ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής άμεσης ανταλλαγής μηνυμάτων για κινητές συσκευές με λειτουργικό σύστημα Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Απόστολου Χ. Γναρδέλλη

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής άμεσης ανταλλαγής μηνυμάτων για κινητές συσκευές με λειτουργικό σύστημα Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Απόστολου Χ. Γναρδέλλη

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Ιουνίου 2018.

.....	.....	.....
Ιάκωβος Βενιέρης Καθηγητής Ε.Μ.Π.	Δήμητρα Θεοδώρα Κακλαμάνη Καθηγήτρια Ε.Μ.Π.	Γιώργος Ματσόπουλος Αναπληρωτής Καθηγητής Ε.Μ.Π

Αθήνα, Ιούνιος 2018

.....

Απόστολος Χ. Γναρδέλλης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Για την εκπόνηση της παρούσας διπλωματικής εργασίας, την ανάπτυξη της εφαρμογής, αλλά και γενικότερα για την ολοκλήρωση του κύκλου των σπουδών μου, νιώθω την ανάγκη να ευχαριστήσω όλους τους ανθρώπους που με βοήθησαν.

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής μου εργασίας, καθηγητή του Εθνικού Μετσόβιου Πολυτεχνείου, κ. Ιάκωβο Βενιέρη, ο οποίος μου έδωσε τη δυνατότητα να ασχοληθώ με ένα αντικείμενο που πραγματικά επιθυμούσα. Τον ευχαριστώ ιδιαίτερα για τις πολύτιμες γνώσεις και συμβουλές που μου παρείχε κατά την εκπόνηση της εργασίας, αλλά και καθ΄όλη τη διάρκεια των σπουδών μου. Επίσης, ευχαριστώ θερμά τους καθηγητές του Ε.Μ.Π., κ. Δήμητρα Θεοδώρα Κακλαμάνη και κ. Γεώργιο Ματσόπουλο, για τη διάθεσή τους να συμμετέχουν στην τριμελή εξεταστική επιτροπή της διπλωματικής μου.

Στη συνέχεια, θα ήθελα να εκφράσω τις ιδιαίτερες ευχαριστίες μου στον υποψήφιο Διδάκτορα κ. Εμμανουήλ Καραμανή, διότι χωρίς τη βοήθειά του η περάτωση της διπλωματικής εργασίας θα ήταν αδύνατη. Οι συμβουλές του στην ανάπτυξη της εφαρμογής και η καθοδήγησή του στη συγγραφή της εργασίας υπήρξε καθοριστική.

Είμαι εξαιρετικά ευγνώμων στη Δρ. Σοφία Καπελλάκη, μέλος του επιστημονικού προσωπικού του εργαστηρίου, η οποία μου πρόσφερε την απεριόριστη βοήθειά της, όποτε αυτή ζητήθηκε.

Επίσης, θα ήθελα να ευχαριστήσω τους φίλους και τους δικούς μου ανθρώπους, για όλες τις στιγμές που μου πρόσφεραν κατά την διάρκεια των σπουδών μου.

Τέλος, θέλω να ευχαριστήσω τους γονείς μου, που πάντα με στηρίζουν αδιάκοπα και στους οποίους οφείλω ό,τι έχω πετύχει στη μέχρι τώρα πορεία μου.

Περίληψη

Η εντατικοποίηση των ρυθμών της καθημερινότητας των ανθρώπων, με άμεση συνέπεια τη μείωση του ελεύθερου χρόνου τους, έχει περιορίσει σημαντικά τις κοινωνικές επαφές τους, κάνοντας την ανάγκη για επικοινωνία διαρκώς μεγαλύτερη. Τη λύση σε αυτό το πρόβλημα ήρθαν να δώσουν οι «έξυπνες» συσκευές, οι οποίες παρέχοντας μια πληθώρα εφαρμογών, περιλαμβάνουν και εφαρμογές προορισμένες για επικοινωνία και κοινωνική δικτύωση.

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός, η ανάπτυξη και η υλοποίηση μίας εφαρμογής άμεσης ανταλλαγής μηνυμάτων για κινητά που χρησιμοποιούν το λειτουργικό σύστημα Android. Η εφαρμογή που δημιουργήσαμε ονομάστηκε "***Social Network***".

Στην εφαρμογή αυτή ο χρήστης έχει τη δυνατότητα να ανταλλάσει γραπτά μηνύματα και φωτογραφίες με τους φίλους του (όχι με όλους τους χρήστες), οι οποίοι θα έχουν διαμορφωθεί μετά από κάποιο αίτημα φιλίας (friend request) που θα έχει γίνει αποδεκτό (accept request). Ο χρήστης μπορεί οποιαδήποτε στιγμή να αναδιαμορφώσει τη λίστα των φίλων του, διαγράφοντας κάποιον από αυτούς και κάνοντας νέους. Επίσης, έχει τη δυνατότητα να ακυρώσει ένα αίτημα φιλίας, να το ξαναστείλει ή να το απορρίψει. Προκειμένου ο χρήστης να χρησιμοποιήσει την εφαρμογή, θα πρέπει πρωτίστως να έχει δημιουργήσει λογαριασμό. Έχει τη δυνατότητα να ανανεώνει την φωτογραφία προφίλ (κάνοντας crop στην επιθυμητή φωτογραφία, πριν την ανεβάσει) και το στάτους του. Τέλος, μπορεί να βλέπει ποιοι φίλοι του είναι συνδεδεμένοι οποιαδήποτε στιγμή ή, σε διαφορετική περίπτωση, πριν πόση ώρα συνδέθηκαν.

Η υλοποίηση της εφαρμογής έγινε με τη βοήθεια ενός υπολογιστή Dell (με λειτουργικό σύστημα Windows 10) και του προγραμματιστικού περιβάλλοντος ανάπτυξης εφαρμογών Android Studio (Android SDK) της Google. Για τη δοκιμή της εφαρμογής χρησιμοποιήθηκε ο προσομοιωτής Android Emulator καθώς και μια συσκευή Samsung Galaxy S7 με έκδοση λειτουργικού συστήματος Android 7.0.0. Το ρόλο του server είχε η πλατφόρμα της Google, Firebase, η οποία παρείχε τις υπηρεσίες της ταυτοποίησης στοιχείων, της βάσης δεδομένων και της αποθήκευσης.

Λέξεις κλειδιά: Android, εφαρμογή άμεσης ανταλλαγής μηνυμάτων, Android Studio (Android SDK), Firebase, JSON, βάση δεδομένων πραγματικού χρόνου, ταυτοποίηση στοιχείων, αποθήκευση.

Abstract

The constant accelerating rhythms of daily life, with immediate impact on the reduction of leisure time, has considerably diminished social contacts, making the need for communication even greater. The solution to this problem came from "smart" devices, which provide a wide range of applications, including those designed for communication and social networking.

The subject of the certain thesis is the designing, development and implementation of an instant messaging application for mobile phones, running under Android operating system. The performed application was called "Social Network".

In “Social Network” the user can exchange text messages and images with friends (not all users) who have been made through a request/accept process. The user can edit his friend list at any time by deleting one of them and making new ones. He also has the ability to cancel a friend request, resend or reject it. In order to use the application, the user must first create an account. Moreover, he has the ability to update the profile photo (by cropping the desired photo before uploading it) and his status. Finally, the user is able to see whether his friends are online or alternatively, when was the last time they were.

The development of the application was performed using a Dell computer (running under Windows 10) and the official Integrated Development Environment “Android Studio” (Android SDK), provided by Google. The application was tested in the simulator “Android Emulator”, as well as in a Samsung Galaxy S7 device, running Android 7.0.0. As a server platform, Google’s Firebase was used, providing services of Authentication, Real-Time Database and Storage.

Keywords: Android, instant messaging application, Android Studio (Android SDK), Firebase, JSON, Real-Time Database, Authentication, Storage.

Πίνακας περιεχομένων

Πίνακας εικόνων

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1. Η εξέλιξη των τηλεπικοινωνιών και τα “smartphones”

Ήταν το 1844 όταν στάλθηκε το πρώτο τηλεγραφικό μήνυμα. Ταξίδεψε 40 μίλια, γεγονός αδιανόητο για εκείνη την εποχή, και το σύστημα μηνυμάτων που χρησιμοποιούσε έστελνε ηλεκτρικά σήματα μέσω καλωδίων.

Το 1861 οι δύο ακτές των Ηνωμένων Πολιτειών ενώνονται μέσω τηλεγράφου, ενώ υπάρχουν ήδη 2250 γραφεία τηλεγράφου σε όλη τη χώρα. Το γεγονός αυτό έθεσε τις βάσεις για ό,τι οι εμπειρογνώμονες αποκάλεσαν «επανάσταση των επικοινωνιών». Βασισμένα σε αυτό το σύστημα καλωδίων, τα σταθερά τηλέφωνα κατασκευάστηκαν το 1876, όταν ο Graham Bell εισήγαγε αυτή την τεχνολογία. Τα καλώδια κρατήθηκαν σε ψηλούς πυλώνες ή τοποθετήθηκαν κάτω από το έδαφος και έδωσαν τη δυνατότητα στους ανθρώπους να επικοινωνούνε από χιλιάδες μίλια μακριά. Αν και τα σταθερά τηλέφωνα έχουν αρχίσει πλέον να καταργούνται, πολλά είναι τα νοικοκυριά και οι επιχειρήσεις που συνεχίζουν να χρησιμοποιούν τη σταθερή τηλεφωνία, η οποία αποτελεί τον βασικότερο τρόπο επικοινωνίας από το 1950 και μετά.

Τα καλώδια ξεπεράστηκαν ως μέσο μετάδοσης ηλεκτρικών σημάτων, χάρη στην κυψελοειδή τεχνολογία. Το 1946, η σουηδική αστυνομία ήταν η πρώτη που τηλεφώνησε μέσω κινητού τηλεφώνου, καθώς ανακαλύφθηκε ότι η ίδια τεχνολογία που μεταδίδει τη φωνή μέσω καλωδίων θα μπορούσε να γίνει ασύρματα, μέσω δορυφορικών κυμάτων. Παρά το πείραμα του 1946, το πρώτο κινητό τηλέφωνο για χρήση ιδιωτών αναπτύχθηκε το 1973 από τον Martin Cooper, μηχανικό της Motorola.

Η εξέλιξη της επικοινωνίας δεν σταμάτησε εκεί και το 1981 υλοποιήθηκε το dial-up Internet, δηλαδή μια μορφή πρόσβασης στο διαδίκτυο, η οποία, βασισμένη στις λειτουργίες του δημόσιου τηλεφωνικού δικτύου μεταγωγής, πραγματοποιούσε σύνδεση σε έναν πάροχο υπηρεσιών διαδικτύου (ISP), καλώντας έναν τηλεφωνικό

αριθμό σε μια συμβατική γραμμή τηλεφώνου [3]. Το επίτευγμα αυτό κατέστησε το email ως ένα δημοφιλή τρόπο για να συνδεθεί κάποιος με τους φίλους του, την οικογένειά του και τον εργασιακό του χώρο. Δέκα χρόνια αργότερα η dial-up σύνδεση άρχισε να αντικαθίσταται από τη χρήση του WiFi και λίγο καιρό αργότερα, το τελευταίο, έδωσε για πρώτη φορά την ευκαιρία στους επιστήμονες να αρχίσουν να πειραματίζονται με το «έξυπνο τηλέφωνο» (smartphone) [2].

Η εξέλιξη της κινητής τηλεφωνίας ήταν τόσο ραγδαία, που μόλις σε 21 χρόνια, από το 1990 μέχρι το 2011, κατάφερε να διεισδύσει στο 87% του παγκόσμιου πληθυσμού, με 6 δισεκατομμύρια συνδέσεις [1].

Σήμερα, 45 χρόνια μετά την πρώτη κλήση, η επικοινωνία έχει αλλάξει ριζικά. Με την ανάπτυξη της τεχνολογίας και του διαδικτύου, οι ανάγκες των ανθρώπων για επικοινωνία τροποποιούνται και παίρνουν διαφορετικές μορφές. Στο διαδίκτυο πλέον κυριαρχούν ιστοσελίδες κοινωνικής δικτύωσης (όπως το Twitter, Facebook, Google+), ιστοσελίδες προορισμένες για συζήτηση (chat) μεταξύ των χρηστών και ιστοσελίδες αναπαραγωγής πολυμέσων (YouTube), οι οποίες ελέγχονται από εταιρίες κολοσσούς όπως είναι το Facebook , η Google, η Microsoft και η Yahoo. Γι’ αυτό το λόγο, η κατασκευή και η χρήση των smartphones, την τελευταία δεκαετία έμοιαζε απαραίτητη, προκειμένου να καλυφθούν όλες αυτές οι ανάγκες και να δημιουργηθούν νέες. Στις νέες αυτές ανάγκες συγκαταλέγονται: η χρήση εφαρμογών άμεσης ανταλλαγής μηνυμάτων (Messenger, Viber, WhatsApp), πλοήγησης (GPS), παιχνιδιών, φωτογραφικής μηχανής και πολλές άλλες, οι οποίες προσφέρουν εκτός από κάλυψη των αναγκών επικοινωνίας, εξίσου, ψυχαγωγία, ενημέρωση καθώς και τη δυνατότητα οι χρήστες να μοιράζονται πράγματα (με εμπειρία χρήσης ανάλογη ενός υπολογιστή, αλλά αυτή την φορά εν κινήσει).

Έτσι, το 2007 η Apple παρουσίασε το πρώτο της smartphone, το iPhone, ενώ το παράδειγμα της ακολούθησαν οι μεγάλες εταιρίες του χώρου, όπως η Samsung, η Sony, η Nokia και η HTC. Με τη νέα αυτή τεχνολογία, οι άνθρωποι απέκτησαν τη δυνατότητα να κρατάνε στα χέρια τους συσκευές με υπολογιστική ισχύ μεγαλύτερη από πολλούς υπολογιστές [1]. Η δυνατότητα των χρηστών να περιηγηθούν στο διαδίκτυο ξεκίνησε με την ανάπτυξη των δικτύων τρίτης γενιάς (3G), συνέχισε με τη δημιουργία και ενσωμάτωση των δικτύων τέταρτης γενιάς (4G) και, πλέον, βρισκόμαστε στο σημείο όπου περιμένουμε την εισαγωγή των δικτύων πέμπτης γενιάς (5G). Τα τελευταία αναμένεται να αποτελέσουν την κινητήρια δύναμη τις

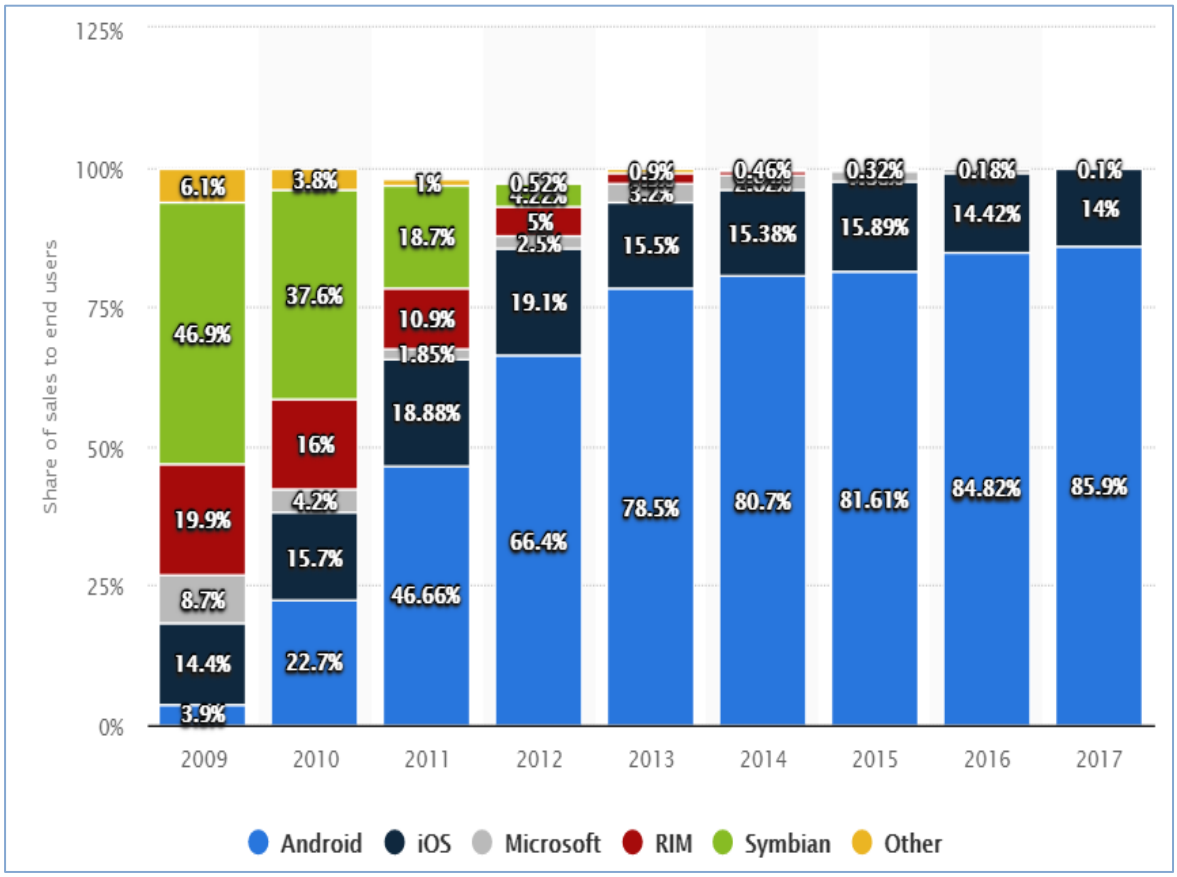
παγκόσμιας οικονομίας (τα αυτοκίνητα χωρίς οδηγό, η εικονική πραγματικότητα, οι έξυπνες πόλεις, τα ρομπότ, είναι μόνο κάποια παραδείγματα εφαρμογών τους).

