ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Android εφαρμογής για τις υπηρεσίες του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων»

Του φοιτητή
Ραφαήλ Μονογιός
Αρ. Μητρώου: 164707

Επιβλέπων
Αντώνης Σιδηρόπουλος
Επίκουρος Καθηγητής

Σεπτέμβριος 2020

Τίτλος Π.Ε. Ανάπτυξη Android εφαρμογής για τις υπηρεσίες του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

Κωδικός Π.Ε. 20122

Ονοματεπώνυμο φοιτητή Ραφαήλ Μονογιός

Ονοματεπώνυμο εισηγητή Αντώνης Σιδηρόπουλος

Ημερομηνία ανάληψης Π.Ε. 03-04-2020

Ημερομηνία περάτωσης Π.Ε. 18-09-2020

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.

Πρόλογος

Ένας σημαντικός λόγος που με έκανε επιλέξω αυτή την Πτυχιακή Εργασία ήταν η περιέργεια για το πως δούλευε το IT_API σαν σύνολο και πως θα μπορούσα να αξιοποιήσω αυτήν την υπηρεσία στο να βελτιώσω τις γνώσεις μου γύρω από στην ανάπτυξη εφαρμογών Android. Ένας δεύτερος λόγος είναι ότι η πρόσβαση στις ανακοινώσεις του τμήματος έπρεπε να γίνεται αποκλειστικά μέσω της ιστοσελίδας apps.it.teithe.gr και η διαδικασία ταυτοποιήσης και επιλογής της καρτέλας των ανακοινώσεων κάθε φορά ήταν κουραστική, οπότε με το IT_API η ανάπτυξη μιας Android εφαρμογής ήταν μονόδρομος.

Περίληψη

Αντικείμενο της παρούσας Πτυχιακής Εργασίας είναι η μελέτη και η ανάπτυξη μιας εφαρμογής για κινητές συσκευές Android η οποία θα παρέχει πρόσβαση στους φοιτητές στις υπηρεσίες του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνές Πανεπιστημίου της Ελλάδος. Μελετήθηκαν και αναλύθηκαν γλώσσες προγραμματισμού, τεχνολογίες, τεχνικές και υπηρεσίες για το καλύτερο πιθανό αποτέλεσμα. Έγινε χρήση το ανοιχτού κώδικα RESTful API, IT API και η υλοποίηση έγινε στην γλώσσα προγραμματισμού Java μέσα στο περιβάλλον ανάπτυξης Android Studio. Επίσης χρησιμοποιήθηκαν προϊόντα από την πλατφόρμα του Firebase. Επιπλέον παρουσιάζονται οδηγίες χρήσης της εφαρμογής με λεκτική περιγραφή και γραφική αναπαράσταση.

«Development of an Android application for the services of Information and Electronic Engineering department»

«Rafail Monogios»

Abstract

The subject of this Thesis is the study and development of an application for Android mobile devices which will provide access to student to the services of the Department of Information and Electronic Engineering of the International University of Greece. Programming languages, technologies, techniques and services were studied and analyzed for the best possible result. The open source RESTful API, IT API was used and implemented in the Java programming language within the Android Studio development environment. Products from the Firebase platform were also used. In addition, instructions for use of the application are presented with a verbal description and graphic representation.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επίκουρο καθηγητή Αντώνη Σιδηρόπουλο για την εμπιστοσύνη που μου έδειξε προτείνοντας μου να εκπονήσω αυτή την Πτυχιακή Εργασία. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου για την υποστήριξη που μου παρείχε κατά την διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
OHA	Open Handset Alliance
REST	Representational State Transfer
IDE	Integrated Development Environment
JVM	Java Virtual Machine
XML	eXtensible Markup Language
MVVM	Model View ViewModel
MVC	Model View Controller
UI	User Interface
IEE	Information and Electronic Engineering

Κεφάλαιο 1ο: Εισαγωγή

1.1 Σκοπός Πτυχιακής Εργασίας

Ο σκοπός της πτυχιακής εργασίας είναι η διευκόλυνση των φοιτητών στο να έχουν πιο γρήγορη και εύκολη επαφή με τις ανακοινώσεις του τμήματος μέσω του Android Smartphone τους όπως επίσης η λήψη ειδοποιήσεων είναι απαραίτητη για την άμεση ενημέρωση του φοιτητή για νέες ανακοινώσεις που τον αφορούν. Για αυτό το λόγο, η δημιουργία μιας Android εφαρμογής είναι απαραίτητη. Αυτή η εφαρμογή θα πρέπει να είναι βασισμένη στην υπηρεσία Apps του τμήματος και θα περιέχει τις ανακοινώσεις, προφίλ χρήστη και δυνατότητα λήψης ειδοποιήσεων σε κατηγορίες που θα επιλέγει ο χρήστης.

1.2 Περιγραφή εφαρμογής

Η εφαρμογή θα αποτελείται από 3 αρχικές οθόνες οι οποίες θα είναι προσβάσιμες από το κάτω μέρος της εφαρμογής με προεπιλεγμένη οθόνη τις ανακοινώσεις. Ο χρήστης θα μπορεί να επιλέξει την ανακοίνωση που τον ενδιαφέρει πατώντας πάνω σε αυτήν και θα τον μεταφέρει σε μια άλλη οθόνη όπου θα μπορεί να δει ολόκληρο το κείμενο της ανακοίνωσης όπως επίσης και αρχεία που θα χρειαστεί να κατεβάσει. Επιπλέον, στην οθόνη ανακοινώσεων ο χρήστης θα μπορεί να αναζητήσει κάποια ανακοίνωση με λέξεις κλειδιά ή να δει τις ειδοποιήσεις από ανακοινώσεις που τον αφορούν. Στην δεύτερη οθόνη θα βρίσκετε το προσωπικό προφίλ του χρήστη με πληροφορίες για αυτόν, όπου θα μπορεί να επεξεργαστεί. Τέλος, στην τρίτη αρχική οθόνη της εφαρμογής θα βρίσκονται η ρυθμίσεις όπου ο χρήστης θα μπορεί να βρει ρυθμίσεις για τον λογαριασμό του, της εφαρμογής αλλά και πληροφορίες.

Η εφαρμογή ακολουθεί το MVVM architectural pattern και χρησιμοποιεί τις βιβλιοθήκες Nagivation, ViewModel, LiveData, Paiging, DataBinding της σουίτας Android Jetpack. Επίσης χρησιμοποιεί τα Adapter, Singleton, Observable design patterns και κάνει χρήση του IT_API. Επιπλέων χρησιμοποιεί τις βιβλιοθήκες Realtime Database, Cloud Functions, Cloud Messaging, Analytics και Crashlytics της πλατφόρμας Firebase.

1.3 Δομή εργασίας

Το πρώτο κεφάλαιο αφορά την εισαγωγή της Πτυχιακής Εργασίας και περιέχει τον βασικό λόγο και σκοπό για την δημιουργία της εφαρμογής. Επίσης περιέχει μια περιγραφή για το τι θα χρησιμοποιηθεί για την υλοποίηση της εφαρμογής και για το πως θα φαίνεται οπτικά.

Το δεύτερο κεφάλαιο αφορά το λειτουργικό σύστημα Android και περιέχει το ιστορικό, ιστορικό εκδόσεων, χαρακτηριστικά, αρχιτεκτονική και άλλες πληροφορίες που το αφορούν. Επίσης περιέχει και άλλες τεχνολογίες, τεχνικές και υπηρεσίες που θα χρησιμοποιηθούν.

Το τρίτο κεφάλαιο αφορά τα Application Programming Interface (API) και αναλύει τους ορισμούς και τεχνολογίες πίσω από το IT_API, το API του τμήματος που θα χρησιμοποιηθεί για τα δεδομένα της εφαρμογής.

Το τέταρτο κεφάλαιο αφορά την υλοποίηση της εφαρμογής και αναλύει το πώς έχει υλοποιηθεί η εφαρμογή με βάση το κεφάλαιο 2.

Το πέμπτο κεφάλαιο αφορά την παρουσίαση της εφαρμογής με κείμενο και εικόνες για όλες τις λειτουργίες που διαθέτει.

Το έκτο κεφάλαιο αφορά τα συμπεράσματα και τις προτάσεις βελτίωσης της εφαρμογής.

Κεφάλαιο 2ο: Λειτουργικό Android και άλλες τεχνολογίες

2.1 Εισαγωγή στο Android

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.

Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear).

Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο και συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και MacOS μαζί. Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance.

Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού. Το λογότυπο για το λειτουργικό σύστημα Android είναι ένα ρομπότ σε χρώμα πράσινου μήλου και σχεδιάστηκε από τη γραφίστρια Ιρίνα Μπλόκ.

Σχήμα 1: Το λογότυπο του Android σήμερα

2.1.1 Ιστορική αναδρομή

Το λειτουργικό σύστημα Android ιδρύθηκε στο Palo Alto της Καλιφόρνια τον Οκτώβριο του 2003 από τους Andy Rubin, Rich Miner, Nick Sears και Chris White. Οι πρώτες προθέσεις της εταιρείας ήταν να αναπτύξει ένα προηγμένο λειτουργικό σύστημα για ψηφιακές φωτογραφικές μηχανές και αυτή ήταν η βάση του πεδίου για τους επενδυτές τον Απρίλιο του 2004. Η εταιρεία αποφάσισε τότε ότι η αγορά των φωτογραφικών μηχανών δεν ήταν αρκετά μεγάλη για τους στόχους της, πέντε μήνες αργότερα είχε εκτρέψει τις προσπάθειές της και ανέβαζε το Android ως λειτουργικό σύστημα χειρός που θα ανταγωνιζόταν τα Symbian και τα Microsoft Windows Mobile.

Τον Ιούλιο του 2005, η Google απέκτησε το Android Inc. για τουλάχιστον 50 εκατομμύρια δολάρια. Στο Google, η ομάδα με επικεφαλής τον Rubin ανέπτυξε μια πλατφόρμα κινητών συσκευών που τροφοδοτείται από τον πυρήνα του Linux. Η Google προώθησε την πλατφόρμα στους κατασκευαστές και τους μεταφορείς με την υπόσχεση παροχής ενός ευέλικτου, αναβαθμίσιμου συστήματος. Αργότερα άλλαξε τα έγγραφα προδιαγραφών Android για να δηλώσει ότι θα υποστηρίζονται οι "Οθόνες αφής", αν και "το Προϊόν σχεδιάστηκε με την παρουσία διακριτών φυσικών κουμπιών ως υπόθεση, επομένως μια οθόνη αφής δεν μπορεί να αντικαταστήσει πλήρως τα φυσικά κουμπιά".

Μέχρι το 2008, τόσο η Nokia όσο και το BlackBerry ανακοίνωσαν έξυπνα τηλέφωνα με βάση το touch για να ανταγωνιστεί το iPhone 3G και η εστίαση του Android τελικά μετατράπηκε σε απλές οθόνες αφής. Το πρώτο εμπορικά διαθέσιμο smartphone που τρέχει το Android ήταν το HTC Dream, επίσης γνωστό ως T-Mobile G1, που ανακοινώθηκε στις 23 Σεπτεμβρίου 2008.Στις 5 Νοεμβρίου 2007, η Open Handset Alliance, μια κοινοπραξία εταιρειών τεχνολογίας όπως η Google, κατασκευαστές συσκευών όπως HTC, Motorola και Samsung, ασύρματες εταιρείες όπως η Sprint και η T-Mobile και κατασκευαστές chipset όπως η Qualcomm και η Texas Instruments με στόχο την ανάπτυξη της "πρώτης πραγματικά ανοικτής και ολοκληρωμένης πλατφόρμας για κινητές συσκευές".

Από το 2008, το Android έχει δει πολλές αναβαθμίσεις, οι οποίες έχουν βελτιώσει σταδιακά το λειτουργικό σύστημα, προσθέτοντας νέα χαρακτηριστικά και διορθώνοντας σφάλματα σε προηγούμενες κυκλοφορίες. Κάθε μεγάλη κυκλοφορία ονομάζεται με αλφαβητική σειρά μετά από επιδόρπιο ή ζαχαρούχο σκεύασμα, με τις πρώτες εκδόσεις του Android να ονομάζονται "Cupcake", "Donut", "Eclair" και "Froyo", με αυτή τη σειρά.

Τον Ιούνιο του 2014, η Google ανακοίνωσε το Android One, ένα σύνολο «μοντέλων αναφοράς υλικού» που θα «επιτρέψουν στους κατασκευαστές συσκευών να δημιουργήσουν εύκολα τηλέφωνα υψηλής ποιότητας με χαμηλό κόστος», σχεδιασμένα για καταναλωτές στις αναπτυσσόμενες χώρες.

Σχήμα 2: HTC Dream, το πρώτο εμπορικά διαθέσιμο smartphone με Android

2.1.2 Ιστορικό εκδόσεων

Από την πρώτη κυκλοφορία τις πρώτης έκδοσης του Android έως σήμερα, το λειτουργικό σύστημα έχει μεταμορφωθεί οπτικά, εννοιολογικά αλλά και λειτουργικά.

2.1.2.1 Έκδοση 1.0 - 1.1

Η πρώτη έκδοση του λειτουργικό συστήματος που κυκλοφόρησε το 2008 και δεν έχει κωδική ονομασία. Υπήρχαν τα απολύτως βασικά μαζί με κάποιες εφαρμογές όπως το Gmail, Maps και Youtube.

2.1.2.2 Έκδοση 1.5 (Cupcake)

Το Cupcake κυκλοφόρησε στις αρχές του 2009 και γεννήθηκε η παράδοση των κωδικών ονομάτων έκδοσης. Εισήγαγε πολλές βελτιώσεις στη διεπαφή χρήστη, συμπεριλαμβανομένου του πρώτου πληκτρολογίου στην οθόνη κάτι που ήταν απαραίτητο καθώς τα τηλέφωνα απομακρύνθηκαν από το μοντέλο φυσικού πληκτρολογίου.

2.1.2.3 Έκδοση 1.6 (Donut)

Το Donut κυκλοφόρησε το φθινόπωρο του 2009. Γέμισε σημαντικές τρύπες στο κέντρο του Android συμπεριλαμβανομένης της δυνατότητας λειτουργίας του λειτουργικού συστήματος σε μια ποικιλία διαφορετικών μεγεθών και αναλύσεων οθόνης, ένας παράγοντας που θα ήταν κρίσιμος στα επόμενα χρόνια. Πρόσθεσε επίσης υποστήριξη για δίκτυα CDMA όπως το Verizon, τα οποία θα διαδραματίσουν βασικό ρόλο στην επικείμενη έκρηξη του Android.

2.1.2.4 Έκδοση 2.0 - 2.1 (Eclair)

Το Eclair εμφανίστηκε μόλις έξι εβδομάδες μετά το Donut και η ενημέρωση 2.1, που ονομάζεται κυκλοφόρησε μερικούς μήνες αργότερα. Προστέθηκε η δυνατότητα φωνητικής turn-by-turn καθοδήγησης με πληροφορίες κίνησης σε πραγματικό χρόνο κάτι που δεν είχε δεί προηγουμένως ο κόσμος των smartphone. Επίσης έφερε ζωντανές ταπετσαρίες στο Android καθώς και την πρώτη λειτουργία speech-to-text και έκανε κύματα για την υλοποίηση της μοναδικής δυνατότητας pinch-to-zoom που ήταν αποκλειστικά iOS για Android, μια κίνηση που συχνά θεωρείται ως η σπίθα που πυροδότησε τον μακροχρόνιο "θερμοπυρηνικό πόλεμο" της Apple εναντίον της Google.

2.1.2.5 Έκδοση 2.2 (Froyo)

Το Froyo ήρθε 4 μήνες μετά το 2.1 Eclair το οποίο περιστράφηκε σε μεγάλο βαθμό γύρω από τις βελτιώσεις στην απόδοση. Παρέδωσε μερικές σημαντικές λειτουργίες στο μπροστινό μέρος (διεπιφάνεια), συμπεριλαμβανομένης της προσθήκης της πλέον τυπικής βάσης στο κάτω μέρος της αρχικής οθόνης καθώς και της πρώτης ενσάρκωσης των φωνητικών εντολών. Επίσης έφερε την υποστήριξη για το Flash στο πρόγραμμα περιήγησης ιστού, μια επιλογή που ήταν σημαντική τόσο λόγω της εκτεταμένης χρήσης του Flash εκείνη την εποχή όσο και λόγω της στάσης της Apple να μην το υποστηρίζει στις δικές τις κινητές συσκευές.

2.1.2.6 Έκδοση 2.3 (Gingerbread)

Το Gingerbread κυκλοφόρησε το 2010 και άρχισε να επικεντρώνεται η πρώτη πραγματική οπτική ταυτότητα του Android.. Το φωτεινό πράσινο ήταν από καιρό το χρώμα της μασκότ ρομπότ του Android και με το Gingerbread, έγινε αναπόσπαστο μέρος της εμφάνισης του λειτουργικού συστήματος. Μαύρο και πράσινο σε όλη τη διεπαφή χρήστη καθώς το Android ξεκίνησε την αργή πορεία προς τον διακριτικό σχεδιασμό.

2.1.2.7 Έκδοση 3.0 - 3.2 (Honeycomb)

Η περίοδος Honeycomb του 2011 ήταν μια περίεργη εποχή για το Android. Το Android 3.0 ήρθε στον κόσμο ως έκδοση μόνο για tablet για να συνοδεύσει την κυκλοφορία του Motorola Xoom και μέσω των επακόλουθων ενημερώσεων 3.1 και 3.2 παρέμεινε μια οντότητα αποκλειστική για tablet (και κλειστού κώδικα). Παρουσίασε ένα δραματικά επαναπροσδιορισμένο UI και είχε ένα διαστημικό "ολογραφικό" σχέδιο που ανταλλάσσει το εμπορικό σήμα της πλατφόρμας με πράσινο χρώμα μπλε και έδωσε έμφαση στην αξιοποίηση του χώρου οθόνης ενός tablet. Ενώ η έννοια μιας διεπαφής για tablet δεν κράτησε πολύ, πολλές από τις ιδέες της Honeycomb έθεσαν τα θεμέλια για

το Android που γνωρίζουμε σήμερα. Το λογισμικό ήταν το πρώτο που χρησιμοποίησε κουμπιά στην οθόνη για τις κύριες εντολές πλοήγησης του Android. σημείωσε την αρχή του τέλους για το κουμπί μόνιμης υπερχείλισης-μενού και εισήγαγε την έννοια ενός UI που μοιάζει με κάρτα με τη σειρά του στη λίστα πρόσφατες εφαρμογές.

2.1.2.8 Έκδοση 4.0 (Ice Cream Sandwich)

Το Ice Cream Sandwich κυκλοφόρησε επίσης το 2011 και χρησίμευσε ως επίσημη είσοδος της πλατφόρμας στην εποχή του μοντέρνου σχεδιασμού. Βελτίωσε τις οπτικές έννοιες που εισήχθησαν με το Honeycomb και επαναένωσε τα tablet και τα τηλέφωνα με ένα ενιαίο, ενοποιημένο όραμα UI. Πέταξε μεγάλο μέρος της "ολογραφικής" εμφάνισης της Honeycomb αλλά κράτησε τη χρήση του μπλε ως highlight σε όλο το σύστημα και άλλα βασικά στοιχεία συστήματος. Επίσης έκανε το swiping μια πιο αναπόσπαστη μέθοδο περιήγησης στο λειτουργικό σύστημα, με την τότε επαναστατική αίσθηση να απομακρύνει πράγματα όπως ειδοποιήσεις και πρόσφατες εφαρμογές και ξεκίνησε την αργή διαδικασία για να φέρει ένα τυποποιημένο πλαίσιο σχεδίασης γνωστό ως "Holo" σε όλο το λειτουργικό σύστημα και στο οικοσύστημα εφαρμογών του Android.

2.1.2.9 Έκδοση 4.1 - 4.3 (Jelly Bean)

Οι κυκλοφορίες του Jelly Bean το 2012-2013 πήραν το νέο ίδρυμα της ICS και σημείωσαν σημαντική πρόοδο στην τελειοποίηση και την ανάπτυξη. Οι κυκλοφορίες πρόσθεσαν πολλά πράγματα και λειτουργίες στο λειτουργικό σύστημα και προχώρησαν πολύ στο να κάνουν το Android πιο ελκυστικό για τον μέσο χρήστη. Έφερε την πρώτη γεύση στο Google Now που δυστυχώς από τότε μετατράπηκε σε μια ροή ειδήσεων. Έδωσε επεκτάσιμες και διαδραστικές ειδοποιήσεις, ένα διευρυμένο σύστημα φωνητικής αναζήτησης και ένα πιο προηγμένο σύστημα για την εμφάνιση αποτελεσμάτων αναζήτησης με έμφαση στα αποτελέσματα βάση καρτών. Η υποστήριξη πολλαπλών χρηστών άρχισε επίσης να μπαίνει στο παιχνίδι, αν και σε εκείνο το σημείο ήταν μόνο σε tablet όπως επίσης έκανε την πρώτη της εμφάνιση η πρώτη έκδοση του πίνακα γρήγορων ρυθμίσεων του Android.

2.1.2.10 Έκδοση 4.4 (KitKat)

Το KitKat κυκλοφόρησε στα τέλη του 2013 και σηματοδότησε το τέλος της σκοτεινής εποχής του Android, καθώς τα μαύρα του Gingerbread και τα μπλε της Honeycomb έφυγαν από το λειτουργικό σύστημα. Ελαφρύτερα φόντα και πιο ουδέτερα στιγμιότυπα πήραν τη θέση τους με μια διαφανή γραμμή κατάστασης και λευκά εικονίδια που δίνουν στο λειτουργικό σύστημα μια πιο σύγχρονη εμφάνιση. Επίσης είδε την πρώτη έκδοση της υποστήριξης "OK, Google" αλλά στο KitKat η προτροπή ενεργοποίησης ανοιχτής ακρόασης λειτούργησε μόνο όταν η οθόνη ήταν ήδη ενεργοποιημένη και η συσκευή ήταν είτε στην αρχική σας οθόνη είτε μέσα στην εφαρμογή Google.

2.1.2.11 Έκδοση 5.0 - 5.1 (Lollipop)

Το Lollipop κυκλοφόρησε το φθινόπωρο του 2014 και με αυτή την έκδοση η Google ξανά “εφηύρε” το Android. Κυκλοφόρησε το πρότυπο Material Design που εξακολουθεί να υπάρχει σήμερα, το οποίο έφερε μια εντελώς νέα εμφάνιση που επεκτάθηκε σε όλο το Android, τις εφαρμογές της και ακόμη και σε άλλα προϊόντα της Google. Εισήγαγε πολλές νέες λειτουργίες στο Android, συμπεριλαμβανομένου του φωνητικού ελέγχου hands-free μέσω της εντολής "OK, Google", υποστήριξη για πολλούς χρήστες σε τηλέφωνα και λειτουργία προτεραιότητας για καλύτερη διαχείριση ειδοποιήσεων. Δυστυχώς, άλλαξε τόσο πολύ που εισήγαγε επίσης πολλά ενοχλητικά σφάλματα, πολλά από τα οποία δεν θα εξολοθρευτούν πλήρως μέχρι την κυκλοφορία του 5.1 τον επόμενο χρόνο.

