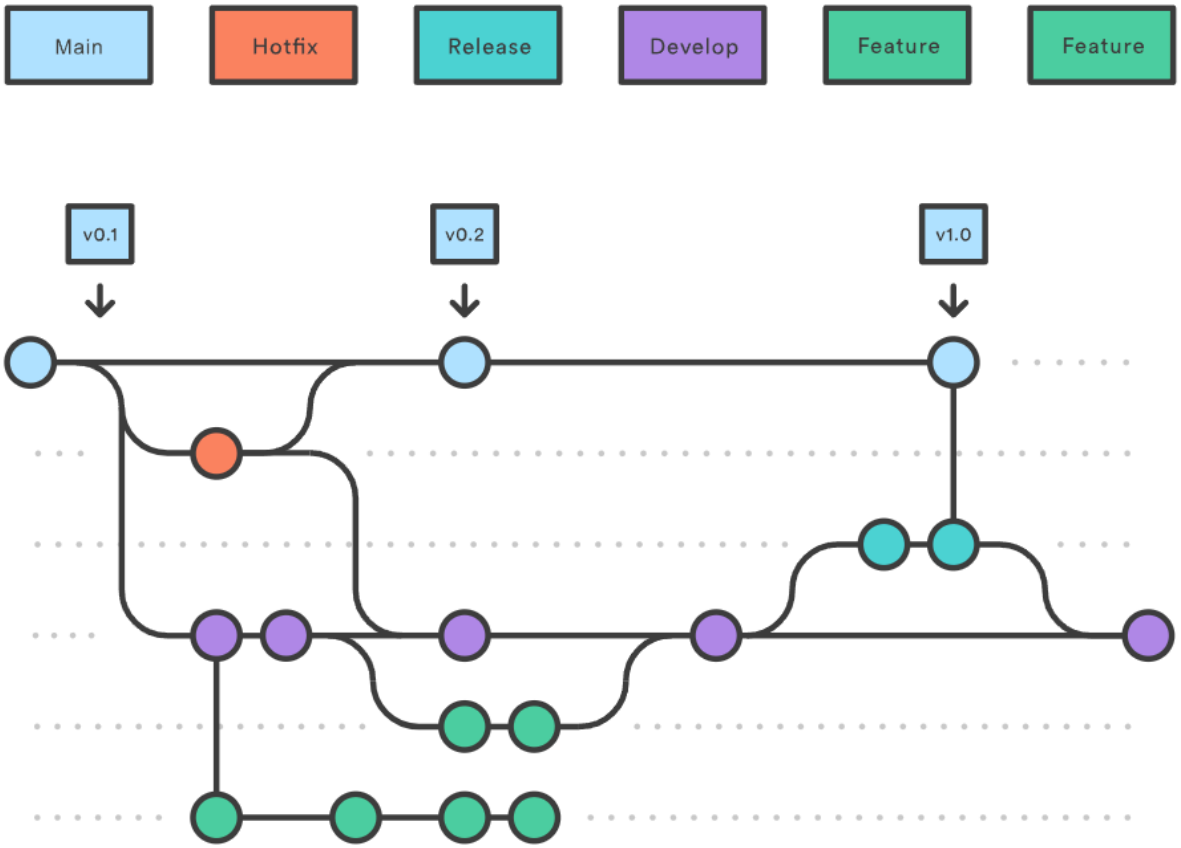
την αρχική βάση κώδικα. Με το starring projects, οι χρήστες μπορούν να αναδείξουν τα αγαπημένα τους, ενώ οι συζητήσεις παρέχουν ένα φόρουμ για πιο εμπεριστατωμένες συζητήσεις σχετικά με την ανάπτυξη έργων.

Η δημοτικότητα του GitHub μεταξύ των προγραμματιστών πηγάζει από την ικανότητά του να απλοποιεί τις συνεργατικές ροές εργασίας, ειδικά για τις ομάδες που είναι κατανεμημένες σε παγκόσμιο επίπεδο. Το μοντέλο διακλάδωσης που διαθέτει επιτρέπει την παράλληλη ανάπτυξη, διατηρώντας παράλληλα τη σταθερότητα του κύριου κλάδου, διευκολύνοντας τον πειραματισμό χωρίς να διακινδυνεύεται η ακεραιότητα του έργου.

Η πλατφόρμα φιλοξενεί επίσης εκατομμύρια έργα ανοικτού κώδικα, επιτρέποντας στους προγραμματιστές να μαθαίνουν και να συνεισφέρουν σε ευρέως χρησιμοποιούμενο λογισμικό, όπως το React, το TensorFlow και το Linux. Αυτές οι συνεισφορές όχι μόνο ενισχύουν τις δεξιότητες αλλά βοηθούν επίσης τους προγραμματιστές να αποκτήσουν προβολή στην κοινότητα, οι δυνατότητες ενσωμάτωσης της πλατφόρμας ενισχύουν περαιτέρω τη χρησιμότητά της. Το GitHub συνεργάζεται άψογα με εργαλεία όπως το Android Studio, Visual Studio Code, το Jenkins, το Travis CI, το Trello και το Jira, δημιουργώντας ένα καλό οικοσύστημα ανάπτυξης. Δίνει επίσης έμφαση στην ασφαλή ανάπτυξη, προσφέροντας λειτουργίες όπως σάρωση εξαρτήσεων, μυστική σάρωση και συμβουλές ασφαλείας, ώστε να βοηθήσει τους προγραμματιστές να εντοπίσουν και να αντιμετωπίσουν τα τρωτά σημεία στον κώδικα τους.

Οι εφαρμογές του GitHub εκτείνονται πέρα από την τεχνολογία λογισμικού. Στην εκπαίδευση, χρησιμοποιείται για τη διδασκαλία δεξιοτήτων κωδικοποίησης και διαχείρισης έργων, προσφέροντας στους μαθητές πρακτική εμπειρία με συνεργατικά εργαλεία. Οι ερευνητές επωφελούνται επίσης από το GitHub, καθώς παρέχει μια πλατφόρμα για την κοινή χρήση συνόλων δεδομένων, σεναρίων και μεθοδολογιών, προωθώντας τη διαφάνεια και την αναπαραγωγιμότητα στην επιστημονική έρευνα.

Το GitHub έχει μεταμορφώσει τον τρόπο με τον οποίο κατασκευάζεται, διαμοιράζεται και συντηρείται το λογισμικό. Ο ισχυρός συνδυασμός του με τον έλεγχο εκδόσεων, τα εργαλεία συνεργασίας, τις δυνατότητες αυτοματοποίησης και την ενσωμάτωση με άλλους πόρους ανάπτυξης το έχει καταστήσει πολύ σημαντικό στις σύγχρονες πρακτικές ανάπτυξης. Πέρα από τα τεχνικά του πλεονεκτήματα, το GitHub έχει καλλιεργήσει ένα ακμάζον οικοσύστημα έργων και συνεργατών ανοικτού κώδικα, προωθώντας την καινοτομία και την ανταλλαγή γνώσεων σε όλους τους κλάδους. Ως ζωτικό εργαλείο στην ανάπτυξη λογισμικού, την εκπαίδευση και την έρευνα, ο αντίκτυπος του GitHub συνεχίζει να αυξάνεται, διαμορφώνοντας το μέλλον της συνεργατικής τεχνολογίας.



Σχήμα 2.4: Github workflow

2.3.1 Git

Το Git είναι ένα κατανεμημένο σύστημα ελέγχου εκδόσεων που έχει σχεδιαστεί για να παρακολουθεί τις αλλαγές σε αρχεία και να διευκολύνει τη συνεργασία μεταξύ προγραμματιστών. Δημιουργήθηκε το 2005 από τον Linus Torvalds, τον δημιουργό του πυρήνα του Linux, το Git αναπτύχθηκε για να αντιμετωπίσει τους περιορισμούς των υφιστάμενων συστημάτων ελέγχου εκδόσεων, όπως η ταχύτητα, η αξιοπιστία και η επεκτασιμότητα. Με την πάροδο του χρόνου, το Git έγινε το βιομηχανικό πρότυπο για τον έλεγχο εκδόσεων, τροφοδοτώντας αμέτρητα έργα παγκοσμίως.

Ένα από τα καθοριστικά χαρακτηριστικά του Git είναι η κατανεμημένη αρχιτεκτονική του. Σε αντίθεση με τα συγκεντρωτικά συστήματα όπου ένας μόνο διακομιστής φιλοξενεί το repository, το Git επιτρέπει σε κάθε προγραμματιστή να έχει ένα πλήρες αντίγραφο του repository, συμπεριλαμβανομένου του ιστορικού του. Αυτό εξασφαλίζει υψηλή διαθεσιμότητα και επιτρέπει στους προγραμματιστές να εργάζονται εκτός σύνδεσης χωρίς να χάνουν την πρόσβαση στο πλήρες ιστορικό ή τα χαρακτηριστικά του έργου. Οι αλλαγές μπορούν αργότερα να συγχρονιστούν με άλλους μέσω ενός απομακρυσμένου repository.

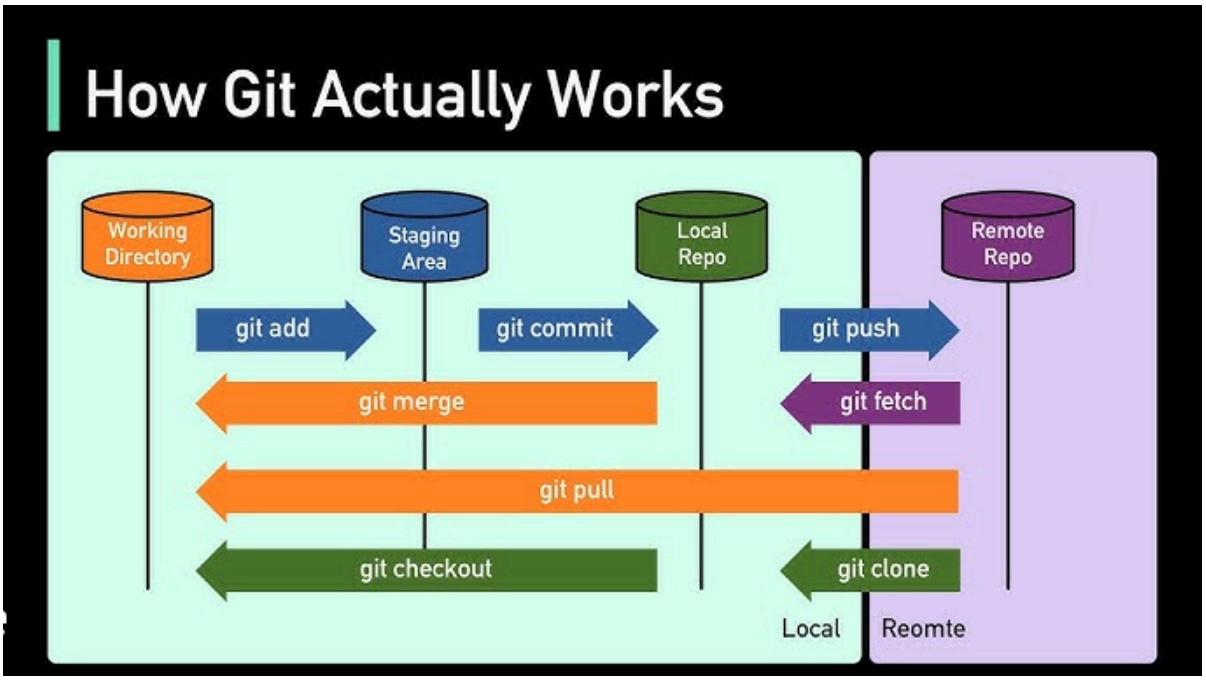
Το Git ξεχωρίζει επίσης για τις δυνατότητες διακλάδωσης, branch, και συγχώνευσης, επιτρέποντας στους προγραμματιστές να εργάζονται σε χαρακτηριστικά ή διορθώσεις ανεξάρτητα. Οι διακλαδώσεις επιτρέπουν στους προγραμματιστές να πειραματίζονται και να καινοτομούν χωρίς να επηρεάζουν την κύρια βάση κώδικα. Μόλις ολοκληρωθεί η εργασία, ο κλάδος, branch, μπορεί να συγχωνευτεί πίσω

στην κύρια βάση κώδικα, συχνά με ελάχιστες συγκρούσεις, conflicts, λόγω των αποτελεσματικών αλγορίθμων συγχώνευσης του Git. Αυτές οι δυνατότητες καθιστούν τη συνεργασία σε μεγάλα έργα απρόσκοπτη.

Ένα άλλο σημαντικό χαρακτηριστικό είναι η περιοχή σταδιοποίησης του Git, η οποία παρέχει μια προεπισκόπηση των αλλαγών πριν αυτές δεσμευτούν στο repository. Αυτό προσφέρει λεπτομερή έλεγχο του τι περιλαμβάνεται σε κάθε δέσμευση, προωθώντας την καλύτερη οργάνωση και τεκμηρίωση των αλλαγών.

Επιπλέον, το Git διασφαλίζει την ακεραιότητα των δεδομένων, χρησιμοποιώντας έναν αλγόριθμο κατακερματισμού (SHA-1) για την ασφαλή και μοναδική ταυτοποίηση κάθε commit, καθιστώντας πρακτικά αδύνατη την αλλαγή του ιστορικού χωρίς ανίχνευση.

Το Git έχει γίνει η κύρια πλατφόρμα όπως το GitHub και το GitLab, οι οποίες προσθέτουν εργαλεία συνεργασίας και αυτοματοποίησης στη βασική του λειτουργικότητα. Η ευελιξία, η ταχύτητα και η αξιοπιστία του το έχουν καταστήσει απαραίτητο για την ανάπτυξη λογισμικού, από τα προσωπικά έργα έως τις εταιρικές εφαρμογές.



Σχήμα 2.5: Πως λειτουργεί το Git

2.4 Αρχιτεκτονική

Ο αρχιτεκτονικός σχεδιασμός μιας εφαρμογής παίζει σημαντικό ρόλο στον καθορισμό της συντηρησιμότητας, της επεκτασιμότητας και της δυνατότητας ελέγχου της. Στη σύγχρονη ανάπτυξη λογισμικού, ο συνδυασμός του προτύπου σχεδίασης Model-View-ViewModel (MVVM) με την clean architecture προσφέρει ένα ισχυρό πλαίσιο για την επίτευξη αυτών των στόχων.

Αυτά τα πρότυπα, τα οποία αναγνωρίζονται ξεχωριστά για την ικανότητά τους να διαχωρίζουν τις ανησυχίες και να βελτιώνουν την οργάνωση του κώδικα, λειτουργούν συνεργικά για τη δημιουργία εφαρμογών που είναι αρθρωτές, ευέλικτες και εύκολα εξελισσόμενες με την πάροδο του χρόνου.

Τα επόμενα κεφάλαια θα διερευνήσουν τις αρχές και την εφαρμογή του MVVM και της Καθαρής Αρχιτεκτονικής, τόσο ανεξάρτητα όσο και ως ολοκληρωμένη προσέγγιση. Το MVVM διευκολύνει την αποσύνδεση της διεπαφής χρήστη (UI) από την επιχειρησιακή λογική(Business logic), ενώ η clean architecture εισάγει καλά καθορισμένα επίπεδα για να διαχωρίσει τις ευθύνες και να διασφαλίσει ότι οι εξαρτήσεις ρέουν προς τα μέσα προς το επίπεδο τομέα.

Μαζί, αυτά τα παραδείγματα αντιμετωπίζουν τις κοινές προκλήσεις του στενά συνδεδεμένου κώδικα και των δύσκολα συντηρήσιμων έργων, επιτρέποντας στους προγραμματιστές να δημιουργούν ανθεκτικά και ελέγξιμα συστήματα λογισμικού. Η συζήτηση θα αναδείξει επίσης τον τρόπο με τον οποίο ο συνδυασμός του MVVM με την clean architecture συμβάλλει στη δομική ακεραιότητα της εφαρμογής που αναπτύσσεται. Με την ευθυγράμμιση των δύο μεθοδολογιών, η προσέγγιση αυτή διασφαλίζει ότι κάθε συστατικό έχει έναν ξεχωριστό ρόλο, προωθώντας έναν συντηρήσιμο και επεκτάσιμο σχεδιασμό εφαρμογών.

Αυτή η ενοποίηση είναι ιδιαίτερα πολύτιμη για το χειρισμό πολύπλοκης επιχειρηματικής λογικής, διατηρώντας παράλληλα ένα καθαρό και διαισθητικό UI. Μέσω αυτού, η πτυχιακή έχει ως στόχο να παρέχει έναν ολοκληρωμένο οδηγό για την υιοθέτηση αυτών των αρχιτεκτονικών αρχών για τη βελτίωση της ποιότητας και της αποτελεσματικότητας της ανάπτυξης εφαρμογών.

2.4.1 MVVM

Το πρότυπο σχεδίασης Model-View-ViewModel [30] (MVVM) έχει αναδειχθεί ως μια ισχυρή αρχιτεκτονική για την ανάπτυξη κλιμακούμενων, συντηρήσιμων και ελέγξιμων εφαρμογών, ιδίως σε τομείς όπως η ανάπτυξη κινητών, επιτραπέζιων και διαδικτυακών εφαρμογών.

Επεκτείνει τις αρχές του κλασικού προτύπου MVC (Model-View-Controller) εισάγοντας ένα επίπεδο ViewModel, εξασφαλίζοντας σαφή διαχωρισμό των ανησυχιών μεταξύ της διεπαφής χρήστη (UI) και της υποκείμενης επιχειρηματικής λογικής.

Στον πυρήνα της, η MVVM περιλαμβάνει τρία κύρια συστατικά:

1. Το Μοντέλο, το οποίο αναπαριστά το επίπεδο δεδομένων και χειρίζεται την επιχειρησιακή λογική
2. Την Προβολή(view), η οποία είναι υπεύθυνη για την απόδοση της διεπαφής χρήστη και την καταγραφή των αλληλεπιδράσεων του χρήστη
3. Το ViewModel, το οποίο χρησιμεύει ως μεσολαβητής μεταξύ του Μοντέλου και της Προβολής.

Το ViewModel ενθυλακώνει τη λογική που σχετίζεται με το UI και εκθέτει ροές δεδομένων στο view μέσω δεσμεύσεων ή παρατηρητών. Αυτό διασφαλίζει ότι οι αλλαγές στα δεδομένα αντανακλώνται αυτόματα στο UI και οι αλληλεπιδράσεις του χρήστη μεταφέρονται απρόσκοπτα πίσω στο ViewModel, αυτός ο μηχανισμός δέσμευσης αποτελεί χαρακτηριστικό γνώρισμα του MVVM. Πλατφόρμες όπως το Android αξιοποιούν εργαλεία όπως το LiveData, MutableState ή το Jetpack Compose για να απλοποιήσουν αυτόν τον συγχρονισμό.

Το μοτίβο προωθεί επίσης τη μονόδρομη ροή δεδομένων, καθιστώντας την συμπεριφορά της εφαρμογής προβλέψιμη και ευκολότερη στην αποσφαλμάτωση. Η κύρια δύναμη του MVVM έγκειται στην ικανότητά του να αποσυνδέει το UI από την επιχειρησιακή λογική. Αυτή η αρθρωτότητα ενισχύει τη συντηρησιμότητα του κώδικα και καθιστά τα επιμέρους στοιχεία ανεξάρτητα ελέγξιμα.