Εικόνα 1 Το πρώτο κινητό τηλέφωνο ιδιωτικής χρήσης (1973).

Εικόνα 2 Το Samsung Galaxy S8 (2017).

1.2. Τα λειτουργικά συστήματα και οι εφαρμογές

Οι «έξυπνες» συσκευές λειτουργούν συνήθως χρησιμοποιώντας ένα από τα δύο δημοφιλέστερα λειτουργικά συστήματα, το Android (Google) και το iOS (Apple). Η Google έχει κατασκευάσει κάποιες συσκευές που διαθέτουν το Android, ενώ συγχρόνως το λειτουργικό σύστημα αυτό χρησιμοποιείται εκτενώς από τις Samsung, Sony, HTC, Motorola, LG, Xiaomi, Lenovo, Nokia και πολλές άλλες. Από την άλλη, η Apple χρησιμοποιεί αποκλειστικά η ίδια το iOS, με τη σειρά κινητών τηλεφώνων iPhone. Κάποια άλλα λειτουργικά συστήματα τα οποία πριν μερικά χρόνια ήταν στο προσκήνιο, αλλά πλέον έχουν σχεδόν εξαλειφθεί λόγω της συντριπτικής κυριαρχίας των δύο προαναφερθέντων συστημάτων, είναι το Symbian της Nokia, το Blackberry OS της RIM, το Windows OS της Microsoft και το Open WebOS της Palm/HP [4]. Τα τελευταία χρόνια, το Android και το iOS, εμφανίζονται να έχουν σχεδόν όλο το μερίδιο της αγοράς των λειτουργικών συστημάτων, με ποσοστό που αγγίζει το 99,9%, με το 0,1% να αντιστοιχεί σε Windows Phone [5].

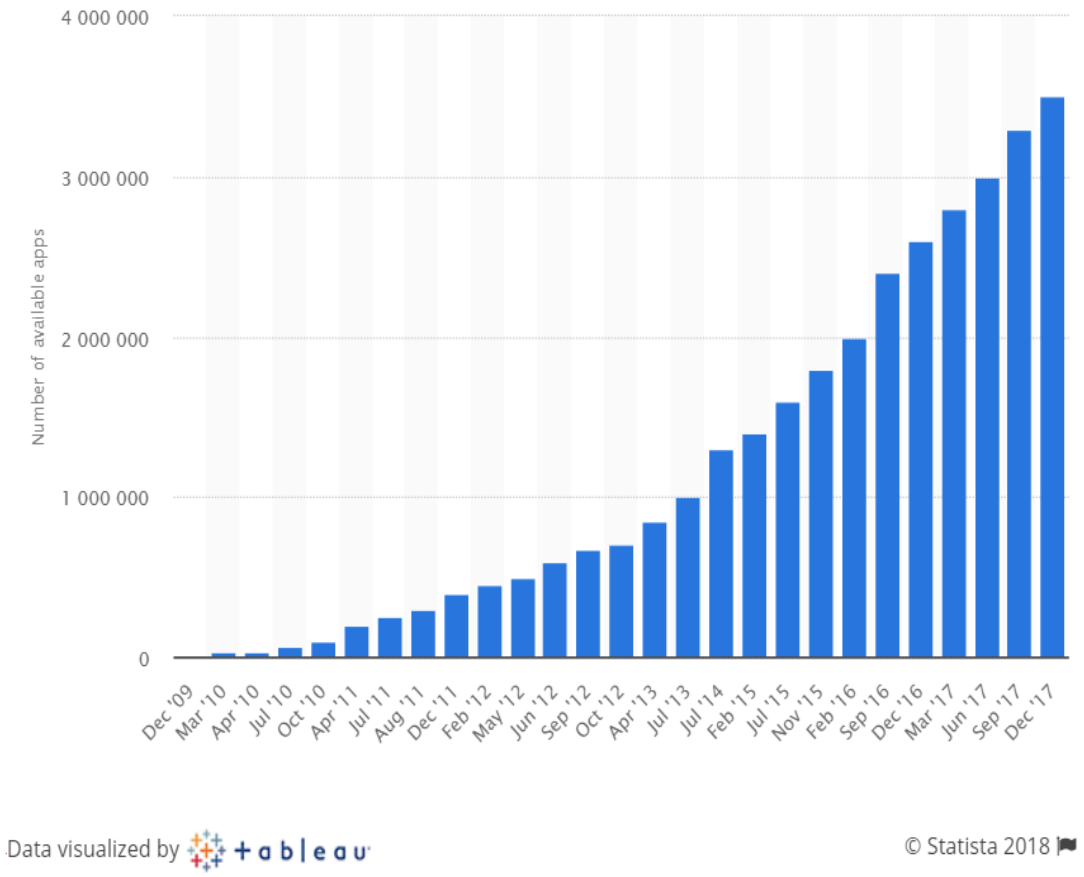


Εικόνα 3 Το μερίδιο της αγοράς των λειτουργικών συστημάτων από το 2009 μέχρι το 2017.

Τα λειτουργικά συστήματα κινητών τηλεφώνων (Mobile OS) προσφέρουν μια πληθώρα εφαρμογών, που έχουν τη δυνατότητα να καλύψουν τις ανάγκες και του πιο απαιτητικού χρήστη. Οι πιο γνωστές από αυτές είναι οι εφαρμογές τηλεφώνου, μηνυμάτων (SMS), ηλεκτρονικού ταχυδρομείου (email), λήψης φωτογραφιών, περιήγησης ιστού, προβολής καιρού, αναπαραγωγής πολυμέσων, ημερολογίου, αριθμομηχανής, εγγραφής φωνής, πλοήγησης και παρακολούθησης των καθημερινών δραστηριοτήτων (π.χ. καρδιακοί παλμοί, χιλιόμετρα που έχει διανύσει , μέτρηση του άγχους).

Τα λειτουργικά αυτά συστήματα όμως δίνουν την ευκαιρία και στους προγραμματιστές να δημιουργήσουν τις δικές τους εφαρμογές. Για αυτόν το λόγο, τα ηλεκτρονικά καταστήματα (εφαρμογών), περιέχουν εκατομμύρια εφαρμογές, από παιχνίδια και εφαρμογές ενημέρωσης μέχρι σύνθετες εφαρμογές επεξεργασίας φωτογραφιών και πολυμέσων. Διαθέτουν εφαρμογές για όλα τα κοινωνικά δίκτυα, στα οποία ο χρήστης μπορεί να συνδεθεί ανά πάσα στιγμή, ενώ σημαντικό χαρακτηριστικό τους είναι η συνδεσιμότητα με τους ηλεκτρονικούς υπολογιστές και με άλλες παρεμφερείς συσκευές, όπως τα tablets και οι smart TVs. Τέλος, οι χρήστες μπορούν να χρησιμοποιήσουν Cloud υπηρεσίες. Οι τελευταίες, τους δίνουν τη δυνατότητα να μην αποθηκεύουν τα δεδομένα τους στη συσκευή τους και να τα μεταφέρουν σε ένα διαδικτυακό «νέφος», στο οποίο έχουν πρόσβαση μέσω του διαδικτύου από οποιαδήποτε συσκευή και τοποθεσία [1].

Τα λειτουργικά συστήματα διαθέτουν, όπως προαναφέραμε, ηλεκτρονικά καταστήματα, στα οποία ο χρήστης μπορεί να περιηγηθεί στις διαθέσιμες για τη συσκευή του εφαρμογές και να επιλέξει να εγκαταστήσει όποια από αυτές επιθυμεί. Οι περισσότερες εφαρμογές έχουν ως αντίτιμο ένα μικρό ποσό της τάξης των 1-5 ευρώ, ή διατίθενται δωρεάν. Στο κατάστημα της Google, το Google Play Store, υπάρχουν τουλάχιστον 3.500.000 εφαρμογές και ο συνολικός αριθμός των λήψεων υπολογίζεται στα 70 δισεκατομμύρια (μόνο για τα Android OS!). Πρόκειται λοιπόν για έναν τομέα που απασχολεί εκατομμύρια προγραμματιστές σε όλο τον κόσμο, συνδέοντας καθημερινά εκατομμύρια χρήστες, οι οποίοι, χρησιμοποιώντας τα «έξυπνα» κινητά τους τηλέφωνα, επικοινωνούν και αλληλεπιδρούν μεταξύ τους [6].



Εικόνα 4 Ο συνολικός αριθμός διαθέσιμων εφαρμογών στο Google Play Store μέχρι τον Δεκέμβριο του 2017.

Η χρήση των smartphones και των εφαρμογών τους έχει διεισδύσει μέσα στην καθημερινότητα του ανθρώπου και επηρεάζει πολλές πτυχές της ζωής του. Για την επικοινωνία με τους συνανθρώπους του, τη δουλειά, τη διασκέδαση, την ενημέρωση, τη μόρφωση, αλλά πολλές φορές ακόμη και την υγεία του, ο σύγχρονος άνθρωπος χρησιμοποιεί το κινητό του τηλέφωνο. Το γεγονός αυτό, τον ωθεί να επιζητεί ακόμα πιο αποδοτικές συσκευές, οι οποίες θα διαθέτουν εφαρμογές πιο χρηστικές που θα ικανοποιούν ολοένα και περισσότερες ανάγκες του, όσο πολύπλοκες κι αν είναι αυτές. Ως αποτέλεσμα, πολλές εταιρείες επενδύουν σε αυτόν τον τομέα, δημιουργώντας έναν ανταγωνισμό τεράστιας κλίμακας με τελικό κερδισμένο τον καταναλωτή [1].

Εικόνα 5 Οι τυπικές αρχικές οθόνες των δύο δημοφιλέστερων λειτουργικών συστημάτων (δεξία είναι το iOS 11 στο iPhone X και αριστερά το Android 8.0 στο Samsung Galaxy S9).

1.3. Η εξέλιξη των μέσων ανταλλαγής μηνυμάτων και των κοινωνικών δικτύων

Η κοινωνική δικτύωση εμφανίσθηκε για πρώτη φορά το 1971, όταν δημιουργήθηκε το πρώτο μήνυμα ηλεκτρονικού ταχυδρομείου. Οι δύο υπολογιστές ήταν τοποθετημένοι ακριβώς ο ένας δίπλα στον άλλον και το μήνυμα έγραφε "qwertyuiop". Το 1978 δημιουργήθηκε το BBS ή Bulletin Board System, το οποίο φιλοξενήθηκε σε προσωπικούς υπολογιστές και στο οποίο οι χρήστες καλούσαν το μόντεμ ενός κεντρικού υπολογιστή, για να ανταλλάξουν δεδομένα μέσω τηλεφωνικών γραμμών. Αργότερα εκείνη την χρονιά, δημιουργήθηκε το Usenet, στο οποίο οι χρήστες δημοσίευαν ειδήσεις, άρθρα και αστείες δημοσιεύσεις, χωρίς πλέον να χρειάζονται έναν κεντρικό εξυπηρετητή. Η πρώτη έκδοση εφαρμογής άμεσης ανταλλαγής μηνυμάτων υλοποιήθηκε το 1988 και ονομάστηκε IRC ή Internet Relay Chat. Το IRC ήταν βασισμένο στο Unix και συνεπώς αφορούσε μόνο λίγους ανθρώπους. Το 1992, στάλθηκε το πρώτο μήνυμα SMS (Short Message Service) από το Vodafone GSM network, με περιεχόμενο “Merry Christmas”, ενώ το 1994

δημιουργήθηκε ο πρώτος ιστότοπος κοινωνικής δικτύωσης, με όνομα Geocities. Το Geocities επέτρεπε στους χρήστες να δημιουργήσουν και να προσαρμόσουν τις δικές τους ιστοσελίδες, ομαδοποιώντας τες σε διαφορετικές «πόλεις» με βάση το περιεχόμενο του ιστοτόπου. Ένα χρόνο αργότερα, το TheGlobe.com εγκαινιάστηκε στο κοινό, δίνοντας στους χρήστες τη δυνατότητα να αλληλεπιδρούν με ανθρώπους που έχουν τα ίδια χόμπι και ενδιαφέροντα, καθώς και να δημοσιεύουν το δικό τους περιεχόμενο. Λίγα χρόνια αργότερα, το 1997, ξεκίνησαν το AOL Instant Messenger και το SixDegrees.com. Η άμεση ανταλλαγή μηνυμάτων πλέον είχε υλοποιηθεί, δίνοντας στους χρήστες της, την ελευθερία να συνομιλούν με τους φίλους τους και να δημιουργούν το προφίλ τους. Ο πρώτος σύγχρονος ιστότοπος κοινωνικής δικτύωσης, όπως τον γνωρίζουμε σήμερα, ήταν ο Friendster (παρόμοια ιδέα με το SixDegrees). Σύντομα την ιδέα αυτή πήρε το MySpace, δίνοντας στους χρήστες μεγαλύτερη ελευθερία σε θέματα μουσικής και βίντεο. Το Myspace παρέχοντας ένα πιο όμορφο διαδικτυακό περιβάλλον, βρέθηκε στην κορυφή των μέσων κοινωνικής δικτύωσης. Ακολούθησε η δημιουργία του Linkedln (2003), προσανατολισμένου κυρίως σε θεάματα επαγγελματικής δικτύωσης και του Facebook (2004), το οποίο κατέληξε να έχει εκατομμύρια χρήστες παγκοσμίως, φέρνοντάς το σε καθολική θέση ισχύος μέχρι και σήμερα. Την ίδια εποχή, παρόλο που το AOL Instant Messenger κατείχε το 52% της αγοράς της άμεσης ανταλλαγής μηνυμάτων, γνώρισε ραγδαία πτώση, όταν το 2006 ήρθε αντιμέτωπο με το σκληρό ανταγωνισμό νέων υπηρεσιών, όπως το Google Talk, το Yahoo! Chat, το MSN Messenger και το Skype, για να καταλήξουμε στη σημερινή εποχή, με την κυριαρχία του Whats App και του Viber [7].