2.1.2.12 Έκδοση 6.0 (Marshmallow)

Το Marshmallow κυκλοφόρησε και σε σύγκριση με άλλες εκδόσεις ήταν μια μικρή έκδοση. Ο λόγο είναι επειδή η Google ξεκίνησε την τάση να κυκλοφορεί μια σημαντική έκδοση Android ανά έτος και αυτή η έκδοση λαμβάνει πάντα τον δικό της ολόκληρο αριθμό. Έφερε ορισμένα πράγματα με διαρκή αντίκτυπο, ωστόσο, συμπεριλαμβανομένων πιο λεπτομερών αδειών εφαρμογής, υποστήριξη για αναγνώστες δακτυλικών αποτυπωμάτων και υποστήριξη για USB-C.

2.1.2.13 Έκδοση 7.0 - 7.1 (Nougat)

Οι εκδόσεις Nougat κυκλοφόρησαν το 2016 και παρείχαν στο Android μια εγγενή λειτουργία διαχωρισμού οθόνης, ένα νέο σύστημα συνδυασμένης εφαρμογής για την οργάνωση ειδοποιήσεων και μια λειτουργία εξοικονόμησης δεδομένων. Τα Nougat πρόσθεσαν επίσης μερικές μικρότερες αλλά ακόμα σημαντικές δυνατότητες, όπως μια συντόμευση Alt-Tab για εφαρμογή μεταξύ εφαρμογών και την κυκλοφορία του βοηθού Google.

2.1.2.14 Έκδοση 8.0 - 8.1 (Oreo)

Το Oreo κυκλοφόρησε το 2017 και πρόσθεσε μια εγγενή λειτουργία εικόνας-σε-εικόνα, μια επιλογή αναβολής ειδοποιήσεων και κανάλια ειδοποιήσεων που προσφέρουν καλό έλεγχο του τρόπου με τον οποίο οι εφαρμογές μπορούν να ειδοποιήσουν τους χρήστες. Επίσης περιλάμβανε ορισμένα αξιοσημείωτα στοιχεία που προωθούσαν τον στόχο της Google να ευθυγραμμίσει το Android και το Chrome OS και να βελτιώσει την εμπειρία χρήσης εφαρμογών Android σε Chromebook και ήταν η πρώτη έκδοση Android που παρουσίασε το Project Treble, μια φιλόδοξη προσπάθεια να δημιουργήσει μια αρθρωτή βάση για τον κώδικα του Android με την ελπίδα να διευκολύνει τους κατασκευαστές συσκευών να παρέχουν έγκαιρες ενημερώσεις λογισμικού.

2.1.2.15 Έκδοση 9 (Pie)

Το Pie κυκλοφόρησε τον Αύγουστο του 2018 και η μεγαλύτερη αλλαγή ήταν το υβριδικό σύστημα πλοήγησης με κουμπιά, το οποίο ανταλλάσσει τα παραδοσιακά πλήκτρα Android, Back, Home και Overview για ένα μεγάλο, πολυλειτουργικό κουμπί Home και ένα μικρό κουμπί Back που εμφανίστηκε δίπλα του, όπου απαιτείται. Περιελάμβανε επίσης ένα καθολικό σύστημα προτεινόμενης απάντησης για ειδοποιήσεις ανταλλαγής μηνυμάτων, έναν νέο πίνακα ελέγχου ψηφιακών στοιχείων ελέγχου ευεξίας και πιο έξυπνα συστήματα για διαχείριση ισχύος και φωτεινότητας οθόνης. Επίσης παρουσιάσε έναν εξυπνότερο τρόπο χειρισμού σημείων πρόσβασης Wi-Fi, μια αλλαγή στη λειτουργία εξοικονόμησης μπαταρίας του Android και μια ποικιλία βελτιώσεων απορρήτου και ασφάλειας

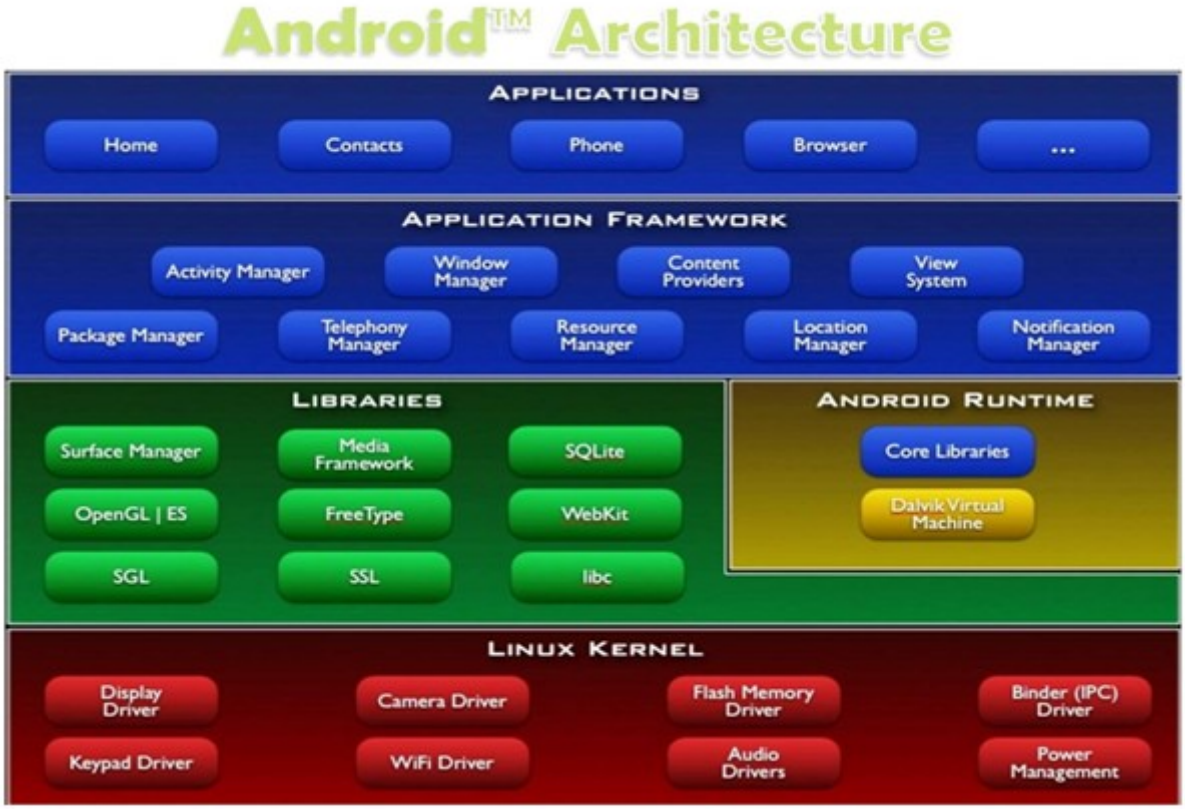
2.1.2.16 Έκδοση 10

Η έκδοση 10 κυκλοφόρησε τον Σεπτέμβριο του 2019 και έφερε το τέλος στις κωδικές ονομασίες. Φέρνει μια εντελώς επαναπροσδιορισμένη διεπαφή στις κινήσεις οθόνης, αυτή τη φορά καταργώντας εντελώς το κουμπί “πίσω” και βασίζεται σε μια πλήρως οδηγημένη προσέγγιση για πλοήγηση συστήματος. Εισάγει μια νέα ρύθμιση για ενημερώσεις σε στυλ επείγουσας επιδιόρθωσης που θα επιτρέψουν τελικά γρηγορότερες και πιο συνεπές εκδόσεις μικρών, στενά εστιασμένων μπαλωμάτων. Επίσης έχει και άλλες σημαντικές βελτιώσεις, συμπεριλαμβανομένου ενός ενημερωμένου συστήματος δικαιωμάτων που σας δίνει περισσότερο έλεγχο για το πώς και πότε οι εφαρμογές έχουν πρόσβαση σε δεδομένα τοποθεσίας, ένα εκτεταμένο σύστημα για την προστασία μοναδικών αναγνωριστικών συσκευών και ένα σκοτεινό θέμα σε όλο το σύστημα.

2.1.3 Αρχιτεκτονική

Το Android είναι ένα λειτουργικό σύστημα και είναι μια στοίβα στοιχείων λογισμικού που χωρίζεται σε πέντε ενότητες και τέσσερα βασικά επίπεδα

- Linux kernel
- Libraries
- Android Runtime
- Application framework
- Applications and Features



Σχήμα 3: Η αρχιτεκτονική Android

2.1.3.1 Linux kernel

Το Android χρησιμοποιεί τον ισχυρό πυρήνα Linux και υποστηρίζει ένα ευρύ φάσμα οδηγών (drivers) υλικού. Ο πυρήνας είναι η καρδιά του λειτουργικού συστήματος που διαχειρίζεται αιτήματα εισόδου και εξόδου από το λογισμικό. Αυτό παρέχει βασικές λειτουργίες συστήματος όπως διαχείριση διεργασιών, διαχείριση μνήμης, διαχείριση συσκευών όπως κάμερα, πληκτρολόγιο, οθόνη κλπ. Το Linux είναι πολύ καλό στη δικτύωση και δεν είναι απαραίτητο να συνδεθεί με το περιφερειακό υλικό. Ο ίδιος ο πυρήνας δεν αλληλεπιδρά άμεσα με τον χρήστη, αλλά αλληλεπιδρά με το κέλυφος και άλλα προγράμματα, καθώς και με τις συσκευές υλικού του συστήματος.

2.1.3.2 Libraries

Στην κορυφή του πυρήνα Linux υπάρχει ένα σύνολο βιβλιοθηκών που περιλαμβάνουν προγράμματα περιήγησης ιστού ανοιχτού κώδικα όπως το WebKit, την βιβλιοθήκη libc. Αυτές οι βιβλιοθήκες χρησιμοποιούνται για την αναπαραγωγή και εγγραφή ήχου και βίντεο. Το SQLite είναι μια βάση δεδομένων που είναι χρήσιμη για την αποθήκευση και την κοινή χρήση δεδομένων εφαρμογής. Οι βιβλιοθήκες SSL είναι υπεύθυνες για την ασφάλεια στο Διαδίκτυο κλπ.

2.1.3.3 Android Runtime

Το Android Runtime παρέχει ένα βασικό στοιχείο που ονομάζεται Dalvik Virtual Machine, το οποίο είναι ένα είδος εικονικής μηχανής Java. Είναι ειδικά σχεδιασμένο και βελτιστοποιημένο για Android. Το Dalvik VM είναι η εικονική μηχανή διεργασίας στο λειτουργικό σύστημα Android. Είναι ένα λογισμικό που τρέχει εφαρμογές σε συσκευές Android και χρησιμοποιεί βασικά χαρακτηριστικά Linux όπως διαχείριση μνήμης και multithreading που είναι σε γλώσσα Java. Επίσης το Dalvik VM επιτρέπει σε κάθε εφαρμογή Android να εκτελεί τη δική της διαδικασία και εκτελεί τα αρχεία σε μορφή .dex.

2.1.3.4 Application framework

Το Application framework παρέχει πολλές υπηρεσίες υψηλότερου επιπέδου σε εφαρμογές όπως διαχειριστής παραθύρων, σύστημα προβολής, διαχειριστής πακέτων, διαχειριστής πόρων κλπ. Οι προγραμματιστές εφαρμογών επιτρέπεται να κάνουν χρήση αυτών των υπηρεσιών στην εφαρμογή τους.

2.1.3.5 Applications and Features

Θα βρείτε όλες τις εφαρμογές Android στο επάνω επίπεδο και θα γράψετε την εφαρμογή σας και θα την εγκαταστήσετε σε αυτό το επίπεδο. Παραδείγματα τέτοιων εφαρμογών είναι επαφές, βιβλία, προγράμματα περιήγησης, υπηρεσίες κλπ. Κάθε εφαρμογή έχει διαφορετικό ρόλο στις συνολικές εφαρμογές.

Παράδειγμα:

- Διάταξη ακουστικών
- Αποθήκευση
- Συνδεσιμότητα: GSM / EDGE, IDEN, CDMA, Bluetooth, WI-FI, EDGE, 3G, NFC, LTE, GPS.
- Μηνύματα: SMS, MMS, C2DM, GCM
- Πολυγλωσσική υποστήριξη
- Δυνατότητα πολλαπλής αφής
- Βιντεοκλήση
- Αποτύπωση οθόνης
- Εξωτερική αποθήκευση
- Υποστήριξη ροής πολυμέσων
- Βελτιστοποιημένα γραφικά

2.1.4 Στοιχεία του Android

Τα βασικά στοιχεία οποιασδήποτε εφαρμογής Android είναι τα εξής:

- Activities
- Intent and broadcast receivers
- Services
- Content Providers
- Widgets and Notifications

2.1.4.1 Activities

Μια δραστηριότητα (activity) είναι το πρώτο βήμα για τη δημιουργία μιας εφαρμογής Android. Παρέχει το χώρο στον χρήστη για να κάνει οτιδήποτε και τα πάντα. Για παράδειγμα, το άνοιγμα μιας επαφής, η κλήση ενός αριθμού, κλπ. όλα γίνονται αλληλεπιδρώντας με ένα παράθυρο και αυτό το παράθυρο παρέχεται από μια δραστηριότητα. Παρέχεται ένα παράθυρο σε κάθε δραστηριότητα όπου γίνεται διεπαφή χρήστη. Γενικά, κάθε εφαρμογή Android έχει περισσότερες από μία δραστηριότητες. Υπάρχει μια "κύρια" δραστηριότητα. Όλες οι άλλες δραστηριότητες είναι δραστηριότητες "παιδιά". Υπάρχει μια στοίβα που ονομάζεται back stack. Κάθε φορά που ξεκινά ένα νέο παράθυρο, η προηγούμενη δραστηριότητα ωθείται στην πίσω στοίβα και διακόπτεται έως ότου ολοκληρωθεί η νέα δραστηριότητα. Μόλις πατηθεί το πίσω πλήκτρο της συσκευής σας, η νέα δραστηριότητα εμφανίζεται από τη στοίβα και καταστρέφεται και η προηγούμενη δραστηριότητα συνεχίζεται.

2.1.4.2 Intent and broadcast receivers

Το Android Intents είναι το μέσο επικοινωνίας, δηλαδή τα στοιχεία της εφαρμογής στέλνουν μηνύματα το ένα στο άλλο. Μπορεί να χρησιμοποιηθεί για το ερώτημα μιας ενέργειας από άλλο στοιχείο της εφαρμογής και μπορεί να χρησιμοποιηθεί για τη δημιουργία μιας νέας δραστηριότητας ή για την επίτευξη αποτελέσματος από άλλη δραστηριότητα. Μια υπηρεσία μπορεί να ξεκινήσει μεταβιβάζοντας πρόθεση για εκτέλεση μίας μόνο λειτουργίας. Μια μετάδοση μπορεί να σταλεί σε άλλες εφαρμογές περνώντας intents.

Τα intents έχουν 2 τύπους::

- **Implicit Intents:** Χρησιμοποιούνται για να δηλώσουν τις γενικές ενέργειες που πρέπει να εκτελεστούν, έτσι ώστε μέρος μιας άλλης εφαρμογής να μπορεί να το χειριστεί.
- **Explicit intents:** Χρησιμοποιούνται γενικά για να ξεκινήσει ένα νέο στοιχείο μέσα στην ίδια εφαρμογή. Αυτά τα στοιχεία ξεκινούν με το όνομά τους, δηλαδή πλήρως αναγνωρισμένο όνομα κλάσης.

Το Android Broadcast είναι ένα μήνυμα που εξαπλώνεται όταν συμβαίνει οποιοδήποτε συμβάν και λαμβάνετε από εφαρμογές. Τα intents μπορούν να χρησιμοποιηθούν για τη μετάδοση εκπομπών σε άλλες εφαρμογές, για παράδειγμα, όταν η συσκευή σας εκκινεί ή ενεργοποιείται το σύστημα δημιουργεί μια μετάδοση σε όλες τις εφαρμογές. Πρέπει να υπάρχει μια διαδικασία ή πρέπει να είναι κάτι που μπορεί να λάβει αυτές τις εκπομπές. Αυτοί οι υποδοχείς καλούνται ραδιοφωνικοί δέκτες. Για αυτό πρέπει να δηλώσετε έναν δέκτη στη δραστηριότητα που θα ασχοληθούμε κατά τον προγραμματισμό για τον ίδιο.

Υπάρχουν δύο τύποι broadcast:

- **Normal Broadcasts:** Είναι ασύγχρονης φύσης και πολλοί δέκτες μπορούν να ενεργοποιηθούν ταυτόχρονα που δεν έχουν καθορισμένη σειρά. Αλλά είναι πολύ αποτελεσματικοί.
- **Ordered Broadcasts:** Είναι σύγχρονης φύσης και η μετάδοση που λαμβάνεται από έναν δέκτη το μεταδίδει σε άλλους δέκτες. Οι εκπομπές παραδίδονται στον δέκτη σε ένα προς ένα και διαδοχική βάση. Οποιοσδήποτε δέκτης θα μεταβιβάσει το αποτέλεσμα σε άλλο δέκτη ή μπορεί να καταστρέψει εντελώς τη μετάδοση.

2.1.4.3 Services

Ένα άλλο σημαντικό στοιχείο μιας εφαρμογής Android είναι οι υπηρεσίες. Δεν παρέχει διεπαφή χρήστη. Κάνει μακροχρόνιες λειτουργίες στο παρασκήνιο και δεν τερματίζεται ακόμη και αν το στοιχείο που ξεκίνησε τερματίστηκε ή άλλαξε σε άλλη εφαρμογή. Μια υπηρεσία μπορεί να συνδεθεί με ένα στοιχείο που μπορεί ακόμη και να κάνει επικοινωνία μεταξύ διεργασιών. Μια υπηρεσία μπορεί να έχει δύο μορφές:

- **Ενεργοποιημένη (Started):** Μετά την έναρξη μιας υπηρεσίας, μπορεί να εκτελεστεί επ 'αόριστον και συνήθως εκτελεί μία λειτουργία. Κανένα αποτέλεσμα δεν επιστρέφεται στον χρήστη και μετά την ολοκλήρωση της εργασίας, θα πρέπει να τερματιστεί.
- **Δεσμευμένη (Bound):** Σε αυτήν την περίπτωση, ένα στοιχείο συνδέεται με μια υπηρεσία ώστε να μπορεί να ολοκληρωθεί μια συγκεκριμένη εργασία. Αυτός ο τύπος υπηρεσίας παρέχει διεπαφή πελάτη-διακομιστή και τα αιτήματα μπορούν να αποσταλούν, να λάβουν αιτήματα και να επιστρέψουν το αποτέλεσμα στον χρήστη. Η επικοινωνία μεταξύ διεργασιών επιτυγχάνεται μέσω αυτής της υπηρεσίας και το στοιχείο της εφαρμογής μπορεί να δεσμευτεί σε μια υπηρεσία. Πολλά στοιχεία μπορούν να δεσμευτούν σε αυτόν τον τύπο υπηρεσίας. Μετά την καταστροφή του στοιχείου, η υπηρεσία τερματίζεται.

2.1.4.4 Content Providers

Αυτά τα στοιχεία Android φέρνουν την αντικειμενοστρεφή λειτουργικότητα στο σύστημα. Οι πάροχοι περιεχομένου ενσωματώνουν δεδομένα και όπως υποδηλώνει το όνομα παρέχουν περιεχόμενο μιας διαδικασίας σε μια άλλη ως εκ τούτου ενεργεί ως διεπαφή. Παρέχει μια πύλη για πρόσβαση σε δεδομένα από ένα δομημένο σύνολο και για την πρόσβαση σε δεδομένα σε έναν πάροχο περιεχομένου πρέπει να δημιουργηθεί ένα αντικείμενο και αυτό το αντικείμενο ενεργεί ως πελάτης ενώ ο πάροχος περιεχομένου ενεργεί ως διακομιστής. Αυτό το αντικείμενο θα λάβει τα αιτήματα, ανακτά τα αποτελέσματα και επιστρέφει το αποτέλεσμα. Το Android διαθέτει παρόχους περιεχομένου που διαχειρίζεται βίντεο, ήχο κλπ. Αυτοί οι πάροχοι περιεχομένου είναι εσωτερικοί σε εφαρμογές Android.

2.1.4.5 Widgets and Notifications

Τα widget Android App είναι οι μικρές προβολές εφαρμογών. Αυτές οι προβολές μπορούν να ενσωματωθούν σε άλλες εφαρμογές. Μπορούν να λαμβάνουν ενημερώσεις σε περιοδική βάση. Είναι ένα γραφικό στοιχείο είναι μια γρήγορη προβολή της λειτουργικότητας και των δεδομένων μιας εφαρμογής. Αυτή η προβολή είναι προσβάσιμη από την αρχική οθόνη της συσκευής.

Τα widget έχουν τους ακόλουθους τύπους:

- **Informational Widget:** Αυτά τα widget θα εμφανίζουν μόνο τις πληροφορίες που είναι σημαντικές. Για παράδειγμα, οι πληροφορίες που εμφανίζονται στην αρχική οθόνη λέγοντας ότι ο χρόνος και η κατάσταση του καιρού είναι ένα widget αυτού του τύπου

- **Collection Widgets:** Αυτά τα widget μετακινούνται με κατεύθυνση από πάνω προς τα κάτω. Συλλογή πληροφοριών του ίδιου τύπου και δίνει δυνατότητα στον χρήστη να ανοίξει οποιοδήποτε από αυτά με πλήρη λεπτομέρεια. Παράδειγμα είναι η εφαρμογή e-mail που θα εμφανίζει όλα τα μηνύματα στα εισερχόμενα και στη συνέχεια επιτρέπει να ανοιχτεί οποιοδήποτε από αυτά
- **Control Widgets:** Εμφανίζει τις πιο συχνά χρησιμοποιούμενες λειτουργίες τις οποίες ο χρήστης μπορεί να θέλει να ελέγξει από την αρχική οθόνη. Για παράδειγμα, σε μια εφαρμογή βίντεο, μπορεί να κάνει παύση, αναπαραγωγή, διακοπή, μετάβαση στο προηγούμενο κομμάτι, μετακίνηση στο επόμενο κομμάτι
- **Hybrid Widgets:** Αυτά τα widget συνδυάζουν τα χαρακτηριστικά όλων των παραπάνω

2.2 IDE

IDE είναι μια συντομογραφία του Integrated Development Environment δηλαδή ένα ολοκληρωμένο περιβάλλον ανάπτυξης. Σε έναν IDE ένας προγραμματιστής συνήθως μπορεί να βρει ένα περιβάλλον ανάπτυξης κώδικα (editor), ένα σύστημα μεταγλώττισης (build) του κώδικα και έναν εντοπιστή σφαλμάτων (debugger) στην ίδια διεπιφάνεια. Ένας από τους σημαντικούς λόγους που ένας προγραμματιστής χρησιμοποιεί έναν IDE είναι ότι δεν χρειάζεται να αλλάζει από το ένα παράθυρο στο άλλο και μπορεί να οργανώνει την ροή της ανάπτυξης λογισμικού με αποτέλεσμα όλη η διαδικασία να γίνετε πιο γρήγορα.