Για παράδειγμα, το ViewModel μπορεί να δοκιμαστεί χωρίς να απαιτείται ένα πλαίσιο UI, απλοποιώντας την υλοποίηση.

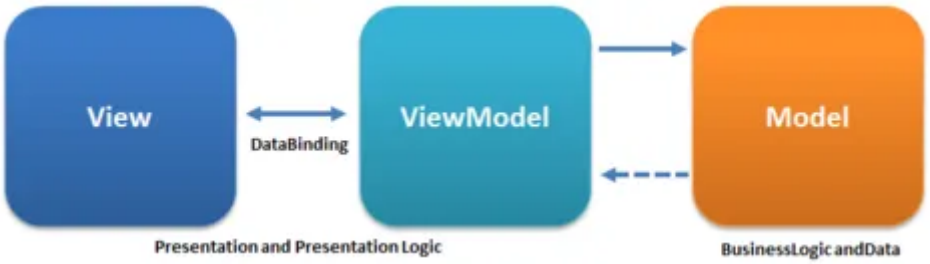
Επιπλέον, απομονώνοντας τη λογική δεδομένων στο επίπεδο Μοντέλου, το MVVM διευκολύνει την απρόσκοπτη ενσωμάτωση με εξωτερικές πηγές δεδομένων, όπως APIs ή βάσεις δεδομένων.

Το πρότυπο MVVM έχει υιοθετηθεί ευρέως σε διάφορες τεχνολογίες. Στο Android, το MVVM είναι αναπόσπαστο μέρος της αρχιτεκτονικής Jetpack, χρησιμοποιώντας κλάσεις ViewModel και Data Binding για αποτελεσματική διαχείριση της κατάστασης (State). Ομοίως, στις εφαρμογές του Windows Presentation Foundation (WPF), το MVVM συνεργάζεται άψογα με το XAML για τη δημιουργία ευέλικτων και δυναμικών UI.

Παρά τα πλεονεκτήματά της, η MVVM δεν είναι χωρίς προκλήσεις. Η εξάρτηση από τη δέσμευση δεδομένων μπορεί να εισάγει συμφόρηση στις επιδόσεις αν δεν γίνει σωστή διαχείριση, ειδικά σε εφαρμογές μεγάλης κλίμακας με πολύπλοκα UI.

Επιπλέον, η αφαίρεση που εισάγεται από το επίπεδο ViewModel μπορεί να οδηγήσει σε υπερβολικους υπολογισμούς σε μικρά έργα, όπου απλούστερες αρχιτεκτονικές όπως η MVC ή η MVP[32] θα μπορούσαν να είναι επαρκείς.

Το MVVM έφερε επανάσταση στην ανάπτυξη εφαρμογών προωθώντας μια αρθρωτή και συντηρήσιμη αρχιτεκτονική. Η ικανότητά της να αποσυνδέει το UI και τη λογική, μαζί με τους ισχυρούς μηχανισμούς δέσμευσης δεδομένων, την καθιστά ιδανική για σύγχρονες, κλιμακούμενες εφαρμογές [12].



Σχήμα 2.6: Δομή MVVM

2.4.2 Clean Architecture

Η Clean Architecture [31], η οποία εισήχθη από τον Robert C. Martin (κοινώς γνωστό ως Uncle Bob), είναι μια μεθοδολογία σχεδιασμού λογισμικού που επικεντρώνεται στη δημιουργία εύρωστων, επεκτάσιμων και συντηρήσιμων συστημάτων με την τήρηση των αρχών του διαχωρισμού των προβλημάτων και της αντιστροφής των εξαρτήσεων.

Η προσέγγιση προωθεί μια δομημένη διαστρωμάτωση των στοιχείων λογισμικού, διασφαλίζοντας ότι η βασική επιχειρησιακή λογική είναι ανεξάρτητη από εξωτερικά συστήματα όπως βάσεις

δεδομένων, πλαίσια ή στοιχεία διεπαφής χρήστη (UI). Επιβάλλοντας σαφή όρια, η Clean Architecture δημιουργεί ένα ευέλικτο πλαίσιο όπου τα επιμέρους επίπεδα μπορούν να τροποποιηθούν ή να αντικατασταθούν χωρίς να επηρεαστεί η υπόλοιπη εφαρμογή. Στη βάση της, η Clean Architecture οργανώνει το σύστημα σε ομόκεντρα επίπεδα με καθορισμένες αρμοδιότητες.

2.4.2.1 Entity Layer

Οι οντότητες (Entity) είναι τα βασικά επιχειρησιακά αντικείμενα μιας εφαρμογής, αντιπροσωπεύουν τους θεμελιώδεις κανόνες και τη δομή της. Τα αντικείμενα αυτά ενθυλακώνουν τα βασικά δεδομένα και τις συμπεριφορές που καθορίζουν το domain layer.

Οι οντότητες χρησιμοποιούνται συνήθως στο domain layer, ανεξάρτητα από οποιοδήποτε πλαίσιο ή εξωτερική τεχνολογία, διασφαλίζοντας ότι μπορούν να αντέξουν στη δοκιμασία του χρόνου και να παραμείνουν ανεπηρέαστες από αλλαγές στο περιβάλλον σύστημα.

Για παράδειγμα, σε μια εφαρμογή ηλεκτρονικού εμπορίου, μια οντότητα Παραγγελίας ή Προϊόντος μπορεί να ορίζει χαρακτηριστικά όπως η τιμή, η περιγραφή ή το επίπεδο αποθέματος.

Οι οντότητες αλληλεπιδρούν με τις περιπτώσεις χρήσης για την επιβολή επιχειρηματικών κανόνων, αλλά δεν εξαρτώνται από τα επίπεδα UI ή δεδομένων. Αυτή η ανεξαρτησία τις καθιστά επαναχρησιμοποιήσιμες σε πολλαπλά πλαίσια και ελέγξιμες μεμονωμένα.

2.4.2.2 Use Case Layer

Τα Use case (που αναφέρονται επίσης ως interactors) ενσωματώνουν την επιχειρησιακή λογική για συγκεκριμένες λειτουργίες ή ροές εργασίας εντός μιας εφαρμογής. Αυτές βρίσκονται στο επίπεδο Domain layer, αλληλεπιδρώντας με οντότητες και repositories για τον συντονισμό εργασιών.

Για παράδειγμα, μια περίπτωση χρήσης για την είσοδο χρήστη θα επικυρώσει τα διαπιστευτήρια έναντι του repository και θα επιστρέψει τα αποτελέσματα στο επίπεδο presentation layer (UI). Τα use case γεφυρώνουν το χάσμα μεταξύ της διεπαφής χρήστη και των επιχειρησιακών κανόνων, διασφαλίζοντας ότι η επιχειρησιακή λογική είναι επαναχρησιμοποιήσιμη και αποσυνδεδεμένη από την υλοποίηση της διεπαφής χρήστη και των δεδομένων. Αυτή η απομόνωση επιτρέπει τον εύκολο έλεγχο, καθώς η λογική μπορεί να επαληθευτεί χωρίς να απαιτείται πρόσβαση σε εξωτερικά συστήματα ή στοιχεία του UI.

2.4.2.3 Interface Adapters

Τα interfaces είναι υπεύθυνοι για τη μετατροπή των μορφών δεδομένων μεταξύ του Domain layer και των εξωτερικών συστημάτων (π.χ. UI ή βάσεις δεδομένων). Βρίσκονται στο Data layer και χρησιμεύουν ως ενδιάμεσος μεταξύ της clean architecture μεταξύ του domain που σχετίζεται με την εξωτερική αλληλεπίδραση. Εξασφαλίζουν ότι τα δεδομένα που μεταφέρονται από και προς το domain layer είναι σε μορφή που μπορεί να κατανοήσει το domain.

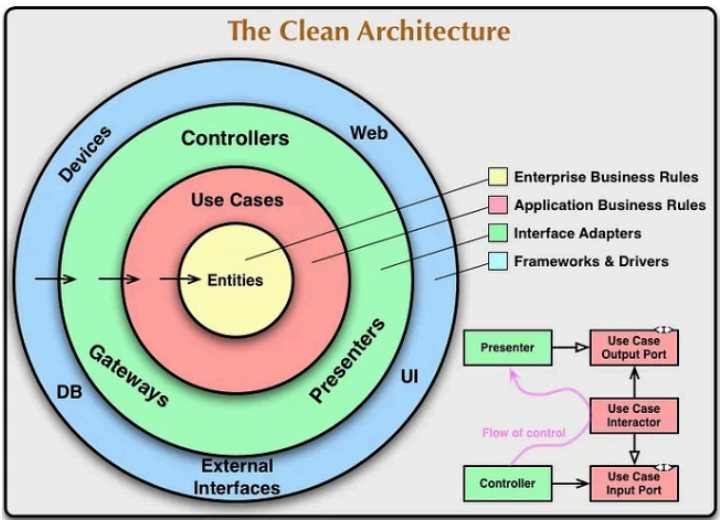
Για παράδειγμα, τα δεδομένα που ανακτώνται από ένα απομακρυσμένο API μπορούν να μετατραπούν σε domain items μέσω μιας κλάσης αντικειμένου (data class) ή μέσω ενός mapper, ο mapper μετατρέπει τα remote δεδομένα σε domain δεδομένα. Αυτός ο διαχωρισμός διασφαλίζει ότι οι αλλαγές στα εξωτερικά συστήματα, όπως ένα νέο σχήμα βάσης δεδομένων ή μια νέα μορφή API, δεν επηρεάζουν τη βασική επιχειρησιακή λογική.

2.4.2.4 Frameworks

Το Frameworks αντιπροσωπεύει το εξωτερικό επίπεδο της clean architecture. Αυτό το στρώμα περιέχει τις τεχνολογίες και τα εργαλεία που είναι απαραίτητα για τη λειτουργία της εφαρμογής, όπως στοιχεία του Android SDK, βιβλιοθήκες όπως το Retrofit[33] για κλήσεις δικτύου ή η Room[34] για τη διαχείριση τοπικών βάσεων δεδομένων. Αυτό το στρώμα αλληλεπιδρά με το hardware του συστήματος και τα εξωτερικά dependencies.

Για παράδειγμα, τα activities και οι υπηρεσίες σε μια εφαρμογή Android ανήκουν σε αυτό το στρώμα. Το framework είναι υπεύθυνο για την παράδοση των αλληλεπιδράσεων του χρήστη στο domain layer και τη λήψη των απαντήσεων απο αυτό, αλλά δεν θα πρέπει να περιέχει επιχειρησιακή λογική.

Αυτή η δομή διασφαλίζει ότι οι αλλαγές στην τεχνολογία (π.χ. η μετάβαση από το Room σε μια άλλη βιβλιοθήκη βάσεων δεδομένων) δεν διαταράσσουν τα εσωτερικά στρώματα, διατηρώντας την ακεραιότητα της εφαρμογής.



Σχήμα 2.7: Δομή Clean Architecture

2.4.3 MVVM με Clean Architecture

Η ενσωμάτωση του μοντέλου MVVM και των αρχών του clean architecture έχει γίνει μια βασική προσέγγιση για τη σχεδίαση συντηρήσιμων, επεκτάσιμων και ελέγξιμων συστημάτων λογισμικού. Τα δύο αυτά μοντέλα δίνουν έμφαση στο διαχωρισμό, αλλά ο συνδυασμός τους ενισχύει την

αρθρωτότητα και την προσαρμοστικότητα μιας εφαρμογής, διασφαλίζοντας ότι τόσο η διεπαφή χρήστη (UI) όσο και η βασική επιχειρησιακή λογική παραμένουν αποσυνδεδεμένα από εξωτερικά dependencies.

Η Clean Architecture εισάγει ομόκεντρα στρώματα ευθύνης, όπου τα εσώτερα στρώματα ενθυλακώνουν τους βασικούς επιχειρηματικούς κανόνες και είναι απομονωμένα από τα εξωτερικά στρώματα, τα οποία διαχειρίζονται το UI, τα πλαίσια και την υποδομή. Το MVVM συμπληρώνει αυτό με τη διάρθρωση του presentation layer σε σαφή συστατικά: το Μοντέλο, το οποίο αλληλεπιδρά με το domain ή data layer. Το ViewModel, το οποίο χρησιμεύει ως ενδιάμεσος φορέας διαχείρισης της κατάστασης και της έκθεσής της στο view, η οποία είναι αποκλειστικά υπεύθυνη για την απόδοση του UI. Μαζί διασφαλίζουν ότι οι αλλαγές σε ένα επίπεδο δεν επηρεάζουν ολόκληρο το σύστημα, προωθώντας μια πιο ανθεκτική και προσαρμόσιμη βάση κώδικα.

Πρακτικά, η Clean Architecture επιτρέπει στους προγραμματιστές να ορίζουν σαφή όρια μεταξύ των domain, application και infrastructure layers[13]. Το MVVM ταιριάζει άψογα στα εξωτερικά στρώματα, δομώντας τον τρόπο με τον οποίο το UI αλληλεπιδρά με την εφαρμογή και τη λογική του Domain.

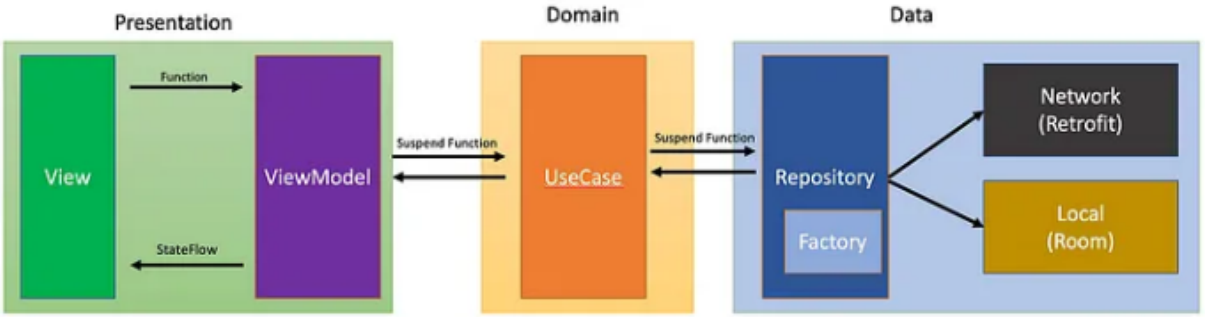
Για παράδειγμα, στην ανάπτυξη Android, το ViewModel αξιοποιεί τα LiveData και τα MutableStates για να προβάλει αλλαγές κατάστασης στο view, επιτρέποντας ενημερώσεις του UI σε πραγματικό χρόνο. Η Clean Architecture διασφαλίζει ότι η επιχειρησιακή λογική, που στεγάζεται στο Domain layer, παραμένει ανεξάρτητη.

Ο συνδυασμός MVVM και Clean Architecture ενισχύει επίσης τη δυνατότητα ελέγχου, Δεδομένου ότι τόσο η λογική του domain όσο και το ViewModel λειτουργούν ανεξάρτητα από το view. Αυτός ο διαχωρισμός διευκολύνει τη συνεχή ολοκλήρωση και δοκιμή, μια κρίσιμη πτυχή της σύγχρονης ανάπτυξης λογισμικού.

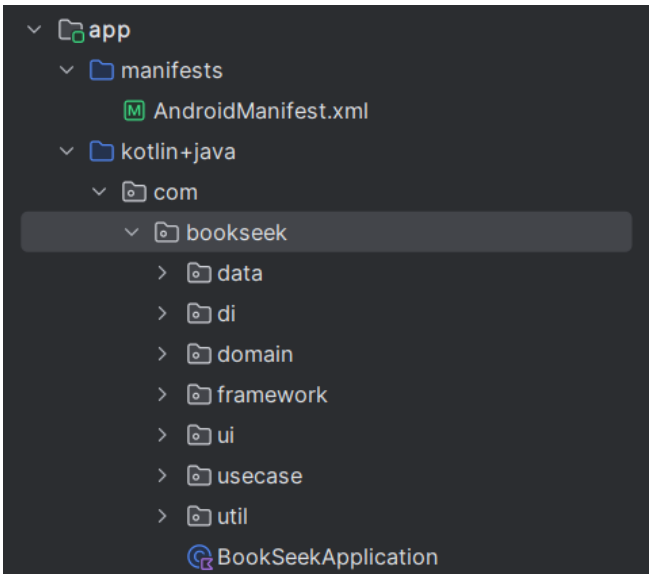
Επιπλέον, η ενοποιημένη προσέγγιση απλοποιεί τη συνεργασία και τη συντήρηση. Για παράδειγμα, ενώ μια ομάδα UI εστιάζει στη βελτίωση του View, μια άλλη ομάδα μπορεί να ασχολείται με την ενίσχυση των επιχειρηματικών κανόνων στο domain layer. Αυτή η αρθρωτότητα μειώνει τον κίνδυνο συγκρούσεων (conflicts) και επιταχύνει τους χρόνους ανάπτυξης.

Παρά τα πλεονεκτήματά της, η ενσωμάτωση του MVVM με την Clean Architecture απαιτεί πειθαρχημένη τήρηση των αρχών σχεδίασης. Η υπερβολική πολυπλοκότητα ή οι περιττές αφαιρέσεις μπορεί να δυσκολέψουν μικρότερα έργα.

Ωστόσο, σε εφαρμογές που απαιτούν επεκτασιμότητα και μακροπρόθεσμη συντήρηση, αυτός ο συνδυασμός παρέχει μια ισχυρή βάση. Τονίζεται πώς οι αρχές της Clean Architecture και η διαχωριστικότητα του MVVM αντιμετωπίζουν κοινά ζητήματα ανάπτυξης εφαρμογών, όπως η πολυπλοκότητα του κώδικα και η δύσκολη συντήρηση. Η σύνθεση του MVVM και της Clean Architecture δημιουργεί ένα ισχυρό πλαίσιο για τη σύγχρονη ανάπτυξη εφαρμογών, χρησιμοποιώντας τη συνδυαστική τους ισχύ[14].



Σχήμα 2.8: MVVM συνδιαστηκά με Clean Architecture



Σχήμα 2.8.1: Android Studio tree MVVM με clean architecture

2.5 Jetpack Compose - UI State

Το Jetpack Compose εισάγει μια σύγχρονη, δηλωτική προσέγγιση για τη δημιουργία διεπαφών χρήστη Android, μεταμορφώνοντας τον τρόπο με τον οποίο οι προγραμματιστές διαχειρίζονται την κατάσταση διεπαφής χρήστη (Ui State)[35].

Μια κρίσιμη πτυχή της λειτουργικότητας του Compose είναι η ικανότητά του να ενσωματώνει απρόσκοπτα μηχανισμούς διαχείρισης κατάστασης όπως το MutableState και τα Flows, προσφέροντας αντιδραστικά και αποτελεσματικά εργαλεία για τη δημιουργία δυναμικών εφαρμογών με απόκριση. Το MutableState είναι μια ειδική κατασκευή του Compose που λειτουργεί ως παρατηρητής του UI State. Επιτρέπει στη διεπαφή χρήστη να ανταποκρίνεται αυτόματα σε αλλαγές κατάστασης ενεργοποιώντας το recompose, διασφαλίζοντας ότι η διεπαφή χρήστη αντικατοπτρίζει τα πιο πρόσφατα δεδομένα.