Στις μέρες μας η κοινωνική δικτύωση αποτελεί αναπόσπαστο κομμάτι της ζωής των ανθρώπων σε ολόκληρο τον κόσμο. Ο όρος κοινωνική δικτύωση (ή κοινωνικό δίκτυο) αναφέρεται σε μια ηλεκτρονική μορφή κοινωνικής αλληλεπίδρασης η οποία χρησιμοποιείται για κοινωνικούς, εκπαιδευτικούς, ενημερωτικούς ή ψυχαγωγικούς σκοπούς. Τα κοινωνικά αυτά δίκτυα εμφανίζονται σε πολλαπλές μορφές: blogs, φόρουμ, podcasts, κοινή χρήση φωτογραφιών, κοινωνικό bookmarking, widgets, βίντεο, είναι μόνο μερικές από αυτές. Οι ιστότοποι κοινωνικής δικτύωσης επιτρέπουν στους χρήστες να δημιουργούν προφίλ, να ανεβάζουν φωτογραφίες και βίντεο, να αλληλεπιδρούν με τους φίλους και την οικογένειά τους, να συμμετέχουν σε ομάδες, να μαθαίνουν τα τελευταία νέα και τις εκδηλώσεις, να παίζουν παιχνίδια, να κουβεντιάζουν, να μοιράζονται μουσική και, γενικότερα, να συμμετέχουν σε ένα μεγάλο εύρος δραστηριοτήτων.

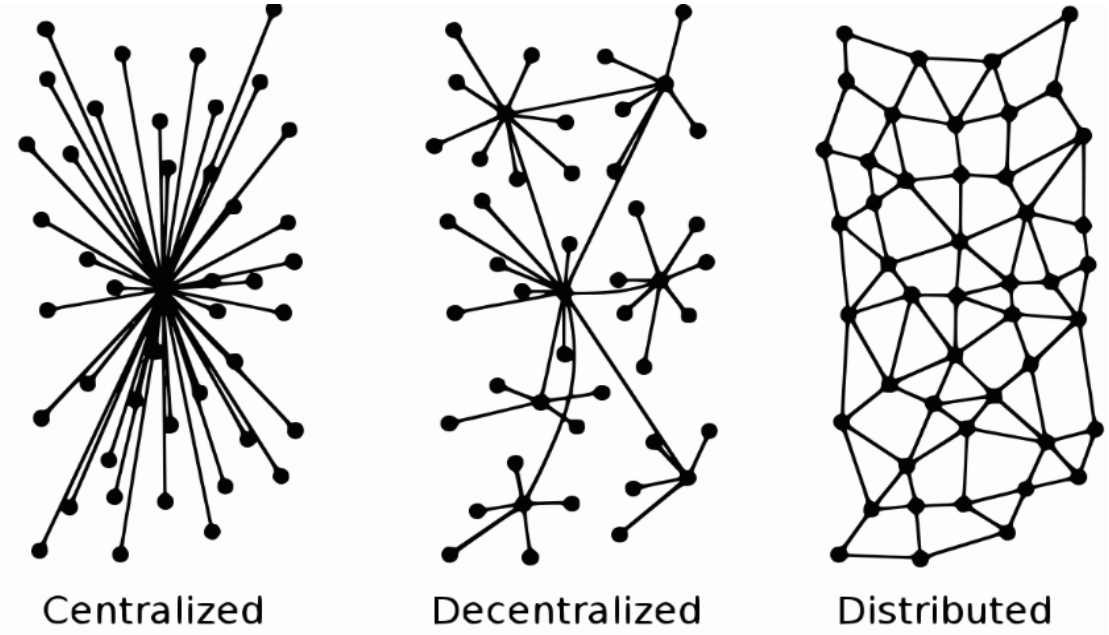
Εικόνα 6 Το AOL Instant Messenger.

1.4. Κατανεμημένο δίκτυο (σύστημα)

Ένα κατανεμημένο δίκτυο είναι ένας τύπος δικτυακής σύνδεσης στο οποίο η διαχείριση των δεδομένων γίνεται σε πολλαπλές υπολογιστικές μονάδες, παρέχοντας όμως στους χρήστες την εικόνα ενός μοναδικού συνεκτικού συστήματος. Παρέχει ένα ενιαίο σύστημα επικοινωνίας δεδομένων, η διαχείριση του οποίου γίνεται, είτε από κοινού, είτε χωριστά σε κάθε υπολογιστική μονάδα. Εκτός από την ενιαία επικοινωνία εντός του δικτύου, μία κατανεμημένη υπηρεσία μοιράζει επίσης και τις λειτουργίες της, αποφεύγοντας την επιβάρυνση ενός και μόνο κεντρικού server [8]. Οι πόροι του δικτύου τοποθετούνται και διαχειρίζονται από διαφορετικές γεωγραφικές τοποθεσίες, προστατεύοντας τη συνολική λειτουργία του δικτύου από μία πιθανή καταστροφή του εξυπηρετητή (στην περίπτωση που είχαμε μόνο έναν κεντρικό εξυπηρετητή) [9]. Μια αρχιτεκτονική υπολογιστών client/server είναι ένα παράδειγμα

κατανεμημένου δικτύου, όπου ο διακομιστής είναι ο παραγωγός ενός πόρου και πολλοί διασυνδεδεμένοι απομακρυσμένοι χρήστες είναι οι καταναλωτές που έχουν πρόσβαση στην εφαρμογή από διαφορετικά δίκτυα. Ο εξυπηρετητής π.χ. του Firebase δεν είναι μόνο μια εικονική μηχανή, αλλά αποτελείται από πολλούς εξυπηρετητές, καθένας σε διαφορετικό μέρος του κόσμου, ώστε όλοι μαζί να συνιστούν ένα κατανεμημένο δίκτυο.

Στην εφαρμογή άμεσης ανταλλαγής μηνυμάτων που θα παρουσιάσουμε, η άντληση των πληροφοριών γίνεται από τη βάση δεδομένων (database) που μας παρέχει ο εξυπηρετητής (server) του Firebase. Η βάση δεδομένων αυτή είναι κατανεμημένη, δηλαδή είναι ένας τύπος διαμόρφωσης βάσης δεδομένων που αποτελείται από χαλαρά συζευγμένα αποθετήρια δεδομένων [10]. Ο χρήστης είναι αυτός που δημιουργεί τα δεδομένα, μέσω της εγγραφής, της σύνδεσης, της ανταλλαγής μηνυμάτων κειμένου και φωτογραφιών, της αποστολής/ ακύρωσης/αποδοχής/απόρριψης αιτημάτων φιλίας, της αλλαγής της φωτογραφίας προφίλ και του status του, ενώ ταυτόχρονα είναι και αυτός που τα λαμβάνει. Κάθε δραστηριότητα του χρήστη έχει αντίκτυπο στο περιεχόμενο της βάσης δεδομένων του Firebase και κάθε χρήστης έχει πρόσβαση σε αυτά τα δεδομένα από οποιοδήποτε σημείο του πλανήτη. Η δυνατότητα αυτή, δηλαδή οποιοσδήποτε χρήστης να μπορεί να συνδεθεί με τη βάση δεδομένων του εξυπηρετητή του Firebase από οποιαδήποτε γωνιά του πλανήτη, είναι μια δυνατότητα που παρέχουν τα κατανεμημένα δίκτυα.



Εικόνα 7 Τύποι δικτύων. Στα δεξιά απεικονίζεται ένα κατανεμημένο δίκτυο.

1.5. Αντικείμενο διπλωματικής εργασίας

Το αντικείμενο της διπλωματικής εργασίας είναι ο σχεδιασμός, η ανάπτυξη και η υλοποίηση μίας εφαρμογής άμεσης ανταλλαγής μηνυμάτων στην πλατφόρμα του Android. Ο χρήστης θα μπορεί να ανταλλάσει μέσω αυτής της εφαρμογής, γραπτά μηνύματα και φωτογραφίες μόνο με τους φίλους του (όχι με όλους τους χρήστες), οι οποίοι θα έχουν δημιουργηθεί μετά από κάποιο αίτημα φιλίας (friend request) που θα έχει γίνει αποδεκτό (accept request). Προκειμένου ο χρήστης να χρησιμοποιήσει την εφαρμογή, θα πρέπει πρωτίστως να έχει δημιουργήσει λογαριασμό (εγγραφή) σε αυτή. Επιπλέον, θα έχει την δυνατότητα να ανανεώνει τη φωτογραφία προφίλ και το στάτους του, καθώς και να βλέπει ποιοί φίλοι του είναι συνδεδεμένοι εκείνη τη στιγμή ή πριν πόση ώρα έχουν συνδεθεί.

Για την ολοκλήρωση της εργασίας αυτής, αρχικά, θα πρέπει να εξοικειωθούμε με τη χρήση του λειτουργικού συστήματος Android. Γνωρίζοντας το πώς λειτουργούν οι πιο δημοφιλείς εφαρμογές και ποια είναι η επιθυμητή εμπειρία χρήσης, θα μπορέσουμε να σχεδιάσουμε την εφαρμογή έτσι ώστε να πληροί τα απαραίτητα κριτήρια [1].

Αφού η ιδέα υπάρχει και έχει γίνει η σχεδίαση της εφαρμογής, θα πρέπει να έρθουμε σε επαφή με τα εργαλεία ανάπτυξης της. Τα εργαλεία αυτά είναι το Android Studio και το Android SDK. Το Android Studio είναι το επίσημο ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) που παρέχει η Google για ανάπτυξη εφαρμογών σε Android [11], ενώ το Android SDK περιλαμβάνει όλα τα απαραίτητα πλαίσια (frameworks) για την ανάπτυξη ακόμα και της πιο σύνθετης εφαρμογής. Θα χρησιμοποιηθούν εργαλεία ελέγχου της λειτουργίας της εφαρμογής, όπως είναι ο Android Emulator καθώς και ένα σύστημα αυτοματισμού για το «χτίσιμο» της εφαρμογής, το Build Gradle. Το καινοτόμο στοιχείο της εφαρμογής αυτής είναι ότι θα αποφευχθεί ο back-end κώδικας για την ανάπτυξη κάποιου server, και όλο το κομμάτι αυτό (Authentication, Real-Time Database, Storage), θα πραγματοποιηθεί με τη χρήση του Firebase. Τέλος, απαραίτητη είναι η γνώση αντικειμενοστραφούς προγραμματισμού. Τα frameworks του Android SDK απαιτούν καλή γνώση της γλώσσας προγραμματισμού Java. Απαιτείται επίσης, γνώση των αρχείων μορφής JSON και των χαρακτηριστικών του Firebase, προκειμένου να συγχωνευθεί σωστά με το Android SDK.

1.6. Διάρθρωση της εργασίας

Στην παρούσα διπλωματική εργασία, προσπαθήσαμε να υλοποιήσουμε την εφαρμογή, ακολουθώντας με σωστό τρόπο τα στάδια που προβλέπονται. Τα στάδια αυτά παρουσιάζονται στα επόμενα κεφάλαια της. Στο 2^{ο} κεφάλαιο, περιγράφονται οι τεχνολογίες τις οποίες πρέπει να γνωρίζει ένας προγραμματιστής, έτσι ώστε να αναπτύξει μία εφαρμογή τέτοιου είδους. Στο 3^{ο} κεφάλαιο, γίνεται η ανάλυση των απαιτήσεων που πρέπει να ικανοποιεί η εφαρμογή και η περιγραφή όλων των σεναρίων χρήσης. Στο 4^{ο} κεφάλαιο, περιγράφεται η σχεδίαση της εφαρμογής και ο τρόπος με τον οποίο συγχωνεύτηκε με το Firebase. Στη συνέχεια, στο 5^{ο} κεφάλαιο, γίνεται αναφορά στην υλοποίηση της εφαρμογής (τα σημαντικότερα σημεία του κώδικα) και περιγράφεται ένα ολοκληρωμένο σενάριο χρήσης της. Τέλος, στο 6^{ο} κεφάλαιο, καταγράφονται τα συμπεράσματα που προέκυψαν μετά την ολοκλήρωση της διπλωματικής εργασίας και παρουσιάζονται πιθανές μελλοντικές επεκτάσεις.

ΚΕΦΑΛΑΙΟ 2

Τεχνολογίες

2.1. Το Android της Google

Το Android είναι ένα λειτουργικό σύστημα κινητής τηλεφωνίας που αναπτύχθηκε από την Google, βασισμένο σε μια τροποποιημένη έκδοση του Linux kernel. Αρχικά σχεδιάστηκε για φορητές συσκευές αφής, όπως τα «έξυπνα» τηλέφωνα (smartphones) και τα tablets και, στη συνέχεια, χρησιμοποιήθηκε για τηλεοράσεις (Android TV), για αυτοκίνητα (Android Auto) και για ρολόγια (Wear OS) [12]. Το λειτουργικό σύστημα διαχειρίζεται το υλικό (hardware) της συσκευής και παρέχει τις τεχνολογίες που είναι απαραίτητες για τη λειτουργία των εγκατεστημένων εφαρμογών [1]. Η συσκευή πωλείται έχοντας προεγκατεστημένες στο λειτουργικό της σύστημα κάποιες βασικές εφαρμογές, όπως το Τηλέφωνο, τα Μηνύματα, το Gmail (εφαρμογή ηλεκτρονικού ταχυδρομείου), το Google Chrome (περιηγητής ιστού), το Google Play Store και αρκετές άλλες, που παρέχουν τις τυπικές υπηρεσίες του συστήματος στο χρήστη.

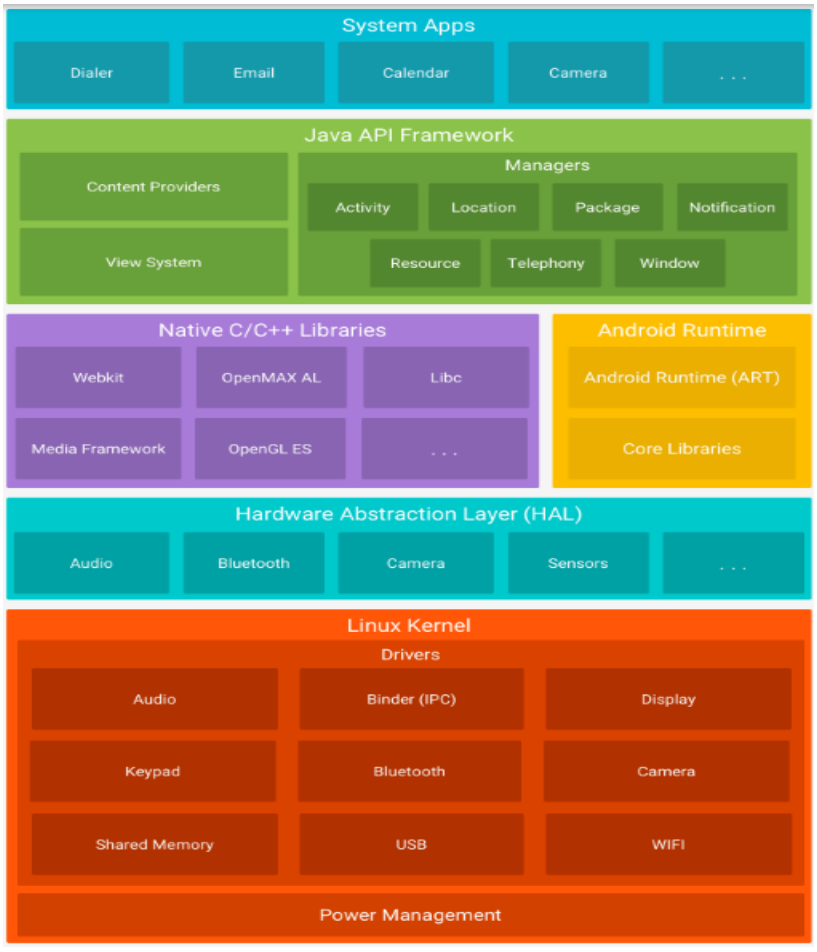
Το Android Software Development Kit (SDK) περιέχει τα εργαλεία και τις διεπαφές που απαιτούνται για την ανάπτυξη, εγκατάσταση, λειτουργία και τη δοκιμή των εφαρμογών που εμφανίζονται στην αρχική οθόνη μιας Android συσκευής. Όλες οι εφαρμογές αναπτύσσονται χρησιμοποιώντας τα πλαίσια (frameworks) του συστήματος του Android και την αντικειμενοστραφή γλώσσα προγραμματισμού Java. Οι εφαρμογές εγκαθίστανται και τοποθετούνται δίπλα στις ήδη εγκατεστημένες εφαρμογές στην αρχική οθόνη της συσκευής και είναι πάντα διαθέσιμες στο χρήστη. Η κατανόηση των τεχνολογιών και των εργαλείων που συνθέτουν το Android SDK είναι απαραίτητη για να σχεδιαστεί και να υλοποιηθεί αποτελεσματικά μία εφαρμογή τέτοιου τύπου.