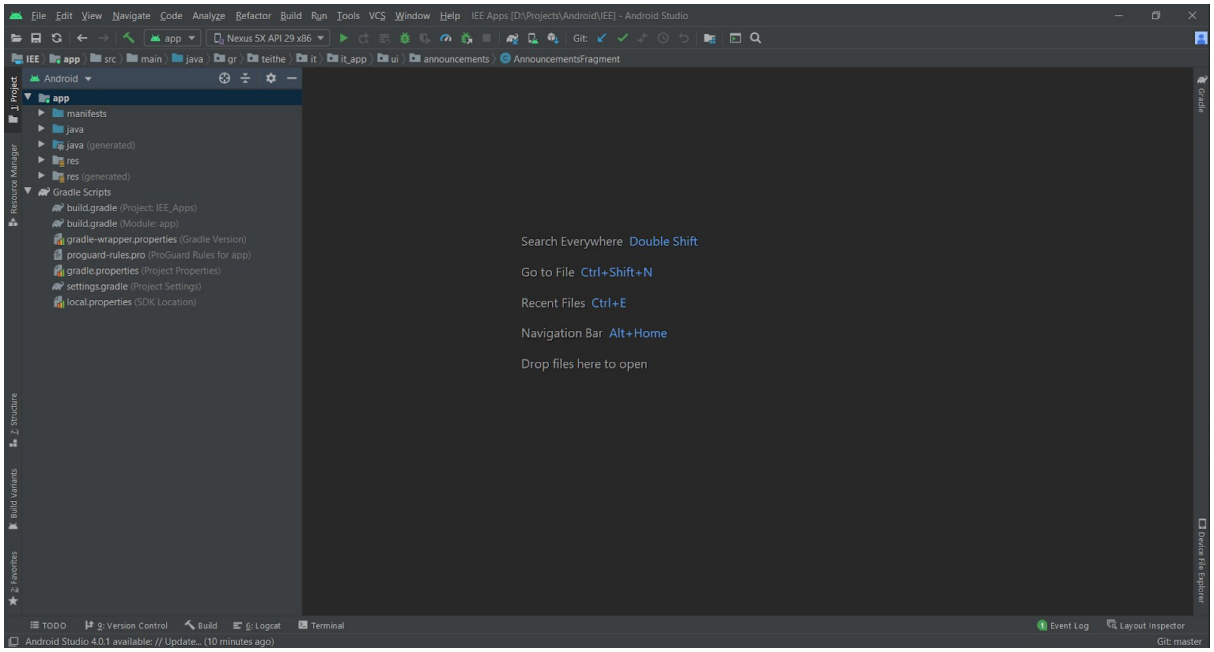
2.2.1 Android Studio

Το Android Studio είναι ο επίσημος IDE του λειτουργικού android για την ανάπτυξη και υλοποίηση εφαρμογών. Σχεδιάστηκε και αναπτύχθηκε αποκλειστικά για το android από την Jetbrains και η πρώτη δοκιμαστική έκδοση κυκλοφόρησε τον Μάιο του 2013 και μετέπειτα η επίσημη τον Δεκέμβριο του 2014. Επίσης το Android Studio μπορεί να χρησιμοποιηθεί από Windows, MacOS, Linux και μέχρι τον Μάιο του 2019 η κύρια γλώσσα ήταν η Java όπου την αντικατέστησε η Kotlin, ωστόσο μπορεί να χρησιμοποιηθεί και η C++.

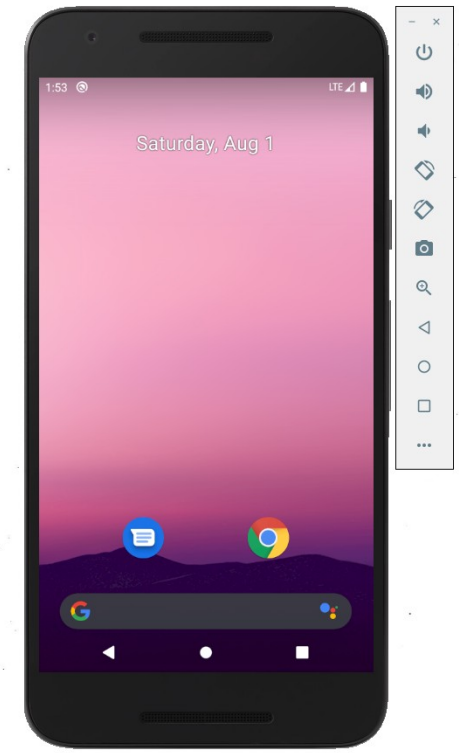
Κύρια χαρακτηριστικά του Android Studio είναι:

- Υποστήριξη του Gradle
- Επαναπροσδιορισμός (refactoring) και γρήγορες επιδιορθώσεις
- Εργαλεία Lint για έλεγχο απόδοσης, τη χρηστικότητα, τη συμβατότητα έκδοσης ή/και άλλα προβλήματα
- Δυνατότητες υλοποίησης του ProGuard και υπογραφής εφαρμογών
- Οδηγούς βάσει προτύπων για τη δημιουργία κοινών σχεδίων (designs) και στοιχείων (components)
- Μια πλούσια παλέτα σχεδίασης διεπιφάνειας που επιτρέπει στους χρήστες να μεταφέρουν στοιχεία στην οθόνη όπως και επιλογή προεπισκόπησης της οθόνης σε διαφορετικές διατάξεις
- Υποστήριξη για τη δημιουργία εφαρμογών Android Wear
- Ενσωματωμένη υποστήριξη για το Google Cloud Platform

- Εικονική συσκευή Android (Emulator) για την εκτέλεση εφαρμογών και εντοπισμό των σφαλμάτων



Σχήμα 4: Το Android Studio



Σχήμα 5: Εικονική συσκευή (emulator)

2.2.2 Java

Η Java είναι μία γλώσσα προγραμματισμού όπου η ανάπτυξη της ξεκίνησε από τον James Gosling και την Sun Microsystems το 1991 και η πρώτη επίσημη παρουσίαση της έγινε το 1996. Συντακτικά μοιάζει με την γλώσσα προγραμματισμού C και C++ και διατηρεί μεγάλη συγγένεια με την C++ αλλά με ποιο έντονο αντικειμενοστρεφή χαρακτήρα. Ο αρχικός σκοπός της Java ήταν να μπορούν να αναπτυχθούν και να εκτελεστούν εφαρμογές σε μικροσυσκευές. Το 2007 η Sun Microsystems κάνει την εικονική μηχανή (Java Virtual Machine - JVM) της Java ανοιχτού κώδικα (open source) και μετέπειτα εξαγοράστηκε από την Oracle το 2010.

Η Java είναι μια γλώσσα μεταγλωττιζόμενη (compiled) και διερμηνευμένη (interpreted). Μεταγλωττιζόμενη (compiled) είναι μια γλώσσα όπου ένας μεταγλωττιστής (compiler) μετατρέπει τον πηγαίο κώδικα (source code) σε γλώσσα μηχανής και διερμηνευόμενη (interpreted) είναι όταν ένας διερμηνέας εκτελεί οδηγίες απευθείας σε μια ακολουθία ενός ή περισσότερων υπορουτίνων και, στη συνέχεια, σε γλώσσα μηχανής. Στην Java ο κώδικας γίνετε compile σε bytecode και μετά γίνεται interpreted απο την εικονική μηχανή JVM.

Υποστηρίζει το μοντέλο του αντικειμενοστραφή προγραμματισμού και είναι μια γλώσσα υψηλού επιπέδου, δηλαδή, η σύνταξη του κώδικα είναι πιο κοντά στην ανθρώπινη λογική αντί της μηχανής. Κάνει αυστηρό έλεγχο τύπων με αποτέλεσμα κάθε μεταβλητή να δημιουργείτε με τον τύπο της και δεν μπορεί να αλλάξει μελλοντικά. Η εκτέλεση προγραμμάτων ελέγχεται από μηχανισμούς ασφαλείας αποτρέποντας την εκτέλεση κακόβουλου κώδικα. Είναι από τις μοναδικές γλώσσες που έχει ενσωματωμένη υποστήριξη multi-threading.

Η διαχείριση της μνήμης γίνεται από ένα ενσωματωμένο πρόγραμμα, τον συλλέκτη απορριμμάτων (garbage collector), με αποτέλεσμα να μην χρειάζεται η παρέμβαση του προγραμματιστή. Τέλος, ένα προγρογραμμα γραμμένο σε Java μπορεί να εκτελεστεί εξίσου το ίδιο καλά σε Windows, Linux, Unix, Macintosh χωρίς να γίνει κάποια αλλαγή στον πηγαίο κώδικα.

2.2.3 Kotlin

Η Kotlin είναι μια γλώσσα προγραμματισμού γενικού τύπου που μπορεί να δουλεύει σε όλα τα λειτουργικά συστήματα (cross-platform). Η κατασκευή της ξεκίνησε τον Ιούλιο του 2011 απο την JetBrains και τον Φεβρουάριο του 2012 έγινε ανοιχτού κώδικα (open source). Η πρώτη επίσημη έκδοση κυκλοφόρησε τον Φεβρουάριο του 2016 και το 2017 η Google ανακοίνωσε την πλήρη υποστήριξη της Kotlin στο λειτουργικό Android. Τον Μάιο του 2019 η Kotlin γίνεται πλέον η επιθυμητή γλώσσα για προγραμματιστές και την ανάπτυξη εφαρμογών μετά από την ανακοίνωση της Google. Σχεδιάστηκε να λειτουργεί πλήρως με την εικονική μηχανή της Java (JVM) αλλά η σύνταξη είναι πιο περιεκτική.

2.3 Architectural Patterns

Ένα αρχιτεκτονικό μοτίβο είναι μια γενική, επαναχρησιμοποιήσιμη λύση σε ένα συχνά εμφανιζόμενο πρόβλημα στην αρχιτεκτονική λογισμικού σε ένα δεδομένο πλαίσιο. Τα αρχιτεκτονικά

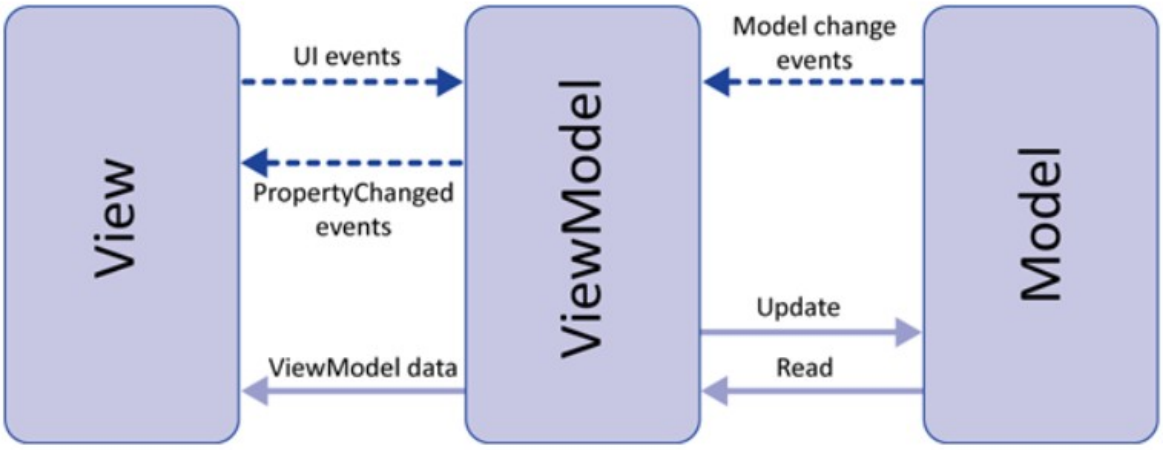
μοτίβα αντιμετωπίζουν διάφορα ζητήματα στη μηχανική λογισμικού, όπως περιορισμούς απόδοσης υλικού υπολογιστή, υψηλή διαθεσιμότητα και ελαχιστοποίηση επιχειρηματικού κινδύνου. Ορισμένα αρχιτεκτονικά σχέδια έχουν εφαρμοστεί σε πλαίσια λογισμικού.

2.3.1 MVVM

Το MVVM (Model-View-ViewModel) είναι ένα από τα αρχιτεκτονικά σχέδια που βελτιώνει τον διαχωρισμό των λειτουργιών, επιτρέπει τον διαχωρισμό της λογικής διεπαφής χρήστη (UI) από την επιχειρηματική λογική (business logic).

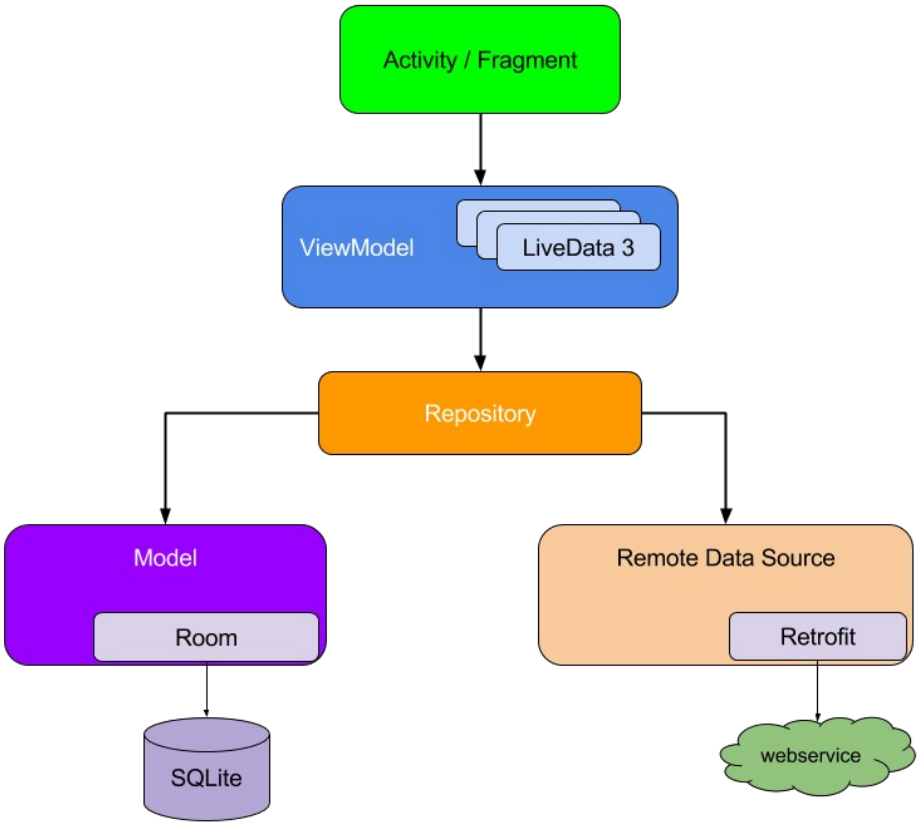
Κατά κύριο λόγο το MVVM έχει 3 επίπεδα:

- **Μοντέλο (Model)** - Το μοντέλο είναι υπεύθυνο για τα δεδομένα της εφαρμογής. Μια καλή στρατηγική είναι η υλοποίηση τους με παρατηρητές (observables) για την επικοινωνία με το ViewModel
- **ViewModel** - Το ViewModel επικοινωνεί με το Model και προετοιμάζει την επικοινωνία με το View με την χρήση των observers. Λειτουργεί σαν μεσολαβητής και καλή στρατηγική είναι να μην ξέρει με ποιο View επικοινωνεί.
- **View** - Το View είναι η διεπαφή που βλεπει και αλληλεπιδρά ο χρήστης. Επικοινωνεί με το ViewModel με τους observers και ανανεώνει στοιχεία διεπαφής αναλόγως.



Σχήμα 6: Το MVVM

Η υλοποίηση του MVVM στην Android εφαρμογή θα γίνει με την βοήθεια της σουίτας Jetpack όπου υλοποιεί την κλάση ViewModel και LiveData για το observable κομμάτι. Θα χρησιμοποιηθεί η βιβλιοθήκη Retrofit για την υλοποίηση της επικοινωνίας με το API και το DataBinding για την δέσμευση των μεταβλητών στο UI.



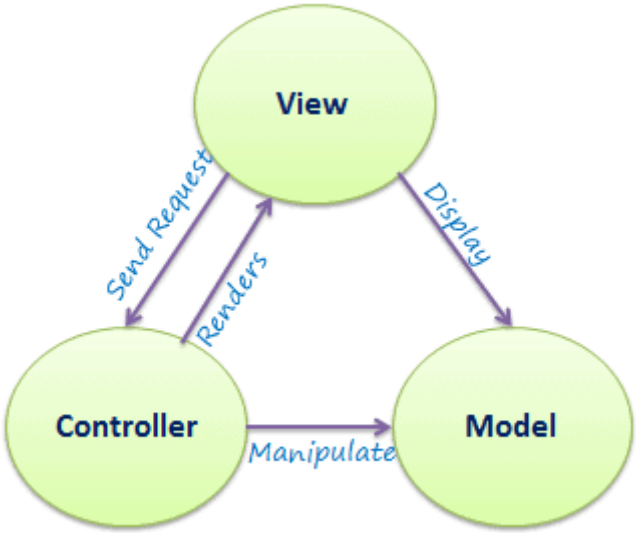
Σχήμα 7: Το MVVM με την χρήση του Jetpack

2.3.2 MVC

Το MVC (Model-View-Controller) είναι ένα αρχιτεκτονικό σχέδιο που διαχωρίζει μια εφαρμογή σε τρία κύρια λογικά στοιχεία: το μοντέλο (model), την προβολή (view) και τον ελεγκτή (controller). Κάθε ένα από αυτά τα στοιχεία είναι κατασκευασμένο για να χειρίζεται συγκεκριμένες πτυχές ανάπτυξης μιας εφαρμογής. Το MVC είναι ένα από τα πιο συχνά χρησιμοποιούμενα βιομηχανικά πρότυπα ανάπτυξης ιστοσελίδων για τη δημιουργία επεκτάσιμων και επεκτάσιμων έργων. Το λειτουργικό σύστημα του Android είναι βασισμένο στο MVC.

- **Model:** Το Model αντιστοιχεί σε όλη τη λογική που σχετίζεται με τα δεδομένα με την οποία εργάζεται ο χρήστης. Αυτό μπορεί να αντιπροσωπεύει είτε τα δεδομένα που μεταφέρονται μεταξύ των στοιχείων Προβολή και Ελεγκτή ή οποιαδήποτε άλλα δεδομένα που σχετίζονται με τη λογική της επιχείρησης. Για παράδειγμα, ένα αντικείμενο πελάτη θα ανακτήσει τις πληροφορίες πελατών από τη βάση δεδομένων, θα το χειριστεί και θα τα ενημερώσει ξανά στη βάση δεδομένων ή θα τα στείλει για εμφάνιση.
- **View:** Το View χρησιμοποιείται για όλη τη λογική διεπαφής χρήστη της εφαρμογής. Για παράδειγμα, η προβολή πελάτη θα περιλαμβάνει όλα τα στοιχεία διεπαφής χρήστη, όπως πλαίσια κειμένου, αναπτυσσόμενα μενού κ.λπ. με τα οποία αλληλεπιδρά ο τελικός χρήστης.

- **Controller:** Οι controllers λειτουργούν ως διεπαφή μεταξύ των στοιχείων του model και του view για την επεξεργασία όλων των επιχειρηματικών λογικών, εισερχόμενων αιτημάτων, χειρισμό δεδομένων χρησιμοποιώντας το στοιχείο μοντέλου και αλληλεπίδραση με τα view για απόδοση της τελικής παραγωγής. Για παράδειγμα, ο ελεγκτής πελατών θα χειριστεί όλες τις αλληλεπιδράσεις και τις εισόδους από την προβολή πελατών και θα ενημερώσει τη βάση δεδομένων χρησιμοποιώντας το μοντέλο πελάτη. Ο ίδιος controllre θα χρησιμοποιηθεί για την προβολή των δεδομένων πελατών.



Σχήμα 8: Η λειτουργία του MVC

2.4 Design patterns

Τα μοτίβα σχεδιασμού (design patterns) είναι τυπικές λύσεις για προβλήματα που εμφανίζονται συνήθως στη σχεδίαση λογισμικού. Είναι σαν προκατασκευασμένα σχεδιαγράμματα που μπορούν να προσαρμοστούν για να λύσουν ένα πρόβλημα. Στην υλοποίηση της εφαρμογής έχουν εφαρμοστούν τα μοτίβα Singleton, Adapter και Observer.

2.4.1 Singleton

Το Singleton είναι ένα δημιουργικό μοτίβο σχεδίασης που επιτρέπει στον προγραμματιστή να διασφαλίσει ότι μια κλάση (class) έχει μόνο μία παρουσία (instance) παρέχοντας παράλληλα ένα δημόσιο σημείο πρόσβασης σε αυτήν την παρουσία. Υλοποιείται με 2 βασικά βήματα:

- Ορίζοντας τον δομιτή (constructor) ως ιδιωτικό (private) για να αποτρέψει από άλλες κλάσεις την χρήση του χειριστή (operator) new
- Δημιουργόντας μία στατική (static) μέθοδο που θα λειτουργεί σαν δομιτής αποθηκεύοντας το instance σε μια τοπική στατική μεταβλητή

```
class SingletonDemo
{
    private static SingletonDemo instance;

    private SingletonDemo()
    {

    }

    public static SingletonDemo getInstance()
    {
        if(instance == null)
        {
            instance = new SingletonDemo();
        }

        return instance;
    }

    public String getDemo()
    {
        return "Demo";
    }
}

class Main
{
    public static void main(String[] args)
    {
        System.out.println(SingletonDemo.getInstance().getDemo());
    }
}
```

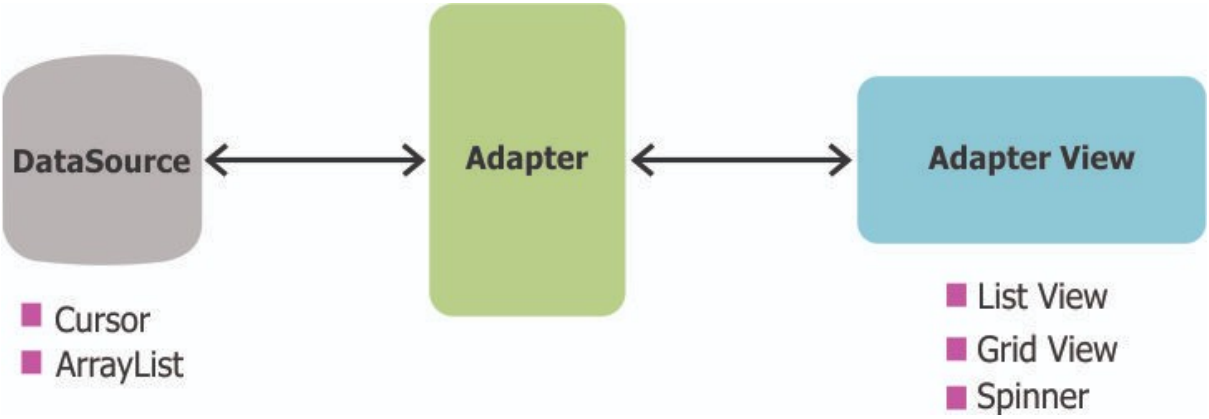
Σχήμα 9: Υλοποίηση του Singleton σε Java

2.4.2 Adapter

Ο Apdater είναι ένα δομικό μοτίβο σχέδιο που επιτρέπει σε αντικείμενα με ασύμβατες διεπαφές να συνεργάζονται. Υλοποιείται ως εξής:

- Ο adapter λαμβάνει μια διεπαφή που είναι συμβατή με ένα από τα αντικείμενα
- Το υπάρχων αντικείμενο χρησιμοποιώντας την διεπαφή μπορεί να καλέσει τις μεθόδους του adapter
- Ο adapter μεταβιβάζει το αίτημα στο δεύτερο αντικείμενο στην μορφή που περιμένει

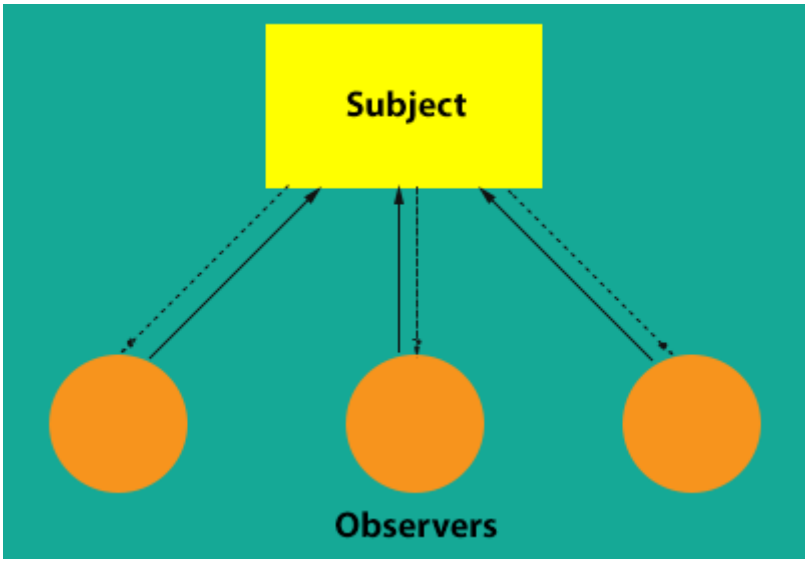
Είναι πιθανόν ο adapter να είναι διπλής κατεύθυνσης, δηλαδή να μετατρέπει αιτήματα και από τις 2 πλευρές.