Το MutableState λειτουργεί καλά για απλά σενάρια διαχείρισης κατάστασης όπου η κατάσταση είναι εντοπισμένη σε ένα συγκεκριμένο στοιχείο διεπαφής χρήστη.

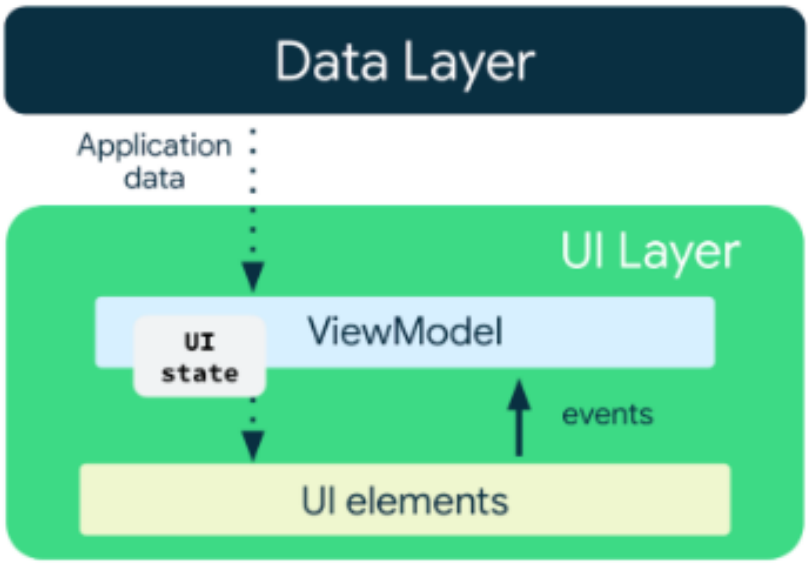
Για παράδειγμα, κατά τη διαχείριση ενός μετρητή σε μια διεπαφή χρήστη, { mutableStateOf(0) } διασφαλίζει ότι η διεπαφή χρήστη ενημερώνεται σε πραγματικό χρόνο καθώς αλλάζει η τιμή. Αυτή η τοπική διαχείριση κατάστασης εξαλείφει την ανάγκη για μη αυτόματες ανανεώσεις, προωθώντας μια πιο διαδραστική και δηλωτική προσέγγιση.

Για πιο σύνθετα και ασύγχρονα σενάρια διαχείρισης κατάστασης, τα flows παρέχουν μια πιο ισχυρή και επεκτάσιμη εναλλακτική λύση, αποτελούν μέρος της βιβλιοθήκης coroutines της Kotlin και έχουν σχεδιαστεί για ασύγχρονο χειρισμό ροών δεδομένων. Όταν ενσωματώνονται στο Compose, τα flows λειτουργούν ως αγωγοί δεδομένων που εκπέμπουν ενημερώσεις κατάστασης, οι οποίες στη συνέχεια μπορούν να συλλεχθούν χρησιμοποιώντας το collectAsState. Αυτή η προσέγγιση είναι ιδιαίτερα επωφελής για το χειρισμό ενημερώσεων σε πραγματικό χρόνο από απομακρυσμένα API, βάσεις δεδομένων ή εισόδους χρηστών.

Για παράδειγμα, μια εφαρμογή που εμφανίζει ενημερώσεις καιρού σε πραγματικό χρόνο μπορεί να χρησιμοποιήσει τα flows για να ανακτά δεδομένα συνεχώς και να αντικατοπτρίζει τις αλλαγές άμεσα στη διεπαφή χρήστη. Ο συνδυασμός του MutableState και του Flow παρέχει ευελιξία για τη διαχείριση της κατάστασης διεπαφής χρήστη σε διαφορετικά επίπεδα. Το MutableState είναι ιδανικό για τοπική κατάσταση, ενώ το Flows υπερέχει στη διαχείριση κατάστασης που προέρχεται από ασύγχρονες ή εξωτερικές πηγές. Η ενσωμάτωση αυτών των εργαλείων στο Compose δημιουργεί ένα ενιαίο παράδειγμα διαχείρισης κατάστασης, διασφαλίζοντας τη συνέπεια και μειώνοντας τον κώδικα. Η αντιδραστικότητα που παρέχεται από το MutableState και το Flows ευθυγραμμίζεται καλά με το δηλωτικό μοντέλο του Compose. Αυτό διασφαλίζει ότι οι αλλαγές κατάστασης διαδίδονται πάντα προβλέψιμα μέσω της ιεραρχίας της διεπαφής χρήστη.

Η έρευνα που παρουσιάστηκε σε πρόσφατες μελέτες IEEE (Narh & Ranjan, 2022) [15] υπογραμμίζει την αποτελεσματικότητα και την επεκτασιμότητα αυτής της προσέγγισης στην ανάπτυξη σύγχρονων εφαρμογών. Οι μελέτες τονίζουν ότι η αξιοποίηση αυτών των εργαλείων διαχείρισης κατάστασης βελτιώνει την αναγνωσιμότητα και τη συντηρησιμότητα του κώδικα, ιδιαίτερα σε εφαρμογές με πολύπλοκες διεπαφές χρήστη που βασίζονται σε δεδομένα. Παρά τα πλεονεκτήματά του, η χρήση αυτών των εργαλείων απαιτεί προσεκτική διαχείριση για την αποφυγή υπερβολικής χρήσης ή περιττα recompose, οι οποίες θα μπορούσαν να επηρεάσουν την απόδοση.

Συμπερασματικά, η ενσωμάτωση των MutableState και Flows στο Jetpack Compose παρέχει ένα ισχυρό πλαίσιο για το Ui State, συνδυάζοντας την απλότητα με την επεκτασιμότητα.



Σχήμα 2.9: Ui State Stages

2.5.1 Compose Navigation

Το Compose Navigation[36] είναι μια βιβλιοθήκη Android που παρέχει έναν ισχυρό και δηλωτικό τρόπο για την υλοποίηση της πλοήγησης σε εφαρμογές που έχουν δημιουργηθεί με το Jetpack Compose. Αποτελεί μέρος της Android Jetpack[37], που έχει σχεδιαστεί ειδικά για να χειρίζεται την πλοήγηση σε περιβάλλοντα χρήστη που βασίζονται σε compose.

Σε αντίθεση με τις παραδοσιακές προσεγγίσεις πλοήγησης που χρησιμοποιούν γραφήματα και navigation fragments σε XML, το Compose Navigation βασίζεται εξ ολοκλήρου στον κώδικα Kotlin. Στον πυρήνα του, το Compose Navigation διαχειρίζεται τη στοίβα πλοήγησης (navigation stack), επιτρέποντας στους προγραμματιστές να ορίζουν συνθέτους προορισμούς και μεταβάσεις μεταξύ τους. Το πλαίσιο απλοποιεί τον χειρισμό εργασιών που σχετίζονται με την πλοήγηση, όπως η μετάδοση δεδομένων μεταξύ οθονών, η διαχείριση καταστάσεων πλοήγησης και η υποστήριξη συνδέσμων σε βάθος. Οι προγραμματιστές ορίζουν τη λογική πλοήγησής τους σε ένα κεντρικό γράφημα πλοήγησης γραμμένο σε Kotlin, το οποίο διασφαλίζει μια ασφαλή και δομημένη προσέγγιση για τον τύπο.

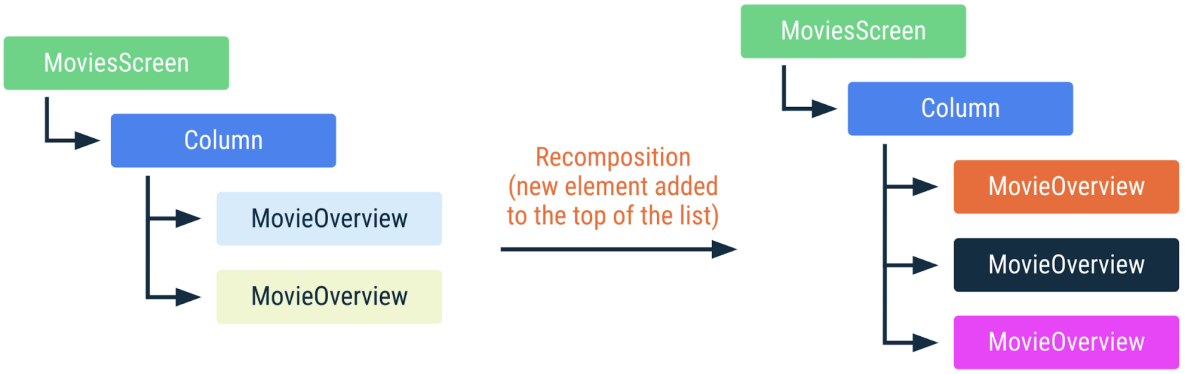
Το Compose Navigation χρησιμοποιεί το NavHost, μια συνθετική συνάρτηση που χρησιμεύει ως κοντέινερ για τη διαχείριση του γραφήματος πλοήγησης. Κάθε προορισμός μέσα στο γράφημα αναπαρίσταται ως συνάρτηση που μπορεί να ενταχθεί στο NavHost. Χρησιμοποιώντας το NavController, ενεργοποιούνται ενέργειες πλοήγησης μέσω προγραμματισμού. Αυτή η δηλωτική δομή προσφέρει ένα σημαντικό πλεονέκτημα έναντι της παραδοσιακής πλοήγησης που βασίζεται σε fragments, μειώνοντας τον κώδικα και εξαλείφοντας τις πολυπλοκότητες του lifecycle που σχετίζονται με τα fragments.

Ένα άλλο βασικό χαρακτηριστικό του Compose Navigation είναι η ικανότητά του να χειρίζεται τη διέλευση ορισμάτων (arguments) και να χρησιμοποιεί safe-type routes. Κατα την πλοήγηση γινεται να

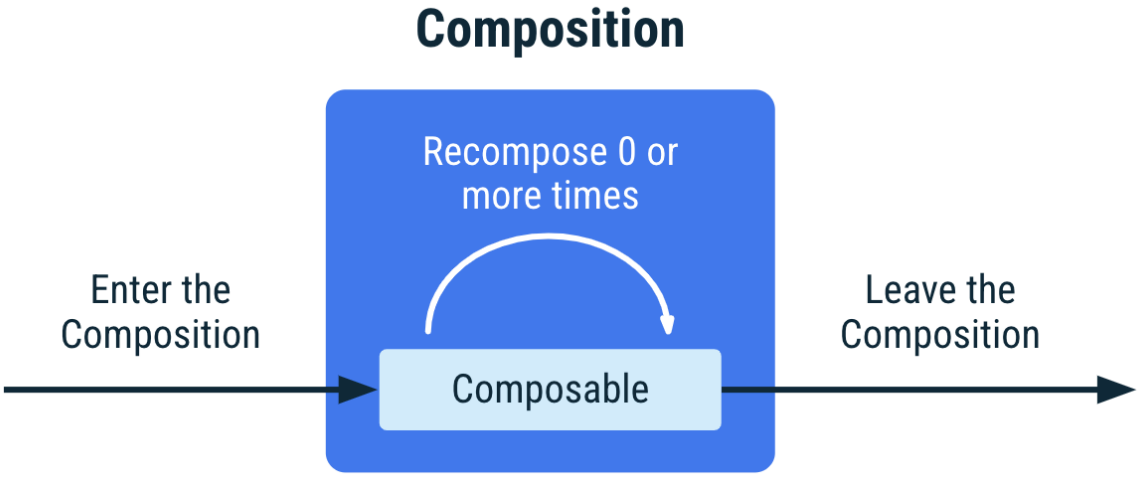
δηλωθούν και arguments ούτως ώστε να χρησιμοποιηθεί στην επόμενη διεπαφή χρήστη που θα είναι ο προορισμός της πλοήγησης, έτσι, διασφαλίζεται η μεταφορά δεδομένων μεταξύ των οθονών.

Επιπλέον, το Compose Navigation υποστηρίζει ένθετα γραφήματα πλοήγησης (nested graphs), επιτρέποντας την ανάλυση σύνθετων ροών πλοήγησης σε διαχειρίσιμα υπογραφήματα. Το Compose Navigation ενσωματώνεται με άλλα στοιχεία Jetpack, όπως το ViewModel, το LiveData και τα MutableStates επιτρέποντας τη διαχείριση κατάστασης σε όλους τους προορισμούς. Η υποστήριξή του σε deeplinks[38] επιτρέπει στους χρήστες να πλοηγούνται απευθείας σε συγκεκριμένες οθόνες από εξωτερικές πηγές, όπως ειδοποιήσεις ή URL. Η δηλωτική φύση του Compose Navigation ευθυγραμμίζεται καλά με τις σύγχρονες πρακτικές ανάπτυξης εφαρμογών, παρέχοντας έναν διαισθητικό τρόπο χειρισμού της πλοήγησης σε εφαρμογές που βασίζονται στη Σύνθεση.

Έρευνες, όπως οι Lin et al. (2024)[17], υπογραμμίζει την αποτελεσματικότητά του στη διαχείριση των ροών πλοήγησης σε δυναμικές αρχιτεκτονικές διεπαφής χρήστη, τονίζοντας τη συμβολή του στη μείωση της πολυπλοκότητας και στη βελτίωση της δυνατότητας συντήρησης.



Σχήμα 2.9.1: Lifecycle and recompositions



Σχήμα 2.9.2: Lifecycle and recompositions

2.6 Dependency Injection

Το Dependency Injection (DI) [39] είναι μια αρχή σχεδιασμού λογισμικού που αντιμετωπίζει την πρόκληση της διαχείρισης dependency μεταξύ στοιχείων σε μια εφαρμογή. Είναι μια μορφή Αντιστροφής Ελέγχου (IoC) [40] όπου τα αντικείμενα δεν δημιουργούν άμεσα dependency. Αντίθετα, παρέχονται dependency σε αντικείμενα από ένα εξωτερικό σύστημα, επιτρέποντας καλύτερη modularity, δυνατότητα testing και συντήρηση των συστημάτων λογισμικού.

Στην ανάπτυξη Android, το DI είναι ιδιαίτερα χρήσιμο επειδή οι σύγχρονες εφαρμογές συχνά περιλαμβάνουν περίπλοκες αλληλεπιδράσεις μεταξύ διαφορετικών επιπέδων, όπως το UI, η επιχειρηματική λογική και η πρόσβαση σε δεδομένα. Δημοφιλή πλαίσια όπως το Dagger, το Hilt και το Koin εφαρμόζουν το DI στο Android για να βελτιστοποιήσουν την εισαγωγή dependency σε μια εφαρμογή. Το DI βοηθά στη διαχείριση αυτών των dependency ορίζοντας τες κεντρικά, αποφεύγοντας τη στενή σύζευξη και κάνοντας τον κώδικα πιο επεκτάσιμο.

Στον πυρήνα του, το DI περιλαμβάνει τρία στοιχεία: το αντικείμενο που απαιτεί dependency, τα ιδια τα dependency και το injection, ο οποίος είναι υπεύθυνος για την παροχή των απαιτούμενων dependency.

Στο Android, τα DI εισάγουν αυτόματα dependency σε activities, fragments, ViewModels ή υπηρεσίες. Αυτό εξαλείφει τον κώδικα boilerplate[41] για τη δημιουργία και τη διαχείριση dependency με μη αυτόματο τρόπο, μειώνοντας τον κίνδυνο σφαλμάτων και βελτιώνοντας την παραγωγικότητα.

- Βελτιωμένη δυνατότητα δοκιμή (testing): Το DI διευκολύνει τη δοκιμή της μονάδας επιτρέποντας την εισαγωγή ψευδών dependency στα components. Για παράδειγμα, αντί να δοκιμάζουν ένα repository απευθείας με μια ζωντανή βάση δεδομένων, γίνετε να μπει μια dummy βάση δεδομένων με ψεύτικα δεδομένα, κάνοντας τις δοκιμές πιο γρήγορες και απομονωμένες από εξωτερικά συστήματα.
- Μειωμένη σύζευξη: Βασιζόμενη σε αφαιρέσεις και παρέχοντας dependency εξωτερικά, τα στοιχεία παραμένουν χαλαρά συνδεδεμένα. Αυτό διευκολύνει την τροποποίηση ή την αντικατάσταση μεμονωμένων στοιχείων χωρίς να επηρεάζεται ολόκληρη η εφαρμογή.
- Επαναχρησιμοποίηση κώδικα και Modularity: Με το DI, τα στοιχεία μπορούν να επαναχρησιμοποιηθούν σε διαφορετικά περιβάλλοντα. Για παράδειγμα, το ίδιο παράδειγμα πελάτη δικτύου ή βάσης δεδομένων μπορεί να κοινοποιηθεί σε πολλά ViewModels ή δραστηριότητες, μειώνοντας τον πλεονασμό και τη χρήση πόρων.
- Κεντρική διαμόρφωση (Configuration): Η διαχείριση των διαμορφώσεων εξάρτησης γίνεται συνήθως σε ενα module ή ένα αρχείο διαμόρφωσης. Αυτή η συγκέντρωση απλοποιεί τη διαχείριση των dependency και διασφαλίζει συνεπή συμπεριφορά σε όλη την εφαρμογή.

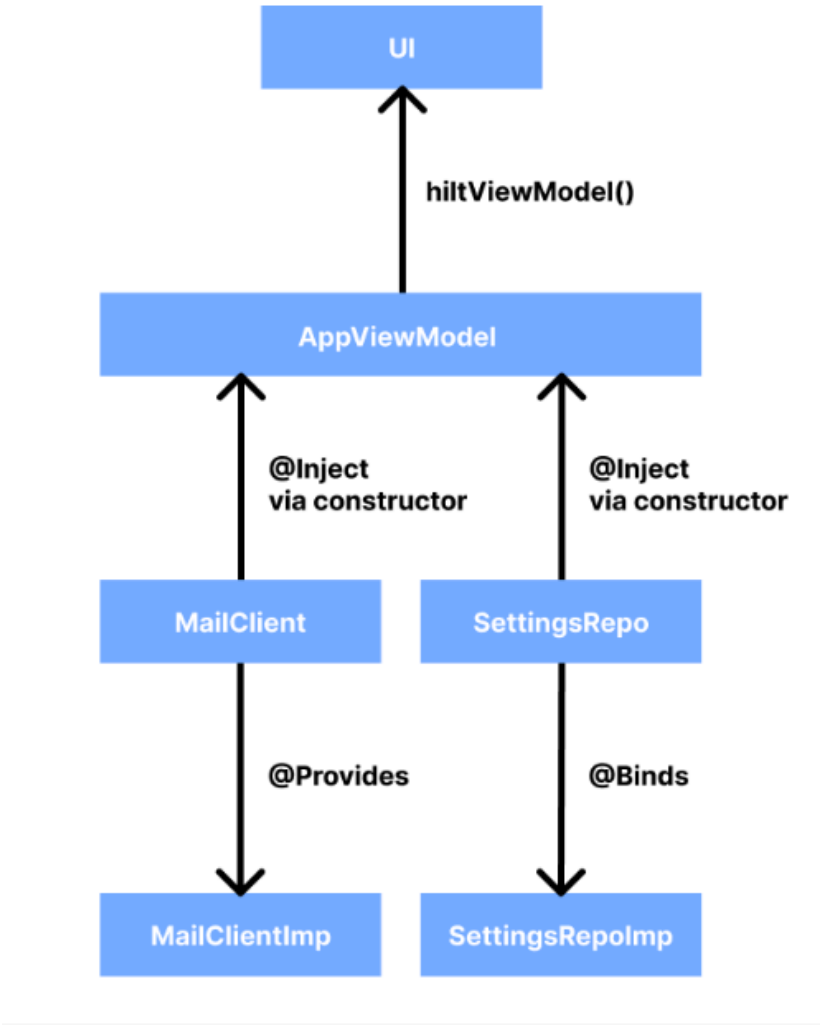
Ενώ το DI προσφέρει σημαντικά πλεονεκτήματα, η υιοθέτησή του μπορεί να δημιουργήσει πολυπλοκότητα, ειδικά σε μικρότερα έργα όπου μπορεί να αρκούν απλούστερες τεχνικές διαχείρισης εξαρτήσεων. Πρέπει να σχεδιαστεί προσεκτικά το γράφημα των dependency για να αποφευχθεί η υπερβολικά περίπλοκη αρχιτεκτονική.