Τη στιγμή που γράφεται η εργασία, το λειτουργικό σύστημα Android βρίσκεται στην έκδοση 8.0-8.1 (Oreo), με την έκδοση 9.0 να βρίσκεται σε μορφή beta.

2.1.1. Η διαστρωματωμένη αρχιτεκτονική του Android

Στο υψηλότερο επίπεδο, το Android ενεργεί ως διαμεσολαβητής μεταξύ του υποκείμενου υλικού (hardware) και των εφαρμογών που εμφανίζονται στην οθόνη. Οι εφαρμογές σπάνια επικοινωνούν άμεσα με το hardware της συσκευής. Αντιθέτως, η επικοινωνία γίνεται μέσω μιας σειράς καλά καθορισμένων διεπαφών (interfaces) του συστήματος, γεγονός που προστατεύει την εφαρμογή από αλλαγές στο hardware. Αυτό σημαίνει ότι οι εφαρμογές μπορούν να λειτουργούν σε συσκευές με διαφορετικό hardware [1]. Οι τεχνολογίες του Android μπορούν να συγκεντρωθούν στα παρακάτω στρώματα:

1. System Apps
2. Java API Framework
3. Native Libraries
4. Android Runtime
5. Hardware Abstraction Layer (HAL)
6. Linux Kernel



Εικόνα 8 Android Multi-layer Architecture.

1) Το στρώμα System Apps

Σε αυτό το στρώμα βρίσκονται όλες οι Android εφαρμογές. Το Android διαθέτει ένα σύνολο βασικών εφαρμογών όπως: μηνύματα ηλεκτρονικού ταχυδρομείου, μηνύματα SMS, ημερολόγιο, περιήγηση στο Internet, επαφές κ.λπ.. Οι εφαρμογές που περιλαμβάνονται ήδη στην πλατφόρμα δεν έχουν κάποια ειδικό status σε σχέση με αυτές που επιλέγει ο χρήστης να εγκαταστήσει. Έτσι, μία εφαρμογή που δεν είναι σχεδιασμένη από την κατασκευάστρια εταιρία του κινητού (third-party app), μπορεί να γίνει π.χ. το προεπιλεγμένο πρόγραμμα ιστού ή ο αποστολέας SMS ή ακόμη και το προεπιλεγμένο πληκτρολόγιο (ισχύουν κάποιες εξαιρέσεις, όπως για την εφαρμογή «Ρυθμίσεις», η οποία δεν μπορεί να αντικατασταθεί από άλλη).

Οι εφαρμογές του συστήματος λειτουργούν είτε ως εφαρμογές για τους χρήστες, είτε για να παρέχουν υπηρεσίες στις οποίες οι προγραμματιστές μπορούν να έχουν πρόσβαση από την εφαρμογή που έχουν δημιουργήσει. Για παράδειγμα, αν η εφαρμογή που αναπτύσσει ένας προγραμματιστής πρόκειται να παραδίδει ένα μήνυμα SMS, δεν χρειάζεται να κατασκευάσει αυτήν την υπηρεσία μόνος του, αλλά μπορεί απλά να καλέσει όποια εφαρμογή SMS είναι ήδη εγκατεστημένη και να αποστείλει το μήνυμα που επιθυμεί.

2) Το στρώμα Java API Framework

Το σύνολο των χαρακτηριστικών του Android OS είναι διαθέσιμο σε μας, μέσω των API (Application Programming Interfaces) που είναι γραμμένα σε γλώσσα Java. Αυτά τα API αποτελούν τις δομικές μονάδες που χρειάζεται κάποιος για να δημιουργήσει εφαρμογές Android, απλοποιώντας την επαναχρησιμοποίηση των βασικών διαρθρωτικών στοιχείων και των υπηρεσιών του συστήματος. Τα στοιχεία αυτά είναι:

- View System: ένα επεκτάσιμο σύνολο όψεων (views) που χρησιμοποιείται για τη δημιουργία των διεπαφών του χρήστη (user interfaces) της εφαρμογής.
- Activity Manager: ελέγχει όλες τις πτυχές του κύκλου ζωής της εφαρμογής και της στοίβας δραστηριοτήτων.
- Content Providers: επιτρέπει στις εφαρμογές να εκδίδουν και να μοιράζονται δεδομένα με άλλες εφαρμογές.
- Notifications Manager: επιτρέπει στην εφαρμογή να εμφανίζει ειδοποιήσεις στο χρήστη.

- Resource Manager: παρέχει πρόσβαση σε ενσωματωμένους πόρους (χωρίς κώδικα), όπως γραμματοσειρές, ρυθμίσεις χρωμάτων και διατάξεις διασύνδεσης χρήστη.

Οι προγραμματιστές έχουν πλήρη πρόσβαση στο ίδιο πλαίσιο (framework) API, που το σύστημα Android χρησιμοποιεί [13].

3) Το στρώμα Native Libraries

Πολλά βασικά στοιχεία και υπηρεσίες του συστήματος Android, όπως το ART και το HAL, είναι χτισμένα από ενσωματωμένο κώδικα που απαιτεί έμφυτες/μητρικές βιβλιοθήκες (native libraries) γραμμένες σε C και C ++. Η πλατφόρμα Android παρέχει το API Java Framework για να εκθέσει τις λειτουργίες ορισμένων εκ των μητρικών βιβλιοθηκών στις εφαρμογές. Για παράδειγμα, υπάρχει πρόσβαση στο OpenGL ES μέσω του Android Java API Framework OpenGL, στην περίπτωση που ένας χρήστης θέλει να προσθέσει υποστήριξη για τη σχεδίαση και το χειρισμό γραφικών 2D και 3D στην εφαρμογή του. Αν ένας προγραμματιστής αναπτύσσει μια εφαρμογή που χρειάζεται C ή C++ κώδικα, μπορεί να χρησιμοποιήσει το Android NDK (Native Development Kit) για να έχει πρόσβαση σε μερικές από αυτές τις μητρικές βιβλιοθήκες, απευθείας από το μητρικό κώδικα.

4) Το στρώμα Android Runtime

Για Android συσκευές με έκδοση 5.0 (API level 21) ή υψηλότερη, η κάθε εφαρμογή εκτελείται μέσω του δικού της Android Runtime (ART). Το ART είναι γραμμένο για να «τρέχει» σε πολλαπλές εικονικές μηχανές συσκευών χαμηλής μνήμης, εκτελώντας αρχεία DEX (μια μορφή byte-κώδικα σχεδιασμένη ειδικά για Android, η οποία έχει βελτιστοποιηθεί για ελάχιστο αποτύπωμα μνήμης). Δημιουργεί εργαλεία, όπως το Jack και μεταγλωττίζει (compile) κομμάτια Java σε DEX bytecode, τα οποία μπορούν να τρέξουν στην πλατφόρμα του Android.

Μερικά από τα κύρια χαρακτηριστικά του ART είναι τα εξής:

- Ahead-of-time (AOT) και just-in-time (JIT) μεταγλώττιση
- Βελτιστοποιημένη συλλογή απορριμμάτων (garbage collection GC)
- Καλύτερη υποστήριξη εντοπισμού σφαλμάτων (προφίλ δειγματοληψίας, λεπτομερείς διαγνωστικές εξαιρέσεις, αναφορές σφαλμάτων) και δυνατότητα καθορισμού των σημείων παρακολούθησης για καταγραφή συγκεκριμένων πεδίων.

Πριν από την έκδοση 5.0, τη θέση του Android Runtime κατείχε ο Dalvik. Συνεπώς, αν η εφαρμογή εκτελείται σωστά στο ART, τότε θα πρέπει να δουλεύει σωστά και στον Dalvik. Το αντίστροφο μπορεί να μην ισχύει.

5) Το στρώμα Hardware Abstraction Layer (HAL)

Το στρώμα αφαίρεσης υλικού (HAL) παρέχει τυποποιημένες διεπαφές που εκθέτουν τις δυνατότητες του υλικού (hardware) των συσκευών στο ανώτερο επίπεδο Java API Framework. Το HAL αποτελείται από πολλαπλές ενότητες βιβλιοθήκης, καθεμία από τις οποίες υλοποιεί μια διεπαφή για έναν συγκεκριμένο τύπο hardware, όπως η κάμερα ή το Bluetooth. Όταν ένα API framework κάνει μια κλήση για πρόσβαση στο hardware της συσκευής, το σύστημα Android φορτώνει την ενότητα της βιβλιοθήκης του εν λόγω στοιχείου του hardware.

6) Το στρώμα Linux Kernel

Το Linux Kernel αποτελεί το θεμέλιο της πλατφόρμας του Android. Για παράδειγμα, το Android Runtime βασίζεται στον πυρήνα των Linux (Linux kernel) για τις βασικές λειτουργίες, όπως είναι η διαχείριση μνήμης χαμηλού επιπέδου. Η χρήση του πυρήνα των Linux επιτρέπει στο Android να εκμεταλλευτεί τα βασικά χαρακτηριστικά ασφάλειας του και, ταυτόχρονα, δίνει τη δυνατότητα στους κατασκευαστές συσκευών να αναπτύξουν προγράμματα οδήγησης υλικού (hardware drivers) για ένα δεδομένο πυρήνα. Παρέχει ένα επίπεδο διαχωρισμού μεταξύ του hardware της συσκευής και περιέχει όλους τους απαραίτητους οδηγούς υλικού (drivers), όπως της κάμερας, του πληκτρολογίου, της οθόνης κλπ. Ο πυρήνας διαχειρίζεται όλα τα θέματα στα οποία το Linux είναι πραγματικά καλό σε αυτά, όπως η δικτύωση και η τεράστια ποικιλία οδηγών συσκευών (drivers) που διευκολύνουν τη διασύνδεση με το περιφερειακό υλικό [14].

2.1.2. Τα Android components και οι “Activities”

Τα συστατικά μέρη (components) μιας Android εφαρμογής είναι τα βασικά δομικά της στοιχεία. Κάθε στοιχείο είναι ένα σημείο εισόδου, μέσω του οποίου το σύστημα ή κάποιος χρήστης μπορεί να μπει στην εφαρμογή. Κάποια στοιχεία εξαρτώνται από άλλα.

Υπάρχουν τέσσερις βασικοί τύποι τέτοιων στοιχείων:

- Activities: καθορίζουν το UI (User Interface) και χειρίζονται την αλληλεπίδραση του χρήστη με την οθόνη του κινητού τηλεφώνου.
- Services: υπηρεσίες που χειρίζονται διεργασίες οι οποίες συμβαίνουν στο πίσω μέρος (background) της εφαρμογής.
- Broadcast receivers: Διαχειρίζονται την επικοινωνία μεταξύ του Android OS και των εφαρμογών.
- Content providers: Ασχολούνται με θέματα διαχείρισης δεδομένων και βάσεων δεδομένων.

Το στοιχείο στο οποίο θα εστιάσουμε εμείς είναι οι "***Activities***", γιατί είναι και το μόνο βασικό δομικό στοιχείο που έχουμε χρησιμοποιήσει στην εφαρμογή μας. Μία δραστηριότητα (Activity) είναι το σημείο εισόδου αλληλεπίδρασης της εφαρμογής με το χρήστη. Στην πραγματικότητα, το σημείο αυτό είναι μία οθόνη που αντιστοιχεί σε ένα UI (User Interface) [15]. Για παράδειγμα, μια εφαρμογή ηλεκτρονικού ταχυδρομείου μπορεί να έχει μια δραστηριότητα που εμφανίζει τη λίστα με τα νέα μηνύματα ηλεκτρονικού ταχυδρομείου, μια άλλη δραστηριότητα για τη σύνταξη ενός μηνύματος ηλεκτρονικού ταχυδρομείου και μια άλλη δραστηριότητα για την ανάγνωση μηνυμάτων ηλεκτρονικού ταχυδρομείου. Παρόλο που οι δραστηριότητες συνεργάζονται για να σχηματίσουν μια συνολική εμπειρία χρήσης στην εφαρμογή, κάθε μία είναι ανεξάρτητη από τις άλλες. Ως εκ τούτου, μια διαφορετική εφαρμογή μπορεί να ξεκινήσει οποιαδήποτε από αυτές τις τρεις δραστηριότητες, αν η εφαρμογή ηλεκτρονικού ταχυδρομείου το επιτρέπει. Για παράδειγμα, μια εφαρμογή κάμερας μπορεί να ξεκινήσει τη δραστηριότητα που συντάσσει νέα μηνύματα ηλεκτρονικού ταχυδρομείου, με στόχο να επιτρέψει στο χρήστη να μοιραστεί μια φωτογραφία.

Μια δραστηριότητα διευκολύνει τις ακόλουθες βασικές αλληλεπιδράσεις μεταξύ συστήματος και εφαρμογής:

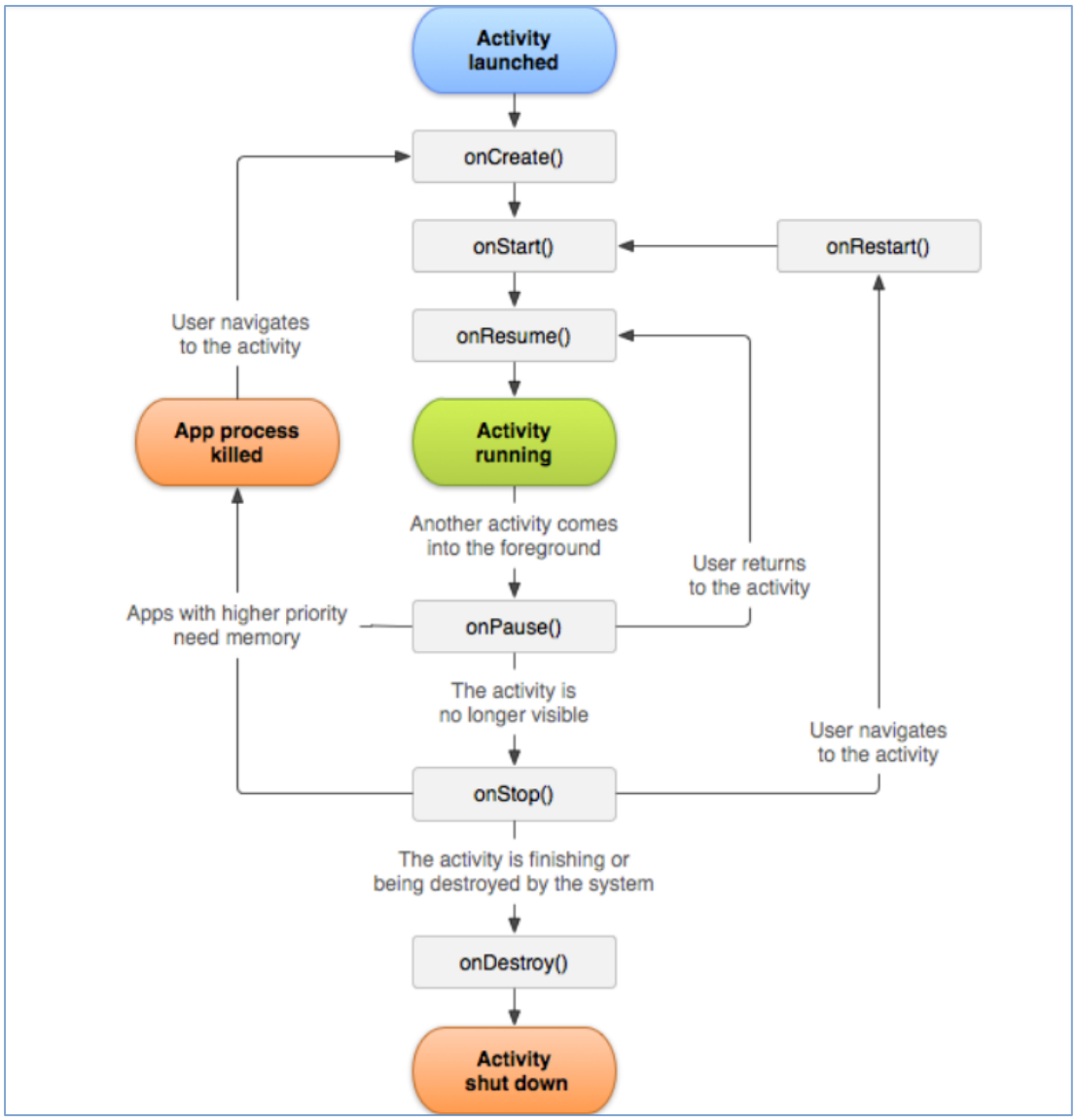
- Παρακολουθεί τί ενδιαφέρει το χρήστη εκείνη τη στιγμή (τί υπάρχει στην οθόνη), προκειμένου να διασφαλίσει ότι το σύστημα εκτελεί τη διεργασία που τη φιλοξενεί.
- Γνωρίζοντας ότι προηγουμένως χρησιμοποιημένες διαδικασίες περιέχουν πράγματα στα οποία ο χρήστης μπορεί να επιστρέψει, δίνει προτεραιότητα στη διατήρηση αυτών των διαδικασιών.