Σχήμα 10: Παράδειγμα χρήσης του adapter

2.4.3 Observer

Το Observer είναι ένα μοτίβο συμπεριφοράς που επιτρέπει να οριστεί ένα μηχανισμός παρατηρητικότητας (observe) σε ένα αντικείμενο έτσι ώστε να ειδοποιεί άλλα αντικείμενα για αλλαγές το αντικείμενο που παρατηρούν.



Σχήμα 11: Το observer pattern

2.5 Android Jetpack

Το Jetpack είναι μια σειρά από βιβλιοθήκες που βοηθούν τους προγραμματιστές να ακολουθούν τις βέλτιστες πρακτικές, να μειώνουν τον περιττό κώδικα και να γράφουν κώδικα που λειτουργεί με συνέπεια σε όλες τις εκδόσεις και συσκευές Android με αποτέλεσμα να μπορούν να επικεντρώνονται στον κώδικα που τους ενδιαφέρει.

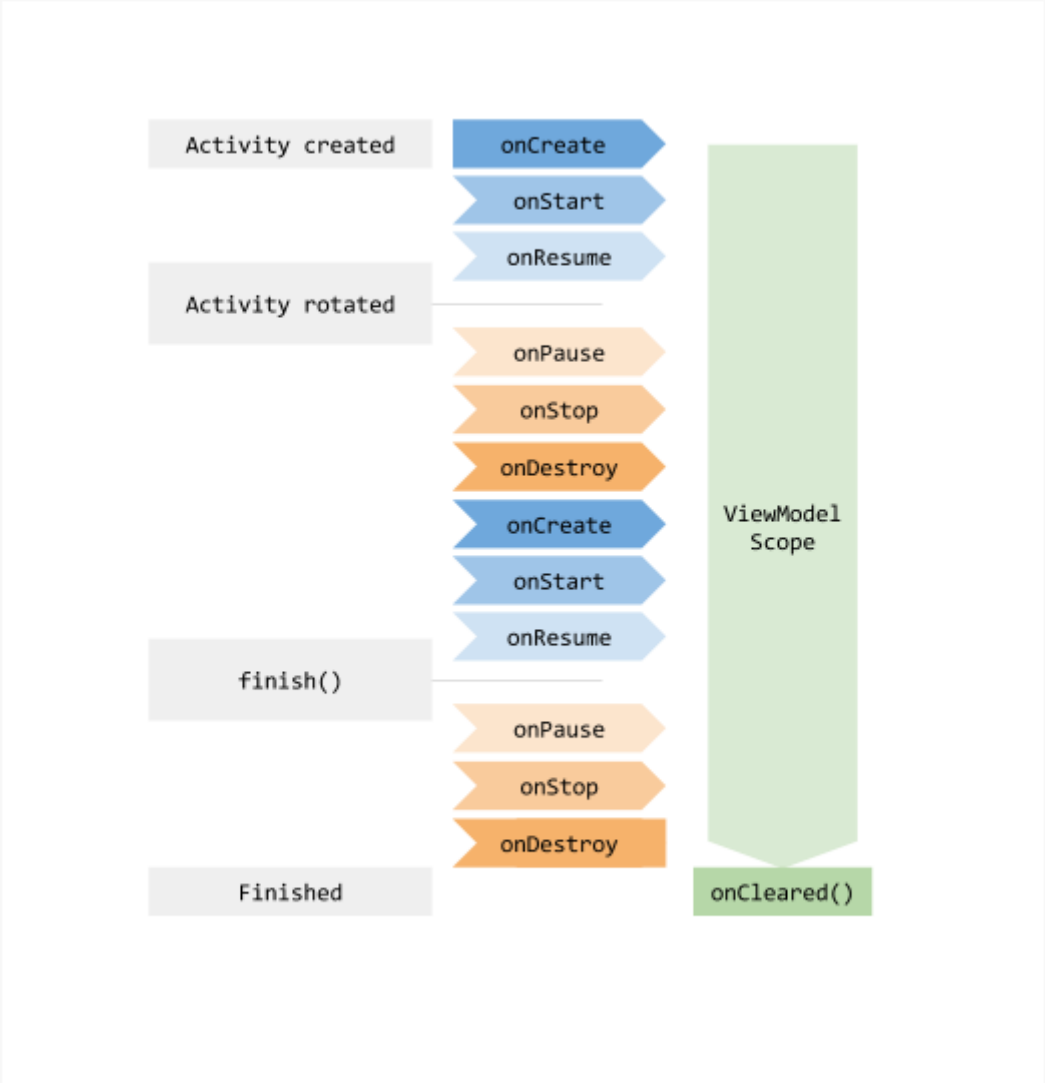
2.5.1 ViewModel

Το Android Jetpack παρέχει την βοηθητική κλάση ViewModel για τον ελεγκτή διεπαφής (UI controller) που είναι υπεύθυνος για την προετοιμασία δεδομένων για τη διεπαφή χρήστη. Τα αντικείμενα ViewModel διατηρούνται αυτόματα κατά τη διάρκεια αλλαγών διαμόρφωσης (configuration changes), έτσι ώστε τα δεδομένα που κατέχουν να είναι άμεσα διαθέσιμα στην επόμενη παρουσία (instance).

Το Android διαχειρίζεται τους κύκλους ζωής των ελεγκτών διεπαφής χρήστη, όπως δραστηριότητες (activities) και τμήματα (fragments) και ενδέχεται να αποφασίσει να καταστρέψει ή να δημιουργήσει ξανά έναν ελεγκτή διεπαφής χρήστη ως απάντηση σε συγκεκριμένες ενέργειες χρήστη ή συμβάντα συσκευών, παράδειγμα η περιστροφή οθόνης, κάτι που δεν μπορεί να ελέγξει ο προγραμματιστής. Εάν το σύστημα καταστρέψει ή δημιουργήσει εκ νέου έναν ελεγκτή διεπαφής χρήστη, τυχόν προσωρινά δεδομένα που σχετίζονται με το περιβάλλον εργασίας χρήστη θα χαθούν.

Ένα άλλο πρόβλημα είναι ότι οι ελεγκτές διεπαφής χρήστη πρέπει συχνά να πραγματοποιούν ασύγχρονες κλήσεις που ενδέχεται να χρειαστούν λίγο χρόνο για να επιστρέψουν. Ο ελεγκτής διεπαφής χρήστη πρέπει να διαχειρίζεται αυτές τις κλήσεις και να διασφαλίζει ότι το σύστημα τις καθαρίζει αφού καταστραφεί για να αποφευχθούν πιθανές διαρροές μνήμης (memory leaks). Αυτή η διαχείριση απαιτεί πολλή συντήρηση και σε περίπτωση που το αντικείμενο ξαναδημιουργηθεί για αλλαγή διαμόρφωσης, είναι σπατάλη πόρων, καθώς το αντικείμενο ενδέχεται να χρειαστεί να ξανά καλέσει κλήσεις που έχει ήδη κάνει.

Αυτά τα προβλήματα λύνει το ViewModel διαχωρίζοντας τα δεδομένα από την λογική του ελεγκτή χρήστη. Τα αντικείμενα ViewModel καλύπτονται στον κύκλο ζωής που περνά στο ViewModelProvider κατά τη λήψη του ViewModel και παραμένει στη μνήμη μέχρι τον κύκλο ζωής που σκοπεύει να φύγει μόνιμα σε περίπτωση δραστηριότητας (activity), όταν τελειώσει (terminate), στην περίπτωση τμήματος (fragment), όταν αποσυνδέεται (detached).



Σχήμα 12: Ο κύκλος ζωής ενός ViewModel

2.5.2 LiveData

LiveData είναι μια κλάση που υλοποιεί ένα παρατηρήσιμο (observable) αντικείμενο . Σε αντίθεση με ένα συνηθισμένο παρατηρήσιμο, το LiveData έχει επίγνωση του κύκλου ζωής πράγμα που σημαίνει ότι σέβεται τον κύκλο ζωής άλλων στοιχείων της εφαρμογής, όπως activities, fragments ή services. Αυτή η επίγνωση διασφαλίζει ότι το LiveData ενημερώνει μόνο τους παρατηρητές που βρίσκονται σε κατάσταση ενεργού κύκλου ζωής.

Το LiveData θεωρεί ότι ένας παρατηρητής ο οποίος αντιπροσωπεύεται από την κλάση Observer, βρίσκεται σε ενεργή κατάσταση εάν ο κύκλος ζωής του βρίσκεται στην κατάσταση ΕΝΑΡΞΗ ή ΕΚΚΙΝΗΣΗ. Το LiveData ειδοποιεί τους ενεργούς παρατηρητές μόνο για ενημερώσεις. Οι ανενεργοί παρατηρητές που έχουν εγγραφεί για να παρακολουθούν αντικείμενα LiveData δεν ειδοποιούνται για αλλαγές.

Μπορεί ένας παρατηρητής να καταχωρηθεί σε συνδυασμό με ένα αντικείμενο που εφαρμόζει τη διεπαφή LifecycleOwner όπου αυτή η σχέση επιτρέπει στον παρατηρητή να αφαιρεθεί όταν η κατάσταση του αντίστοιχου αντικειμένου κύκλου ζωής αλλάξει σε ΚΑΤΑΒΟΛΗ. Αυτό είναι

ιδιαίτερα χρήσιμο για activities και fragments επειδή μπορούν να παρατηρήσουν με ασφάλεια αντικείμενα LiveData και να μην ανησυχούν για διαρροές μνήμης.

2.5.3 Navigation

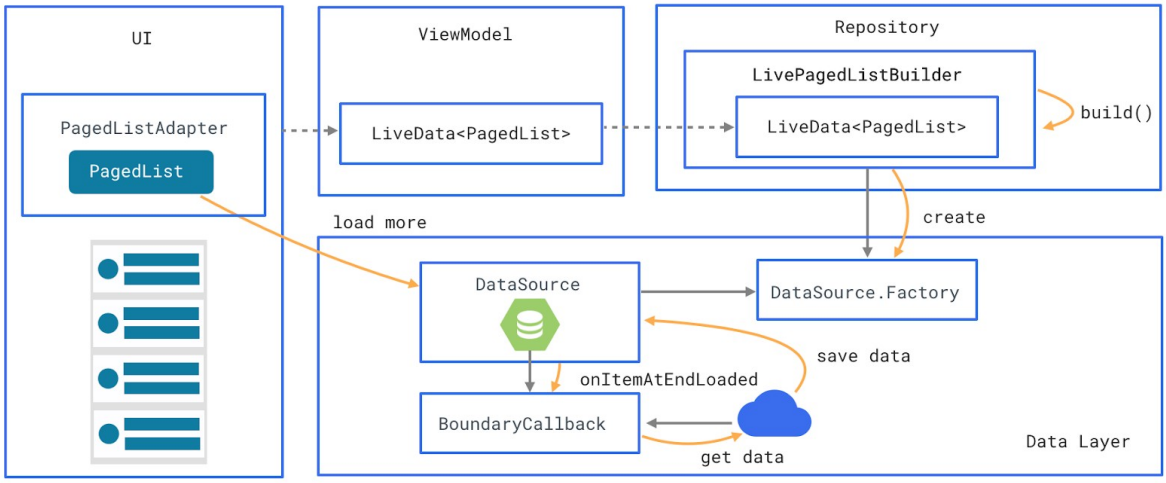
Το Navigation Component αναφέρεται στις αλληλεπιδράσεις που επιτρέπουν στους χρήστες να πλοηγούνται σε διάφορα σημεία περιεχομένου μιας εφαρμογής. Βοηθά τον προγραμματιστή να εφαρμόσει την πλοήγηση από απλά κλικ σε κουμπιά έως πιο περίπλοκα μοτίβα, όπως στο appbar και στο navigation drawer. Το Navigation διασφαλίζει επίσης μια συνεπή και προβλέψιμη εμπειρία χρήστη ακολουθώντας ένα καθιερωμένο σύνολο κανόνων και αρχών και αποτελείται από τα παρακάτω τρία βασικά μέρη:

- **Navigation graph** - Ένα XML αρχείο που περιέχει όλες τις πληροφορίες που σχετίζονται με την πλοήγηση σε μια κεντρική τοποθεσία. Περιλαμβάνει όλες τις μεμονωμένες περιοχές περιεχομένου σε μια εφαρμογή που ονομάζονται προορισμοί, καθώς και τις πιθανές διαδρομές που μπορεί να ακολουθήσει ένας χρήστης.
- **NavHost** - Ένα κενό κοντέινερ που εμφανίζει τους προορισμούς από το navigation graph. Το navigation component περιέχει μια προεπιλεγμένη εφαρμογή NavHost, το NavHostFragment, η οποία εμφανίζει τους προορισμούς.
- **NavController** - Ένα αντικείμενο που διαχειρίζεται την πλοήγηση της εφαρμογής σε ένα NavHost. Το NavController ενορχηστρώνει την ανταλλαγή περιεχομένου προορισμού στο NavHost καθώς οι χρήστες μετακινούνται σε ολόκληρη την εφαρμογή. Κατά την περιήγηση στην εφαρμογή συγκεκριμένες ενέργειες λένε στον NavController ότι ο χρήστης θέλει ν πλοηγηθεί είτε κατά μήκος μιας συγκεκριμένης διαδρομής στο navigation graph είτε απευθείας σε έναν συγκεκριμένο προορισμό και στη συνέχεια το NavController εμφανίζει τον κατάλληλο προορισμό στο NavHost.

2.5.4 Paging

Η βιβλιοθήκη Paging βοηθάει να φορτώνονται και να εμφανίζονται μικρά κομμάτια δεδομένων κάθε φορά, σε περίπτωση που υπάρχει μεγάλος όγκος δεδομένων. Η φόρτωση μερικών δεδομένων μειώνει:

- Την χρήση του εύρους ζώνης του δικτύου
- Τους πόρους του συστήματος
- Τον χρόνο φόρτωσης των δεδομένων
- Τον χρόνο εμφάνισης στο UI



Σχήμα 13: Η υλοποίηση του paging

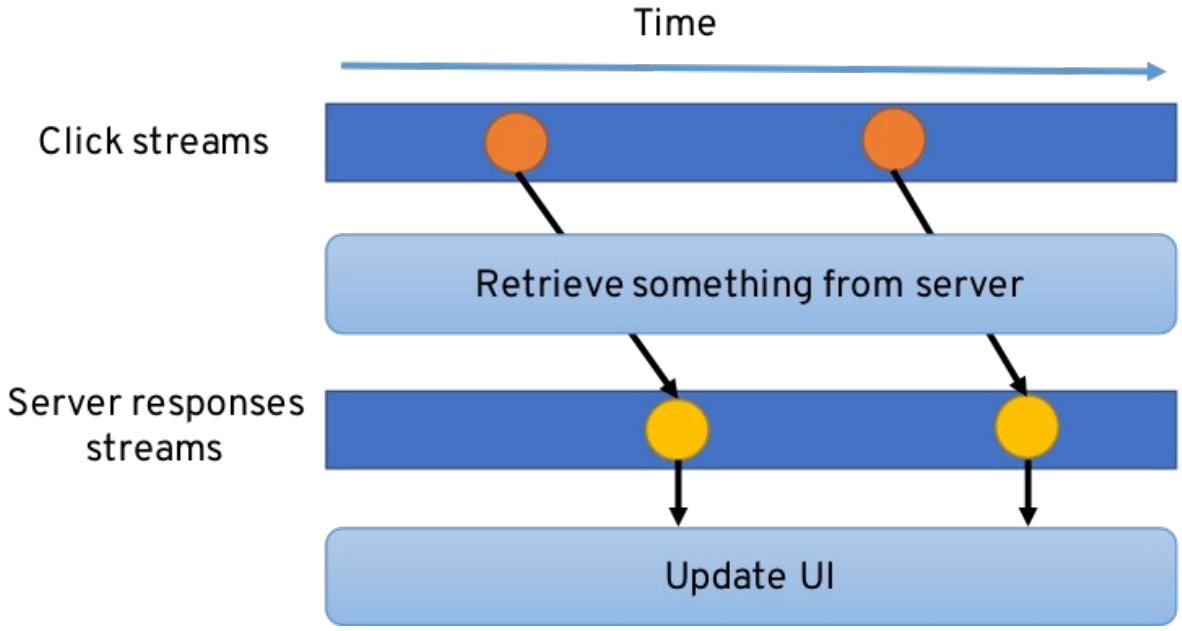
2.5.5 DataBinding

Το DataBinding είναι μια βιβλιοθήκη υποστήριξης που επιτρέπει στον προγραμματιστή να συνδέσει τα στοιχεία διεπαφής χρήστη (UI components) στα layouts με πηγές δεδομένων χρησιμοποιώντας μια δηλωτική μορφή και όχι μέσω προγραμματισμού. Αυτό ελαχιστοποιεί τον απαραίτητο κώδικα στη λογική της εφαρμογής για σύνδεση με τα στοιχεία διεπαφής χρήστη.

Το DataBinding δημιουργεί αυτόματα μια δεσμευτική κλάση με βάση αυτή του layout. Αυτή η κλάση διατηρεί όλες τις συνδέσεις από τις ιδιότητες διάταξης, για παράδειγμα την καθορισμένη μεταβλητή στα αντίστοιχα στοιχεία διεπαφής. Παρέχει επίσης ρυθμιστές (setters) για τα στοιχεία δεδομένων απο το layout. Το όνομα της δημιουργούμενης κλάσης βασίζεται στο όνομα του αρχείου layout. Αυτό το όνομα μετατρέπεται στην σύμβαση ονομασίας (naming convention) Pascal και προστίθεται το Binding στο τέλος. Για παράδειγμα, εάν το αρχείο διάταξης ονομάζεται activity_main.xml, η κλάση θα ονομάζεται ActivityMainBinding.

2.6 Reactive Programming

Reactive Programming είναι προγραμματισμός που προγραμματίζει με ασύγχρονες ροές δεδομένων. Η ροή είναι μια ακολουθία συνεχιζόμενων συμβάντων που έχουν παραγγελθεί στο χρόνο. Μπορεί να εκπέμπει τρία διαφορετικά πράγματα: μια τιμή (κάποιου τύπου), ένα σφάλμα ή ένα “ολοκληρωμένο” σήμα. Καταγράφουμε αυτά τα εκπεμπόμενα συμβάντα μόνο ασύγχρονα, καθορίζοντας μια συνάρτηση που θα εκτελείται όταν εκπέμπεται μια τιμή, μια άλλη λειτουργία όταν εκπέμπεται ένα σφάλμα και μια άλλη λειτουργία όταν εκπέμπεται “ολοκληρωμένο”. Η “ακρόαση” στη ροή ονομάζεται εγγραφή (subscribing). Οι συναρτήσεις που ορίζουμε είναι παρατηρητές. Η ροή είναι το αντικείμενο που παρατηρείται.



Σχήμα 14: Παράδειγμα του Reactive Programming

2.6.1 RxJava

Το RxJava είναι μια υλοποίηση Java VM του ReactiveX και στον πυρήνα του απλοποιεί την ανάπτυξη κώδικα επειδή αυξάνει το επίπεδο της αφαίρεσης (abstraction) γύρω από τα νήματα (threads). Δηλαδή ένας προγραμματιστής δεν χρειάζεται να ανησυχεί για τις λεπτομέρειες του τρόπου εκτέλεσης λειτουργιών που πρέπει να εμφανίζονται σε διαφορετικά νήματα. Αυτό είναι ιδιαίτερα ελκυστικό δεδομένου ότι το νήμα είναι δύσκολο να γίνει σωστό και εάν δεν εφαρμοστεί σωστά, μπορεί να σφάλματα που είναι δύσκολα στην εύρεση και διόρθωση.

2.7 Firebase

Το Firebase είναι μια πλατφόρμα που αναπτύχθηκε από την Google για τη δημιουργία εφαρμογών για κινητά και ιστούς. Αρχικά ήταν μια ανεξάρτητη εταιρεία που ιδρύθηκε το 2011. Το 2014, η Google απέκτησε την πλατφόρμα και είναι πλέον η κορυφαία προσφορά τους για ανάπτυξη εφαρμογών. Η πλατφόρμα Firebase έχει 18 προϊόντα χωρισμένα σε τρεις ομάδες: Ανάπτυξη (Develop), Ποιότητα (Quality) και Ανάπτυξη (Grow).

2.7.1 Realtime Database

Η βάση δεδομένων Realtime είναι μια βάση δεδομένων που φιλοξενείται από το cloud. Τα δεδομένα αποθηκεύονται ως JSON και συγχρονίζονται σε πραγματικό χρόνο σε κάθε συνδεδεμένο πελάτη. Όταν ένας χρήστης δημιουργεί εφαρμογές πολλαπλών πλατφορμών με SDK iOS, Android και JavaScript, όλοι οι πελάτες (clients) μοιράζονται μία παρουσία της βάσης δεδομένων και λαμβάνουν αυτόματα ενημερώσεις με τα νεότερα δεδομένα.

Βασικές δυνατότητες:

- **Σε πραγματικό χρόνο**: Αντί για τυπικά αιτήματα HTTP, το Firebase Realtime Database χρησιμοποιεί συγχρονισμό δεδομένω δλαδή κάθε φορά που αλλάζουν τα δεδομένα, οποιαδήποτε συνδεδεμένη συσκευή λαμβάνει αυτήν την ενημέρωση εντός χιλιοστών του δευτερολέπτου.
- **Εκτός σύνδεσης:** Οι εφαρμογές Firebase παραμένουν αποκριτικές ακόμη και όταν είναι εκτός σύνδεσης, επειδή το SDK Firebase Realtime Database διατηρεί τα δεδομένα στο δίσκο. Μόλις αποκατασταθεί η συνδεσιμότητα, η συσκευή πελάτη λαμβάνει τις αλλαγές που έχασε, συγχρονίζοντας την με την τρέχουσα κατάσταση της βάσης δεδομένων.
- **Προσβάσιμο από συσκευές πελάτη:** Η Firebase Realtime Database μπορεί να προσεγγιστεί απευθείας από κινητή συσκευή ή πρόγραμμα περιήγησης ιστού και δεν χρειάζεται διακομιστής. Η ασφάλεια και η επικύρωση δεδομένων διατίθενται μέσω των κανόνων ασφαλείας του Firebase Realtime Database, κανόνες βάσει έκφρασης που εκτελούνται όταν διαβάζονται ή γράφονται τα δεδομένα.