Επιπλέον, η ακατάλληλη χρήση των DI μπορεί να οδηγήσει σε προβλήματα απόδοσης, ιδιαίτερα κατά την εκκίνηση της εφαρμογής, λόγω του χρόνου που απαιτείται για την επίλυση των dependency. Για την αντιμετώπιση αυτών των προκλήσεων, πλαίσια όπως το Hilt βελτιστοποιούν το DI ειδικά για Android.

Το Hilt παρέχει προκαθορισμένα στοιχεία για κοινές κατηγορίες Android, όπως activities, μειώνοντας τον χρόνο και τον boilerplate κώδικα. Έρευνα, όπως αυτή που παρουσιάζεται από τους Cao, C., Meng, C., Ge, H., & Yu, P (2017) [16] στην αξιολόγηση των frameworks DI σε εφαρμογές

Android, υπογραμμίζει πώς αυτά τα frameworks ενισχύουν το modularity και την αξιοπιστία της εφαρμογής.

Το Dependency Injection είναι το μέλλον της σύγχρονης ανάπτυξης εφαρμογών Android, προσφέροντας μια δομημένη προσέγγιση για τη διαχείριση dependency και ενισχύοντας τη συντηρησιμότητα, την επεκτασιμότητα και τη δυνατότητα δοκιμής των συστημάτων λογισμικού. Η ενσωμάτωσή του σε εφαρμογές Android μέσω πλαισίων όπως το Dagger και το Hilt απλοποιεί τη διαδικασία ανάπτυξης και διασφαλίζει καθαρό κώδικα.



Σχήμα 2.10 Dependency injection workflow

2.6.1 Hilt

Το Hilt είναι ένα dependency framework ειδικά σχεδιασμένο για Android, χτισμένο πάνω από στο Dagger[42], μια δημοφιλή βιβλιοθήκη DI. Το Hilt, το οποίο εισήχθη από την Google, απλοποιεί τη διαδικασία ενσωμάτωσης του DI σε εφαρμογές Android. Παρέχει μια βελτιστοποιημένη προσέγγιση για τη διαχείριση dependency σε στοιχεία Android. Το Hilt αυτοματοποιεί μεγάλο μέρος της διαδικασίας DI, προσφέροντας προκαθορισμένα εξαρτήματα που συνδέονται με τον κύκλο ζωής(

lifecycle) του Android.

Αυτά τα στοιχεία περιλαμβάνουν το ActivityComponent, το FragmentComponent και το ViewModelComponent, μεταξύ άλλων, διασφαλίζοντας ότι τα dependency καλύπτονται κατάλληλα με το lifecycle των αντίστοιχων στοιχείων τους. Αυτή η επίγνωση του lifecycle ελαχιστοποιεί τον κίνδυνο διαρροής μνήμης (leak memory)[43] και βελτιστοποιεί τη χρήση των πόρων.

Ένα από τα βασικά πλεονεκτήματα της Hilt είναι η διαμόρφωσή του που βασίζεται σε annotation. τα annotation είναι της μορφής @Inject, @HiltAndroidApp, @Module και βοηθούν στη δήλωση του dependency, εξαλείφοντας την ανάγκη για εκτεταμένο κώδικα boilerplate. Το annotation @HiltAndroidApp εφαρμόζει το Hilt στην κατηγορία εφαρμογής(application level), έτσι το DI έχει πρόσβαση σε ολόκληρη την εφαρμογή. Το Hilt βελτιώνει σημαντικά την παραγωγικότητα μειώνοντας την πολυπλοκότητα της εφαρμογής χάρης στο DI. Ενσωματώνεται άψογα με άλλα στοιχεία του Jetpack, όπως το ViewModels και το Navigation, καθιστώντας το ιδανική επιλογή για σύγχρονη ανάπτυξη Android. Το Hilt έχει μεταμορφώσει το DI στο Android παρέχοντας ένα απλοποιημένο πλαίσιο με επίγνωση του lifecycle.

Η ενσωμάτωσή του στο οικοσύστημα Android προσφέρει καθαρότερη αρχιτεκτονική και βελτιωμένη συντηρησιμότητα, καθιστώντας το ένα πολύτιμο εργαλείο για την δημιουργία επεκτάσιμων και αποτελεσματικών εφαρμογών. Για έργα που χρησιμοποιούν ήδη το Dagger, η μετάβαση στο Hilt προσφέρει μια πιο φιλική και συνεκτική εμπειρία για προγραμματιστές.

2.7 ProGuard - R8

Το ProGuard και το R8[46] είναι εργαλεία που χρησιμοποιούνται στην ανάπτυξη Android για τη βελτιστοποίηση, τη συρρίκνωση και τη απόκρυψη - επισκίαση του κώδικα, βελτιώνοντας τόσο την απόδοση όσο και την ασφάλεια. Ενώ εξυπηρετούν παρόμοιους σκοπούς, διαφέρουν ως προς την εφαρμογή και την αποτελεσματικότητα, με το R8 να είναι το πιο προηγμένο εργαλείο και ο προεπιλεγμένος συρρικνωτής κώδικα για Android από το Android Studio 3.4.

Το ProGuard είναι ένας βελτιστοποιητής και επισκιαστής bytecode Java. Λειτουργεί αφαιρώντας τον αχρησιμοποίητο κώδικα και τους πόρους από μια εφαρμογή, μειώνοντας το μέγεθός της. Επιπλέον, το ProGuard αποκρύπτει τον κώδικα μετονομάζοντας κλάσεις, πεδία και μεθόδους με σύντομα, χωρίς νόημα ονόματα, καθιστώντας την αντίστροφη μηχανική (reverse engineering) [44] πιο δύσκολη.

Η Συρρίκνωση κώδικα Εξαλείφει τον αχρησιμοποίητο κώδικα και τους πόρους για τη μείωση του μεγέθους του APK, η απόκρυψη κώδικα είναι η μετονομασία του κώδικα για να τον κάνει δυσανάγνωστο και η Προεπαλήθευση Προετοιμάζει τον bytecode για εκτέλεση σε παλαιότερες συσκευές.

Το ProGuard ήταν το προεπιλεγμένο εργαλείο στην ανάπτυξη Android για πολλά χρόνια, αλλά οι επιδόσεις και οι δυνατότητες βελτιστοποίησης του είναι περιορισμένες σε σύγκριση με σύγχρονα εργαλεία όπως το R8.

Το R8 είναι μια πιο αποτελεσματική αντικατάσταση του ProGuard, που έχει σχεδιαστεί για να χειρίζεται τις ίδιες εργασίες ενώ ενσωματώνεται απευθείας στη διαδικασία build της εφαρμογής Android. Το R8 συνδυάζει τη συρρίκνωση, τη απόκρυψη σε ένα μόνο βήμα, βελτιώνοντας το build time και την απόδοση της εφαρμογής.

1. Βελτιωμένη απόδοση: Ταχύτερη και πιο αποτελεσματική από το ProGuard.
2. Απρόσκοπτη ενσωμάτωση: Λειτουργεί ως μέρος της προσθήκης Android Gradle χωρίς πρόσθετη ρύθμιση.

3. Καλύτερη βελτιστοποίηση: Χρησιμοποιεί προηγμένους αλγόριθμους για να αφαιρέσει τον αχρησιμοποίητο κώδικα και να μειώσει περαιτέρω το μέγεθος της εφαρμογής[45].

Τα δύο εργαλεία συμβάλλουν στη βελτίωση της απόδοσης της εφαρμογής μειώνοντας το μέγεθος APK, το οποίο είναι ιδιαίτερα σημαντικό για χρήστες με περιορισμένο χώρο αποθήκευσης. Η διαδικασία απόκρυψης παρέχει ένα επίπεδο ασφάλειας, καθιστώντας πιο δύσκολο για τους εισβολείς να απομεταγλωττίσουν και να κατανοήσουν τον κώδικα.

Ωστόσο, ενώ αυτά τα εργαλεία ενισχύουν την ασφάλεια, δεν κάνουν τις εφαρμογές πλήρως ανθεκτικές στην αντίστροφη μηχανική. Το ProGuard και το R8 είναι απαραίτητα εργαλεία για τη βελτιστοποίηση εφαρμογών Android, με το R8 να λειτουργεί πλέον ως η προεπιλεγμένη και πιο προηγμένη επιλογή. Γίνεται η διαμόρφωση αυτών των εργαλείων μέσω κανόνων ProGuard, προσαρμόζοντας το επίπεδο βελτιστοποίησης και απόκρυψης κώδικα στις ανάγκες του κάθε έργου.

2.8 Firebase - Crashlytics

Τα Crashlytics[50] είναι μέρος εργαλείων Firebase της Google, είναι μια ισχυρή λύση αναφοράς σφαλμάτων που έχει σχεδιαστεί για να βοηθά στην παρακολούθηση και διάγνωση προβλημάτων στις εφαρμογές. Παρέχοντας πληροφορίες σε πραγματικό χρόνο για σφάλματα εφαρμογών, επιτρέπει τον γρήγορο εντοπισμό και την επίλυση προβλημάτων, διασφαλίζοντας μια πιο ομαλή εμπειρία χρήστη και διατηρώντας την αξιοπιστία της εφαρμογής.

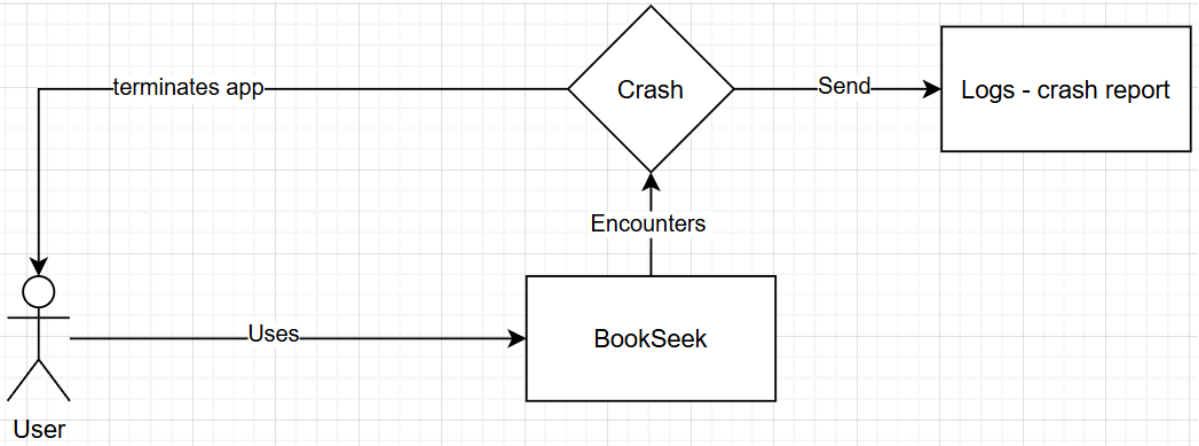
Στον πυρήνα του, το Crashlytics καταγράφει λεπτομερείς πληροφορίες σχετικά με σφάλματα και μη θανατηφόρα σφάλματα που συμβαίνουν σε μια εφαρμογή, θανατηφόρα σφάλματα είναι όταν κλείνει η εφαρμογή. Συλλέγει δεδομένα όπως ίχνη στοίβας, πληροφορίες συσκευής, έκδοση λειτουργικού συστήματος και ενέργειες χρήστη που οδήγησαν στο crash. Αυτό επιτρέπει στους προγραμματιστές να εντοπίσουν τη βασική αιτία των προβλημάτων με ελάχιστη προσπάθεια. Η πλατφόρμα παρέχει επίσης έναν πίνακα εργαλείων με οργανωμένες αναφορές που επισημαίνουν τα πιο κρίσιμα προβλήματα, ταξινομημένα με βάση τη συχνότητα και το αντίκτυπό τους στους χρήστες.

Ένα από τα ξεχωριστά χαρακτηριστικά των Crashlytics είναι οι ειδοποιήσεις σε πραγματικό χρόνο. Όταν εντοπιστεί ένα νέο ζήτημα, το σύστημα μπορεί να ειδοποιήσει μέσω email, Slack[52] ή άλλων ενσωματωμένων εργαλείων επικοινωνίας. Αυτός ο άμεσος βρόχος ανατροφοδότησης διασφαλίζει ότι τα προβλήματα μπορούν να αντιμετωπιστούν έγκαιρα, μειώνοντας τον κίνδυνο δυσαρέσκειας ή εγκατάλειψης των χρηστών.

Τα Crashlytics είναι ιδιαίτερα χρήσιμα επειδή υπερβαίνουν τις αναφορές σφαλμάτων. Προσφέρει επίσης εργαλεία για την παρακολούθηση μη θανατηφόρων σφαλμάτων και την ανάλυση των τάσεων απόδοσης, όπως οι χρόνοι εκκίνησης εφαρμογών ή οι καθυστερήσεις απόδοσης μιας οθόνης. Αυτό το πρόσθετο πλαίσιο βοηθά στον εντοπισμό σημείων συμφόρησης ή περιοχών για βελτίωση της συνολικής απόδοσης της εφαρμογής.

Ένα άλλο σημαντικό πλεονέκτημα είναι η απρόσκοπτη ενσωμάτωσή του με άλλες υπηρεσίες Firebase, όπως το Analytics[53]. Τα Crashlytics ενσωματώνονται εύκολα σε δημοφιλή περιβάλλοντα ανάπτυξης όπως το Android Studio, καθιστώντας το απλό στη ρύθμιση και τη χρήση του. Τα Crashlytics είναι ένα απαραίτητο εργαλείο για τη σύγχρονη ανάπτυξη εφαρμογών, παρέχοντας ολοκληρωμένες αναφορές σφαλμάτων και διαγνωστικά (Diagnostics).

Η ικανότητά του να παρέχει χρήσιμες πληροφορίες σε πραγματικό χρόνο δίνει στους προγραμματιστές τη δυνατότητα να δημιουργούν πιο σταθερές, αξιόπιστες εφαρμογές ενώ αντιμετωπίζουν προληπτικά τις ανησυχίες των χρηστών.



Σχήμα 2.11: Λειτουργικότητα crashlytics

2.9 Επίλογος

Στο κεφάλαιο αυτό είδαμε τις τεχνολογίες, τις αρχές και τη δομή που χρειάζεται η Καθαρή Αρχιτεκτονική, δίνοντας έμφαση στο ρόλο της ως θεμελιώδους σχεδιαστικής προσέγγισης για τη δημιουργία επεκτάσιμων, συντηρήσιμων και προσαρμόσιμων συστημάτων λογισμικού, ιδίως σε εφαρμογές Android.

Οργανώνοντας μια εφαρμογή σε διακριτά επίπεδα – Περιπτώσεις Χρήσης, Προσαρμογείς Διεπαφής και Πλαίσια και Οδηγοί – η αρχιτεκτονική αυτή διασφαλίζει ότι κάθε συστατικό εκπληρώνει έναν συγκεκριμένο σκοπό, ενώ παραμένει ανεξάρτητο από τα υπόλοιπα. Στον πυρήνα αυτής της αρχιτεκτονικής, οι Οντότητες αντιπροσωπεύουν τους βασικούς επιχειρηματικούς κανόνες, ενθυλακώνοντας τη λογική και τα δεδομένα που καθορίζουν τον τομέα. Αυτές παραμένουν σταθερές με την πάροδο του χρόνου και ανεπηρέαστες από αλλαγές σε εξωτερικά συστήματα, παρέχοντας ένα ισχυρό θεμέλιο για την εφαρμογή. Οι Περιπτώσεις Χρήσης ορίζουν ροές εργασίας που συντονίζουν τις αλληλεπιδράσεις μεταξύ των επιπέδων, επιβάλλοντας τους ειδικούς για την εφαρμογή κανόνες και διευκολύνοντας σαφείς και επαναχρησιμοποιήσιμες υλοποιήσεις της βασικής λειτουργικότητας. Το στρώμα Interface Adapters εξασφαλίζει την ομαλή ροή δεδομένων μεταξύ του τομέα της εφαρμογής και εξωτερικών συστημάτων, όπως APIs ή βάσεις δεδομένων. Χειριζόμενο τη μετατροπή των δεδομένων σε μορφές κατάλληλες για τον τομέα, το στρώμα αυτό θωρακίζει την κεντρική λογική από τις εξωτερικές πολυπλοκότητες. Το εξωτερικό στρώμα, Πλαίσια και Οδηγοί, συνδέει το σύστημα με εξωτερικά εργαλεία και τεχνολογίες, όπως το Android SDK ή βιβλιοθήκες τρίτων, επιτρέποντας την αλληλεπίδραση με το περιβάλλον, διατηρώντας παράλληλα την ακεραιότητα των εσωτερικών στρωμάτων.

Η Καθαρή Αρχιτεκτονική προσφέρει μια δομημένη προσέγγιση που προωθεί την αρθρωτότητα και τον σαφή διαχωρισμό των προβλημάτων. Αυτός ο διαχωρισμός διασφαλίζει ότι οι αλλαγές σε ένα επίπεδο δεν επηρεάζουν άλλα επίπεδα, καθιστώντας το σύστημα ανθεκτικό στις τεχνολογικές αλλαγές και τις εξελισσόμενες απαιτήσεις. Ιδιαίτερα στο πλαίσιο των εφαρμογών Android, η προσέγγιση αυτή

παρέχει ένα πλαίσιο που εξισορροπεί την ευρωστία με την ευελιξία, διασφαλίζοντας ότι οι εφαρμογές παραμένουν αποδοτικές και προσαρμόσιμες με την πάροδο του χρόνου.

Κεφάλαιο 3ο: Ανάπτυξη Βάσης Δεδομένων

3.1 Εισαγωγή

Η αρχιτεκτονική back-end μιας σύγχρονης εφαρμογής παίζει καθοριστικό ρόλο στη διασφάλιση της διαχείρισης δεδομένων, της επεκτασιμότητας και της απρόσκοπτης εμπειρίας των χρηστών. Αυτό το κεφάλαιο επικεντρώνεται στον σχεδιασμό και την υλοποίηση του back-end της εφαρμογής, με ιδιαίτερη έμφαση στο Firebase Firestore, μια ισχυρή βάση δεδομένων NoSQL που βασίζεται στο cloud και παρέχεται από την Google.