- Βοηθάει την εφαρμογή να σταματήσει μια διαδικασία, ώστε ο χρήστης να μπορεί να επιστρέψει σε δραστηριότητες, με την προηγούμενη κατάσταση να έχει αποκατασταθεί.
- Παρέχει έναν τρόπο ώστε οι εφαρμογές να υλοποιούν ροές μεταξύ χρηστών και το σύστημα να τις συντονίζει.

Πολύ σημαντικό κομμάτι μιας δραστηριότητας είναι ο κύκλος ζωής της (activity lifecycle). Μία δραστηριότητα μπορεί να βρεθεί σε μία κατάσταση, από ένα σύνολο διαφορετικών καταστάσεων, ανάλογα με το τι συμβαίνει στην αλληλεπίδραση χρήστη-εφαρμογής. Κάθε φορά που η κατάσταση μιας δραστηριότητας αλλάζει, μία από τις παρακάτω μεθόδους του κύκλου ζωής της καλείται:

- **onCreate()**: καλείται όταν η Activity αρχικοποιείται για πρώτη φορά. Αυτή η μέθοδος πρέπει να εφαρμοστεί για να γίνει οποιαδήποτε αρχικοποίηση συγκεκριμένη για τη δραστηριότητα.
- **onStart()**: καλείται την πρώτη φορά που η δραστηριότητα πρόκειται να γίνει ορατή στο χρήστη, καθώς ετοιμάζεται να έρθει στο προσκήνιο για να γίνει διαδραστική με το χρήστη.
- **onResume()**: η δραστηριότητα μεταβαίνει σε αυτήν την κατάσταση, όταν αρχίζει να αλληλεπιδρά με το χρήστη.
- **onStop()**: η μέθοδος αυτή καλείται όταν η δραστηριότητα δεν είναι πλέον ορατή στην εφαρμογή.
- **onPause()**: η μέθοδος αυτή χρησιμοποιείται για την παύση εργασιών που δεν πρέπει να συμβαίνουν όταν η δραστηριότητα βρίσκεται σε κατάσταση παύσης.
- **onDestroy()**: καλείται πριν η δραστηριότητα καταστραφεί, είτε επειδή ένας χρήστης την τερματίζει, είτε επειδή το σύστημα θέλει να εξοικονομήσει χώρο.
- **onRestart()**: καλείται όταν ενεργοποιηθεί μια δραστηριότητα μετά τη διακοπή της (δηλαδή μετά την onStop()).

Σχηματικά το Activity Lifecycle παρατίθεται από κάτω:



Εικόνα 9 Ο κύκλος ζωής μιας "Activity".

2.2 Java

Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού γενικού σκοπού, η οποία σχεδιάστηκε από την εταιρεία Sun Microsystems, ως μέρος ενός ερευνητικού έργου ανάπτυξης λογισμικού, για ηλεκτρονικές συσκευές καταναλωτικού επιπέδου. Ο τρόπος αυτός ανάπτυξής της, τη μετέτρεψε σε μία ιδανική γλώσσα για τη διανομή εκτελέσιμων προγραμμάτων μέσω του παγκόσμιου ιστού καθώς και για την ανάπτυξη προγραμμάτων που θα μπορούν εύκολα να μεταφέρονται σε διαφορετικά λειτουργικά συστήματα [1].

Οι δημιουργοί της Java ήθελαν να σχεδιάσουν μία γλώσσα απλή, αντικειμενοστραφή, διαμοιραζόμενη, εύρωστη, ασφαλή, με ουδέτερη αρχιτεκτονική, εύκολη στη μεταφορά, υψηλής απόδοσης, δυναμική και πολυνηματική [18]. Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος της πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix, Macintosh (και σε άλλες κονσόλες, κυρίως ηλεκτρονικών παιχνιδιών) χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε λειτουργικό σύστημα. Για να επιτευχθεί αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα που είναι γραμμένα σε Java να είναι «κατανοητά» από κάθε υπολογιστή, ανεξάρτητα από το είδος του επεξεργαστή και του λειτουργικού συστήματος. Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Η λύση δόθηκε με την ανάπτυξη της εικονικής μηχανής.

2.2.1. Η εικονική μηχανή JVM

Αφού γραφτεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class σε κώδικα από byte (ή bytecode). Ο κώδικας από byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, η εικονική μηχανή JVM (Java Virtual Machine) που πρέπει να είναι εγκατεστημένη σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια, τα μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να μπορέσει να εκτελεστεί. Πιο σύγχρονες εφαρμογές της εικονικής μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα. Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Πρέπει να σημειωθεί ότι η JVM (εικονική μηχανή) είναι λογισμικό που εξαρτάται από το περιβάλλον στο οποίο τρέχει, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοσή του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές, κ.λπ.

Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης), γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Επιπλέον, ο χρήστης δεν μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον τρέξει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα, όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως [16].

2.2.2. Ο συλλέκτης απορριμμάτων (Garbage Collector)

Ακόμα μία ιδέα που βρίσκεται πίσω από τη Java είναι η ύπαρξη του συλλέκτη απορριμμάτων (Garbage Collector). Συλλογή απορριμμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Υπεύθυνη για αυτό είναι και πάλι η εικονική μηχανή η οποία μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο σωρό) κοντεύει να γεμίσει, ενεργοποιεί το συλλέκτη απορριμμάτων. Έτσι, ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερωθεί ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα σημαντικό γιατί τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης είναι πολύ συχνά στον προγραμματισμό [16].

2.2.3. Οι επιδόσεις της Java

Στην ανάπτυξη λογισμικού, η γλώσσα προγραμματισμού Java θεωρήθηκε ιστορικά πιο αργή από τις ταχύτερες γλώσσες τρίτης γενιάς, όπως η C και η C ++. Ο κύριος λόγος είναι ο διαφορετικός σχεδιασμός της γλώσσας. Στη Java, μετά τη μεταγλώττιση, τα προγράμματα τρέχουν σε μια εικονική μηχανή Java (JVM) και όχι απευθείας στον επεξεργαστή του υπολογιστή ως εγγενής κώδικας, όπως τα προγράμματα C και C ++. Η απόδοση ήταν θέμα ανησυχίας, γιατί μεγάλο μέρος του

λογισμικού για επιχειρήσεις είχε αρχίσει να γράφεται σε Java μετά την ταχεία ανάκαμψή της, στα τέλη της δεκαετίας του 1990 και στις αρχές της δεκαετίας του 2000. Από τα τέλη της δεκαετίας του 1990, η ταχύτητα εκτέλεσης των προγραμμάτων Java βελτιώθηκε σημαντικά με την εισαγωγή της μεταγλώττισης Just-in-Time (JIT) (η οποία μετατρέπει τον κώδικα από byte απευθείας σε γλώσσα μηχανής), την προσθήκη στοιχείων που ενισχύουν την καλύτερη ανάλυση του κώδικα και τις βελτιστοποιήσεις στον JVM (όπως είναι ο HotSpot JVM). Επιπλέον, η επιτάχυνση της εκτέλεσης του Java bytecode (στο hardware) που παρέχεται από την Jazelle του ARM, συμβάλει σε σημαντικές βελτιώσεις στην απόδοση.

Τέλος, η απόδοση ενός Java bytecode μεταγλωττισμένου προγράμματος εξαρτάται από το πόσο βέλτιστα γίνεται η διαχείριση των εργασιών του από την εικονική μηχανή Java (JVM) υποδοχής, καθώς και από το πόσο καλά εκμεταλλεύεται το JVM τις λειτουργίες του υλικού του υπολογιστή και του λειτουργικού συστήματος (OS). Έτσι, οποιαδήποτε δοκιμή ή σύγκριση απόδοσης Java πρέπει πάντα να αναφέρει την αρχιτεκτονική εκδόσεων, τον προμηθευτή, το OS και το υλικό του χρησιμοποιούμενου JVM [17].

2.3. Το εργαλείο ανάπτυξης εφαρμογών “Android Studio”

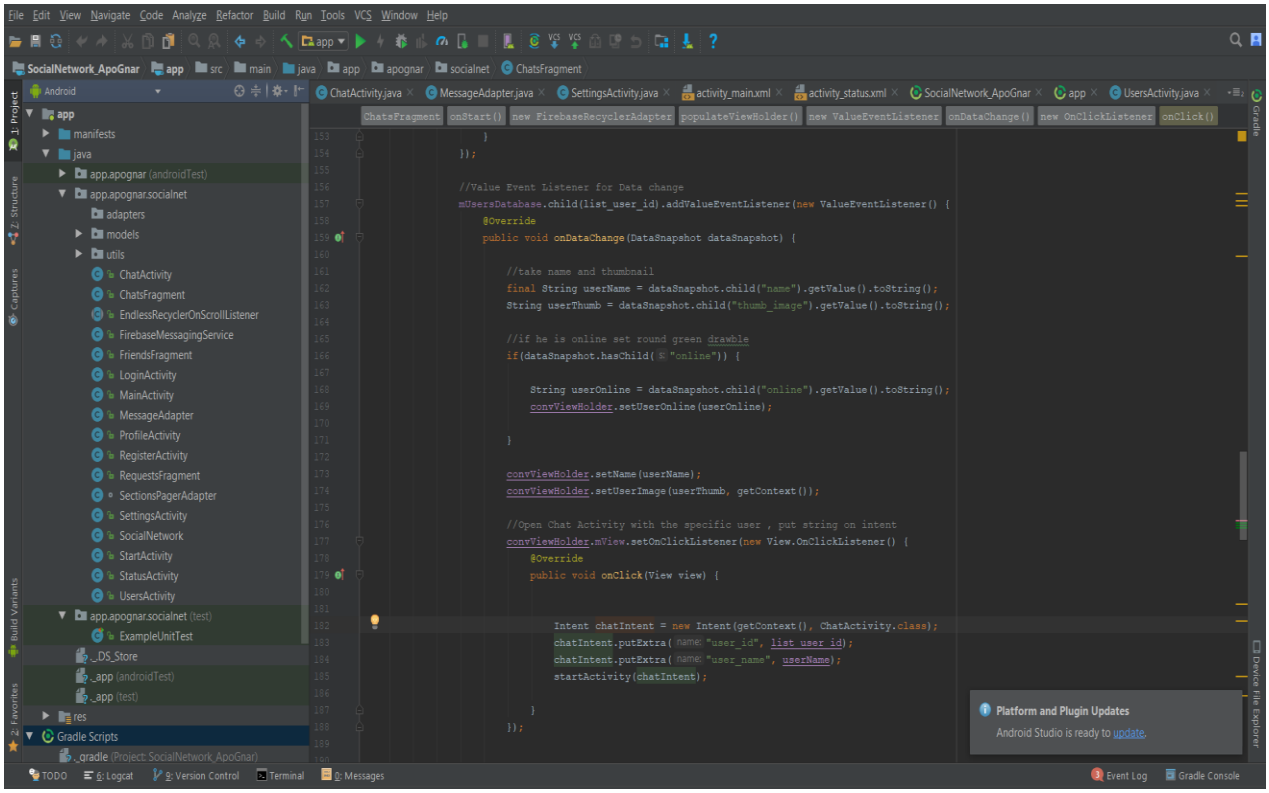
Για την ανάπτυξη εφαρμογών σε συσκευές Android, ο προγραμματιστής πρέπει να διαθέτει έναν υπολογιστή με λειτουργικό σύστημα Windows, macOs ή Linux, στον οποίο θα έχει εγκαταστήσει το Android Studio. Το Android Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE, Integrated Development Environment) για το λειτουργικό σύστημα Android, το οποίο βασίζεται στο λογισμικό IntelliJ IDEA της JetBrains και έχει σχεδιαστεί ειδικά για αυτόν το σκοπό (ανάπτυξη Android εφαρμογών). Αντικατέστησε το Eclipse Android Development Tools, το οποίο αποτέλεσε το αρχικό περιβάλλον ανάπτυξης εφαρμογών Android.

Το Android Studio παρέχει όλα τα εργαλεία που χρειάζονται για τη δημιουργία και διαχείριση των εφαρμογών, τη σχεδίαση και την υλοποίηση της διεπαφής του χρήστη, την αποσφαλμάτωση του κώδικα και πολλά άλλα. Όταν ο προγραμματιστής επιλέγει να «τρέξει» την εφαρμογή του έχει δύο επιλογές. Είτε να χρησιμοποιήσει τον Android Emulator, είτε να χρησιμοποιήσει μία Android συσκευή. Ο Android Emulator είναι ένας προσομοιωτής συσκευών Android που παρέχει στον προγραμματιστή τη δυνατότητα να βεβαιωθεί ότι η εφαρμογή του συμπεριφέρεται

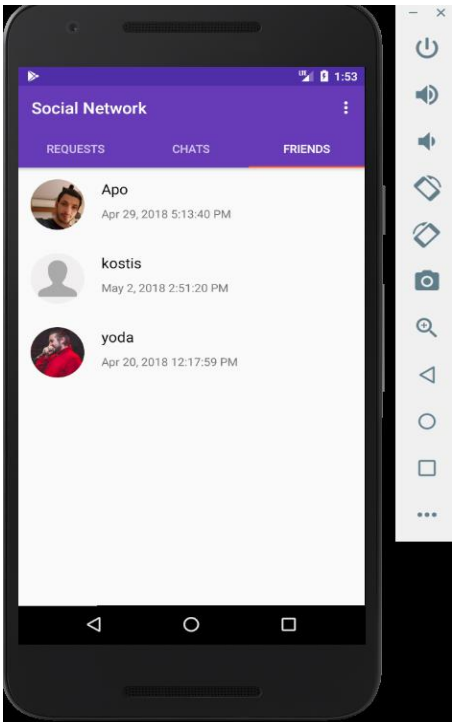
όπως θα ήθελε. Αν η εφαρμογή λειτουργεί ικανοποιητικά, μπορεί να δοκιμαστεί απευθείας σε μία κανονική συσκευή Android που είναι συνδεδεμένη με τον υπολογιστή ή στην οποία έχει εγκατασταθεί το APK αρχείο της εφαρμογής. Η δοκιμή στην κανονική συσκευή (native device) αποτελεί το καλύτερο περιβάλλον εκτέλεσης της εφαρμογής.

Η βασική γλώσσα προγραμματισμού που χρησιμοποιείται στο Android Studio είναι η Java. Επιπλέον, το Android Studio περιλαμβάνει τα εξής εργαλεία [19]:

- visual layout editor: ο προγραμματιστής μπορεί να σχεδιάσει όλες τις διατάξεις της εφαρμογής με το «χέρι», αποφεύγοντας την υλοποίηση με κώδικα σε xml.
- APK Analyzer: μειώνει το μέγεθος της εφαρμογής, παρακολουθώντας τα στοιχεία του APK αρχείου.
- Intelligent code editor: προτείνει διορθώσεις και δίνει πιθανές επιλογές στον προγραμματιστή, καθώς γράφει τον κώδικα.
- Flexible build system (build gradle): επιτρέπει να δημιουργηθούν ποικίλα «χτισίματα» (build) της εφαρμογής, ώστε αυτή να μπορεί να τρέξει σε διαφορετικές συσκευές.
- Realtime profilers: παρέχει στατιστικά για τη δραστηριότητα της μνήμης, της CPU και του δικτύου.