2.7.2 Cloud Functions

Το Cloud Functions for Firebase είναι ένα πλαίσιο χωρίς διακομιστή που επιτρέπει να εκτελεστεί αυτόματα κώδικας backend ως απόκριση σε συμβάντα που ενεργοποιούνται από λειτουργίες Firebase και αιτήματα HTTPS. Ο κώδικας JavaScript ή TypeScript αποθηκεύεται στο cloud της Google και εκτελείται σε διαχειριζόμενο περιβάλλον.

Βασικές Δυνατότητες:

- **Ενσωματώνει την πλατφόρμα Firebase:** Οι συναρτήσεις μπορούν να ανταποκρίνονται σε συμβάντα που δημιουργούνται από διάφορες λειτουργίες του Firebase και του Google Cloud, από ενεργοποιήσεις ελέγχου ταυτότητας Firebase έως Cloud Storage Triggers.
- **Μηδενική συντήρηση:** Ο χρήστης μπορεί να αναπτύξει τον κώδικα σε JavaScript ή TypeScript στους διακομιστές του Firebase με μία εντολή από τη γραμμή εντολών. Το Firebase κλιμακώνει αυτόματα υπολογιστικούς πόρους για να ταιριάζει με τα μοτίβα χρήσης των χρηστών. Ο χρήστης δεν ανησυχεί για διαπιστευτήρια, διαμόρφωση διακομιστή ή παροχή νέων διακομιστών.
- **Διατηρεί τη λογική ιδιωτική και ασφαλή:** Σε πολλές περιπτώσεις, οι προγραμματιστές προτιμούν να ελέγχουν τη λογική εφαρμογών στον διακομιστή για να αποφύγουν την παραβίαση από την πλευρά του πελάτη. Επίσης, μερικές φορές δεν είναι επιθυμητό να επιτρέπεται η αντιστροφή του κώδικα. Το Cloud Functions είναι πλήρως μονωμένο από τον πελάτη, οπότε ο χρήστης μπορεί να είναι σίγουρος ότι είναι ιδιωτικός και πάντα κάνει ακριβώς αυτό που θέλει.

2.7.3 Cloud Messaging

Το Firebase Cloud Messaging είναι μια λύση ανταλλαγής μηνυμάτων μεταξύ πλατφορμών που επιτρέπει στον προγραμματιστή να στέλνει αξιόπιστα μηνύματα χωρίς κόστος. Χρησιμοποιώντας το Cloud Messaging, ο προγραμματιστής μπορεί να ειδοποιήσει μια εφαρμογή

πελάτη ότι νέα email ή άλλα δεδομένα είναι διαθέσιμα για συγχρονισμό. Επίσης μπορεί να στείλει μηνύματα ειδοποίησης για να αυξήση την αφοσίωση και τη διατήρηση του χρήστη.

Βασικές δυνατότητες:

- **Αποστολή μηνυμάτων ειδοποίησης ή μηνυμάτων δεδομένων:** Αποστολή μηνυμάτων ειδοποίησης που εμφανίζονται στον χρήστη. Εναλλακτικά, μπορούν να σταλθούν μηνύματα δεδομένων για να προσδιοριστεί πλήρως τι συμβαίνει στον κώδικα μιας εφαρμογής.
- **Ευέλικτη στόχευση μηνυμάτων:** Διανομή μηνυμάτων στην εφαρμογή πελάτη με οποιονδήποτε από τους 3 τρόπους — σε μεμονωμένες συσκευές, σε ομάδες συσκευών ή σε συσκευές που έχουν εγγραφεί σε θέματα.
- **Αποστολή μηνυμάτων από εφαρμογές πελατών:** Μπορούν να σταλθούν επιβεβαιώσεις, συνομιλίες και άλλα μηνύματα από συσκευές πίσω στον διακομιστή μέσω του αξιόπιστου και αποδοτικού από την μπαταρία καναλιού σύνδεσης της FCM.

2.7.4 Analytics

Στην καρδιά του Firebase βρίσκεται το Google Analytics, μια δωρεάν και απεριόριστη ανάλυση λύσης. Το Analytics ενσωματώνεται σε όλες τις λειτουργίες του Firebase και παρέχει απεριόριστες αναφορές για έως και 500 ξεχωριστά συμβάντα που μπορούν να ορίστούν χρησιμοποιώντας το Firebase SDK. Οι αναφορές του Analytics βοηθούν τον προγραμματιστή να κατανοήσει με σαφήνεια τη συμπεριφορά των χρηστών, γεγονός που επιτρέπει να λαμβάνει ενημερωμένες αποφάσεις σχετικά με το μάρκετινγκ εφαρμογών και τις βελτιστοποιήσεις απόδοσης.

Βασικές δυνατότητες:

- **Απεριόριστη αναφορά:** Το Analytics παρέχει απεριόριστες αναφορές για έως και 500 ξεχωριστά συμβάντα.
- **Τμηματοποίηση κοινού:** Τα προσαρμοσμένα είδη κοινού μπορούν να οριστούν στην κονσόλα Firebase βάσει δεδομένων συσκευής, προσαρμοσμένων συμβάντων ή ιδιοτήτων χρήστη. Αυτά τα είδη κοινού μπορούν να χρησιμοποιηθούν με άλλες λειτουργίες του Firebase κατά τη στόχευση νέων λειτουργιών ή μηνυμάτων ειδοποίησης.

2.7.5 Crashlytics

Το Firebase Crashlytics είναι ένας ελαφρύς και σε πραγματικό χρόνο συντάκτης σφαλμάτων που βοηθά τον προγραμματιστή στην παρακολούθηση έτσι ώστε να δίνει προτεραιότητα και να διορθώνει ζητήματα σταθερότητας που διαβρώνουν την ποιότητα μιας εφαρμογής. Εξοικονομεί χρόνο αντιμετώπισης προβλημάτων ομαδοποιώντας έξυπνα σφάλματα και επισημαίνοντας τις περιστάσεις που τα οδηγούν.

Βασικές δυνατότητες:

- **Επιλεγμένες αναφορές σφαλμάτων:** Το Crashlytics συνθέτει μια χιονοστιβάδα σφαλμάτων σε μια διαχειρίσιμη λίστα ζητημάτων, παρέχει πληροφορίες με βάση τα συμφραζόμενα και υπογραμμίζει τη σοβαρότητα και την επικράτηση των σφαλμάτων, ώστε η βασική αιτία να μπορεί να εντοπιστεί γρήγορα.

- **Λύσεις για κοινό σφάλμα:** Το Crashlytics προσφέρει Crash Insights, χρήσιμες συμβουλές που επισημαίνουν τα κοινά προβλήματα σταθερότητας και παρέχουν πόρους που τους διευκολύνουν στην αντιμετώπιση προβλημάτων, τη δοκιμή και την επίλυση.
- **Ειδοποιήσεις σε πραγματικό χρόνο:** Μπορεί ο προγραμματιστής να λάβει ειδοποιήσεις σε πραγματικό χρόνο για νέα ζητήματα, παλινδρομικά ζητήματα και αυξανόμενα ζητήματα που ενδέχεται να απαιτούν άμεση προσοχή.

Κεφάλαιο 3ο: APIs

3.1 Εισαγωγή

API ή αλλιώς Application programming interface (Διασύνδεση προγραμματισμού εφαρμογών) είναι μια διεπαφή επικοινωνίας μεταξύ πολλών πελατών (clients) με έναν εξυπηρετητή (server) έτσι ώστε να επιτυγχάνεται η διασύνδεση μεταξύ τους. Το API ορίζει τα είδη των κλήσεων και αιτημάτων που μπορεί να πραγματοποιήσει ένας πελάτης σαν μια μέθοδο εισόδου εξόδου. Απλοποιεί τον προγραμματισμό αφαιρώντας την υποκείμενη εφαρμογή και εκθέτοντας μόνο αντικείμενα ή ενέργειες που χρειάζεται ο προγραμματιστής.

3.2 RESTful APIs

Το RESTful API είναι βασισμένο στην υπηρεσία REST. Το REST (Representational State Transfer) είναι ένα στυλ αρχιτεκτονικής και προσέγγισης στις επικοινωνίες που χρησιμοποιούνται συχνά στην ανάπτυξη υπηρεσιών Ιστού. Πρωτοεμφανίστηκε από τον Roy Fielding το 2000 και μεταφέρει τους πόρους από διάφορους πελάτες (clients) με το πρωτόκολλο HTTP. Πιο συγκεκριμένα, για την ανάσυρση ενός πόρου χρησιμοποιείται η μέθοδος GET, για την αλλαγή ή ενημέρωση η μέθοδος PUT και για την δημιουργία ή διαγραφή ενός πόρου χρησιμοποιούνται οι μέθοδοι POST και DELETE αντίστοιχα. Κύρια χαρακτηριστικά της υπηρεσίας REST είναι:

- Αποκλειστική χρήση HTTP αιτημάτων/μεθόδων για την επικοινωνία του χρήστη με τον παροχέα της διαδικτυακής υπηρεσίας.
- Είναι stateless.
- Κάθε κλήση είναι ανεξάρτητη. Απεικόνιση της δομής των καταλόγων σαν URIs.
- Μεταφορά δεδομένων μέσω XML, JSON

3.3 IT_API

Στην υλοποίηση της εφαρμογής χρησιμοποιήθηκε το ανοιχτού κώδικα RESTful API, IT_API. Το IT_API είναι ένα API που ενσωματώνει τις ηλεκτρονικές υπηρεσίες του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνές Πανεπιστημίου της Ελλάδος. Το IT_API προσφέρει 2 ειδών λειτουργίες, τις δημόσιες (public) και τις ιδιωτικές (private). Πιο συγκεκριμένα στις ιδιωτικές υπηρεσίες για να αποκτήσει πρόσβαση ο χρήστης χρειάζεται να ταυτοποιηθεί και να λάβει ένα διακριτικό πρόσβασης (access token). Για την ταυτοποίηση και την διασφάλιση της ασφάλειας των πόρων χρησιμοποιείται το πρωτόκολλο OAuth 2.0.

3.3.1 OAuth 2.0

Το OAuth 2.0 είναι ένα πρωτόκολλο που επιτρέπει σε μια υπηρεσία να παραχωρήσει πρόσβαση σε τρίτους στους προστατευόμενους πόρους της. Στο OAuth, ο πελάτης (client) ζητά

πρόσβαση σε πόρους που ελέγχονται από τον κάτοχο πόρων (resource owner) και φιλοξενείται από τον διακομιστή πόρων με αποτέλεσμα να εκδίδεται ένα διαφορετικό σύνολο διαπιστευτηρίων από αυτά του κατόχου πόρου. Αντί να χρησιμοποιεί τα διαπιστευτήρια του κατόχου πόρων για πρόσβαση σε προστατευμένους πόρους, ο πελάτης αποκτά ένα διακριτικό πρόσβασης (Access Token). Το διακριτικό πρόσβασης είναι μια συμβολοσειρά που δηλώνει ένα συγκεκριμένο σκοπό (scope), διάρκεια ζωής και άλλα χαρακτηριστικά πρόσβασης.

3.3.2 Διαδικασία ταυτοποίησης

Η διαδικασία ταυτοποίησης γίνεται μόνο την πρώτη φορά που ο χρήστης θα μπει στην εφαρμογή και έπειτα με την χρήση του Refresh Token η εφαρμογή έχει την δυνατότητα να εξάγει νέα access tokens κάθε φορά που είναι απαραίτητο. Το πρώτο βήμα η εφαρμογή στέλνει ένα αίτημα στον Login Server όπου στέλνει τα στοιχεία του πελάτη (client) μαζί με τους σκοπούς (scopes) που θα χρησιμοποιήσει η εφαρμογή και ανακατευθήνει τον χρήστη σε μία οθόνη ταυτοποίησης. Με την είσοδο των στοιχείων γίνεται ανακατεύθυνση σε ένα URL απάντησης και σε περίπτωση που το αίτημα γίνει αποδεκτό, επιστρέφει έναν κωδικό εξουσιοδότησης (authorization code) που θα χρησιμοποιηθεί στην απόκτηση του access token μαζί με το refresh token που θα αποθηκευτεί και θα χρησιμοποιηθεί αργότερα, αλλιώς παραμέτρους λάθους.

Πίνακας 1: Παράδειγμα αίτησης για authorization code

```
POST

https://login.iee.ihu.gr/authorization/?
client_id=[CLIENT_ID]&redirect_uri=[RESPONSE_URL]&response_type=code&scope
=[SCOPES]
```

Πίνακας 2: Απάντηση URL επιστροφής

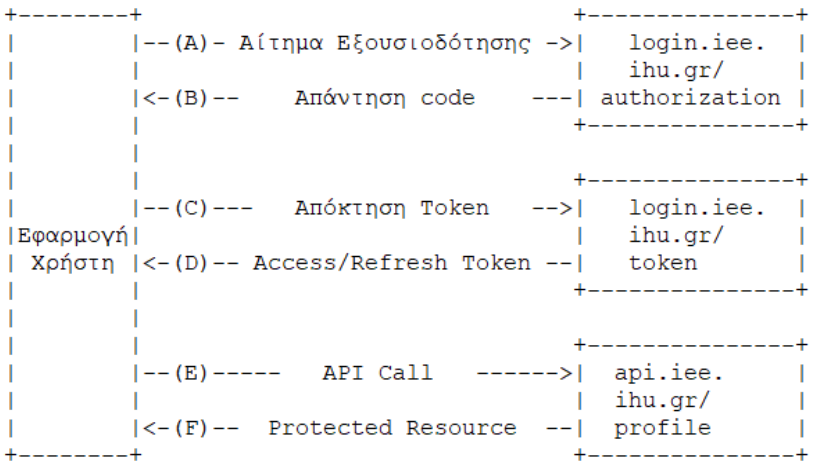
```
[RESPONSE_URL]?code=07729766680546522985774 45&state=undefined
```

Πίνακας 3: Παράδειγμα αίτησης για access token

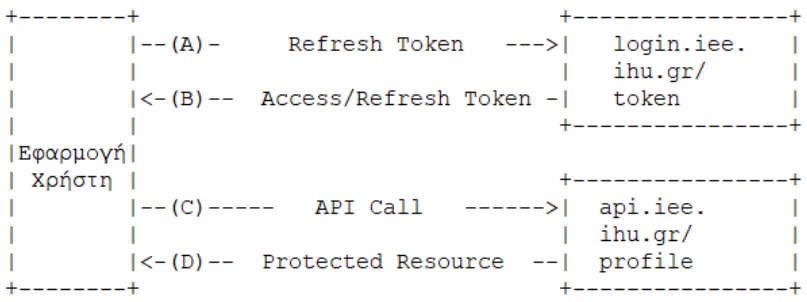
```
POST

https://login.iee.ihu.gr/token

Parameters:

client_id: [CLIENT_ID]

client_secret: [CLIENT_SECRET]

grand_type: authorization_code

code: 07729766680546522985774 45
```

Πίνακας 4: Απάντηση αιτήματος σε μορφή JSON


```
{
      "access_token":
      "eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI1NTI4Iiwic2NvcG
UiOlsiYW5ub3VuY2VtZW50cyIsIm5vdGlmaWNhdGlvbnMiLCJwcm9maWxlIiwiZWRpdF9wcm9m
aWxlIiwiZWRpdF9wYXNzd29yZCIsImVkaXRfbWFpbCIsInJlZnJlc2hfdG9rZW4iXSwiaGFzaC
I6IjFjY2tjOWQ0ZjF6OHhldDExcDI0IiwiaWF0IjoxNTk1MjY0NzQzLCJleHAiOjE1OTUyNjQ4
NjMsImF1ZCI6WyI1YmU5OGFmNmJlMzcxNjU4OTkxMzhhOTYiXX0.R2rXGX5zk-
rXrl2WiYpF2o-
_MTzUXiGWdWWZLuTbdrEHFLDqYfaBcU9SiN7vlsDUaLdwHsDi78qV6X2iHzQ8y2TD4ajbHmXEB
1sgkxDZ_RqQKfiW8lUk6MnDPNDIyHRwrIw5XBnx9awSD7d98y4pE9i0M9vDepyszrkdwUk2sDz
TCPeh32PRACGj3KOdICma5jDxLwlCS4JrxIFZJ5_4P5cR7z2nRNS4_zYSXAAZo9Gpo6ehgrbVW
hTKxl93aH0_WqJzxF2OFaDI6rZAQ4fBu3AUmrefYcqyvSW4vACl_cjm0XAEr2gE2ea82MjZQrm
ql0xPhczODkaT96c7oxS43Ry2UvLsY6RlklxNfhjlkYtoJ4ZzBc5kBZZhOqvKqM__DTScrQclr
byTtJ_NFdPTPx9X37R-U1JxIUxcfGiPjcI2ZMKFd9Q2MbFPsIzrgyFeqQUsphMCpIsli-
VlkqXn9LKNjg1l2iiIQrU_FanD5aLazxpfu32tAEDFRPsYN8seBhaz36uDBwVdyI8es_Sn14FA
FyVHyBvKIdqRfJeTzb_XSwj7NOP_Ka8_fxGPnBhFeeouGG4t4tm6P5WPJihrxBkIb8SC9saJr0
7PC5z6vQfG8HIxBTYZZTD54gagfA0ZvLDAHEg98PscQwe-98xJ-
5KXlsmMbUkkdG8daG5cELQ",

      "user": "[USER_ID]",

      "refresh_token":
      "0u9aio0qsvdc915yfbdz4xl7uppi2osoa52qqyjig83p1bc3aoigvqi9j84aywd4253
xsej7dn0onay0jgairqbdyd3htij0x5ye"

}
```

Σχήμα 15: Διαδικασία ταυτοποίησης την πρώτη φορά



Σχήμα 16: Διαδικασία απόκτησης access token με refresh token

3.3.3 Λειτουργίες IT_API

Το IT_API παρέχει κατά κύριο λόγο, τις ανακοινώσεις του τμήματος. Όλες οι μέθοδοι που υποστηρίζει είναι:

- Ανακοινώσεις (Announcements) - Οι ανακοινώσεις του τμήματος, υπάρχουν δημόσιες και ιδιωτικές
- Κατηγορίες (Categories) - Οι κατηγορίες στις οποίες ανήκουν οι ανακοινώσεις, υπάρχουν δημόσιες και ιδιωτικές
- Αρχεία (Files) - Τα αρχεία που επισυνάπτονται στις ανακοινώσεις, υπάρχουν δημόσια και ιδιωτικά
- Προφίλ (Profile) - Το προφίλ του χρήστη, είναι ιδιωτικό
- Χρήστες (Users) - Χρήστες περιέχουν τα στοιχεία του προφίλ των χρηστών, είναι ιδιωτικό
- Ομάδες (Groups) - Ομάδες είναι ομάδες χρηστών, είναι ιδιωτικό και μόνο για καθηγητές
- Ειδοποιήσεις (Notifications) - Ειδοποιήσεις είναι οι ειδοποιήσεις ανακοινώσεων από τις κατηγορίες που είναι εγγεγραμμένος ένας χρήστης, είναι ιδιωτικές.

* Όπου ιδιωτικές χρειάζεται access token

Οι ανακοινώσεις υποστηρίζουν φιλτρα (filters), σελιδοποίηση (pagination) και ταξινόμηση (sorting).

Πίνακας 5: Παράδειγμα αίτησης για δημόσιες ανακοινώσεις

```
GET
https://api.iee.ihu.gr/announcements/public
Headers:
Content-Type:application/json
```

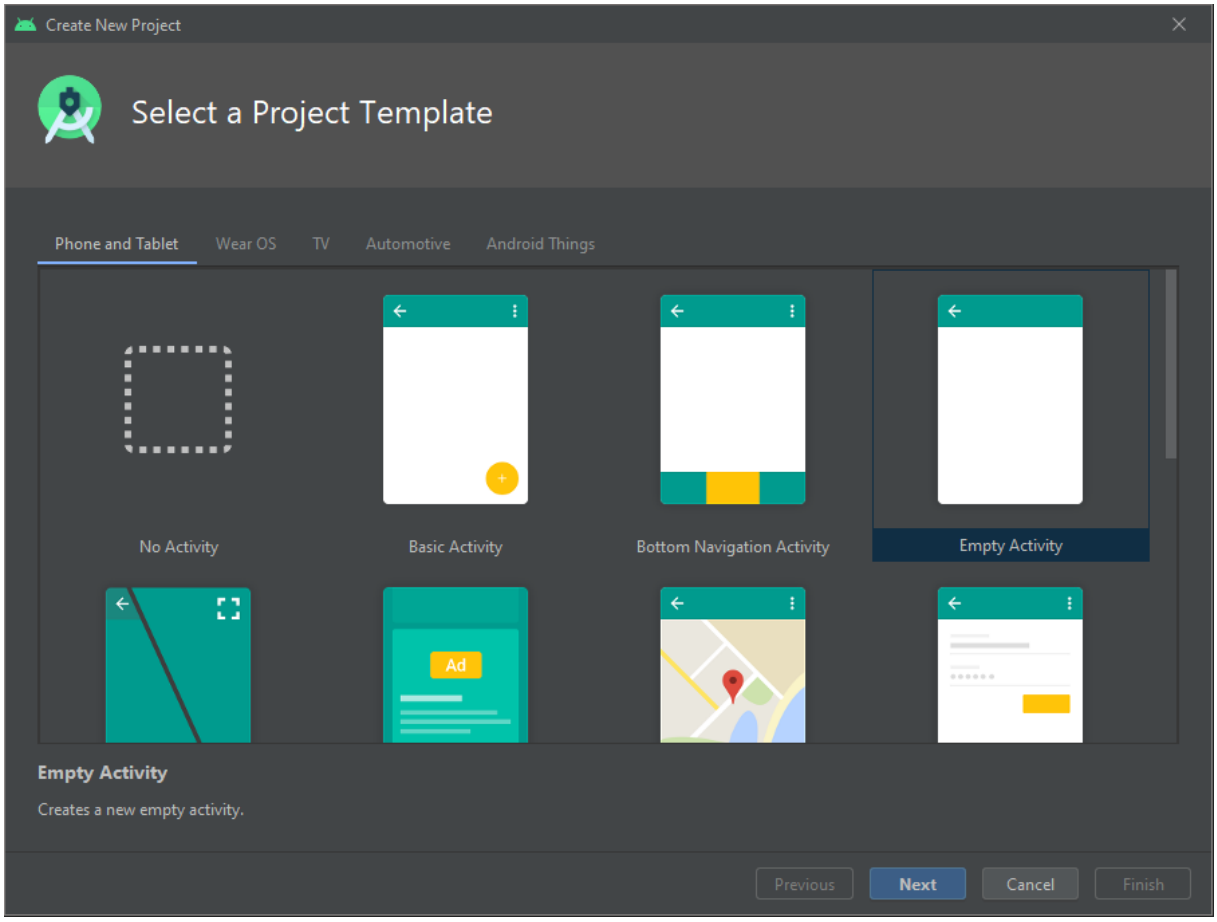
Πίνακας 6: Απάντηση αιτήματος σε μορφή JSON


```
[{
"publisher": {
   "id": "6448",
   "name": "Μελίνα Ιωαννίδου"
   },
   "attachments": ["5f1145f69c9f44304039f22f"],
   "_id": "5f1145f69c9f44304039f22d",
   "_about": "591ca2b75c39e553cdf0a513",
   "titleEn": "Πρόσκληση υποβολής αιτήσεων 2020-21 για το ΠΜΣ Εφαρμοσμένα
   Ηλεκτρονικά Συστήματα",
   "textEn": "Συνημμένα, η πρόσκληση υποβολής υποψηφιοτήτων, εισαγωγής
   ακαδημαϊκού έτους 2020-21, για το Πρόγραμμα Μεταπτυχιακών Σπουδών
   Εφαρμοσμένα Ηλεκτρονικά Συστήματα (Applied Electronic Systems). \n\nΗ
   Διευθύντρια του Π.Μ.Σ.\nΜελίνα Ιωαννίδου\nΑν. Καθηγήτρια\nΤμ. Μηχ/κών
   Πληροφορικής & Ηλεκτρονικών Συστημάτων\nΔιεθνές Πανεπιστήμιο της
   Ελλάδος\n\n",
   "text": "Συνημμένα, η πρόσκληση υποβολής υποψηφιοτήτων, εισαγωγής
   ακαδημαϊκού έτους 2020-21, για το Πρόγραμμα Μεταπτυχιακών Σπουδών
   Εφαρμοσμένα Ηλεκτρονικά Συστήματα (Applied Electronic Systems). \n\nΗ
   Διευθύντρια του Π.Μ.Σ.\nΜελίνα Ιωαννίδου\nΑν. Καθηγήτρια\nΤμ. Μηχ/κών
   Πληροφορικής & Ηλεκτρονικών Συστημάτων\nΔιεθνές Πανεπιστήμιο της
   Ελλάδος\n\n",
   "title": "Πρόσκληση υποβολής αιτήσεων 2020-21 για το ΠΜΣ Εφαρμοσμένα
   Ηλεκτρονικά Συστήματα",
   "date": "2020-07-17T06:32:22.616Z",
   "__v": 0,
   "wordpressId": 9652698
   },
   {
   …
}]
```

Κεφάλαιο 4ο: Υλοποίηση εφαρμογής

4.1 Εισαγωγή

Η υλοποίηση της εφαρμογής έγινε στην γλώσσα προγραμματισμού Java μέσα στο περιβάλλον ανάπτυξης λογισμικού Android Studio. Η διαδικασία ξεκίνησε δημιουργώντας ένα καινούριο κενό Android Project.