Αξιοποιώντας το Firestore, η εφαρμογή επιτυγχάνει αποτελεσματική αποθήκευση δεδομένων, συγχρονισμό σε πραγματικό χρόνο και ασφαλή επικοινωνία μεταξύ του πελάτη και του διακομιστή. Σε αυτό το κεφάλαιο, θα διερευνήσουμε τον τρόπο με τον οποίο το Firestore ενσωματώνεται στην εφαρμογή για να χρησιμεύσει ως το κύριο αποθετήριο δεδομένων. Συζητείται ο σχεδιασμός και η δομή της βάσης δεδομένων προσανατολισμένης στα έγγραφα του Firestore, συμπεριλαμβανομένου του μοντέλου συλλογών και εγγράφων, το οποίο παρέχει ευελιξία και επεκτασιμότητα για διάφορους τύπους δεδομένων.

Το κεφάλαιο εξετάζει επίσης τους μηχανισμούς που διευκολύνουν τις ενημερώσεις δεδομένων σε πραγματικό χρόνο, όπως οι δυνατότητες συγχρονισμού σε πραγματικό χρόνο της Firebase, οι οποίες είναι κρίσιμες για την παροχή μιας δυναμικής και ευέλικτης εμπειρίας χρήστη.

Επιπλέον, αυτό το κεφάλαιο εξετάζει τις πτυχές της ασφάλειας και της επεκτασιμότητας του Firestore. Χρησιμοποιώντας τους κανόνες ασφαλείας και τις δυνατότητες ελέγχου ταυτότητας χρηστών της Firebase, η εφαρμογή διασφαλίζει ότι η πρόσβαση στα δεδομένα είναι τόσο περιορισμένη όσο και καλά διαχειρίσιμη.

Η ενσωμάτωση του Firestore με άλλες υπηρεσίες της Firebase, όπως οι Functions και Analytics, καταδεικνύει το ρόλο του στην οικοδόμηση μιας στιβαρής και διασυνδεδεμένης αρχιτεκτονικής back-end. Μέσα από λεπτομερείς συζητήσεις και πρακτικές γνώσεις, αυτό το κεφάλαιο περιγράφει τον τρόπο με τον οποίο η υποδομή back-end σχεδιάζεται για να υποστηρίζει τις απαιτήσεις της εφαρμογής, τηρώντας παράλληλα τις σύγχρονες αρχές της απόδοσης, της αξιοπιστίας και της συντηρησιμότητας. Παρέχεται έτσι μια ολοκληρωμένη άποψη της υλοποίησης του back-end, αναδεικνύοντας τη σημασία του για τη δημιουργία μιας απρόσκοπτης εμπειρίας του χρήστη και την υποστήριξη της συνολικής λειτουργικότητας της εφαρμογής.

3.2 Firebase - Firestore

Το Firestore, γνωστό και ως Cloud Firestore, είναι μια ευέλικτη, επεκτάσιμη βάση δεδομένων NoSQL που αναπτύχθηκε από την Google ως μέρος της πλατφόρμας Firebase. Σχεδιασμένο για να απλοποιεί την ανάπτυξη backend, το Firestore προσφέρει ισχυρές δυνατότητες για συγχρονισμό δεδομένων σε πραγματικό χρόνο, δυνατότητες εκτός σύνδεσης και απρόσκοπτη ενσωμάτωση με άλλες υπηρεσίες Firebase και Google Cloud. Χρησιμοποιείται ευρέως στη σύγχρονη ανάπτυξη εφαρμογών για αποθήκευση, συγχρονισμό και αναζήτηση δεδομένων.

Το Firestore λειτουργεί σε μια δομή βάσης δεδομένων προσανατολισμένη στα έγγραφα. Τα δεδομένα αποθηκεύονται σε συλλογές, οι οποίες περιέχουν έγγραφα. Κάθε έγγραφο είναι ένα σύνολο ζευγών κλειδιών-τιμών, που επιτρέπουν τη δομημένη ή μη δομημένη αποθήκευση δεδομένων. Οι συλλογές

και τα έγγραφα είναι δυναμικά, που σημαίνει ότι δεν απαιτούν ένα προκαθορισμένο σχήμα. Αυτή η ευελιξία είναι ιδιαίτερα πολύτιμη για εφαρμογές όπου τα μοντέλα δεδομένων ενδέχεται να εξελιχθούν με την πάροδο του χρόνου.

Σε αντίθεση με τις παραδοσιακές σχεσιακές βάσεις δεδομένων, το Firestore δεν χρησιμοποιεί πίνακες ή σειρές, καθιστώντας το εξαιρετικά προσαρμόσιμο σε διαφορετικές περιπτώσεις χρήσης δεδομένων. Η επικοινωνία μεταξύ του Firestore και μιας εφαρμογής πραγματοποιείται μέσω ενός πρωτοκόλλου συγχρονισμού σε πραγματικό χρόνο. Όταν μια εφαρμογή-πελάτης συνδέεται στο Firestore, διατηρεί μια ανοιχτή σύνδεση, επιτρέποντας την άμεση μετάδοση των αλλαγών στη βάση δεδομένων σε όλους τους συνδεδεμένους πελάτες. Αυτός ο συγχρονισμός σε πραγματικό χρόνο είναι ένα από τα ξεχωριστά χαρακτηριστικά του Firestore, που επιτρέπει στους προγραμματιστές να δημιουργούν συνεργατικές εφαρμογές με ελάχιστο κώδικα υποστήριξης.

Το Firestore υποστηρίζει δύο λειτουργίες: εγγενή λειτουργία και λειτουργία αποθήκευσης δεδομένων. Η εγγενής λειτουργία είναι βελτιστοποιημένη για το Firebase, παρέχοντας συγχρονισμό σε πραγματικό χρόνο, κανόνες ασφαλείας και SDK για κινητά. Η λειτουργία αποθήκευσης δεδομένων, από την άλλη πλευρά, είναι συμβατή με παλαιού τύπου εφαρμογές Google Cloud Datastore και εστιάζει σε μαζική επεξεργασία μεγάλης κλίμακας. Το Firestore χρησιμοποιεί κανόνες ασφαλείας για τον έλεγχο της πρόσβασης στη βάση δεδομένων. Ορίζονται κανόνες σε επίπεδο συλλογής ή εγγράφου, προσδιορίζοντας ποιος μπορεί να διαβάζει ή να γράφει δεδομένα βάσει συνθηκών όπως η κατάσταση ελέγχου ταυτότητας ή οι ρόλοι χρήστη. Αυτή η προσέγγιση διασφαλίζει λεπτομερή έλεγχο της πρόσβασης στα δεδομένα και βελτιώνει τη συνολική ασφάλεια της εφαρμογής.

Ένα από τα κύρια πλεονεκτήματα του Firestore είναι η δυνατότητα εκτός σύνδεσης. Το SDK αποθηκεύει δεδομένα τοπικά στη συσκευή, επιτρέποντας στις εφαρμογές να συνεχίσουν να λειτουργούν ακόμη και όταν δεν υπάρχει σύνδεση δικτύου. Οι αλλαγές που γίνονται εκτός σύνδεσης συγχρονίζονται αυτόματα με τον διακομιστή μόλις αποκατασταθεί η σύνδεση. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη για εφαρμογές για κινητές συσκευές, όπου η συνδεσιμότητα μπορεί να είναι αναξιόπιστη. Το Firestore ενσωματώνεται απρόσκοπτα με άλλες υπηρεσίες Firebase, όπως Έλεγχος ταυτότητας, Λειτουργίες και Analytics.

Για παράδειγμα, μπορεί να λειτουργήσει με τον έλεγχο ταυτότητας Firebase για να περιορίσει την πρόσβαση στη βάση δεδομένων με βάση την ταυτότητα χρήστη ή με τις Λειτουργίες Firebase για την ενεργοποίηση κώδικα από την πλευρά του διακομιστή ως απόκριση σε συμβάντα βάσης δεδομένων.

Επιπλέον, το Firestore είναι συμβατό με τις υπηρεσίες Google Cloud, επιτρέποντας περιπτώσεις προηγμένης χρήσης, όπως η μηχανική εκμάθηση και η επεξεργασία μεγάλων δεδομένων. Το σύστημα ερωτημάτων στο Firestore είναι ισχυρό και φιλικό προς το χρήστη. Υποστηρίζει πολύπλοκα ερωτήματα, συμπεριλαμβανομένου του φιλτραρίσματος, της ταξινόμησης και της αλυσίδας, ενώ διατηρεί εξαιρετική απόδοση. Τα ερωτήματα ευρετηριάζονται αυτόματα, διασφαλίζοντας ότι η ανάκτηση δεδομένων παραμένει γρήγορη, ακόμη και όταν μεγαλώνει η βάση δεδομένων. Το Firestore επικοινωνεί με εφαρμογές μέσω SDK για διάφορες πλατφόρμες, συμπεριλαμβανομένων των Android, iOS και web. Αυτά τα SDK παρέχουν μεθόδους αλληλεπίδρασης με τη βάση δεδομένων, όπως προσθήκη, ενημέρωση και διαγραφή εγγράφων, καθώς και ακρόαση για ενημερώσεις σε πραγματικό χρόνο.

Το Firestore υποστηρίζει επίσης REST και gRPC API, καθιστώντας το προσβάσιμο από σχεδόν οποιαδήποτε πλατφόρμα ή γλώσσα προγραμματισμού. Συνοπτικά, το Firestore παρέχει μια ισχυρή και φιλική προς τους προγραμματιστές λύση βάσης δεδομένων για σύγχρονες εφαρμογές. Οι δυνατότητές

του σε πραγματικό χρόνο, η υποστήριξη εκτός σύνδεσης και η απρόσκοπτη ενσωμάτωσή του με άλλες υπηρεσίες Firebase και Google Cloud το καθιστούν ιδανική επιλογή για τη δημιουργία επεκτάσιμων, ασφαλών και αποκριτικών εφαρμογών.

3.3 Room

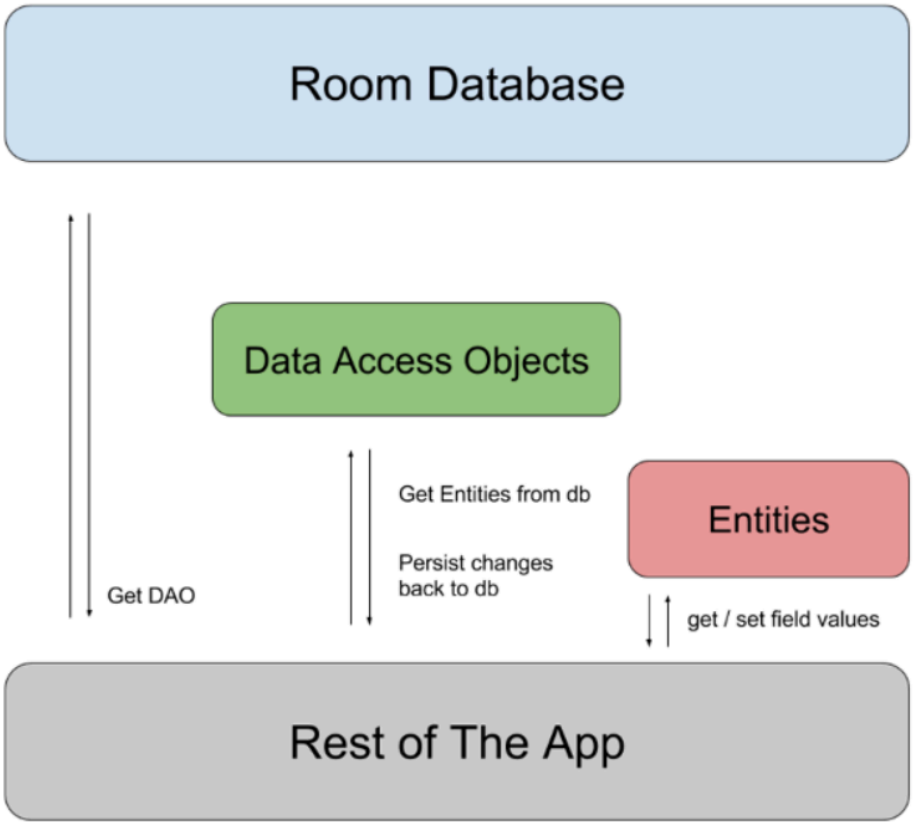
Η Room [34] είναι μια βιβλιοθήκη που παρέχεται από την Google ως μέρος του Android Jetpack. Απλοποιεί τη διαδικασία αποθήκευσης, ανάκτησης και διαχείρισης δεδομένων τοπικά σε μια συσκευή Android, λειτουργώντας ως επίπεδο αφαίρεσης πάνω από την SQLite. Ενώ η SQLite είναι μια ισχυρή βάση δεδομένων, συχνά απαιτεί εκτεταμένο boilerplate κώδικα και χειροκίνητη διαχείριση queries.

Η Room εξαλείφει μεγάλο μέρος αυτής της πολυπλοκότητας, παρέχοντας μια πιο φιλική προς τον προγραμματιστή διεπαφή, ενώ παράλληλα εξασφαλίζει αξιόπιστη απόδοση. Λειτουργεί μέσω τριών κύριων συστατικών: της κλάσης βάσης δεδομένων, των αντικειμένων πρόσβασης σε δεδομένα (DAOs) και των κλάσεων οντοτήτων.

Οι κλάσεις οντοτήτων αναπαριστούν τους πίνακες της βάσης δεδομένων, τα DAOs χειρίζονται τα queries και η κλάση βάσης δεδομένων παρέχει το σημείο σύνδεσης με την ίδια τη βάση δεδομένων. Αυτά τα συστατικά συνεργάζονται απρόσκοπτα, εξασφαλίζοντας τη συνοχή των δεδομένων και μειώνοντας τον κίνδυνο σφαλμάτων.

Ένα από τα αξιοσημείωτα χαρακτηριστικά της Room είναι η ενσωμάτωσή του με τις Kotlin Coroutines και MutableStates, επιτρέποντας ροές δεδομένων και ασύγχρονα queries. Αυτό διασφαλίζει ότι οι λειτουργίες της βάσης δεδομένων δεν μπλοκάρουν το main thread, διατηρώντας μια ομαλή εμπειρία χρήσης ακόμη και κατά το χειρισμό μεγάλων συνόλων δεδομένων. Η Room υποστηρίζει επίσης την επαλήθευση queries κατά τη μεταγλώττιση, η οποία εντοπίζει τα σφάλματα νωρίς στη διαδικασία ανάπτυξης.

Με την Room, η διαχείριση τοπικών δεδομένων γίνεται πιο αποτελεσματική και λιγότερο επιρρεπής σε σφάλματα. Ο συνδυασμός της απλότητας, της απόδοσης και της ενσωμάτωσης με τις σύγχρονες βιβλιοθήκες Android το καθιστά προτιμώμενη επιλογή για την αποθήκευση τοπικών δεδομένων σε εφαρμογές Android.



Σχήμα 3.1: Room workflow

3.4 Υλοποίηση βάσης στο app

Για την βάση δεδομένων μας χρησιμοποιούμε την firestore της google σε συνδυασμό με ενα python script το οποίο φορτώνει ενα pdf βιβλίο στην βάση δεδομένων μας.

Αρχικά το pdf αρχείο του βιβλίου θα πρέπει να υποστεί μια τροποποίηση ούτως ώστε το script να μπορέσει να ομαδοποίηση σωστά τα δεδομένα που έχουμε από το βιβλίο Χωρίζει το pdf σε σελίδες και κεφάλαια, όπως και αναγνωρίζει πότε αλλάζει σελίδα. Αυτό μας βοηθάει να έχουμε καθαρά δεδομένα για να μπορέσουμε να τα προβάλουμε στην εφαρμογή μας. Οταν ο χρήστης κάνει αναζήτηση κάποια λέξη ή φράση το app προσπαθεί να βρει αυτήν την λέξει και να εμφανίσει σε ποιά σελίδα και κεφάλαιο βρίσκεται αυτο που ψάχνει.

Η αναζήτηση έχει δομηθεί με τέτοιο τρόπο ώστε να δίνει προτεραιότητα στις επικεφαλίδες του βιβλίου και μετά αν δεν βρει κάτι να αναζητεί την λέξη σε ολόκληρη την σελίδα και να εμφανίσει ως αποτέλεσμα την σελίδα και πόσες φορές εμφανίστηκε. Σε περίπτωση που η λέξη βρεθεί σε δύο επικεφαλίδες, τότε θα εμφανίσει πρώτα αυτή που περιέχει περισσότερες φορές τη λέξη που ψάχνουμε. Τα αποτελέσματα εμφανίζονται μέχρι 5 καθώς παραπάνω απο αυτο το νούμερο δεν θα είχε νόημα να ψάξει κανείς.

Στην Firestore Αποθηκεύουμε 2 πεδία:

1. Το Βιβλίο όπου απαρτίζεται από το περιεχόμενο εκείνης της σελίδας και το νούμερο της σελίδας.

2. το όνομα του συγγραφέα, ενα εξωφύλλου του βιβλίου, εάν αυτο επιθυμεί ο εκδότης, και ο τίτλος του βιβλίου, η εικόνα του βιβλίου αποθηκεύεται μέσα στην firebase storage και με ενα link το προβάλουμε στο app.

Η συλλογή των δεδομένων γίνεται με ενα API call στην firestore και μας παρέχει τα δεδομένα από την βάση μας. Μόλις ολοκληρωθεί η διαδικασία περισυλλογής των δεδομένων η εφαρμογή στέλνει στην τοπική βάση δεδομένων Room το όνομα του βιβλίου ούτως ώστε να μην χρειάζεται ο χρήστης να σκανάρει το QR code που βρίσκεται στο βιβλίο και να του φέρνει τα δεδομένα απευθείας.

3.5 Επίλογος

Σε αυτό το κεφάλαιο εξετάστηκε λεπτομερώς ο ρόλος της αρχιτεκτονικής back-end και των βάσεων δεδομένων στην υποστήριξη μιας σύγχρονης εφαρμογής Android. Η χρήση του Firebase Firestore σε συνδυασμό με τη βιβλιοθήκη Room ανέδειξε την αποτελεσματικότητα της συνδυασμένης αξιοποίησης cloud-based και τοπικών βάσεων δεδομένων.

Το Firestore παρέχει μια ευέλικτη και επεκτάσιμη πλατφόρμα αποθήκευσης δεδομένων, επιτρέποντας την εύκολη διαχείριση τον συγχρονισμό πληροφοριών σε πραγματικό χρόνο, ενώ η τοπική βάση Room διασφάλισε την απρόσκοπτη εμπειρία του χρήστη ακόμα και σε συνθήκες περιορισμένης συνδεσιμότητας. Η διαδικασία αποθήκευσης, ανάκτησης και εμφάνισης δεδομένων περιγράφηκε με τρόπο που ανέδειξε τη λειτουργικότητα και την αποδοτικότητα του συστήματος. Από την οργάνωση του περιεχομένου του βιβλίου σε συλλογές και έγγραφα στο Firestore μέχρι τη χρήση του Room για την τοπική αποθήκευση του ονόματος του βιβλίου, κάθε τεχνολογική απόφαση εξυπηρέτησε έναν σαφή σκοπό: τη βελτιστοποίηση της απόδοσης και τη βελτίωση της εμπειρίας του χρήστη.