Εικόνα 10 Το περιβάλλον του Android Studio.

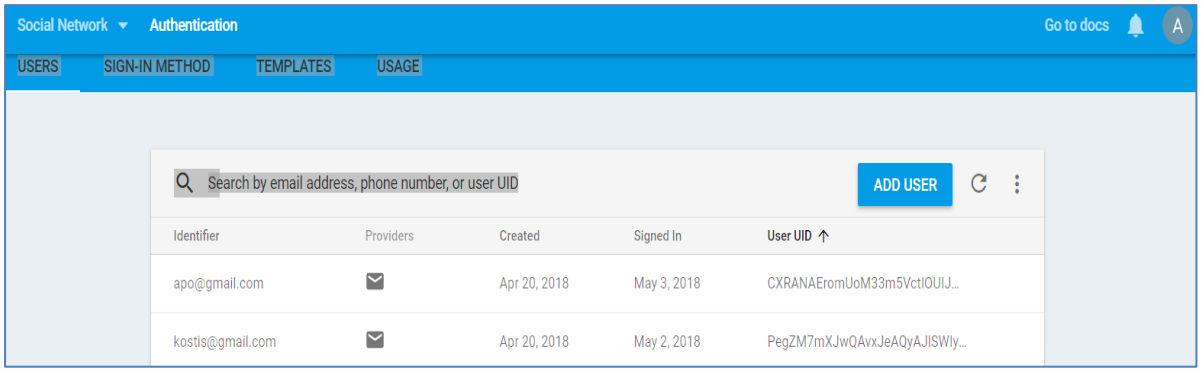


Εικόνα 11 Χρήση του Android Emulator για τον έλεγχο της συμπεριφοράς της εφαρμογής.

2.4. Firebase

Το Firebase είναι μια πλατφόρμα της Google που πραγματοποιεί λειτουργίες server (backend υπηρεσία), δίνοντας τη δυνατότητα στον προγραμματιστή να δημιουργήσει ισχυρές, ασφαλείς και κλιμακούμενες εφαρμογές, χρησιμοποιώντας υποδομή υψηλού επιπέδου. Είναι ένα κατανεμημένο σύστημα που παρέχει ένα δίαυλο επικοινωνίας σε clients (με τον server), μέσω του HTTP πρωτοκόλλου. Παρέχει υπηρεσίες για την ανάπτυξη βελτιωμένων εφαρμογών, αλλά και για την προώθηση μιας εφαρμογής στην αγορά. Ενδεικτικά, κάποιες από αυτές είναι: Authentication, Real-time database, Storage, Cloud Functions, Dynamic Links, Analytics, Hosting [20]. Στην παρούσα εφαρμογή έχουμε επωφεληθεί από τις τρεις πρώτες από αυτές (Authentication, Real-time Database, Storage). Ως εκ τούτου, δεν χρειάστηκε να γραφτεί κώδικας για την υλοποίηση κάποιου server, που θα πραγματοποιούσε την ταυτοποίηση των στοιχείων των χρηστών, την δουλειά της βάσης δεδομένων και την αποθήκευση εικόνων.

Προκειμένου να αξιοποιήσουμε αυτές τις υπηρεσίες που προσφέρει το Firebase απαιτούνται να γίνουν δύο βήματα. Πρώτον, να κατεβάσουμε το google-services.json και να το τοποθετήσουμε στο /app directory (το αρχείο αυτό, περιέχει τους HTTPS πόρους για την εκάστοτε υπηρεσία του server μας). Δεύτερον, να εισάγουμε τις κατάλληλες βιβλιοθήκες (η κάθε μια αντιστοιχεί σε ένα service). Οι επιθυμητές αυτές βιβλιοθήκες εγκαθίστανται στο πρότζεκτ μας στο Android Studio είτε μέσω του Firebase Assistant αυτόματα, είτε χειροκίνητα αν θέλουμε να ελέγξουμε την ανάπτυξη της εφαρμογής εξ ολοκλήρου οι ίδιοι. Από κει και μετά, το Firebase παρέχει συγκεκριμένες εντολές στο Android Studio, οι οποίες δίνουν στον προγραμματιστή πρόσβαση σε κρίσιμα δεδομένα, όπως π.χ. το *user unique identifier (UUID)* ενός χρήστη, που δημιουργείται κατά την εγγραφή (register) του στην εφαρμογή. Αυτό το κλειδί (UUID), καθορίζει πλήρως το χρήστη και μας δίνει πρόσβαση στα δεδομένα του στο database.



Εικόνα 12 Η μορφή του Authentication του χρήστη Apo και το UUID του.

Εικόνα 13 Η μορφή του Real-time Database (JSON format) του χρήστη Apo και η σύνδεση με το UUID του.

2.4.1. JSON

Το JSON (Javascript Object Notation) είναι ένα «ελαφρύ» πρότυπο ανταλλαγής δεδομένων, εύκολο στην ανάγνωση και τη σύνταξή του από τον άνθρωπο και, εξίσου εύκολο, για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Βασίζεται στη γλώσσα προγραμματισμού Javascript, για την αναπαράσταση απλών δεδομένων και συσχετισμένων πινάκων, που ονομάζονται αντικείμενα (object). Ωστόσο, παρά τη σχέση του με τη Javascript, είναι μία ανεξάρτητη γλώσσα, με συμβάσεις που είναι όμοιες με διάφορες γλώσσες προγραμματισμού, κυρίως της «οικογένειας» της C. Το JSON χρησιμοποιείται συχνά για σειριοποίηση και διαβίβαση δεδομένων πάνω από μία σύνδεση δικτύου [1]. Κατά κύριο λόγο, χρησιμοποιείται για τη δόμηση των οντοτήτων που ανταλλάσσονται μεταξύ των απαντήσεων του εξυπηρετητή (server) και των αιτημάτων του πελάτη (client).

Λειτουργεί σαν εναλλακτική λύση στη θέση της XML. Το JSON αποτελείται από δύο δομές:

- Ένα σύνολο από ζεύγη ονόματος-τιμής (name/value pairs). Στις διάφορες γλώσσες αυτό μεταφράζεται ως αντικείμενο, record, struct, dictionary, πίνακας ή λίστα.
- Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού αυτό γίνεται αντιληπτό ως πίνακας ή λίστα.

Αυτές είναι δύο ευρέως αναγνωρίσιμες δομές δεδομένων. Όλες οι σύγχρονες γλώσσες προγραμματισμού μπορούν να τις υποστηρίξουν με τον ένα ή τον άλλον τρόπο. Αυτό φυσικά είναι και το μεγάλο πλεονέκτημα του JSON. Το γεγονός δηλαδή ότι μέσω του JSON τα δεδομένα μπορούν να πάρουν μία μορφή, που είναι αναγνώσιμη από πολλές διαφορετικές γλώσσες προγραμματισμού, λύνει σημαντικά προβλήματα που αφορούν τη μεταφορά δεδομένων [21].

Για παράδειγμα, το Firebase Real-Time Database είναι μία NoSQL βάση δεδομένων, που φιλοξενείται σε ένα «νέφος» (cloud) και επιτρέπει στον προγραμματιστή να αποθηκεύσει και να συγχρονίσει τα δεδομένα μεταξύ των χρηστών σε πραγματικό χρόνο. Η αποθήκευση αυτών των δεδομένων γίνεται σε μορφή αρχείων JSON (όπως φαίνεται και στην Εικόνα 13).

Επιπλέον, για να πετύχουμε επικοινωνία μεταξύ των υπηρεσιών του Firebase, χρειαζόμαστε ένα ακόμη JSON αρχείο, το “google-services.json”. Το αρχείο αυτό περιέχει σε μορφή JSON, όλους τους πόρους που απαιτούνται για τις υπηρεσίες που παρέχει ο server. Όπως βλέπουμε και στην εικόνα παρακάτω, το αρχείο αποτελείται από key/value pairs και εμφωλευμένα JSON objects.

```
{
  "project_info": {
    "project_number": "594620494684",
    "firebase_url": "https://social-network-d3278.firebaseio.com",
    "project_id": "social-network-d3278",
    "storage_bucket": "social-network-d3278.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:594620494684:android:3dc57b3b1d2a3ed7",
        "android_client_info": {
          "package_name": "app.apognar.socialnet"
        }
      },
```

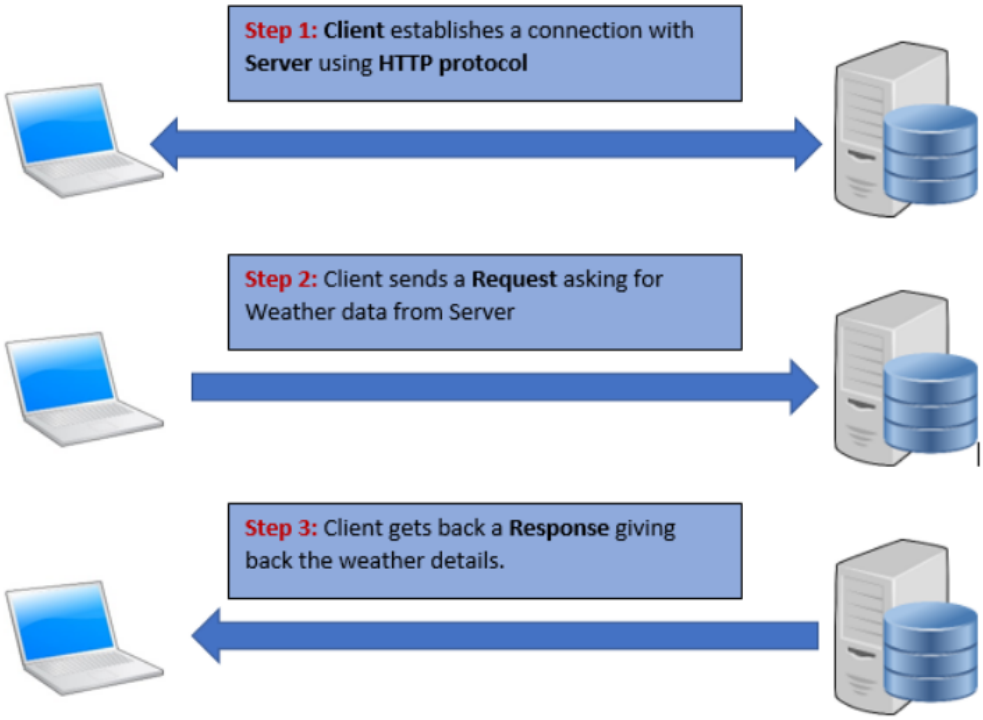
Εικόνα 14 Το “google-services.json” αρχείο.

2.4.2. Η αρχιτεκτονική client-server

Με τον όρο client - server συνήθως εννοούνται δύο (ή περισσότεροι) υπολογιστές που συνδέονται μεταξύ τους μέσω του διαδικτύου. Ωστόσο, πρέπει να επισημανθεί ότι ένα ζεύγος client-server δεν είναι απαραίτητα δύο απομακρυσμένοι υπολογιστές του διαδικτύου, αλλά μπορεί να είναι ακόμη και δύο προγράμματα που εκτελούνται ως διαδικασίες στον ίδιο υπολογιστή. Για να επιτευχθεί η επικοινωνία μεταξύ των δύο συστημάτων χρειαζόμαστε τρία πράγματα:

- Ένα μέσο επικοινωνίας, και συγκεκριμένα ένα πρωτόκολλο ώστε τα δύο συστήματα να αλληλεπιδρούν. Το πρωτόκολλο αυτό είναι το HTTP.
- Ένα πρωτόκολλο για να ζητήσει ο client τις απαραίτητες πληροφορίες από τον server. Αυτό μπορεί να πάρει πολλές μορφές μορφοποιημένων δεδομένων, αλλά οι πιο συχνά χρησιμοποιούμενες είναι το JSON και το XML.
- Ένα πρωτόκολλο το οποίο θα περιλαμβάνει την απάντηση του server και, το οποίο, επίσης μπορεί να είναι μορφής JSON ή XML.

Ανακεφαλαιώνοντας λοιπόν, ένας πελάτης (client) και ένας διακομιστής (server) δημιουργούν μία σύνδεση χρησιμοποιώντας το HTTP πρωτόκολλο. Μόλις δημιουργηθεί η σύνδεση, ο πελάτης στέλνει το αίτημα στον διακομιστή με την μορφή XML ή JSON, την οποία κατανοούν και οι δυο οντότητες. Μετά την κατανόηση του αιτήματος, ο server απαντά με τα κατάλληλα δεδομένα στέλνοντας πίσω μια απάντηση [22].



Εικόνα 15 Η αρχιτεκτονική client-server.

2.4.3. Η πρωτόκολλο TCP/IP

Στην ορολογία των δικτύων, πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν το πώς ανταλλάσσουν μεταξύ τους δεδομένα οι υπολογιστές ενός δικτύου. Το πρωτόκολλο είναι αυτό που καθορίζει το πώς διακινούνται τα δεδομένα, το πώς γίνεται ο έλεγχος και ο χειρισμός των λαθών κ.λπ. Το Internet δεν είναι ένα απλό δίκτυο, αλλά ένα παγκόσμιος ιστός δικτύων (διαδίκτυο). Χρειάζεται επομένως ένα σύνολο από συμβάσεις, οι οποίες να καθορίζουν το πώς ανταλλάσσουν μεταξύ τους δεδομένα, υπολογιστές που μπορεί να είναι διαφορετικού τύπου και να ανήκουν σε διαφορετικά δίκτυα.

Ακριβώς αυτό το σύνολο συμβάσεων προσφέρει το TCP/IP. Όλοι οι υπολογιστές που είναι συνδεδεμένοι στα επιμέρους μικρότερα δίκτυα του Internet τρέχουν το πρωτόκολλο TCP/IP κι έτσι μιλούν μια κοινή γλώσσα που τους επιτρέπει να συνεννοούνται παρά τις διαφορές τους. Το Internet χρησιμοποιεί την τεχνολογία μεταγωγής πακέτων για τη μεταφορά των δεδομένων: τα δεδομένα αποκόπτονται σε τμήματα που ονομάζονται πακέτα και σε κάθε πακέτο μπαίνει μια “επικεφαλίδα” με τις διευθύνσεις του υπολογιστή-αποστολέα και του υπολογιστή-παραλήπτη.

Σημειώνουμε ότι σε κάθε υπολογιστή του Internet αντιστοιχίζεται μία διεύθυνση που ονομάζεται διεύθυνση IP.

Το πρωτόκολλο IP είναι υπεύθυνο για το πέρασμα του πακέτου από υπολογιστή σε υπολογιστή μέσα από το "νέφος" των συνδέσεων του διαδικτύου. Καθώς το IP δρομολογεί το κάθε πακέτο μέσα στο δίκτυο, προσπαθεί να το παραδώσει, αλλά δεν μπορεί να εγγυηθεί ούτε ότι το πακέτο θα φτάσει στον προορισμό του, ούτε ότι τα διάφορα πακέτα που αποτελούν τα αρχικά δεδομένα θα φτάσουν με τη σειρά με την οποία στάλθηκαν, ούτε ότι το περιεχόμενο των πακέτων θα φτάσει αναλλοίωτο.

Το TCP προσφέρει ένα αξιόπιστο πρωτόκολλο πάνω από το IP. Εγγυάται ότι τα πακέτα θα παραδοθούν στον προορισμό τους, ότι θα φτάσουν με τη σειρά με την οποία στάλθηκαν και ότι τα περιεχόμενα των πακέτων θα φτάσουν αναλλοίωτα (δηλ. όπως στάλθηκαν) [23].