Σχήμα 17: Η δημιουργία του Project

4.2 Gradle και Manifest

Προστέθηκαν οι κατάλληλες βιβλιοθήκες στα αρχεία build.gradle και έγιναν οι κατάλληλες αλλαγές στο defaultConfig. Οι αλλαγές στο config αφορούν την ελάχιστη και μέγιστη έκδοση Android που θα υποστηρίζει η εφαρμογή όπως επίσης και η ενεργοποίηση της υποστήριξη της Java 8 και του DataBinding.

```
android {
    compileSdkVersion 30
    buildToolsVersion "30.0.2"

    defaultConfig {
        applicationId "gr.teithe.it.it_app"
        minSdkVersion 23
        targetSdkVersion 30
        versionCode 13
        versionName "1.1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    buildFeatures{
        dataBinding = true
    }
}
```

Σχήμα 18: Το build.gradle

Στο AndroidManifest αρχείο προστέθηκαν οι υπηρεσίες του Firebase που θα χρησιμοποιηθούν για την υλοποιήση των push notifications.

```
<meta-data
    android:name="com.google.firebase.messaging.default_notification_channel_id"
    android:value="default" />

<meta-data
    android:name="com.google.firebase.messaging.default_notification_icon"
    android:resource="@drawable/ic_notifications_black_24dp" />

<meta-data
    android:name="com.google.firebase.messaging.default_notification_color"
    android:resource="@color/colorAccent" />

<service
    android:name=".util.MyFirebaseMessagingService"
    android:stopWithTask="false">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Σχήμα 19: Η υπηρεσία του Firebase στο Manifest

4.3 Θέμα και χρώματα

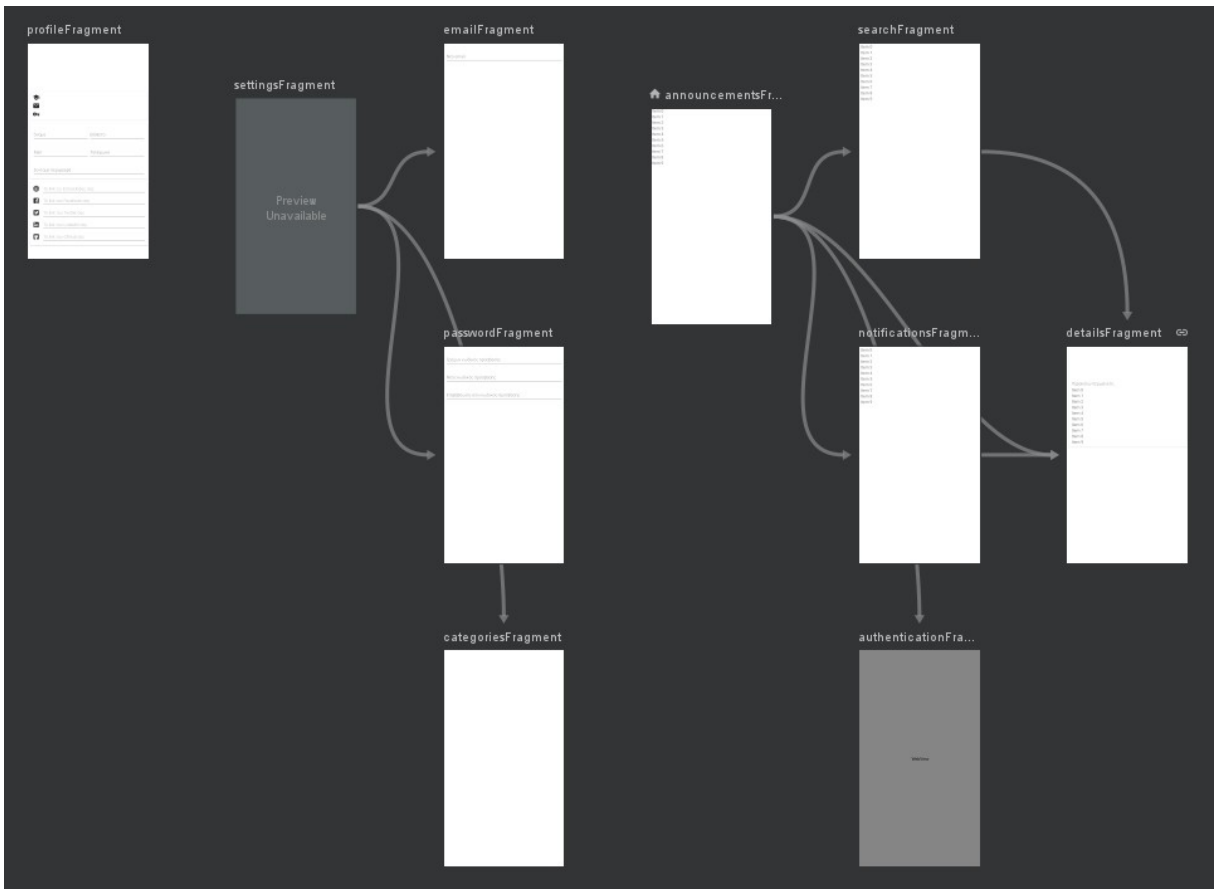
Στην εφαρμογή υπάρχουν 2 θέματα. Το Λευκό με σκούρα γράμματα και το Σκούρο με τα λευκά γράμματα. Για την υλοποίηση του χρησιμοποιήθηκε το DayNight από τα MaterialComponents και τροποποιήθηκαν τα χρώματα σε μερικά στοιχεία και εικονίδια με την επιλογή του να αλλάζουν χρώμα ανάλογα με το θέμα. Η αρχικοποίηση του Θέματος γίνετε στην κλάση MainActivity όταν ξεκινάει η εφαρμογή με μια βοηθητική κλάση ThemeHelper.

```xml
<resources>
    <style name="AppTheme" parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">?android:attr/textColorSecondary</item>

        <item name="textInputStyle">@style/TextInput</item>

        <item name="alertDialogTheme">@style/AlertDialogTheme</item>
    </style>

    <style name="AlertDialogTheme" parent="ThemeOverlay.AppCompat.Dialog.Alert">
        <item name="buttonBarNegativeButtonStyle">@style/NegativeButtonStyle</item>
        <item name="buttonBarPositiveButtonStyle">@style/PositiveButtonStyle</item>
    </style>

    <style name="NegativeButtonStyle" parent="Widget.MaterialComponents.Button.TextButton.Dialog">
        <item name="android:textColor">?android:attr/textColorSecondary</item>
    </style>

    <style name="PositiveButtonStyle" parent="Widget.MaterialComponents.Button.TextButton.Dialog">
        <item name="android:textColor">?android:attr/textColorPrimary</item>
    </style>

    <style name="TextInput" parent="Widget.Design.TextInputLayout">
        <item name="android:backgroundTint">?android:attr/colorBackground</item>
        <item name="android:textColor">?android:attr/textColorPrimary</item>
        <item name="android:textColorHint">?android:attr/textColorSecondary</item>
        <item name="android:colorAccent">?android:attr/textColorSecondary</item>
        <item name="colorControlNormal">?android:attr/textColorSecondary</item>
        <item name="colorControlActivated">?android:attr/textColorPrimary</item>
        <item name="colorControlHighlight">?android:attr/textColorSecondary</item>
    </style>
</resources>
```

Σχήμα 20: Το αρχείο styles

4.4 Navigation

Η εφαρμογή ακολουθεί το μοντέλο του Single Activity Pattern. Υπάρχει το βασικό Activity MainActivity που διαχωρίζεται την πλοήγηση της εφαρμογής μέσω του Navigation Component. Σε κάθε οθόνη που υπάρχει στην εφαρμογή είναι και ένα διαφορετικό Fragment. Τα Fragments μεταξύ τους είναι συνδεμένα σε ένα κεντρικό αρχείο του Navigation Component και μερικοί από τους προορισμούς μεταφέρουν δεδομένα.



Σχήμα 21: Οι προορισμοί στα Fragments της εφαρμογής

4.5 Ταυτοποίηση

Η ταυτοποιήση γίνετε με την χρήση WebView. Το WebView φορτώνει την σελίδα login και περιμένει για ανακατεύθυνση στην σελίδα επιστροφής μαζί με το αποτέλεσμα. Σε περίπτωση κάποιας άλλης ανακατεύθυνσης τότε ξανά εμφανίζει στον χρήστη την σελίδα ταυτοποίηση. Μετά την ταυτοποίηση, η εφαρμογή ενεργοποιεί τις πλήρης λειτουργίες της.

```
mDataBinding.fAuthenticationWeb.getSettings().setJavaScriptEnabled(true);
mDataBinding.fAuthenticationWeb.setWebViewClient(new WebViewClient()
{
    @Override
    public boolean shouldOverrideUrlLoading(WebView wView, String url)
    {
        if(url.contains(Constants.LOGIN_URL))
        {
            mDataBinding.fAuthenticationWeb.loadUrl(url);

            return true;
        }
        else
        {
            if(url.contains(Constants.RESPONSE_URL))
            {
                if(url.contains("error"))
                {
                    Toast.makeText(getContext(), text: "Δεν δώθηκαν δικαιώματα στην εφαρμογή", Toast.LENGTH_SHORT).show();
                    Navigation.findNavController(mDataBinding.getRoot()).popBackStack();
                }
                else
                {
                    Uri uri = Uri.parse(url);
                    final String code = uri.getQueryParameter( key: "code");

                    mViewModel.authenticate(code);
                }

                return false;
            }
            else
            {
                mDataBinding.fAuthenticationWeb.loadUrl(Constants.AUTHORIZATION_URL);

                return true;
            }
        }
    }
});
```

Σχήμα 22: Η διαχείριση του WebView

4.6 Refresh και Access Tokens

Όταν η ταυτοποίηση επιτύχει, τότε αποθηκεύονται στην μνήμη το refresh και το access token με την βοήθεια μιας Singleton κλάσης, PreferencesManager. Το refresh token χρησιμοποιείται κάθε φορά που το access token λήγει, έτσι ώστε να εκδώσει ένα νέο access token και το ίδιο το refresh token να αντικατασταθεί. Στον http client που θα χρησιμοποιηθεί, χρησιμοποιούνται 2 interceptors όπου ο ένας προσθέτει το access token στο αίτημα που θα αποσταλεί στο API και ο άλλος θα εκδώσει και θα ξαναστείλει το αίτημα σε περίπτωση που το access token έχει λήξει.

```
@Nullable
@Override
public Request authenticate(@Nullable Route route, @Nullable Response response) throws IOException
{
    if(response != null && response.code() == 401)
    {
        TokenResponse token = new TokenRepository().refreshToken().execute().body();

        if(token != null && token.getRefreshToken() != null)
        {
            PreferencesManager.setRefreshToken(token.getRefreshToken());
            PreferencesManager.setAccessToken(token.getAccessToken());

            return response.request().newBuilder()
                    .header( name: "x-access-token", token.getAccessToken())
                    .build();
        }
    }

    return null;
}
```

Σχήμα 23: Ο interceptor που θα επανεκδώσει το νέο access token

4.7 Δεδομένα εφαρμογής, Retrofit και Repositories

Την επικοινωνία της εφαρμογής την αναλαμβάνει το retrofit (http client), το οποίο μετατρέπει μια διεπαφή σε αντικείμενα κλήσης. Στην διεπαφή υπάρχουν όλα τα σημεία (endpoints) που προσφέρει το IT_API και με την βοήθεια μιας Singleton κλάσης δημιουργούνται τα αντικείμενα κλήσης. Τα αντικείμενα κλήσης χωρίζονται σε κλάσεις Repository, όπου κατηγοριοποιούνται ανάλογα με τις λειτουργίες που προσφέρουν, πχ, για τις ανακοινώσεις.

```
public class AnnouncementsRepository
{
    private ApiService apiService;

    public AnnouncementsRepository()
    {
        apiService = ApiClient.getClient();
    }

    public Call<List<Announcement>> getPagedAnnouncements(String path, int pageSize, long page)
    {
        return apiService.getPagingAnnouncements(path, pageSize, page);
    }

    public Observable<List<Announcement>> getAnnouncements(String path)
    {
        return apiService.getAnnouncements(path);
    }

    public Observable<Announcement> getAnnouncement(String id)
    {
        return apiService.getAnnouncement(id);
    }

    public Observable<File> getFile(String id)
    {
        return apiService.getFile(id);
    }
}
```

Σχήμα 24: Το repository για τις ανακοινώσεις

4.8 ViewModel, LiveData και RxJava

Τα Repositories χρησιμοποιούνται από τα ViewModels τα οποία λειτουργούν ως μεσολαβητές στα fragments. Ένα ViewModel μπορεί να χρησιμοποιήσει πολλά Repositories. Στα ViewModels υλοποιούνται οι μεθόδου επικοινωνίας με το Repository και τα LiveData αντικείμενα που θα επικοινωνούν με τα fragments. Στις μεθόδους επικοινωνίας, λόγο του ότι τα αιτήματα στο API δεν μπορούν να τρέξουν στο κύριο Thread της εφαρμογής, χρησιμοποιείτε το RxJava για να μεταφερθεί το αίτημα σε ένα διαφορετικό Thread έτσι ώστε η εφαρμογή να μπορεί να τρέχει κανονικά και το αίτημα στο παρασκήνιο. Όταν το αίτημα τελειώσει, τότε μεταφέρετε στο κύριο Thread ενημερώνονται με τα ανάλογα αποτελέσματα τα LiveData αντικείμενα και αυτά με την σειρά τους ενημερώνει το UI.

```
public void loadAnnouncement(String id)
{
    isLoading.setValue(true);

    disposable.add(announcementsRepository.getAnnouncement(id)
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(announce ->
            {
                announcement.setValue(announce);
                isLoading.setValue(false);

                if(announce.getAttachments() != null)
                {
                    loadFiles(announce.getAttachments());
                }
            }, throwable ->
            {
                if(throwable instanceof UnknownHostException)
                {
                    errorMessage.postValue("Δεν υπάρχει πρόσβαση στο διαδίκτυο");
                }
                else if(throwable instanceof SocketTimeoutException)
                {
                    errorMessage.postValue("Η σύνδεση έλειξε, δοκιμάστε ξανά");
                }
                else
                {
                    errorMessage.postValue("Παρουσιάστηκε άγνωστο σφάλμα.\n" + throwable.getMessage());
                }

                isLoading.setValue(false);
            }));
}
```

Σχήμα 25: Παράδειγμα μεθόδου στο στο ViewModel

4.9 Fragment, Layout και DataBinding

Το Fragment, όταν λάβει την ενημέρωση από το LiveData, τότε με την χρήση του DataBinding πραγματοποιεί τις αλλαγές στην διεπιφάνεια χρήστη. Κάθε Fragment περιέχει ένα layout το οποίο χρησιμοποιείτε από το DataBinding και όταν χρειαστεί δηλώνονται μεταβλητές που μπορεί να χρειαστούν για απευθείας ανάθεση από το UI. Ανάμεσα στο layout περιέχονται τα ConstraintLayout, RelativeLayout και LinearLayout ανάλογα από τις ανάγκες της διεπαφής και ανάμεσα σε αυτά τα στοιχεία που μπορεί να χρειαστούν, όπως ένα TextView.

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="file"
            type="gr.teithe.it.it_app.data.model.File" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/l_file"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="sans-serif-light"
            android:text="@{file.name}"
            android:textColor="?android:attr/textColorPrimary"/>
    </LinearLayout>
</layout>
```

Σχήμα 26: Παράδειγμα χρήσης του DataBinding με μεταβλητή

4.10 Adapters

Στους adapters στέλνονται τα δεδομένα που είναι πολλά και σε λίστες, για παράδειγμα οι ειδοποιήσεις. Κληρονομεί και υλοποιεί την κλάση RecyclerView.Adapter και περιέχει μια εσωτερική κλάση που χρησιμοποιεί DataBinding για την εμφάνιση των δεδομένων.

```java
public class NotificationViewHolder extends RecyclerView.ViewHolder
{
    private final NotificationItemBinding binding;

    public NotificationViewHolder(@NonNull NotificationItemBinding binding)
    {
        super(binding.getRoot());

        this.binding = binding;
    }

    public void bind(Notification item)
    {
        binding.getRoot().setOnClickListener(v -> listener.onClicked(binding.getNotification()));
        binding.setNotification(item);
        binding.executePendingBindings();
    }
}
```

Σχήμα 27: Η κλάση ViewHolder για τον Adapter

4.11 Ανακοινώσεις και Paging

Στις ανακοινώσεις, χρησιμοποιείται η βοήθεια της βιβλιοθήκης Paging, η οποία φέρνει 15-15 ανακοινώσεις κάθε φορά. Για την υλοποίηση του paging χρειάζεται μια κλάση PageSource η οποία υλοποιεί τις μεθόδους που θα χρησιμοποιηθούν για την αρχική φόρτωση και την επιπλέον φόρτωση των ανακοινώσεων. Τα αιτήματα σε αυτή την περίπτωση δεν χρειάζονται να τρέξουν σε άλλο Thread μιας και η PageSource τρέχει από μόνη της σε άλλο Thread. Η κλάση PageSource χρησιμοποιείτε στο ViewModel και όλη η λειτουργία της φόρτωσης των δεδομένων γίνεται αυτόματα από το Paging.

```
AnnouncementsDataSourceFactory dataSourceFactory = new AnnouncementsDataSourceFactory();

liveDataSource = dataSourceFactory.getAnnouncementsDataSource();

PagedList.Config config = new PagedList.Config.Builder()
        .setEnablePlaceholders(false)
        .setPageSize(15)
        .build();

announcementsPagedList = new LivePagedListBuilder<>(dataSourceFactory, config).build();

isLoading = Transformations.switchMap(liveDataSource, AnnouncementsDataSource::isLoading);
isLoggedIn = Transformations.switchMap(liveDataSource, AnnouncementsDataSource::isLoggedIn);
errorMessage = Transformations.switchMap(liveDataSource, AnnouncementsDataSource::getErrorMessage);
```

Σχήμα 28: Η αρχικοποίηση του Paging στο ViewModel

4.12 Push Notifications

Τα push notifications έχουν υλοποιηθεί με την χρήση των Firebase Topics. Κάθε κατηγορία στις ανακοινώσεις αντιστοιχεί σε ένα topic με την ενεργοποίηση τους στις ρυθμίσεις, οι κατηγορίες από την λίστα στην “Παρακολούθηση πινάκων” κάνουν έγγραφη ή απ εγγραφή ανάλογα με τις κατηγορίες που επέλεξε ο χρήστης στα αντίστοιχα topics στο Firebase.

Στο Firebase υπάρχει μια Realtime Database στην οποία όταν κάποιος χρήστης ανοίξει μια ανακοίνωση τότε το ID της ανακοίνωσης μαζί με τις πληροφορίες καταγράφονται στην Realtime Database. Η διαδικασία της καταγραφής γίνεται μόνο 1 φορά για κάθε ανακοίνωση με την βοήθεια των κανόνων της Realtime Database.

Οι κανόνες αποτρέπουν την πρόσβαση σε οποιαδήποτε συσκευή να διαβάσει τις εγγραφές που υπάρχουν ήδη στην Realtime Database. Επίσης γίνεται έλεγχος για το ότι υπάρχουν τα σωστά δεδομένα και για το αν έχουν ήδη προστεθεί.

Πίνακας 7: Οι κανόνες του Realtime Database

{

```
  "rules":
  {
    ".read": false,
    "Notifications":
    {
      "$uid":
      {
          ".validate": "newData.hasChildren(['about', 'category', 'title', 'name', 'date']) && newData.child('date').val() > '2020-04-29T00:00:00.000Z'",
          ".write": "!data.exists()"
      }
    }
  }
}
```

Κάθε εγγραφή ενεργοποιεί ένα Firebase Cloud Function το οποίο ενεργοποιείται κάθε φορά που μια εγγραφή προστίθεται στην Realtime Database. Το συγκεκριμένο function είναι γραμμένο στο περιβάλλον της NodeJS και παίρνει τα δεδομένα από την εγγραφή που προστέθηκε στην Realtime Database, δημιουργεί και στέλνει το notification στο topic που αντιστοιχεί η εγγραφή.