Παράλληλα, η αξιοποίηση ενός Python script για την αποθήκευση δεδομένων από το βιβλίο ενίσχυσε την αυτοματοποίηση και την ακρίβεια της βάσης δεδομένων. Το σύστημα αναζήτησης σχεδιάστηκε με προτεραιότητα στην ευχρηστία, παρέχοντας γρήγορα και σχεσιακά αποτελέσματα στον χρήστη. Η χρήση κριτηρίων, όπως οι επικεφαλίδες και οι σελίδες, ανέδειξε μια έξυπνη προσέγγιση στην ταξινόμηση και προβολή των αποτελεσμάτων.

Εν τέλει, η ενσωμάτωση του Firestore και του Room προσέφερε έναν συνδυασμό κεντρικής και τοπικής αποθήκευσης, καλύπτοντας πλήρως τις ανάγκες της εφαρμογής. Το back-end, όπως σχεδιάστηκε και υλοποιήθηκε, αποτελεί ένα ισχυρό θεμέλιο για τη λειτουργικότητα και την επεκτασιμότητα της εφαρμογής, διασφαλίζοντας την ομαλή εμπειρία χρήστη και τη δυνατότητα μελλοντικών βελτιώσεων.

Κεφάλαιο 4ο: Ανάπτυξη UI και λειτουργία εφαρμογής

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε και θα δούμε τις οθόνες της εφαρμογής όπως επίσης και θα αναλύσουμε την λειτουργικότητα τους. Στο κάτω μέρος της οθόνης βλέπουμε το κεντρικό menu πλοήγησης το bottom navigation που παρέχει την πλοήγηση σε αυτές τις οθόνες. Η πλοήγηση αποτελείται από 3 κύριες οθόνες, την αρχική οθόνη σχήμα στο 4.1-4.2, την αναζήτηση στο σχήμα 4.3-4.4 και τέλος, στο σχήμα 4.4 όπου βρίσκονται οι ρυθμίσεις.

4.2 Λειτουργικότητα και κώδικας εφαρμογής

Η εφαρμογή ξεκινά τη λειτουργία της στο επίπεδο framework(βλ. σχήμα 4.10), όπου ξεκινά η συλλογή δεδομένων από τη Firestore. Σε αυτό το στάδιο, η εφαρμογή συνδέεται με τη βάση δεδομένων για να ανακτήσει τα απαραίτητα remote μοντέλα. Όταν αυτά τα δεδομένα ανακτηθούν με επιτυχία χωρίς σφάλματα, μετακινούμαστε στο επίπεδο data, όπου το repository χειρίζεται την πηγή δεδομένων (data source) που λαμβάνεται από το επίπεδο framework.

Στο επίπεδο data, ένας mapper συνδυάζει διαφορετικές πτυχές δεδομένων, όπως λεπτομέρειες και περιεχόμενο βιβλίου, σε ένα ενοποιημένο μοντέλο σε επίπεδο domain. Αυτή η αντιστοίχιση διασφαλίζει ότι τα δεδομένα μετατρέπονται σε μορφή κατάλληλη για επιχειρηματική λογική(business logic), διατηρώντας παράλληλα την ακεραιότητά τους. Σε αυτό το σημείο, το DI (Dependency Injection), χρησιμοποιώντας τη Hilt, συνδέει το repository με την πηγή δεδομένων.Το Injection επιτρέπει στην εφαρμογή να διαχειρίζεται αποτελεσματικά τις εξαρτήσεις (Dependency), διασφαλίζοντας ότι τα στοιχεία αλληλεπιδρούν απρόσκοπτα χωρίς στενή σύζευξη.

Στη συνέχεια, το επίπεδο domain προετοιμάζει τα δεδομένα για περαιτέρω χρήση, εστιάζοντας στην παροχή καθαρών και επαναχρησιμοποιήσιμων μοντέλων δεδομένων. Μόλις ολοκληρωθεί αυτή η διαδικασία, τα δεδομένα μεταβιβάζονται σε ένα UseCase, το οποίο εξάγει πληροφορίες σε επίπεδο domain και τις προετοιμάζει για τη διεπαφή χρήστη.

Τέλος, η διαδικασία φτάνει στο επίπεδο UI, όπου το UseCase κάνει inject σε ένα ViewModel. Το ViewModel εφαρμόζει έναν ακόμη mapper που ονομάζεται UiMapper, για να μετατρέψει τα δεδομένα σε επίπεδο domain σε μια μορφή UiModel. Τα αντιστοιχισμένα δεδομένα παραδίδονται στη συνέχεια στα στοιχεία διεπαφής χρήστη, διασφαλίζοντας ότι η διεπαφή χρήστη αποκρίνεται.

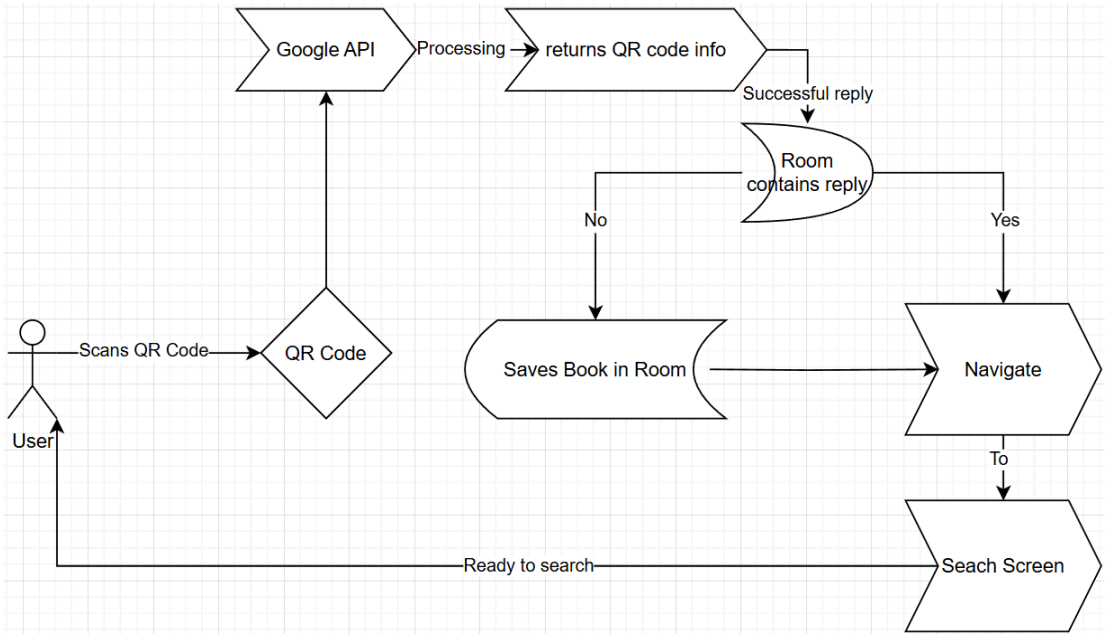
Αυτή η πολυεπίπεδη αρχιτεκτονική, που τηρεί τις αρχές της Καθαρής Αρχιτεκτονικής, διασφαλίζει ότι η εφαρμογή είναι αρθρωτή, διατηρήσιμη και επεκτάσιμη, παρέχοντας ομαλή ροή από τη συλλογή δεδομένων έως την παρουσίαση του χρήστη.

4.3 Προσθήκη βιβλίων

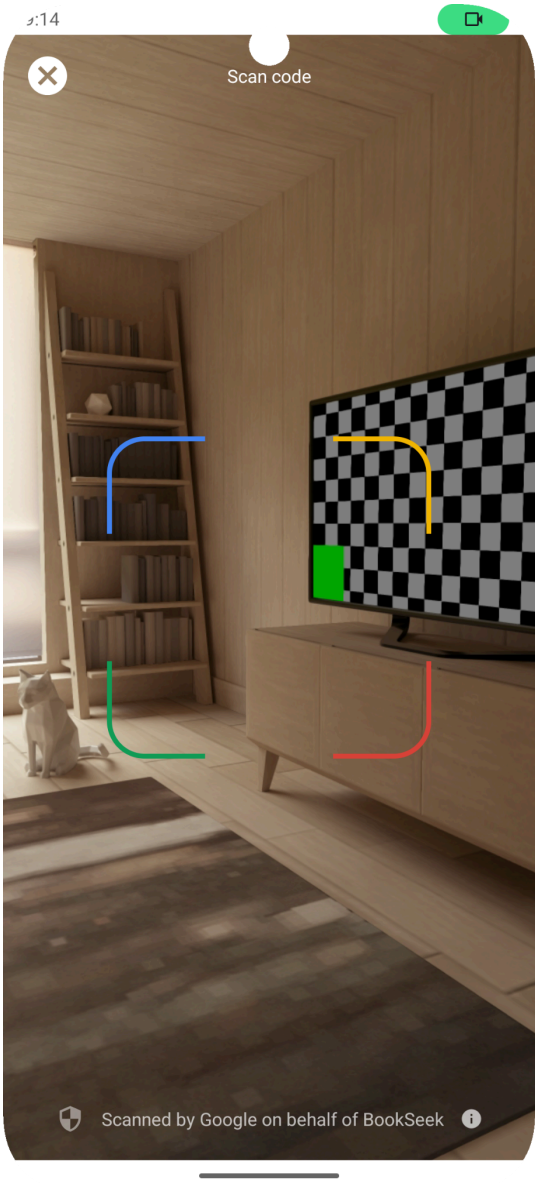
Η Προσθήκη βιβλίων μέσα στην εφαρμογή γίνεται με τον εξής τρόπο:

1. Ο χρήστης εντοπίζει το QR Code που βρίσκεται στο βιβλίο.
2. Ο χρηστης ανοίγει το QR Code scanner [47], βλ. σχήμα 4.2.1, που υπάρχει σε όλες τις βασικές σελίδες και βρίσκεται κάτω δεξιά με το σύμβολο "+". Ανοίγοντας αυτήν την οθόνη ανοίγει η κάμερα χωρίς να χρειάζεται άδεια κάμερας, καθώς είναι ένα API call της google όπου η ανάγνωση του QR Code γίνετε διασφαλίζοντας όλα τα προσωπικά δεδομένα και επιστρέφει τα δεδομένα που περιέχει το QR Code.
3. Όταν ολοκληρωθεί η ανάγνωση του QR Code η εφαρμογή αναζητεί στην βάση δεδομένων το βιβλίο το οποίο σκάναρε ο χρήστης.
4. Ανοίγει η σελίδα search details, βλ. σχήμα 4.5, όπου ο χρήστης κάνει αναζήτηση την λέξη - φράση που θέλει να αναζητήσει στο βιβλίο.

Παρακάτω,βλ. σχήμα 4.1, είναι ένα ER διάγραμμα για το πως λειτουργεί η διαδικασία αποθήκευσης και διαχείρισης του QR Code.



Σχήμα 4.1: Λειτουργικότητα QR scanner - προσθήκη βιβλίου, ER Diagram



Σχήμα 4.2.1: QR code Scanner απο emulator

4.4 Αρχική οθόνη

Στα παρακάτω σχήματα, βλ. 4.2 και 4.3, βλέπουμε την αρχική οθόνη σε 2 διαφορετικές καταστάσεις. Στην πρώτη κατάσταση, βλ. σχήμα 4.2, βλέπουμε την αρχική οθόνη όπου ο χρήστης δεν έχει προσθέσει κάποιο βιβλίο οπότε εμφανίζει το μήνυμα πως δεν βρέθηκαν βιβλία.

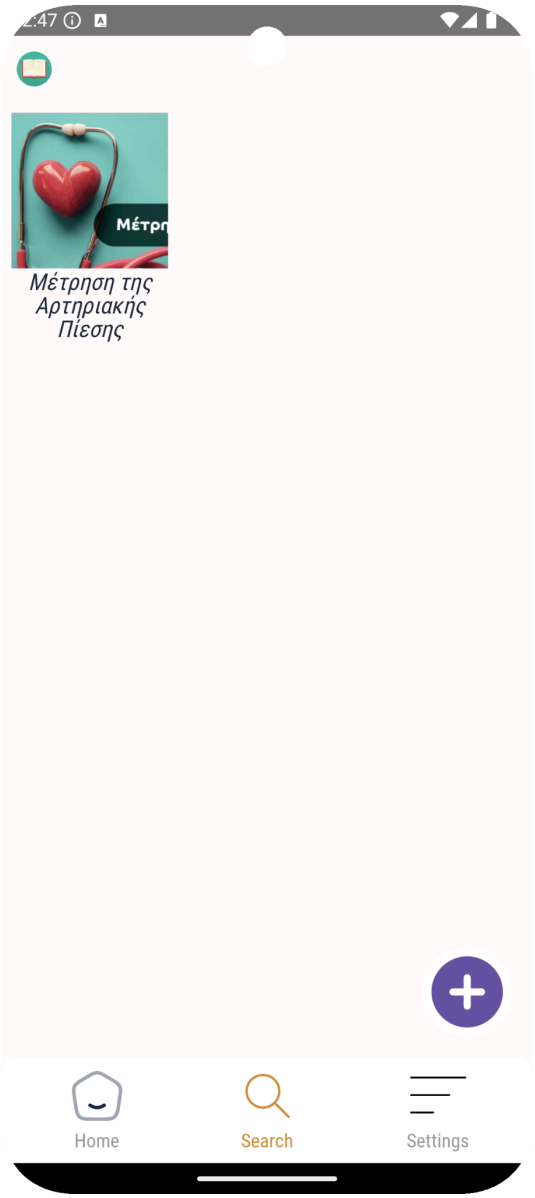
Αυτό που βλέπουμε στην κατάσταση 2, σχήμα 4.3, είναι μια λίστα από τα βιβλία που ο χρήστης έχει σκανάρει. ούτως ώστε να υπάρχουν για να μην χρειάζεται να σκανάρει κάθε φορά που θέλει να ψάξει κάτι. Όταν το βιβλίο μπει στην εφαρμογή η πληροφορία απο το QR Code αποθηκεύεται τοπικά στην βάση δεδομένων Room. Η εμφάνιση αυτών στην αρχική και στην αναζήτηση γίνεται παίρνοντας δεδομένα από την room και κάνοντας ένα API call στην firebase για κάθε εγγραφή που υπάρχει σε αυτή, και αυτή με την σειρά της φέρνει το βιβλίο με τα περιεχόμενα του. Επίσης, ο χρήστης μπορεί να δει ποίος είναι ο συγγραφέας του βιβλίου καθώς και την εικόνα του.

Σχήμα 4.2: Αρχική οθόνη χωρίς βιβλίο

Σχήμα 4.3: Αρχική οθόνη με βιβλίο

4.5 Αναζήτηση

Στην οθόνη αναζήτησης βλέπουμε τη λίστα των βιβλίων που εχει σκανάρει ο χρήστης. Για να μπορέσει να ξεκινήσει η αναζήτηση θα πρέπει ο χρήστης να επιλέξει βιβλίο, μόλις επιλέξει βιβλίο εμφανίζεται η οθόνη στο σχήμα 4.7. Εδώ βλέπουμε μόνο την εικόνα του βιβλίου αλλά και τον τίτλο, σε περίπτωση που δεν έχει εικόνα εμφανίζει μια γενική εικόνα βιβλίου.



Σχημα 4.4: Αναζήτηση με βιβλία

Σχημα 4.5: Αναζήτηση

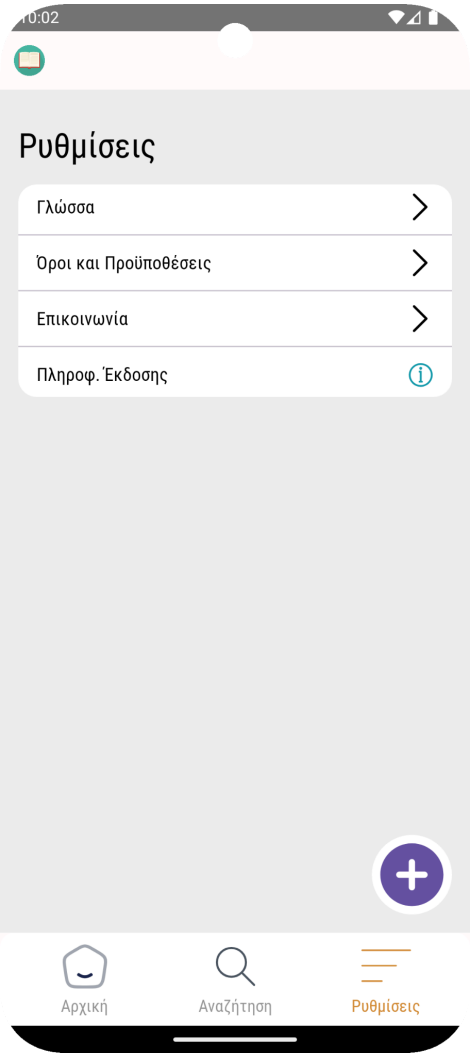
4.6 Ρυθμίσεις

Τελευταία από τις κύριες οθόνες είναι οι ρυθμίσεις της εφαρμογής, βλ. σχήμα 4.6. Απαρτίζεται από τη γλώσσα, τους όρους και προϋποθέσεις, την επικοινωνία και την πληροφορία έκδοσης:

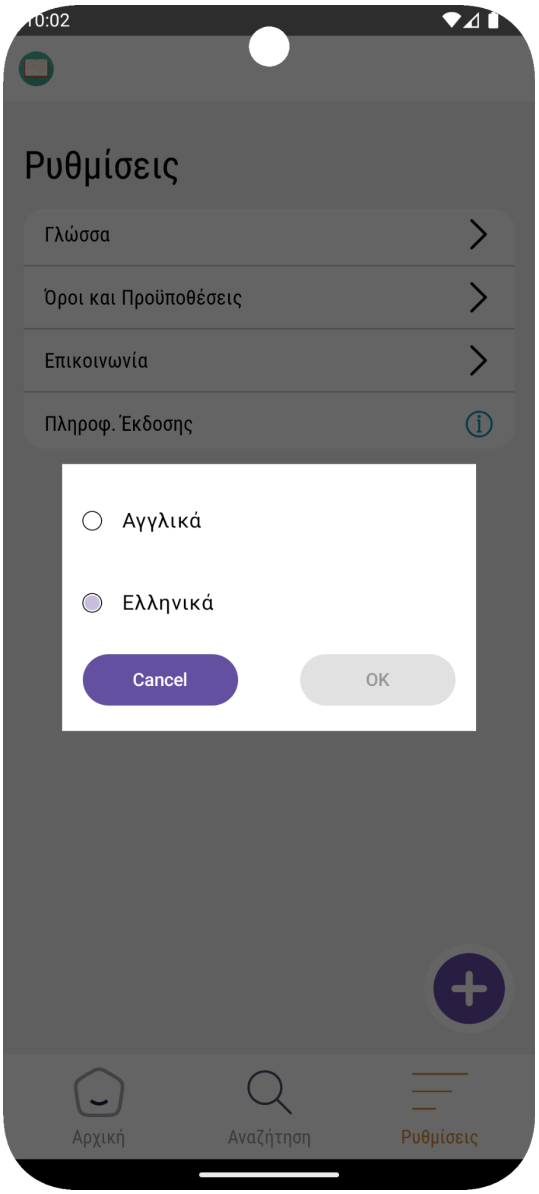
1. Το κουμπί “Γλώσσα” παρέχει τη δυνατότητα στον χρήστη να αλλάξει τη γλώσσα στην οποία η εφαρμογή προβάλλεται, βλ. σχήμα 4.6.1. Έχει τη δυνατότητα να επιλέξει μεταξύ Ελληνικών και Αγγλικών. Η εφαρμογή διαβάζει τη γλώσσα στην οποία είναι η συσκευή, και ως πρώτη εκκίνηση της, θα είναι σε αυτή του συστήματος του κινητού. Εάν επιλεγεί γλώσσα διαφορετική από αυτή που είναι ήδη επιλεγμένη, τότε η εφαρμογή θα κάνει μια γρήγορη επανεκκίνηση και θα προβάλλεται στη αυτή που επέλεξε ο χρήστης. Αυτό μας δίνει τη δυνατότητα να μπορούμε να προσθέσουμε και άλλες γλώσσες για να προσελκύσουμε

περισσότερο κόσμο να δοκιμάσει την εφαρμογή μας. Φυσικά, προϋποθέτει να υπάρχει και το βιβλίο που θα αναζητήσουμε στα αγγλικά ή στη γλώσσα που θα είναι και η εφαρμογή.