ΚΕΦΑΛΑΙΟ 3

Ανάλυση

3.1. Απαιτήσεις εφαρμογής άμεσης ανταλλαγής μηνυμάτων

Οι ρυθμοί της καθημερινότητας των ανθρώπων έχουν αυξηθεί εκθετικά τις τελευταίες δεκαετίες. Οι υποχρεώσεις, είτε εργασιακές, είτε κοινωνικές, συνεχώς πληθαίνουν, με αποτέλεσμα να είναι δύσκολο κάποιος να ανταπεξέλθει σε αυτές. Επιπλέον, ο σύγχρονος άνθρωπος αντιλαμβάνεται με «καθολικό» τρόπο την επιβίωσή του, προσπαθώντας να αναπτύσσεται και να προοδεύει σε όλες τις πτυχές της ζωής του [1]. Για να επιτευχθεί αυτό, πρέπει να ενσωματώσει στην καθημερινότητά του πολλαπλές δραστηριότητες, οι οποίες απαιτούν τη συνεννόηση με τα μέλη των διάφορων κοινωνικών ομάδων, έτσι ώστε αυτές οι δραστηριότητες να οργανωθούν στα μικρά χρονικά πλαίσια που είναι διαθέσιμα. Αυτή η έλλειψη χρόνου καθιστά το τηλέφωνο μια ξεπερασμένη πλέον λύση, αφού μέχρι κάποιος να τελειώσει τη μία δραστηριότητα της ημέρας του ξεκινάει την επόμενη άμεσα. Τη λύση στις ανάγκες της επικοινωνίας έρχονται να δώσουν, οι εφαρμογές άμεσης ανταλλαγής μηνυμάτων.

Οι «έξυπνες» συσκευές έχουν τη δυνατότητα να διαθέτουν στο χρήστη εφαρμογές αυτού του τύπου. Εφόσον τις συσκευές αυτές τις έχουμε συνεχώς μαζί μας, μπορούμε να έχουμε πρόσβαση, με λίγες μόνο κινήσεις, στις εφαρμογές άμεσης ανταλλαγής μηνυμάτων. Γίνεται, επομένως, αντιληπτό, ότι μία εφαρμογή αυτού του είδους μπορεί να φανεί ιδιαίτερα χρήσιμη στο πλαίσιο του σύγχρονου τρόπου ζωής, όπου ο προσωπικός χρόνος μειώνεται διαρκώς και η ανάγκη για επικοινωνία κλιμακώνεται.

Για να είναι χρήσιμο ένα τέτοιο εργαλείο, θα πρέπει να πληροί κάποιες βασικές προδιαγραφές, οι οποίες πρέπει να καλύπτουν κατ' ελάχιστο τις εξής ανάγκες:

- Τα μηνύματα να αποστέλλονται γρήγορα και αποδοτικά (χωρίς δηλαδή η εφαρμογή να «κολλάει»).

- Να παρέχεται ισχυρή ασφάλεια στην επικοινωνία. Κάθε χρήστης δηλαδή να έχει πρόσβαση στα δικά του, και μόνο στα δικά του δεδομένα, και να μη μπορεί να γράψει και να διαβάσει στη βάση δεδομένων κάποιου άλλου.
- Η εφαρμογή να είναι εύχρηστη και ευχάριστη κατά την χρήση.

3.2. Η εφαρμογή άμεσης ανταλλαγής μηνυμάτων “Social Network”

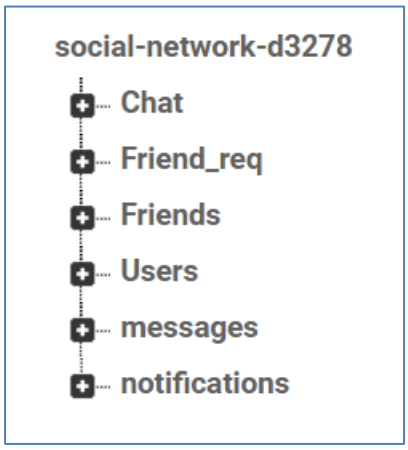
Η εφαρμογή που αναπτύσσουμε στην παρούσα διπλωματική εργασία ονομάστηκε “***Social Network***” και στοχεύει στην ικανοποίηση των βασικών παραμέτρων που αναφέρθηκαν προηγουμένως, στο πλαίσιο των απαιτήσεων των εφαρμογών άμεσης ανταλλαγής μηνυμάτων. Στην εφαρμογή που θα παρουσιάσουμε, ο χρήστης έχει τη δυνατότητα να ανταλλάσει γραπτά μηνύματα και φωτογραφίες μόνο με τη λίστα των φίλων του (όχι με όλους τους χρήστες), η οποία διαμορφώνεται κατόπιν αιτημάτων φιλίας (friend request) που έχουν γίνει αποδεκτά (accept request). Ο χρήστης μπορεί οποιαδήποτε στιγμή να αναδιαμορφώσει τη λίστα των φίλων του, διαγράφοντας κάποιους από αυτούς ή ορίζοντας νέους. Επίσης, έχει τη δυνατότητα να ακυρώσει ένα αίτημα φιλίας, να το ξαναστείλει ή να το απορρίψει. Προκειμένου ο χρήστης να χρησιμοποιήσει την εφαρμογή, θα πρέπει αρχικά να έχει δημιουργήσει ένα λογαριασμό. Στη συνέχεια του δίδεται η δυνατότητα να ανανεώνει τη φωτογραφία προφίλ (κάνοντας crop στην επιθυμητή φωτογραφία πριν την ανεβάσει) και το στάτους του. Τέλος, μπορεί να βλέπει ποιοί φίλοι του είναι συνδεδεμένοι οποιαδήποτε στιγμή, με ένα πράσινο λαμπάκι δίπλα στο όνομα τους, ή, σε διαφορετική περίπτωση, πριν πόση ώρα έχουν συνδεθεί.

Ο στόχος της συγκεκριμένης διπλωματικής είναι να αναπτυχθεί μία εύχρηστη εφαρμογή άμεσης ανταλλαγής μηνυμάτων, η οποία θα βοηθά το χρήστη στην καθημερινότητά του, προσφέροντας του ένα απλό, γρήγορο και ασφαλές μέσο επικοινωνίας με τους φίλους του. Η επικοινωνία μεταξύ των χρηστών δε γίνεται άμεσα, αλλά έμμεσα, χρησιμοποιώντας τον εξυπηρετητή του Firebase. Ο χρήστης, προκειμένου να αξιοποιήσει τις δυνατότητες που προσφέρει ο server, θα πρέπει αρχικά να κάνει μία τυπική εγγραφή, συμπληρώνοντας το όνομα και τη διεύθυνση ηλεκτρονικού ταχυδρομείου.

3.2.1. Η αλληλεπίδραση με τον server του Firebase

Όπως έχουμε αναφέρει, κάθε δραστηριότητα του χρήστη που, είτε αφορά τον ίδιο (π.χ. αλλαγή φωτογραφίας προφίλ) είτε την επικοινωνία με τους άλλους (π.χ. ανταλλαγή μηνύματος), περνάει σαν πληροφορία πάντα από τον server, πριν τα αποτελέσματα γίνουν ορατά στις οθόνες της εφαρμογής.

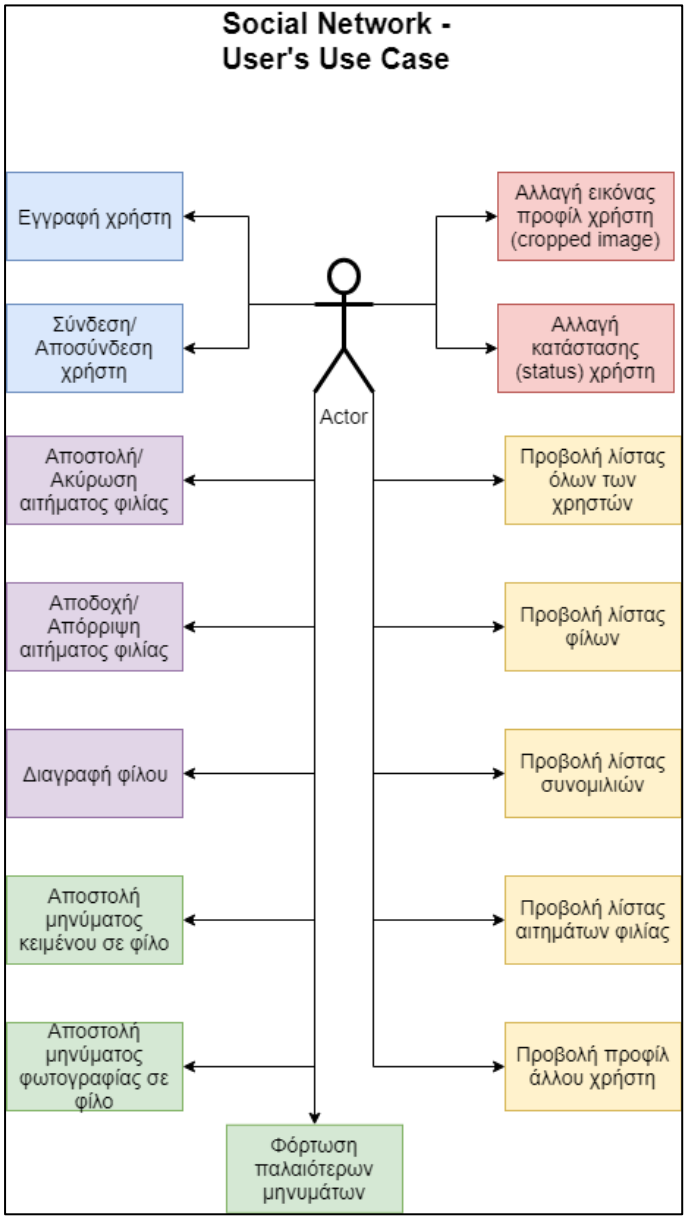
Αρχικά, με τη σύνδεση-εγγραφή ενός χρήστη στην εφαρμογή, δημιουργείται ένας λογαριασμός που αντιστοιχεί σε αυτόν, με σήμα κατατεθέν το UUID key, που παράγεται από κάποια γεννήτρια κρυπτογραφημένων κλειδιών. Με βάση αυτό, διαμορφώνεται το κομμάτι της ταυτοποίησης στοιχείων, μέσω του ζεύγους email/password. Επίσης, με την ίδια διαδικασία, εγγράφονται στη βάση δεδομένων, στο μονοπάτι “Users”, κάποια στοιχεία για το χρήστη, όπως το όνομα, το αν είναι online, το device token κ.λπ., σε μορφή JSON (βλ. Εικόνα 13). Από κει και πέρα, οποιαδήποτε δραστηριότητα πραγματοποιήσει ο χρήστης, είτε αυτή είναι να στείλει ένα μήνυμα τύπου κειμένου, είτε να στείλει ένα μήνυμα τύπου φωτογραφίας, είτε να αλλάξει τη φωτογραφία του προφίλ του, είτε να αλλάξει το στάτους του, είτε να στείλει/ακυρώσει/απορρίψει/δεχτεί κάποιο αίτημα φιλίας, πραγματοποιείται μία εγγραφή/ενημέρωση/διαγραφή στο Real-time Database και στο Storage του Firebase. Παρακάτω φαίνονται όλα τα διαφορετικά μονοπάτια της βάσης δεδομένων που χρειαζόμαστε (τύπου JSON):



Εικόνα 16 Η μορφή της βάσης δεδομένων της εφαρμογής μας

3.3. Σενάρια χρήσης

Για την εξαγωγή των απαιτήσεων της εφαρμογής, αναλύθηκαν τα κυριότερα σενάρια χρήσης της. Στο παρακάτω σχήμα παρουσιάζονται όλες οι δραστηριότητες που μπορεί να πραγματοποιήσει ένας χρήστης:



Εικόνα 17 Το διάγραμμα των σεναρίων χρήσης της εφαρμογής.

3.3.1. Εγγραφή χρήστη

Το σενάριο αυτό περιγράφει τη διαδικασία που ακολουθεί ένας χρήστης για να εγγραφεί στην εφαρμογή.

3.3.1.1. Βασική ροή-Εγγραφή χρήστη

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Εκκίνηση εφαρμογής. 2. Επιλογή συνδέσμου εγγραφής (Sign up).		
	3. Μετάβαση στη φόρμα εγγραφής.	
4. Συμπλήρωση πεδίων φόρμας: i. Όνομα χρήστη. ii. Διεύθυνση ηλεκτρονικού ταχυδρομείου. iii. Κωδικός εισόδου.		
	5. Έλεγχος συμπληρωθέντων στοιχείων. 6. Αποστολή στοιχείων στον Server.	
		7. Έλεγχος μοναδικότητας διεύθυνσης ηλεκτρονικού ταχυδρομείου. 8. Δημιουργία εγγραφής χρήστη στη βάση δεδομένων και στην ταυτοποίηση στοιχείων. 9. Επιστροφή μηνύματος

		επιτυχούς εγγραφής χρήστη.
	10. Μετάβαση στην στην κεντρική οθόνη.	

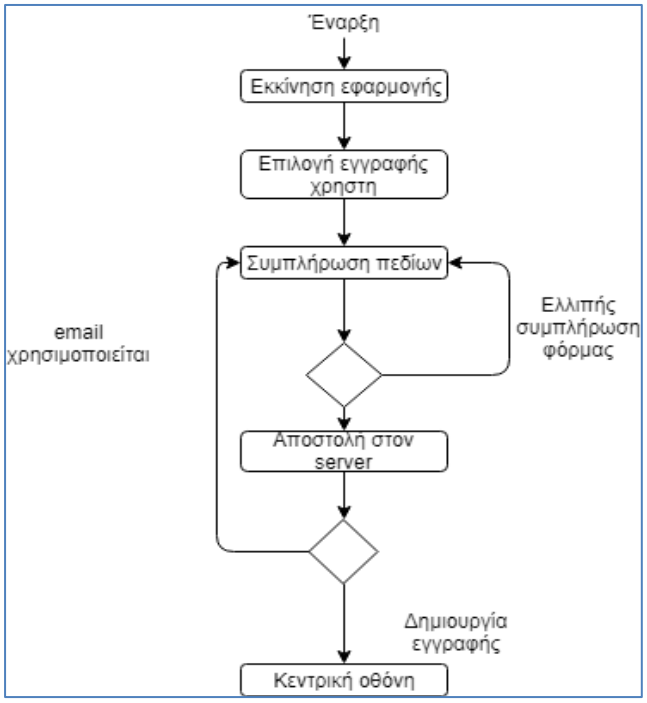
3.3.1.2. Ροή διαχείρισης σφάλματος-Ελλιπής συμπλήρωση φόρμας

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ
	5. Ελλιπής συμπλήρωση φόρμας. 6. Εμφάνιση μηνύματος σφάλματος. 7. Παραμονή στη φόρμα εγγραφής χρήστη.

3.3.1.3. Ροή διαχείρισης σφάλματος-Μη μοναδική διεύθυνση ηλεκτρονικού ταχυδρομείου

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
		7. Διεύθυνση ηλεκτρονικού ταχυδρομείου μη μοναδική. 8. Επιστροφή μη επιτυχούς προσπάθειας εγγραφής.
	9. Εμφάνιση μηνύματος σφάλματος. 10. Παραμονή στη φόρμα εγγραφής χρήστη.	

3.3.1.4. Διάγραμμα δραστηριοτήτων



Εικόνα 18 Εγγραφή χρήστη στον server.

3.3.2 Σύνδεση/Αποσύνδεση χρήστη

Το σενάριο αυτό περιγράφει τη διαδικασία που ακολουθεί ένας χρήστης για να συνδεθεί στην εφαρμογή.

3.3.2.1. Βασική ροή-Σύνδεση χρήστη

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Εκκίνηση εφαρμογής. 2. Επιλογή συνδέσμου σύνδεσης (Login).		
	3. Μετάβαση στη φόρμα σύνδεσης.	
4. Συμπλήρωση των πεδίων: i. Διεύθυνση ηλεκτρονικού		

ταχυδρομείου. ii. Κωδικού πρόσβασης.		
	5. Έλεγχος συμπληρωθέντων στοιχείων. 6. Αποστολή στοιχείων στον server.	
		7. Έλεγχος εγκυρότητας στοιχείων. 8. Επιστροφή μηνύματος επιτυχούς σύνδεσης.
	9. Εμφάνιση κεντρικής οθόνης.	