Πίνακας 8: Ο κώδικας που στέλνει τα push notifications σε NodeJS

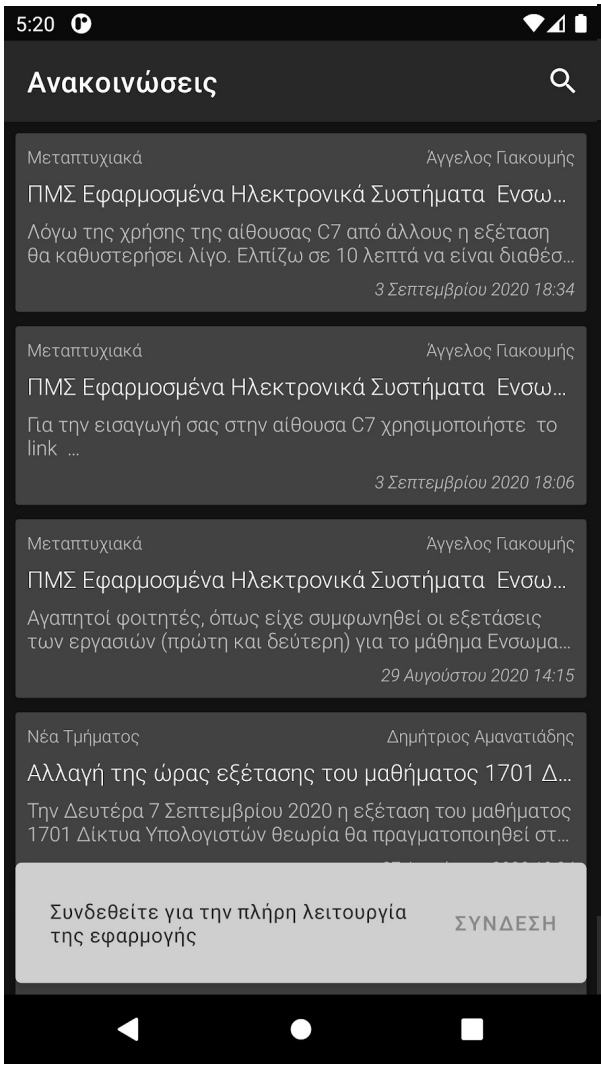
```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp();


exports.sendNotification = functions.database.ref('/Notifications/{id}').onWrite((change, context) =>
{
  if(change.before.exists())
  {
    return null;
  }
  if(!change.after.exists())
```

```
  {
    return null;
  }
  const notification = change.after.val();
  const payload =
  {
    'data':
    {
      'id':context.params.id,
      'category':notification.category,
      'title':notification.title,
      'name':notification.name
    }
  };

  return admin.messaging().sendToTopic(notification.about, payload).then(response =>
  {
    return console.log('Succesfully sent a notification for ', notification.title);
  })
  .catch((error) =>
  {
    return console.error('Failed: ', error);
  });
});
```

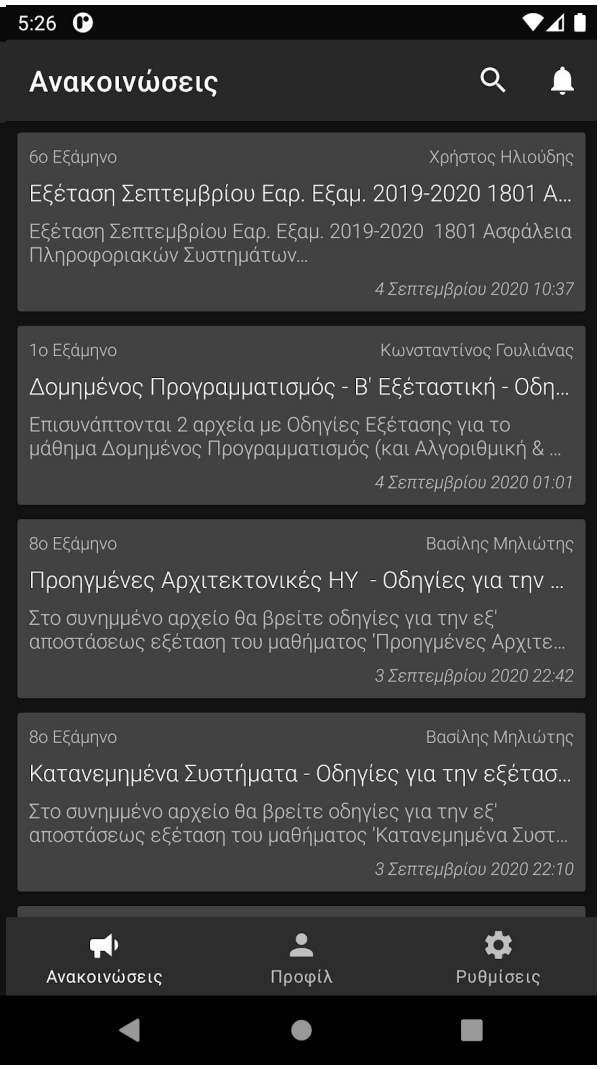
Κεφάλαιο 5ο: Παρουσίαση εφαρμογής - Οδηγίες χρήσεις

5.1 Ανακοινώσεις

Η εφαρμογή έχει 2 διαφορετικές παραλλαγές. Η μία είναι η συνδεδεμένη που προσφέρει όλες τις υπηρεσίες της εφαρμογής και η μη συνδεδεμένη. Στο πρώτο άνοιγμα της εφαρμογής εμφανίζεται η μη συνδεδεμένη οθόνη. Εμφανίζονται μόνο οι δημόσιες ανακοινώσεις με την δυνατότητα αναζήτησης. Επίσης υπάρχει το κουμπί για την σύνδεση που μεταφέρετε στην οθόνη σύνδεσης (βλέπε εικόνα 3) με τον λογαριασμό του Apps. Με την σύνδεση στην εφαρμογή "ξεκλειδώνονται" όλες οι λειτουργίες.



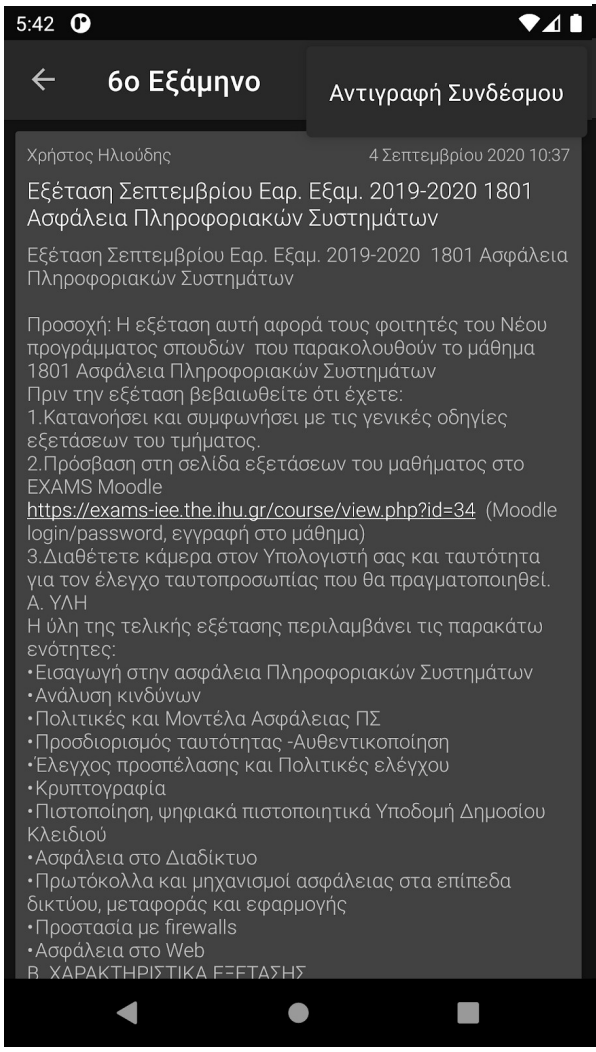
Σχήμα 29: Ανακοινώσεις 1



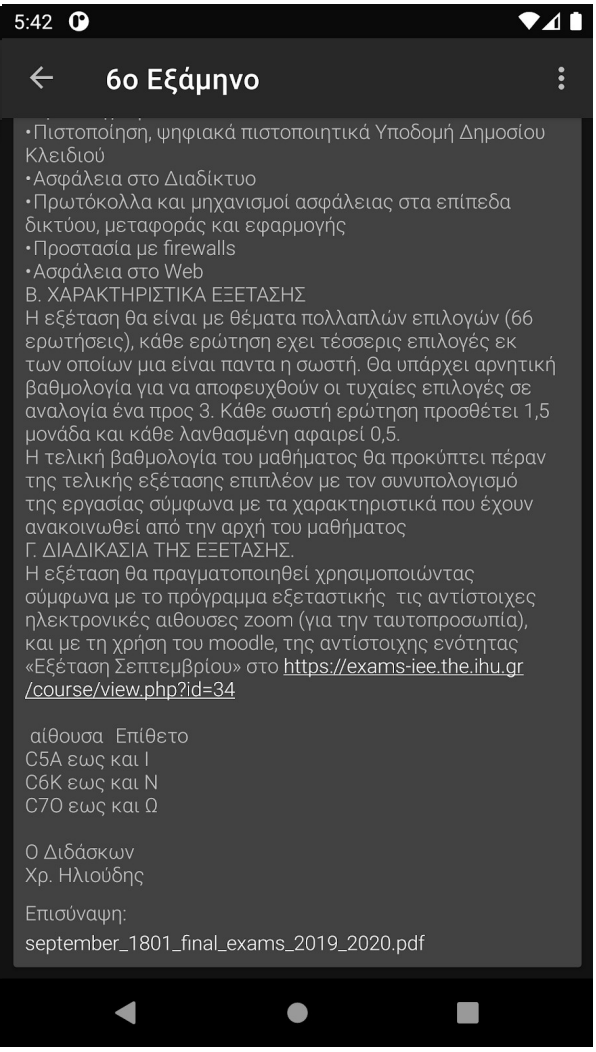
Σχήμα 30: Ανακοινώσεις 2

5.1.1 Λεπτομέρειες ανακοίνωσης

Και στις 2 περιπτώσεις, υπάρχουν οι ανακοινώσεις, μόνο δημόσιες στην μη συνδεδεμένη και όλες στην συνδεδεμένη. Πατώντας σε μια ανακοίνωση εμφανίζεται η οθόνη με ολόκληρο τον τίτλο και κείμενο της ανακοίνωσης, όπως επίσης και πιθανά επισυνημμένα αρχεία. Πατώντας σε ένα επισυνημμένο αρχείο η εφαρμογή ανακατευθύνει στον Browser της συσκευής για την ολοκλήρωση της λήψης του αρχείου. Επιπλέον υπάρχει επιλογή αντιγραφή του συνδέσμου της ανακοίνωσης σε περίπτωση που κάποιος χρήστης θέλει να μοιραστεί την ανακοίνωση με κάποιον.



Σχήμα 31: Λεπτομέρειες 1

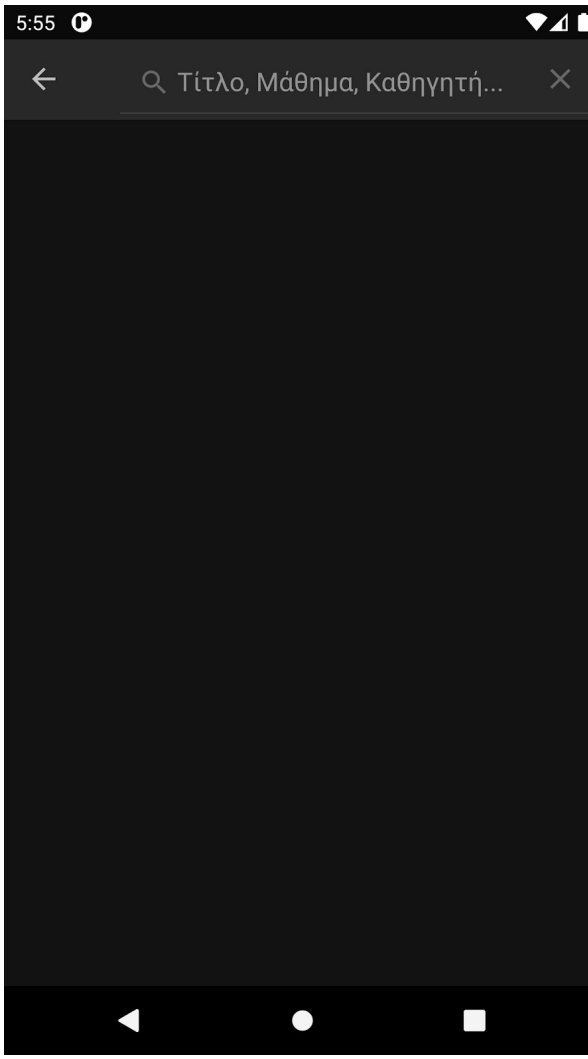


Σχήμα 32: Λεπτομέρειες 2

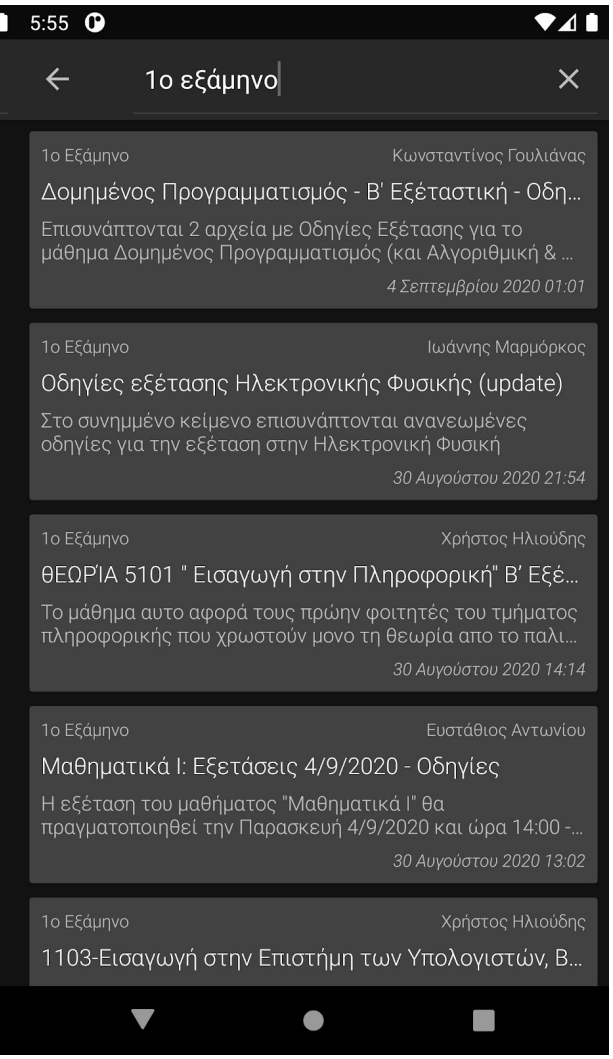
5.1.2 Αναζήτηση

Πατώντας το κουμπί της αναζήτησης εμφανίζεται η οθόνη αναζήτησης όπου ο χρήστης μπορεί να γράψει μια λέξη κλειδί, ένα κομμάτι από τον τίτλο ή το κείμενο, το όνομα του καθηγητή ή

ακόμα και μιας κατηγορίας. Οι ανακοινώσεις που θα περιέχουν την εισαγωγή του χρήστη θα εμφανιστούν στην οθόνη, αν καμία ανακοίνωση δεν βρεθεί τότε θα εμφανιστεί μήνυμα λάθους.



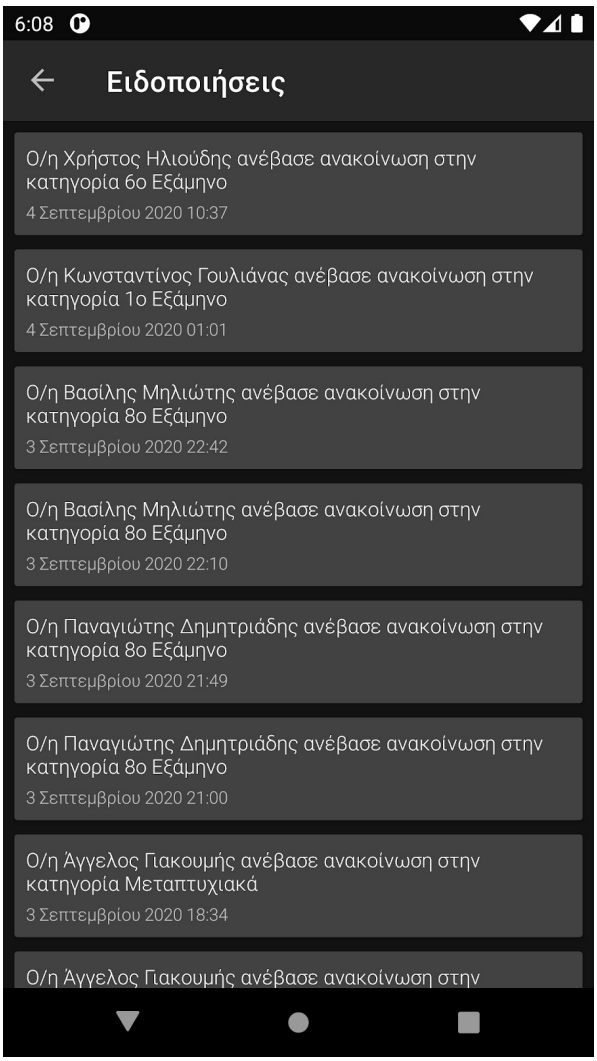
Σχήμα 33: Αναζήτηση 1



Σχήμα 34: Αναζήτηση 2

5.1.3 Ειδοποιήσεις

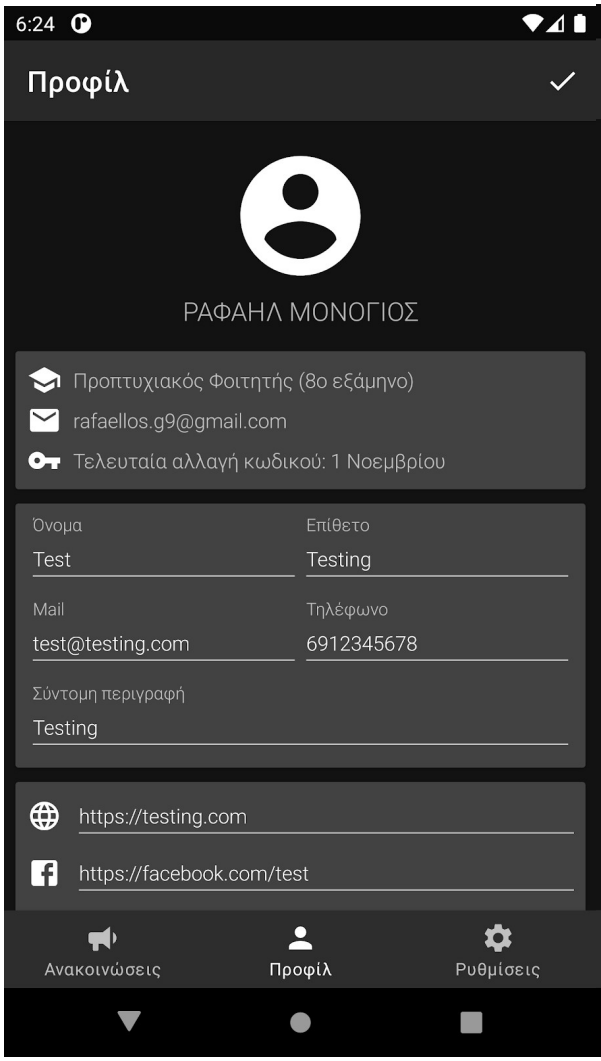
Στο κουμπί ειδοποιήσεις (καμπάνα) εμφανίζεται η οθόνη με τις ειδοποιήσεις. Οι ειδοποιήσεις αφορούν τις κατηγορίες που ο χρήστης επέλεξε να παρακολουθεί στην ρύθμιση “Παρακολούθηση πινάκων” που θα αναλυθεί αργότερα. Οι αδιάβαστες ειδοποιήσεις έχουν πλαγιαστά γράμματα έτσι ώστε ο χρήστης να μπορεί να ξεχωρίζει από τις υπόλοιπες. Πατώντας πάνω σε μια ειδοποίηση, όπως ακριβώς και στην οθόνη με τις ανακοινώσεις, η εφαρμογή μεταφέρετε στις λεπτομέρειες της ανακοίνωσης που αντιστοιχεί με την ειδοποίηση.



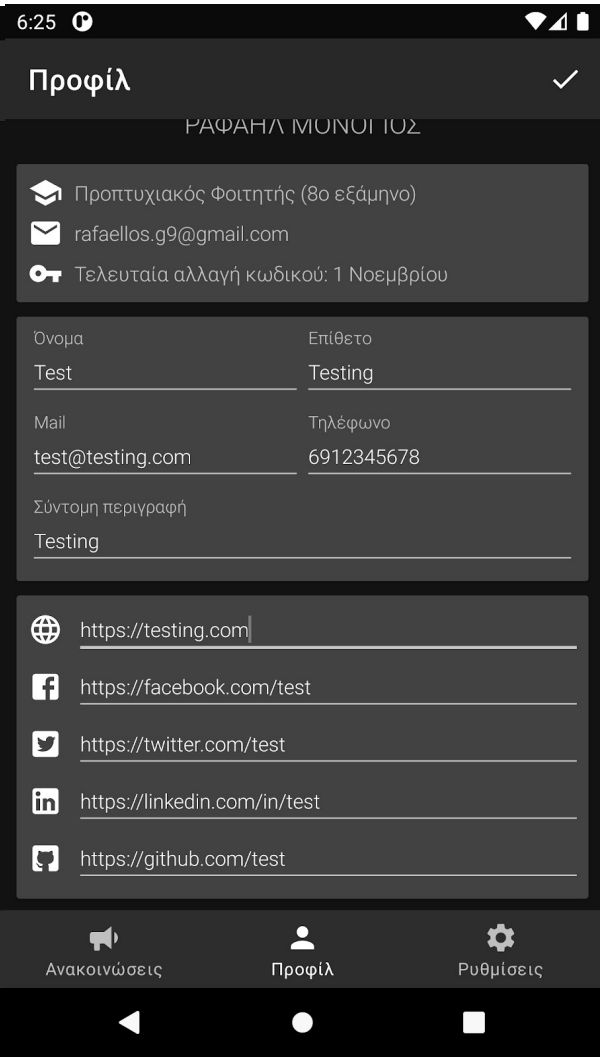
Σχήμα 35: Ειδοποιήσεις

5.2 Προφίλ

Στο μενού με το Προφίλ, εμφανίζονται πληροφορίες που αφορούν τον συνδεδεμένο χρήστη. Επίσης υπάρχουν πληροφορίες για το δημόσιο προφίλ του κάθε χρήστη, για παράδειγμα το προσωπικό email, facebook κτλ, όπου μπορεί να επεξεργαστεί και να κατοχυρώσει τις αλλαγές πατώντας το κουμπί επιβεβαίωσης (νι) πάνω δεξιά.