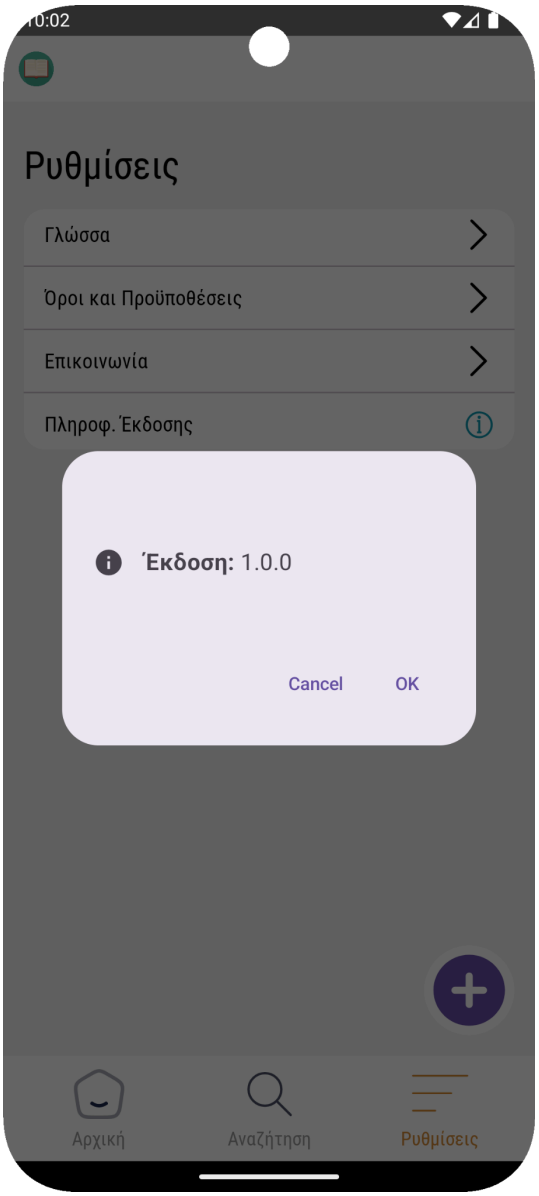
2. το κουμπί "Όροι και Προϋποθέσεις" είναι οι κανόνες και οι όροι για να μπορεί κάποιος να χρησιμοποιεί την εφαρμογή.
3. το κουμπί "Επικοινωνία" δίνει τη δυνατότητα στον χρήστη να μπορεί να στείλει email στον δημιουργό της εφαρμογής. Αυτό είναι πολύ θετικό, καθώς ο χρήστης θα μπορεί να προσφέρει κάποια ανατροφοδότηση για το εάν η εφαρμογή δεν λειτουργεί σωστά ή ότι είναι πολύ καλή, αλλά χρειάζεται και κάποια πράγματα παραπάνω. Έτσι, με βοηθό τον χρήστη, υπάρχει μια μεγάλη βοήθεια στο θέμα της ανάπτυξης - αποσφαλμάτωσης (Debug). Πατώντας το κουμπί, η εφαρμογή ανοίγει κάποια εφαρμογή email, π.χ., Gmail, ο παραλήπτης συμπληρώνεται αυτόματα, καθώς και το θέμα του email, εάν αυτό χρειαστεί για διευκόλυνση του χρήστη.
4. το κουμπί "Πληροφορία έκδοσης", βλ. σχήμα 4.6.2, πατώντας το ανοίγει ένα dialog, όπου μας ενημερώνει για την έκδοση της εφαρμογής. Αυτό μπορεί να μας φανεί χρήσιμο σε συνδυασμό με το email, καθώς ο χρήστης μπορεί να έχει μια παλαιότερη έκδοση της εφαρμογής εγκατεστημένη. Γνωρίζοντας αυτό, ίσως έχουμε διορθώσει το bug σε μια μεταγενέστερη έκδοση και απλά να ενημερώσουμε τον χρήστη να κάνει αναβάθμιση της εφαρμογής.



Σχήμα 4.6 Οθόνη ρυθμίσεων



Σχήμα 4.6.1: Αλλαγή γλώσσας εφαρμογής

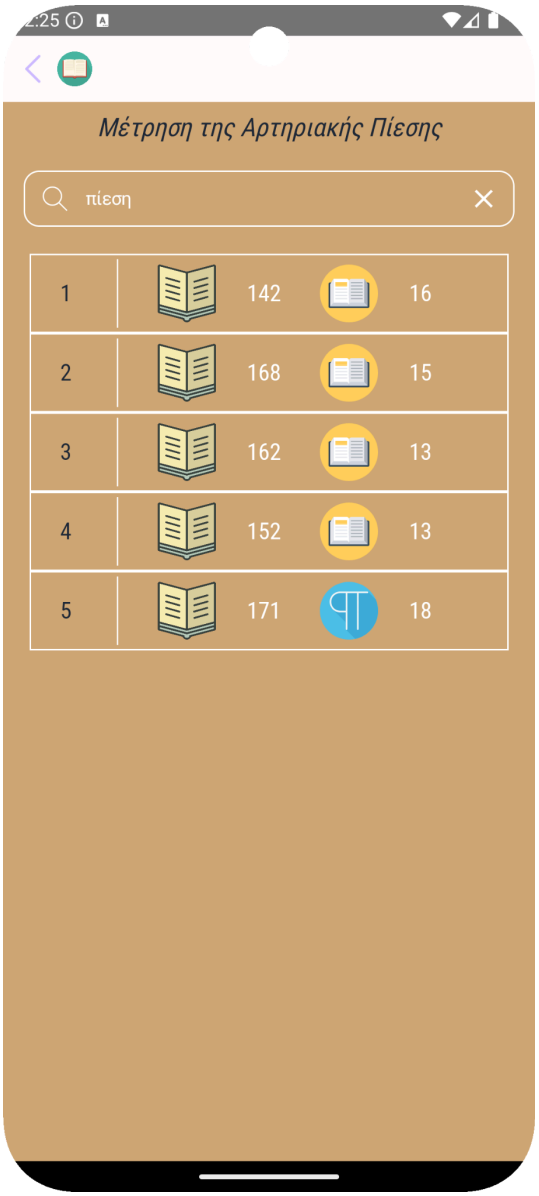


Σχήμα 4.6.1 Εμφάνιση έκδοσης εφαρμογής

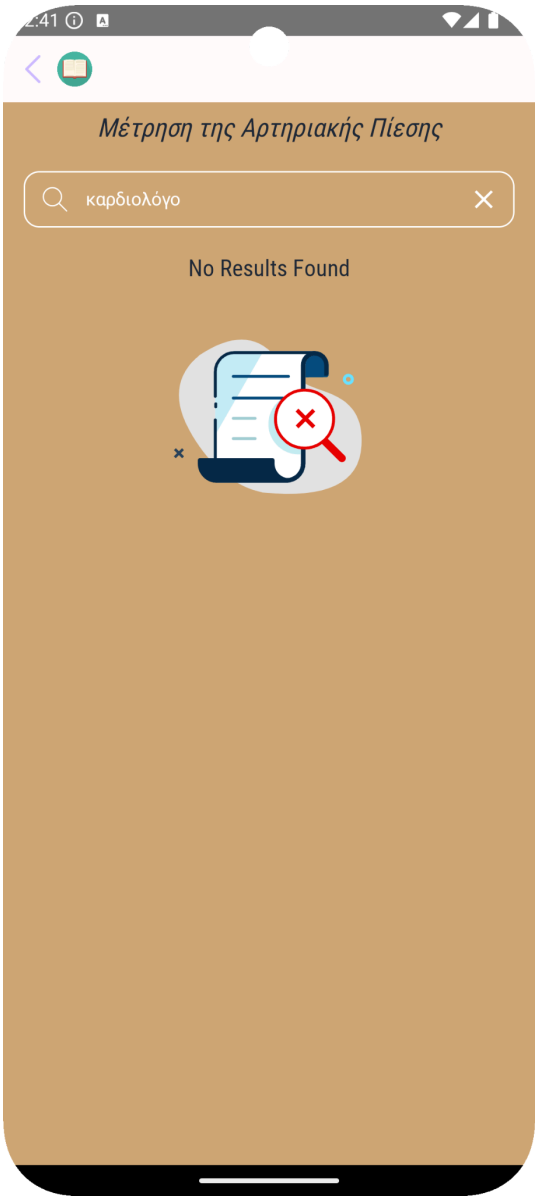
4.7 Λεπτομέρειες αναζήτησης

Στην οθόνη λεπτομέρειες αναζήτησης (search details), βλέπουμε πάνω τον τίτλο του βιβλίου και ένα πεδίο εισαγωγής κειμένου, textfield. Ο χρήστης μπορεί να κάνει αναζήτηση του κειμένου ή της λέξης που υπάρχει και η εφαρμογή θα του εμφανίσει τα αποτελέσματα, εάν αυτά υπάρχουν, με τη μορφή σελίδας όπως και πόσες φορές εμφανίζεται σε αυτή.

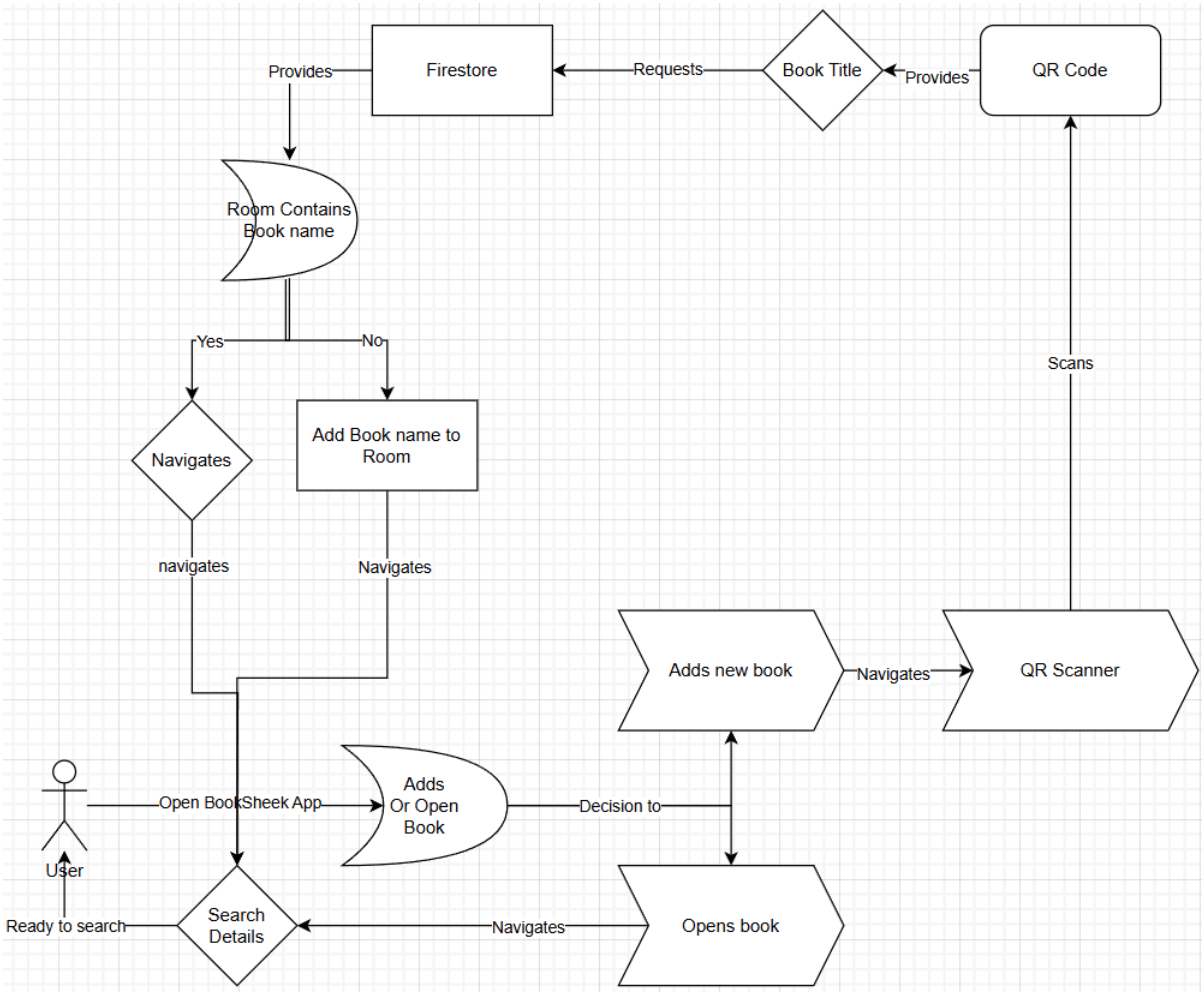
Τα αποτελέσματα έχουν μια ιεραρχία εμφάνισης. Πρώτα, η εφαρμογή αναζητεί τη λέξη ή φράση στις επικεφαλίδες και μετά αναζητεί μέσα στο περιεχόμενο των κειμένων για να βρει αποτελέσματα. Το σύνολο των αποτελεσμάτων μπορεί να είναι έως και πέντε, καθώς δεν θα υπήρχε λόγος για περισσότερα. Στο σχήμα 4.7, βλέπουμε μια αναζήτηση όπου εμφανίζονται αποτελέσματα με επικεφαλίδας αλλά και σελίδας, ενώ στο σχήμα 4.8 βλέπουμε τι μας εμφανίζει εάν δεν βρει κάποιο αποτέλεσμα.



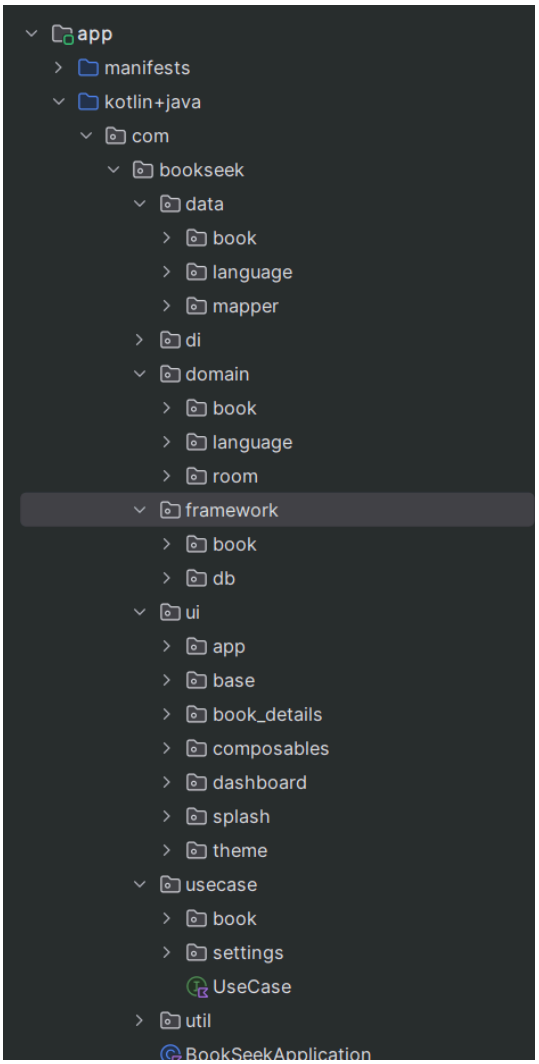
Σχήμα 4.7: Λεπτομέρειες αναζήτησης αποτελέσματα



Σχήμα 4.8: Λεπτομέρειες αναζήτησης χωρίς αποτελέσματα



Σχήμα 4.9: ER Διάγραμμα Λειτουργίας εφαρμογής



Σχήμα 4.10: Component tree

4.8 Επίλογος

Συνοψίζοντας, η εφαρμογή αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας παρέχει μια ολοκληρωμένη εμπειρία διαχείρισης και ανάκτησης περιεχομένου βιβλίων μέσω της χρήσης τεχνολογίας αιχμής, βασικές λειτουργίες, όπως η προσθήκη βιβλίων μέσω σαρωτών κωδικών QR code, η αναζήτηση συγκεκριμένου περιεχομένου και η διαχείριση των προτιμήσεων του χρήστη μέσω ρυθμίσεων, είναι σχεδιασμένες με ευκολία χρήσης και σχεδιασμένη με γνώμονα την αποτελεσματικότητα, η εφαρμογή του σαρωτή κωδικού QR και η διασφάλιση της προστασίας της ιδιωτικότητας καθιστούν την προσθήκη βιβλίων μια ασφαλή και απρόσκοπτη διαδικασία ενσωματώνοντας τοπικές και διαδικτυακές βάσεις δεδομένων, όπως η Room και η Firestore, με τρόπο που μεγιστοποιεί την ταχύτητα και την ακρίβεια και τα δεδομένα μπορούν να αποθηκευτούν και να ανακτηθούν. Η αρχική οθόνη παρέχει μια οπτικά φιλική και λειτουργική προσέγγιση προσαρμοσμένη στις ανάγκες του χρήστη.

Διάφοροι τρόποι προβολής, όπως χωρίς προσθήκη βιβλίων ή με ήδη σαρωμένα βιβλία, παρέχουν μια σαφή και οργανωμένη εμπειρία πλοήγησης. Η λειτουργία αναζήτησης επιτρέπει στους χρήστες να βρίσκουν αποτελεσματικά τις πληροφορίες που χρειάζονται μέσω μιας διαδικασίας που χρησιμοποιεί προηγμένα εργαλεία ταξινόμησης και εμφάνισης αποτελεσμάτων. Τα αποτελέσματα της αναζήτησης εμφανίζονται με ιεραρχικό τρόπο, ώστε να εξασφαλίζεται η καλύτερη δυνατή εμπειρία.

Τέλος, το μενού ρυθμίσεων ενισχύει τη διαδραστικότητα της εφαρμογής και παρέχει τη δυνατότητα αλλαγής της γλώσσας, επικοινωνίας με τον προγραμματιστή και προβολής της έκδοσης της εφαρμογής. Η δυνατότητα προσαρμογής της γλώσσας βελτιώνει την προσβασιμότητα της εφαρμογής σε ένα ευρύτερο κοινό, ενώ οι λειτουργίες επικοινωνίας ενισχύουν τη συνεχή βελτίωση μέσω της ανατροφοδότησης των χρηστών.

Συνολικά, η εφαρμογή ενσωματώνει σύγχρονη αρχιτεκτονική σχεδιασμού και τεχνολογικές λύσεις για να παρέχει μια καινοτόμο και φιλική προς τον χρήστη εμπειρία. Τα χαρακτηριστικά της βελτιώνουν την αναζήτηση περιεχομένου φυσικών βιβλίων και την καθιστούν ένα ισχυρό εργαλείο για τη γεφύρωση παραδοσιακών και ψηφιακών περιβαλλόντων.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Αυτή η πτυχιακή αντιπροσωπεύει έναν συνδυασμό έρευνας, σχεδιασμού και υλοποίησης για τη δημιουργία μιας εξελιγμένης εφαρμογής Android προσαρμοσμένης για την αναζήτηση του περιεχομένου σε έντυπα βιβλία. Ο συνδυασμός τεχνολογιών αιχμής, όπως η Kotlin, το Jetpack Compose, το MVVM με Clean Architecture και το Firebase- Firestore, όχι μόνο απέδειξαν τις δυνατότητες των σύγχρονων πρακτικών ανάπτυξης αλλά και τον ρόλο των καλά δομημένων συστημάτων στη βελτίωση της εμπειρίας των χρηστών.

Κεντρική θέση σε αυτό το έργο ήταν η ενσωμάτωση του Firestore, το οποίο επέτρεψε την επεκτάσιμη αποθήκευση δεδομένων βάσει cloud και τον συγχρονισμό σε πραγματικό χρόνο. Αυτό εξασφάλισε μια απρόσκοπτη σύνδεση μεταξύ του backend και του frontend της εφαρμογής, δημιουργώντας μια δυναμική και αποκριτική εμπειρία χρήστη. Η προσθήκη της Room ως τοπικής βάσης δεδομένων αύξησε αυτό το σύστημα βελτιστοποιώντας την ταχύτητα ανάκτησης δεδομένων, μια κρίσιμη δυνατότητα σε περιβάλλοντα κινητής τηλεφωνίας.

Η εφαρμογή του MVVM και του Clean Architecture παρέχει ένα στιβαρό δομικό πλαίσιο, διασφαλίζοντας και συντηρησιμότητα. Αυτές οι αρχές διευκόλυναν την επεκτασιμότητα λογισμικού που μπορεί να προσαρμοστεί στις μελλοντικές απαιτήσεις. Επιπλέον, η χρήση του Jetpack Compose επαναπροσδιόρισε το επίπεδο διεπαφής χρήστη, προσφέροντας μια δηλωτική προσέγγιση για την ανάπτυξη της διεπαφής χρήστη, βελτιώνοντας την απόδοση και κάνοντας τη βάση κώδικα πιο διαισθητική.

Ένα από τα ξεχωριστά χαρακτηριστικά αυτής της εφαρμογής ήταν η ικανότητά της να αναλύει δεδομένα βιβλίου στο Firestore μέσω ενός Python script, αναλύοντας το περιεχόμενο σε εγγραφές με δυνατότητα αναζήτησης. Αυτή η καινοτόμος προσέγγιση επέτρεψε αποτελεσματικές αναζητήσεις λέξεων-κλειδιών, παρέχοντας ακριβή αποτελέσματα στους χρήστες με ελάχιστη υπολογιστική επιβάρυνση. Η λειτουργία κωδικού QR της εφαρμογής ενίσχυσε περαιτέρω την πρακτικότητά της απλοποιώντας τη ροή εργασίας του χρήστη και επιτρέποντας την άμεση πρόσβαση σε δεδομένα που αφορούν συγκεκριμένα βιβλία.

Η πτυχιακή υπογραμμίζει επίσης τη σημασία της ευθυγράμμισης των τεχνολογικών επιλογών με τους στόχους του έργου. Κάθε εργαλείο και πλαίσιο επιλέχτηκε προσεκτικά όχι μόνο για να ανταποκριθεί στις τρέχουσες απαιτήσεις της εφαρμογής αλλά και για να διασφαλίσει μελλοντική επεκτασιμότητα. Αυτή η διορατικότητα τοποθετεί την εφαρμογή ως μια επεκτάσιμη λύση ικανή να εξελιχθεί παράλληλα με τις εξελίξεις στο οικοσύστημα Android.

Αυτή η εργασία δείχνει πώς μια στοχαστική σύνθεση σύγχρονων τεχνολογιών μπορεί να αντιμετωπίσει προβλήματα του πραγματικού κόσμου, όπως η βελτίωση της πρόσβασης σε πληροφορίες σε έντυπα βιβλία. Πέρα από τα τεχνικά του επιτεύγματα, αυτό το έργο αντικατοπτρίζει την ευρύτερη σημασία της γεφύρωσης των παραδοσιακών μέσων με σύγχρονες ψηφιακές λύσεις, ενισχύοντας την καινοτομία με σεβασμό στις καθιερωμένες πρακτικές. Μέσω της ανάπτυξής της, αυτή η εφαρμογή έχει θέσει τις βάσεις για περαιτέρω διερεύνηση στην ενοποίηση της τεχνολογίας κινητής τηλεφωνίας με εκπαιδευτικούς και πληροφοριακούς πόρους. Αποτελεί ταυτόχρονα ένα λειτουργικό εργαλείο για τους χρήστες και ως απόδειξη της μεταμορφωτικής δύναμης του στοχαστικού σχεδιασμού και της τεχνολογικής εφευρετικότητας.

Η εφαρμογή θα μπορούσε να γίνει καλύτερη με τους εξής τρόπους:

1. **Dedicated backend**: Ένα κεντρικό backend μειώνει την πολυπλοκότητα του χειρισμού της λογικής απευθείας στην εφαρμογή και επιτρέπει την καλύτερη διαχείριση της επιχειρηματικής λογικής και της επεξεργασίας δεδομένων. Χρησιμοποιώντας πλαίσια υποστήριξης όπως το Node.js, το Django ή το Spring Boot[48] για να δημιουργήσουμε RESTful ή GraphQL API[49]. Το backend μπορεί να χειριστεί εργασίες όπως:
   1.1. Επεξεργασία ερωτημάτων αναζήτησης
   1.2. Διαχείριση ελέγχου ταυτότητας χρήστη και ρόλων
   1.3. Οργάνωση και βελτιστοποίηση ανάκτησης δεδομένων

   Οφέλη: Αυτή η προσέγγιση διασφαλίζει τη συνέπεια, επιταχύνει την απόδοση της εφαρμογής εκφορτώνοντας εργασίες από τον πελάτη και υποστηρίζει προηγμένη επεξεργασία από την πλευρά του διακομιστή.

   Σε περίπτωση αλλαγής βάσης θα χρειαστεί μια μετανάστευση βάσεων δεδομένων (migrate) Ενώ το Firestore είναι εξαιρετικό για δεδομένα σε πραγματικό χρόνο και απλές δομές, η μετάβαση σε μια βάση δεδομένων με πιο πλούσια χαρακτηριστικά όπως η PostgreSQL, η MySQL ή η MongoDB μπορεί να προσφέρει καλύτερη ευρετηρίαση, βελτιστοποίηση ερωτημάτων και σχέσεις δεδομένων.

2. **Προσθήκη μηχανής αναζήτησης**: Η βελτιστοποίηση της διαδικασίας αναζήτησης με αποκλειστικές μηχανές αναζήτησης όπως το Elasticsearch ή το Apache Solr[50] μπορεί να κάνει την ανάκτηση λέξεων-κλειδιών και φράσεων ταχύτερη και πιο αποτελεσματική, ακόμη και με μεγάλα σύνολα δεδομένων.Μπορούμε να συγχρονίσουμε δεδομένα (sync) από τη βάση δεδομένων με τη μηχανή αναζήτησης κάνοντας χρήση τα queries. Αυτό μπορεί επίσης να περιλαμβάνει προηγμένες λειτουργίες όπως ασαφείς αναζητήσεις, ανοχή τυπογραφικών λαθών και αναλυτικά στοιχεία αναζήτησης.
3. **Δυνατότητες εκτός σύνδεσης**: Οι χρήστες μπορεί να χρειάζονται πρόσβαση σε περιβάλλοντα χαμηλής ή καθόλου συνδεσιμότητας. Τρόπος: Επέκταση της Room.
4. **Εξατομίκευση χρήστη**: Η ενίσχυση της αφοσίωσης των χρηστών μέσω προσαρμοσμένων εμπειριών μπορεί να αυξήσει το ενδιαφέρον τους. Πώς: Με την προσθήκη λογαριασμών χρηστών όπου αποθηκεύονται οι προτιμήσεις, το ιστορικο. Χρήση μοντέλων μηχανικής εκμάθησης για να επισημανθουν σχετικές ενότητες με βάση το ιστορικό αναζήτησης και χρήσης τους. Πλεονεκτήματα: Προσφέρει εξατομικευμένη και δυναμική εμπειρία χρήστη.
5. **Advanced Analytics**: Η παρακολούθηση της χρήσης της εφαρμογής βοηθά στον εντοπισμό περιοχών προς βελτίωση. Πώς: Ενσωματώστε εργαλεία όπως το Google Analytics, το Firebase Analytics[51] για να παρακολουθείτε η συμπεριφορά των χρηστών, τα μοτίβα αναζήτησης και τη χρήση λειτουργιών. Οφέλη: Οι πληροφορίες που βασίζονται σε δεδομένα μπορούν να καθοδηγήσουν μελλοντικές βελτιώσεις.
6. **Ενσωματώστε AI**: Η προσθήκη δυνατοτήτων AI μπορεί να κάνει την εφαρμογή πιο ισχυρή για ακαδημαϊκούς ή ερευνητικούς σκοπούς. Πώς: Με την χρήση εργαλείων όπως μοντέλα OpenAI GPT ή Google NLP API, συνοψίζοντας το περιεχόμενο του βιβλίου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Arshad Ahmad & Kan Li & Chong Feng & Syed Mohammad Asim & Abdallah Yousif & Shi Ge, "An Empirical Study of Investigating Mobile Applications Development Challenges", IEEE Access, March 2018

[2] M. E. Joorabchi, A. Mesbah, and P. Kruchten, ''Real challenges in mobile app development,'' in Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas., Oct. 2013, pp. 15–24.

[3] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, ''Hybrid mobile apps in the Google Play Store: An exploratory investigation,'' in Proc. 2nd ACM Int. Conf. Mobile Softw. Eng. Syst., 2015, pp. 56–59.

[4] T. Vilček, T. Jakopec, ''Comparative analysis of tools for development of native and hybrid mobile applications'',in IEEE Xplore, 2017

[5] Fattoh Al-Qershi; Muhammad Al-Qurishi; Sk Md Mizanur Rahman; Atif Al-Amri, "Android vs. iOS: The security battle", in IEEE, 2014

[12] Priyadi, Y., & Adrian, M. Usability Measurement in User Interface Design Using Heuristic Evaluation & Severity Rating. IEEE Annual Information Technology Conference. Access Paper, in IEEE, 2022

[13] Verdecchia, R., & Malavolta, I." Guidelines for architecting android apps: A mixed-method empirical study", in IEEE, 2019

[14] Dobrean, D. "Automatic examining of software architectures on mobile applications codebases", in IEEE, 2019

[15] Narh, P. G., & Ranjan, R. "Integration of AI With Mobile Application Development", in IEEE 2022

[16] Cao, C., Meng, C., Ge, H., & Yu, P. "Xdroid: Testing android apps with dependency injection" in IEEE, 2017

[17] Lin, S., Ruan, X., Cai, X., & Chen, Y. "Research and Implementation of Urban Water Supply Pipe Network Management System Based on Mobile GIS" in IEEE, 2024

[6] European Telecommunications Standards Institute, "Digital Video Broadcasting (DVB): Implementation guide for DVB terrestrial services; transmission aspects," *European Telecommunications Standards Institute*, ETSI-TR-101, 2007. [Online]. Available: http://www.etsi.org.

[7] Google, Android History, https://developer.android.com

[8] Google, Rubin, A., et al., Founders of Android, https://www.android.com/what-is-android

[9] "History of Android" https://www.geeksforgeeks.org/history-of-android

[10] Kotlin Lang, “Kotlin docs” https://kotlinlang.org/docs/home.html

[11] “git’ https://git-scm.com/docs/git

[18] Google, “firestore” https://firebase.google.com/docs/firestore

[19] Google, “AAPT2” https://developer.android.com/tools/aapt2

[20] Google, “Android runtime and Dalvik” https://source.android.com/docs/core/runtime

[21] Google, ‘Analyze your build with the APK analyzer” https://developer.android.com/studio/debug/apk-analyzer

[22] Google, “Build App - Gradle” https://developer.android.com/build

[23] Gradle, “Gradle docs” https://docs.gradle.org/current/userguide/gradle_basics.html

[24] Google,”Java versions in Android builds” https://developer.android.com/build/jdks

[25] Medium, “Java: how to avoid the billion dollar mistake?” https://mirakl.tech/java-how-to-avoid-the-billion-dollar-mistake-f84a7189d4dd

[26] Google, “Kotlin Coroutines on Android” https://developer.android.com/kotlin/coroutines

[27] Google, “Processes and threads” https://developer.android.com/guide/components/processes-and-threads

[28] Github, “Github” https://docs.github.com/en

[29] Git, “Git” https://git-scm.com/docs

[30] Medium, “A guide to MVVM for android developers”
https://medium.com/@sammierosado/a-guide-to-mvvm-for-android-developers-79f097c8670e

[31] Medium, “Introduction to clean architecture”
https://celepbeyza.medium.com/introduction-to-clean-architecture-acf25ffe0310

[32] Webmob, “Which Android Architecture Is Best: MVC vs MVP vs MVVM”
https://webmobtech.com/blog/mvc-mvp-mvvm

[33] Medium, “Retrofit in Android”
https://medium.com/@KaushalVasava/retrofit-in-android-5a28c8e988ce

[34] Google, “Save data in a local database using Room”
https://developer.android.com/training/data-storage/room

[35] Google, State holders and Ui State”
https://developer.android.com/topic/architecture/ui-layer/stateholders

[36] Medium, “Navigation in jetpack compose”
https://medium.com/@KaushalVasava/navigation-in-jetpack-compose-full-guide-beginner-to-advanced-950c1133740

[37] Google, Jetpack” https://developer.android.com/jetpack

[38] Medium, “Jetpack Compose Deep Linking: Type-Safe Navigation”
https://medium.com/@shivayogih25/jetpack-compose-deep-linking-type-safe-navigation-done-right-ed79da513c5b

[39] Google, “Dependency Injection in Android”
https://developer.android.com/training/dependency-injection

[40] Medium, ”Dependency Injection (DI) on Android”
https://medium.com/@zorbeytorunoglu/depencency-injection-di-on-android-1dade6c22711

[41] Amazon,”What is boilerplate code?” https://aws.amazon.com/what-is/boilerplate-code

[42] Google," Using Dagger in Android apps"
https://developer.android.com/training/dependency-injection/dagger-android

[43] Medium,"Memory leaks"
https://medium.com/@sarahmaher_01/android-development-memory-leaks-f6b1ed33c029

[44] Medium, "Reverse-Engineering Your Path Towards Software Developer"
https://brianjenney.medium.com/reverse-engineering-your-path-towards-software-developer-25b432eeedf2

[45] Google, "Shrink, obfuscate, and optimize your app"
https://developer.android.com/build/shrink-code

[46] "ProGuard - R8" https://www.guardsquare.com/manual/home

[47] Google," Google code scanner"
https://developers.google.com/ml-kit/vision/barcode-scanning/code-scanner

[48] Medium, "Node.js vs Spring Boot vs Django: Which One Is Best for Beginners?"
https://sandydev.medium.com/node-js-vs-spring-boot-vs-django-which-one-is-best-for-beginners-8782d3be54

[49] Amazon, "What's the Difference Between GraphQL and REST?"
https://aws.amazon.com/compare/the-difference-between-graphql-and-rest/

[50] Medium, "What is Solr? Comparing Apache Solr vs. Elasticsearch"
https://medium.com/@affiliatevasu/what-is-solr-comparing-apache-solr-vs-elasticsearch-a2703f72f64f

[51] Google, "Google Analytics" https://firebase.google.com/docs/analytics

[52] Slack, "What is Slack?" https://slack.com/help/articles/115004071768-What-is-Slack

[53] Google, "Google Analytics" https://firebase.google.com/docs/analytics

ΠΑΡΑΡΤΗΜΑ Α : ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

A) Κώδικας python για εισαγωγή βιβλίου στην firestore

```
import firebase_admin
from firebase_admin import credentials, firestore
from pdfminer.high_level import extract_text

# Initialize Firebase Admin SDK
cred = credentials.Certificate('C:/Users/Mike/Desktop/Python Script/serviceAccountKey.json')
firebase_admin.initialize_app(cred)
db = firestore.client()

def extract_text_from_pdf(pdf_file):
    text = extract_text(pdf_file)
    return text

def upload_book_details_to_firestore():
    book_details = {
        'author': 'Κιοσκερίδης Ιορδάνης',
'bookCover':
'https://firebasestorage.googleapis.com/v0/b/bookseek-c75de.appspot.com/o/book_covers%2Fkef4.pn
g?alt=media&token=91596c2f-f8d2-4b49-aeb0-1ccf00d0565d',
        'title': 'Μέτρηση της Αρτηριακής Πίεσης'
    }
    db.collection('Μέτρηση της Αρτηριακής Πίεσης').document('book_details').set(book_details)

def upload_page_to_firestore(page_text, pageNumber):
    page_data = {
        'pageNumber': pageNumber,
        'content': page_text.strip()
    }
    db.collection('Μέτρηση της Αρτηριακής Πίεσης') \
      .document('book_pages') \
```

```
        .collection('pages') \
        .add(page_data)

def import_pdf_to_firestore(pdf_file_path):
    upload_book_details_to_firestore()

    with open(pdf_file_path, 'rb') as pdf_file:
        text = extract_text_from_pdf(pdf_file)
        pages = text.split('\f')

        for pageNumber, page_text in enumerate(pages, start=141):
            # Strip whitespace and check if the page is empty
            stripped_page = page_text.strip()
            if not stripped_page:
                # Skip empty last page (or any empty pages)
                continue

            upload_page_to_firestore(stripped_page, pageNumber)

if __name__ == "__main__":
    pdf_file_path = 'C:/Users/Mike/Desktop/Python Script/book.pdf'
    import_pdf_to_firestore(pdf_file_path)
```