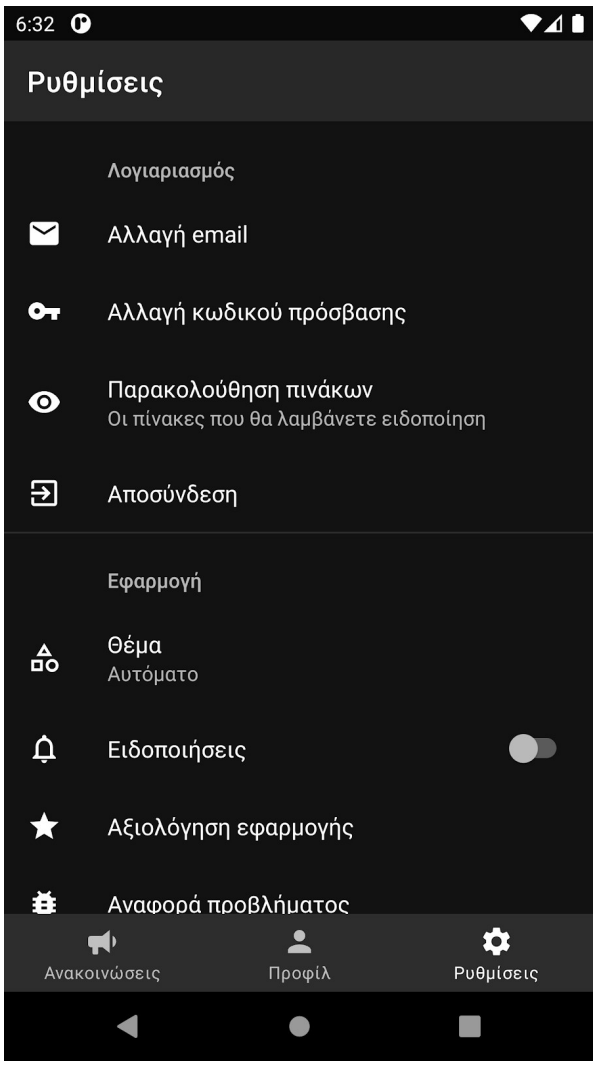
Σχήμα 36: Προφίλ 1



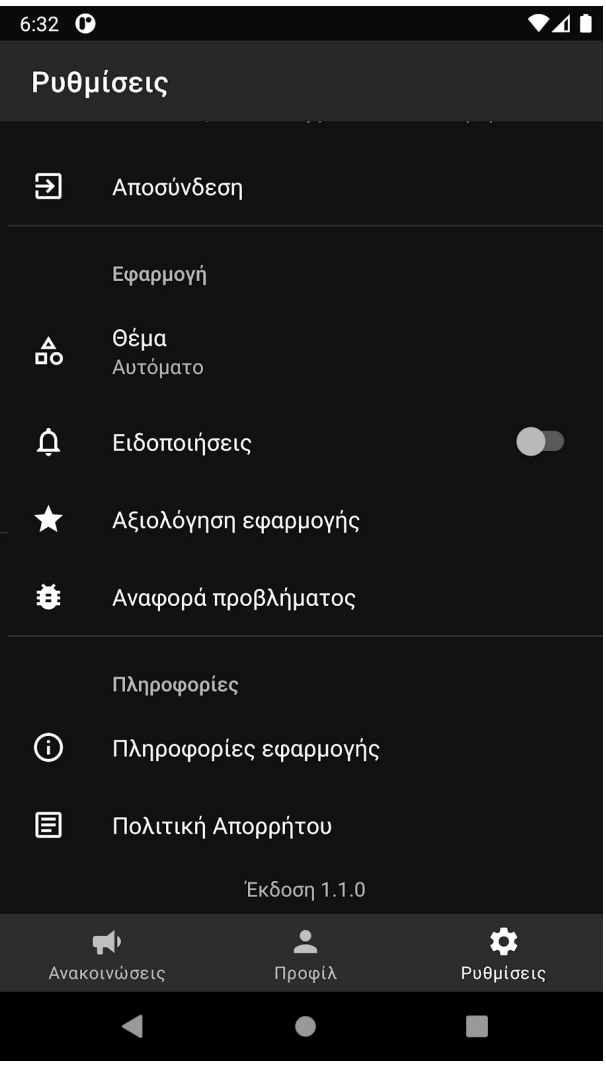
Σχήμα 37: Προφίλ 2

5.3 Ρυθμίσεις

Το τρίτο βασικό μενού είναι οι Ρυθμίσεις. Στις ρυθμίσεις υπάρχουν 3 κατηγορίες, δηλαδή ρυθμίσεις που αφορούν τον λογαριασμό του χρήστη, την εφαρμογή και πληροφορίες.



Σχήμα 38: Ρυθμίσεις 1



Σχήμα 39: Ρυθμίσεις 2

5.3.1 Αλλαγή email

Στην αλλαγή email ο χρήστης έχει την δυνατότητα να αλλάξει το email στο δημόσιο προφίλ του, αλλά όχι το email της σχολής.

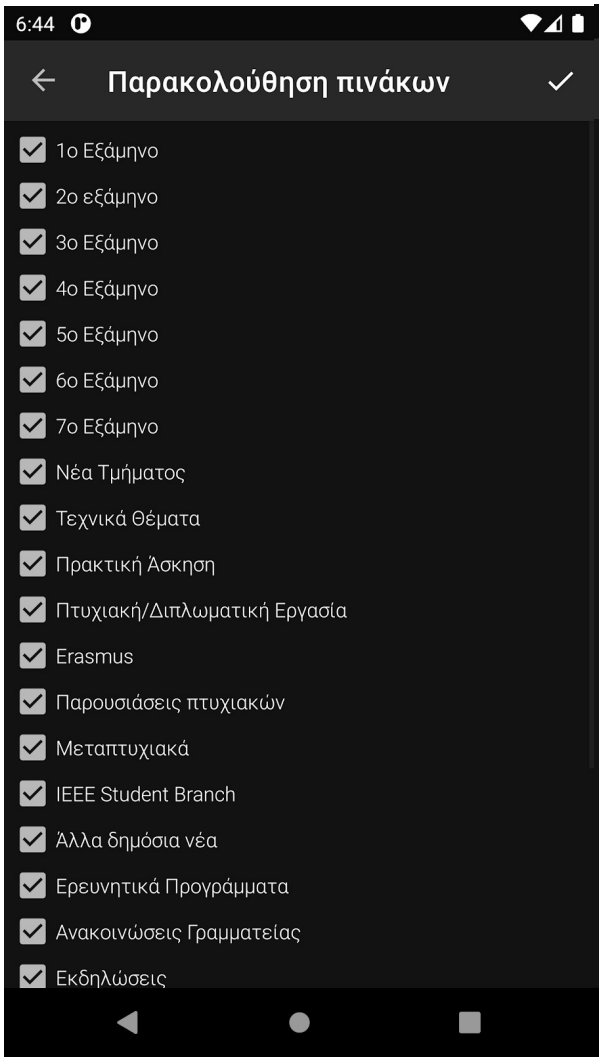
5.3.2 Αλλαγή κωδικού πρόσβασης

Στην αλλαγή κωδικού πρόσβασης ο χρήστης μπορεί να αλλάξει τον κωδικό πρόσβασης του στην υπηρεσία του Apps.

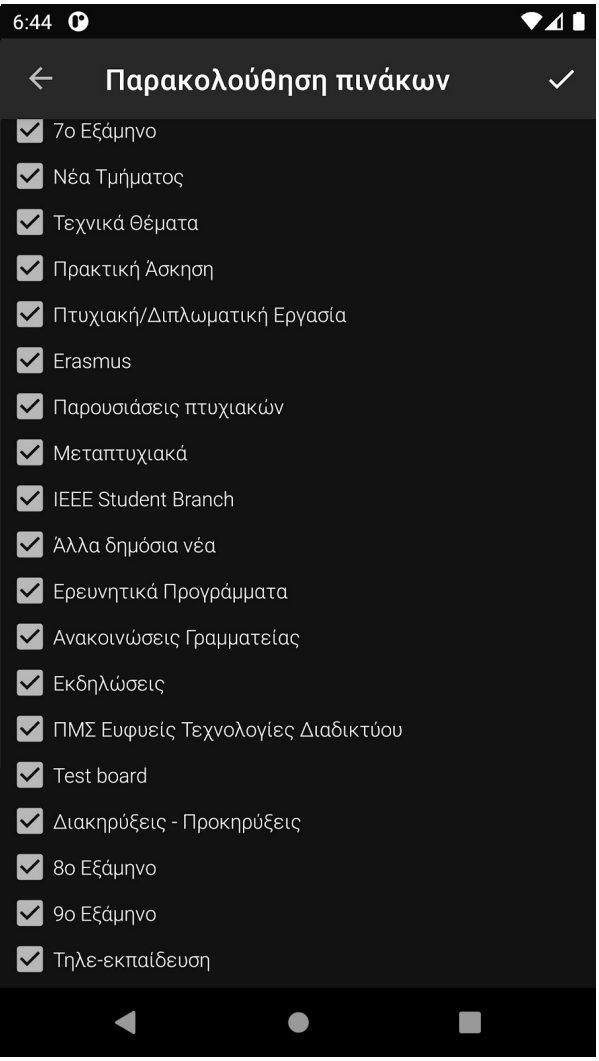
5.3.3 Παρακολούθηση πινάκων

Η παρακολούθηση πινάκων είναι ένα μενού όπου ο χρήστης μπορεί να επιλέξει να λαμβάνει ειδοποιήσεις σε μια κατηγορία που τον ενδιαφέρει.

Προσοχή: αυτή δεν είναι η επιλογή για τα push notifications όπως θα δούμε πιο κάτω.



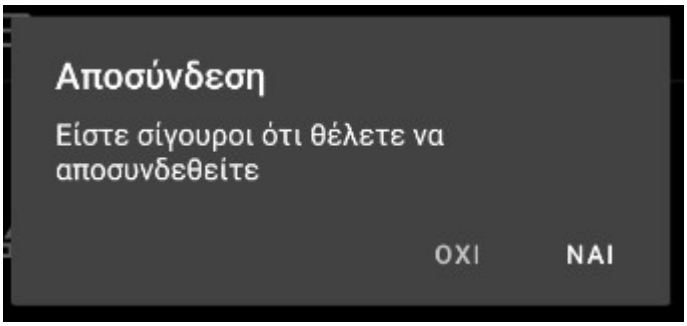
Σχήμα 40: Παρακολούθηση πινάκων 1



Σχήμα 41: Παρακολούθηση πινάκων 2

5.3.4 Αποσύνδεση

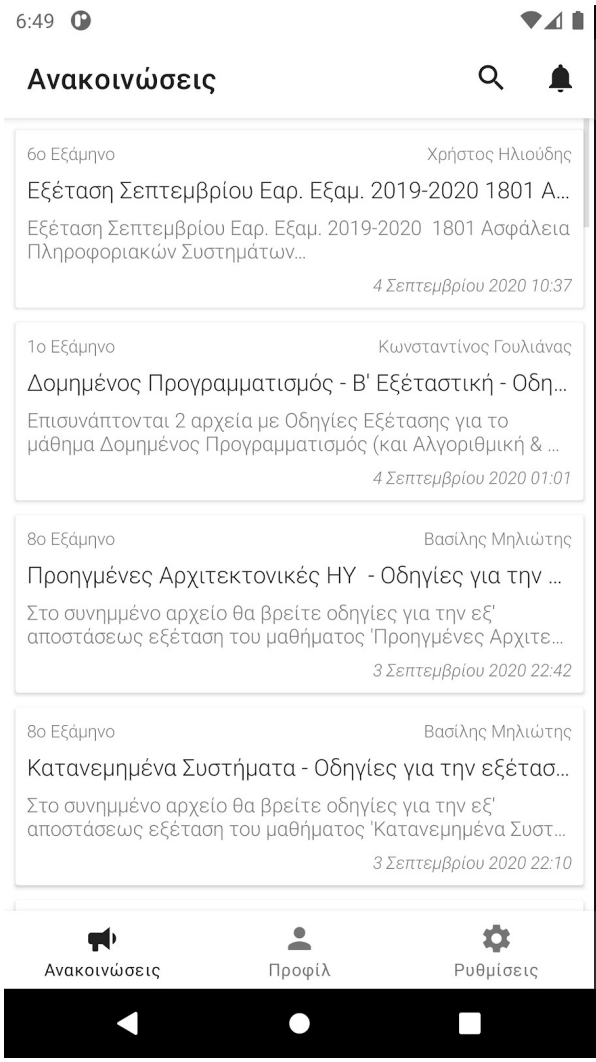
Η επιλογή αποσύνδεση επιτρέπει στον χρήστη να αποσυνδεθεί από την εφαρμογή μετά από επιβεβαίωση της ενέργειας.



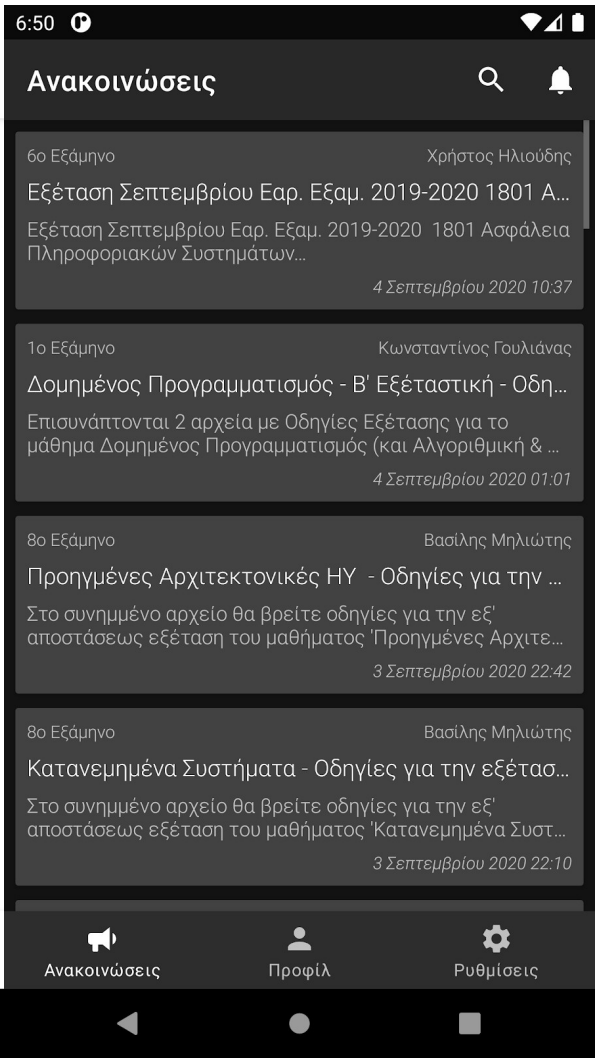
Σχήμα 42: Αποσύνδεση

5.3.5 Θέμα

Η ρύθμιση θέμα δίνει την δυνατότητα στον χρήστη να αλλάξει τα χρώματα της εφαρμογής, σε λευκό ή σκούρο χρώμα. Προεπιλεγμένη επιλογή είναι το αυτόματο, δηλαδή αποφασίζει η εφαρμογή με βάση το θέμα που έχει ο κάθε χρήστης στην Android συσκευή του ή με βάση την επιλογής της εξοικονόμησης ενέργειας.



Σχήμα 43: Θέμα 1



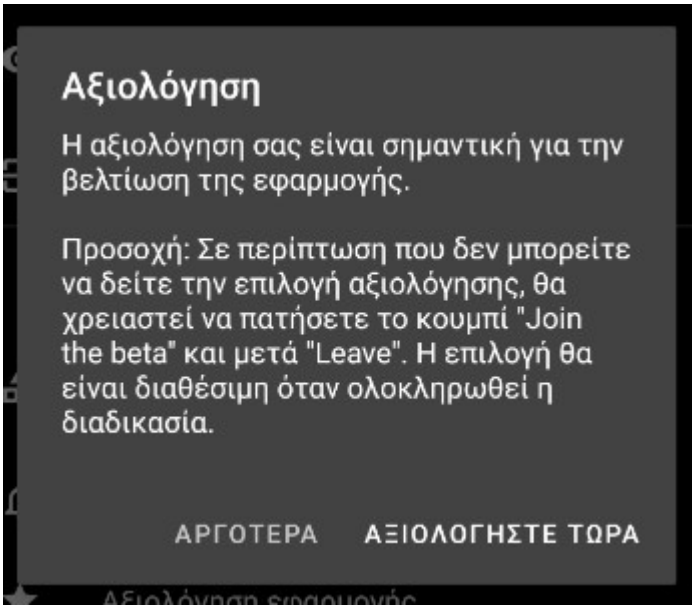
Σχήμα 44: Θέμα 2

5.3.6 Ειδοποιήσεις

Στην επιλογή ειδοποιήσεις, ο χρήστης έχει την επιλογή να ενεργοποιήσει τα push notifications για να λαμβάνει ειδοποιήσεις στην συσκευή του όταν δημιουργείτε μια νέα ανακοίνωση στις επιλεγμένες κατηγορίες του “Παρακολούθηση πινάκων”

5.3.7 Αξιολόγηση εφαρμογής

Η επιλογή αξιολόγηση εφαρμογής εμφανίζει ένα κείμενο με πληροφορίες σχετικά για το πως ο χρήστης μπορεί να αξιολογήσετε την εφαρμογή, δίνοντας του την επιλογή να μεταβεί στο PlayStore και να αξιολογήσει εκείνη την στιγμή.



Σχήμα 45: Αξιολόγηση

5.3.8 Αναφορά προβλήματος

Στην αναφορά προβλήματος, η εφαρμογή ανακατευθύνει τον χρήστη στην εφαρμογή email του, συμπληρώνοντας τα πεδία email και τίτλο έτσι ώστε να μπορεί να αναφέρει το πρόβλημα που μπορεί να αντιμετωπίζει και να το στείλει στον δημιουργό της εφαρμογής.

5.3.9 Πληροφορίες εφαρμογής

Στις πληροφορίες εφαρμογής ο χρήστης μπορεί να διαβάσει πληροφορίες σχετικά με την εφαρμογή.

Σχήμα 46: Πληροφορίες

5.3.10 Πολιτική απορρήτου

Τέλος, η επιλογή πολιτική απορρήτου ανακατευθύνει τον χρήστη στο κείμενο πολιτικής απορρήτου της εφαρμογής.

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

Η εφαρμογή είναι σε λειτουργία από τον Δεκέμβριο του 2018. Από την πρώτη στιγμή εκατοντάδες φοιτητές στήριξαν την εφαρμογή και εξακολουθούν να την στηρίζουν μέχρι και σήμερα. Μέχρι τον Σεπτέμβριο του 2020, 1280 χρήστες κατέβασαν την εφαρμογή συνολικά και 865 συσκευές είναι ενεργές. Από τις 865 ενεργές συσκευές ενεργοί χρήστες είναι 780 και περίπου 500 από αυτούς επισκέπτονται καθημερινά την εφαρμογή σε εξεταστικές περιόδους, 350 σε ενεργά εξάμηνα και 200 στις διακοπές. Βάση τα παραπάνω στατιστικά, συμπεραίνω ότι χρήση της εφαρμογής από φοιτητές είναι αναγκαία για την γρήγορη και σωστή ενημέρωση μέσω των push notifications για νέες ανακοινώσεις που διαθέτει.

Η εφαρμογή σε αυτό το σημείο υλοποιεί τις περισσότερες υπηρεσίες που προσφέρει το IT_API εκτός από τις υπηρεσίες που χρειάζονται μεγαλύτερου επιπέδου πρόσβαση. Μελλοντικά θα μπορούσαν να υλοποιηθούν λειτουργίες για τους καθηγητές έτσι ώστε να μπορούν να δημιουργούν, επεξεργάζονται ή ακόμα και να διαγράφουν μια ανακοίνωση. Επίσης θα μπορούσε να προστεθεί η υπηρεσία καταλόγου έτσι ώστε οι φοιτητές να μπορούν να βρίσκουν πληροφορίες και στοιχεία επικοινωνίας για τους καθηγητές.

Στο άμεσο μέλλον, η εφαρμογή θα γίνει ανοιχτού κώδικα στο github, έτσι ώστε να δοθεί η δυνατότητα σε φοιτητές του τμήματος που ενδιαφέρονται να διορθώσουν, βελτιώσουν ή ακόμα και να προσθέσουν λειτουργίες στην εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "Adapter." https://refactoring.guru/design-patterns/adapter (accessed Sep. 09, 2020).

[2] "Android," *Βικιπαίδεια*. May 17, 2020, Accessed: Sep. 09, 2020. [Online]. Available: https://el.wikipedia.org/w/index.php?title=Android&oldid=8252350.

[3] "Android Jetpack," *Android Developers*. https://developer.android.com/jetpack (accessed Sep. 09, 2020).

[4] "Android Studio," *Wikipedia*. Sep. 01, 2020, Accessed: Sep. 09, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=976143679.

[5] J. R. Raphael, "Android versions: A living history from 1.0 to 11," *Computerworld*, Jun. 26, 2020. https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html (accessed Sep. 09, 2020).

[6] A. Pavlidis, *apavlidi/IT_API*. 2020.

[7] "API," *Wikipedia*. Sep. 08, 2020, Accessed: Sep. 09, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=API&oldid=977377213.

[8] "Architectural pattern," *Wikipedia*. Jun. 27, 2020, Accessed: Sep. 09, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Architectural_pattern&oldid=964696387.

[9] "Basic components of Android Application | Android Programming by Wideskills." https://www.wideskills.com/android/overview-android/principal-ingredients-android (accessed Sep. 09, 2020).

[10]"Cloud Functions for Firebase," *Firebase*. https://firebase.google.com/docs/functions (accessed Sep. 11, 2020).

[11] "Data Binding Library," *Android Developers*. https://developer.android.com/topic/libraries/data-binding (accessed Sep. 09, 2020).

[12] "Firebase," *Wikipedia*. Aug. 17, 2020, Accessed: Sep. 11, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Firebase&oldid=973553570.

[13] "Firebase Cloud Messaging," *Firebase*. https://firebase.google.com/docs/cloud-messaging (accessed Sep. 11, 2020).

[14] "Firebase Crashlytics," *Firebase*. https://firebase.google.com/docs/crashlytics (accessed Sep. 11, 2020).

[15] "Firebase Realtime Database," *Firebase*. https://firebase.google.com/docs/database (accessed Sep. 11, 2020).

[16] "java_book_EMP.pdf." Accessed: Sep. 09, 2020. [Online]. Available: https://people.iee.ihu.gr/~sfetsos/java_book_EMP.pdf.

[17] "Kotlin (programming language)," *Wikipedia*. Sep. 03, 2020, Accessed: Sep. 09, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Kotlin_(programming_language)&oldid=976487642.

[18] "LiveData Overview," *Android Developers*. https://developer.android.com/topic/libraries/architecture/livedata (accessed Sep. 09, 2020).

[19] “Meet RxJava: The Missing Reactive Programming Library for Android,” *Toptal Engineering Blog*. https://www.toptal.com/android/functional-reactive-android-rxjava (accessed Sep. 09, 2020).

[20] “MVC Framework - Introduction - Tutorialspoint.” https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (accessed Sep. 11, 2020).

[21] H. Saleh, “MVVM architecture, ViewModel and LiveData (Part 1),” *Medium*, Oct. 13, 2018. https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1 (accessed Sep. 09, 2020).

[22] “Navigation,” *Android Developers*. https://developer.android.com/guide/navigation (accessed Sep. 09, 2020).

[23] “Paging library overview,” *Android Developers*. https://developer.android.com/topic/libraries/architecture/paging (accessed Sep. 09, 2020).

[24] “Singleton.” https://refactoring.guru/design-patterns/singleton (accessed Sep. 09, 2020).

[25] 262588213843476, “The introduction to Reactive Programming you’ve been missing,” *Gist*. https://gist.github.com/staltz/868e7e9bc2a7b8c1f754 (accessed Sep. 09, 2020).

[26] “ViewModel Overview,” *Android Developers*. https://developer.android.com/topic/libraries/architecture/viewmodel (accessed Sep. 09, 2020).

[27] “What is a RESTful API (REST API) and How Does it Work?,” *SearchAppArchitecture*. https://searchapparchitecture.techtarget.com/definition/RESTful-API (accessed Sep. 09, 2020).

[28] “What is an IDE?” https://www.redhat.com/en/topics/middleware/what-is-ide (accessed Sep. 09, 2020).

[29] “What is Android? Introduction of Android OS & it’s Applications,” *ElProCus - Electronic Projects for Engineering Students*, Oct. 29, 2013. https://www.elprocus.com/what-is-android-introduction-features-applications/ (accessed Sep. 09, 2020).

[30] “What’s a design pattern?” https://refactoring.guru/design-patterns/what-is-pattern (accessed Sep. 09, 2020).