3.3.2.2. Ροή διαχείρισης σφάλματος-Ελλιπής συμπλήρωση φόρμας

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ
	5. Ελλιπής συμπλήρωση φόρμας. 6. Εμφάνιση μηνύματος σφάλματος. 7. Παραμονή στη φόρμα εγγραφής χρήστη.

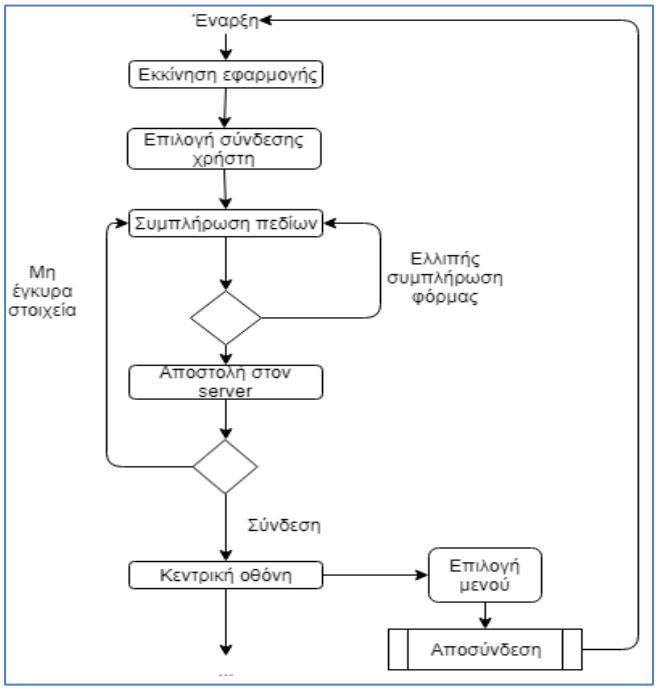
3.3.2.3. Ροή διαχείρισης σφάλματος-Μη έγκυρα στοιχεία χρήστη

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
		7. Μη έγκυρα στοιχεία χρήστη. 8. Μη επιτυχής σύνδεση.
	9. Εμφάνιση μηνύματος σφάλματος. 10. Παραμονή στη φόρμα εγγραφής χρήστη.	

3.3.2.4. Αποσύνδεση από το λογαριασμό

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
10. Επιλογή μενού στην εργαλειοθήκη. 11. Επιλογή συνδέσμου αποσύνδεσης (Log out).		
	12. Επιστροφή στην οθόνη εκκίνησης (Login/Sign up).	

3.3.2.5. Διάγραμμα δραστηριοτήτων



Εικόνα 19 Σύνδεση/αποσύνδεση χρήστη.

3.3.3 Αιτήματα φιλίας και διαγραφή φίλου

Το σενάριο αυτό περιγράφει τη διαδικασία που ακολουθεί ένας χρήστης για να στείλει/ακυρώσει/αποδεχτεί/απορρίψει ένα αίτημα φιλίας ή/και να διαγράψει κάποιον φίλο του.

3.3.3.1. Αποστολή αιτήματος φιλίας

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή μενού εργαλειοθήκης. 3. Επιλογή “All Users”.		
	4. Μετάβαση στη λίστα όλων των χρηστών.	
5. Επιλογή επιθυμητού χρήστη.		
	6. Μετάβαση στο προφίλ του χρήστη.	
7. Επιλογή αποστολής αιτήματος φιλίας.		
		8. Δημιουργία εγγραφής στη βάση δεδομένων.
	9. Εμφάνιση επιλογής ακύρωσης αιτήματος φιλίας.	

3.3.3.2. Ακύρωση αιτήματος φιλίας

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
Ομοίως η αρχή, με την αποστολή αιτήματος φιλίας.		
7. Επιλογή ακύρωσης αιτήματος φιλίας.		
		8. Πραγματοποίηση διαγραφής στη βάση δεδομένων.
	9. Εμφάνιση επιλογής αποστολής αιτήματος.	

3.3.3.3. Βασική ροή- Αποδοχή/απόρριψη αιτήματος φιλίας

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή της καρτέλας “Requests”.		
	3. Εμφάνιση λίστας αιτημάτων φιλίας.	
4. Επιλογή επιθυμητού χρήστη από τη λίστα.		
	5. Άνοιγμα παραθύρου με δύο επιλογές: i) Αποδοχή αιτήματος. ii) Απόρριψη αιτήματος.	
6. Επιλογή αποδοχής ή απόρριψης αιτήματος.		
		7. Δημιουργία εγγραφής ή διαγραφής στη βάση δεδομένων.
	8. Εξαφάνιση αιτήματος από τη λίστα αιτημάτων.	

3.3.3.4. Εναλλακτική ροή- Αποδοχή/απόρριψη αιτήματος φιλίας

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή μενού εργαλειοθήκης. 3. Επιλογή “All Users”.		
	4. Μετάβαση στη λίστα όλων των χρηστών.	
5. Επιλογή χρήστη που έχει στείλει αίτημα φιλίας.		
	6. Μετάβαση στο προφίλ του επιλεγμένου χρήστη.	
7. Επιλογή αποδοχής ή απόρριψης αιτήματος.		.
		8. Δημιουργία εγγραφής ή διαγραφής στη βάση δεδομένων.
	9. Εμφάνιση: i) Επιλογής διαγραφής φίλου, αν το αίτημα έγινε αποδεκτό. ii) Επιλογής αποστολής νέου αιτήματος φιλίας, αν το αίτημα απορρίφθηκε.	

3.3.3.5. Βασική ροή-Διαγραφή φίλου

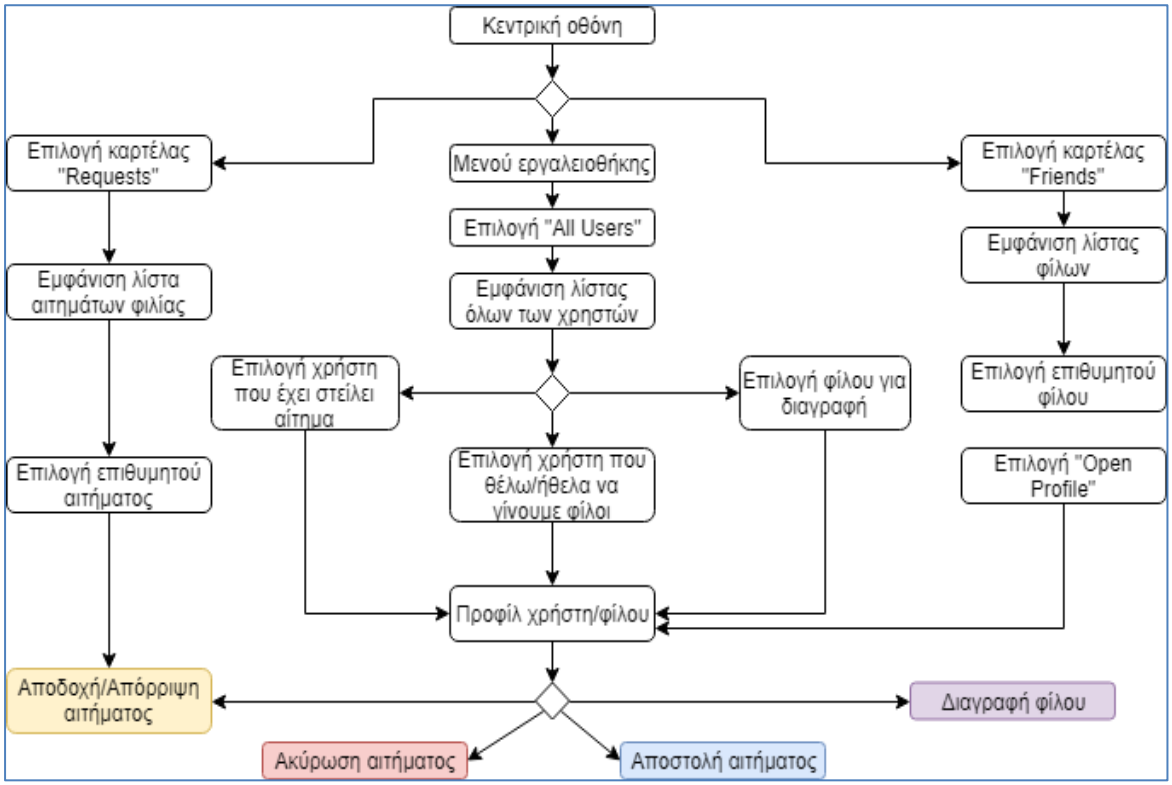
ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή της καρτέλας “Friends”.		
	3. Εμφάνιση της λίστας των φίλων του χρήστη.	
4. Επιλογή επιθυμητού φίλου.		
	5. Άνοιγμα παραθύρου με δύο επιλογές: i) Αποστολή μηνύματος. ii) Προφίλ φίλου.	
6. Επιλογή “Open Profile”.		
	7. Μετάβαση στο προφίλ του φίλου.	
8. Επιλογή διαγραφής φίλου.		
		9. Πραγματοποίηση διαγραφής στη βάση δεδομένων.
	10.Εμφάνιση επιλογής αποστολής αιτήματος φιλίας.	

3.3.3.6. Εναλλακτική ροή-Διαγραφή φίλου

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή μενού εργαλειοθήκης.		

3. Επιλογή “All Users”.		
	4. Μετάβαση στην λίστα όλων των χρηστών.	
5. Επιλογή επιθυμητού φίλου από την λίστα όλων των χρηστών.		
	6. Μετάβαση στο προφίλ του φίλου.	
Ομοίως η συνέχεια, με την βασική ροή.		

3.3.3.7. Διάγραμμα δραστηριοτήτων



Εικόνα 20 Η διαχείριση των αιτημάτων φιλίας και η διαγραφή φίλου.

3.3.4. Ανταλλαγή μηνυμάτων κειμένου και φωτογραφίας

Το σενάριο αυτό περιγράφει τη διαδικασία που ακολουθεί ένας χρήστης για να στείλει ένα μήνυμα κειμένου/φωτογραφίας ή για να δει τα παλαιότερα μηνύματα που έχει ανταλλάξει με έναν άλλον χρήστη. Στη συγκεκριμένη εφαρμογή δύο χρήστες μπορούν να επικοινωνήσουν μόνο εφόσον πρώτα έχουν γίνει φίλοι.

3.3.4.1. Βασική ροή-Ανταλλαγή μηνύματος κειμένου

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή καρτέλας “Chats”.		
	3. Εμφάνιση λίστας όλων των συνομιλιών του χρήστη.	
4. Επιλογή επιθυμητής συνομιλίας.		
	5. Άνοιγμα συνομιλίας και εμφάνιση πρόσφατων μηνυμάτων.	
6. Επιλογή της μπάρας εγγραφής κειμένου.		
	7. Ξεκλείδωμα του πληκτρολογίου.	
8. Εγγραφή του κειμένου που θα σταλεί. 9. Πάτημα του συμβόλου αποστολής.		
		10. Εγγραφή μηνύματος στην βάση δεδομένων.
	11.Εμφάνιση ανανεωμένης συνομιλίας με την προσθήκη του νέου μηνύματος.	

3.3.4.2. Βασική ροή-Ανταλλαγή μηνύματος φωτογραφίας

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
Ομοίως η αρχή, με την βασική ροή ανταλλαγής κειμένου.		
6. Επιλογή του συμβόλου προσθήκης φωτογραφίας.		
	7. Μετάβαση στο Gallery του κινητού.	
8. Επιλογή επιθυμητής φωτογραφίας.		
		9. Αποθήκευση φωτογραφίας στο Storage. 10.Εγγραφή μηνύματος στη βάση δεδομένων.
	11.Εμφάνιση ανανεωμένης συνομιλίας με την προσθήκη του νέου μηνύματος	

3.3.4.3. Εναλλακτική ροή-Ανταλλαγή μηνύματος κειμένου/φωτογραφίας

Αποτελεί το μόνο τρόπο με τον οποίο ο χρήστης μπορεί να ξεκινήσει συνομιλία με κάποιο φίλο του. Με άλλα λόγια, το πρώτο μήνυμα μεταξύ δυο χρηστών, θα ανταλλαγεί με αυτή και μόνο τη ροή.

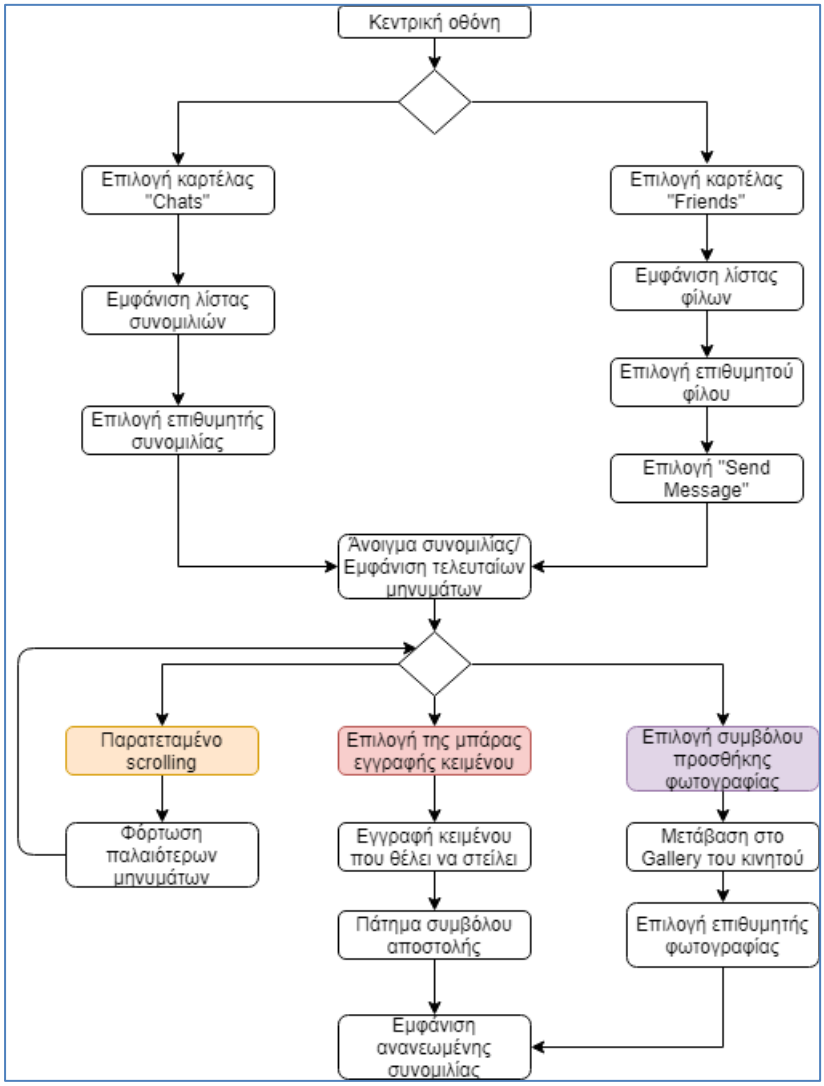
ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή καρτέλας “Friends”.		

	3. Εμφάνιση λίστας φίλων.	
4. Επιλογή επιθυμητού φίλου.		
	5. Άνοιγμα παραθύρου με δύο επιλογές: i) Αποστολή μηνύματος. ii) Προφίλ φίλου.	
6. Επιλογή “Send Message”.		
	7. Άνοιγμα συνομιλίας και εμφάνιση πρόσφατων μηνυμάτων.	
Ομοίως η συνέχεια, με τη βασική ροή, τόσο για μήνυμα κειμένου, όσο και για φωτογραφίας.		

3.3.4.4. Φόρτωση παλαιότερων μηνυμάτων

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
Ομοίως η αρχή, με βασική και εναλλακτική ροή ανταλλαγής μηνυμάτων.		
6. Παρατεταμένο scrolling.		
		7. Ανάγνωση παλαιότερων μηνυμάτων από τη βάση δεδομένων.
	8. Φόρτωση και εμφάνιση παλαιότερων μηνυμάτων.	

3.3.4.5. Διάγραμμα δραστηριοτήτων



Εικόνα 21 Ανταλλαγή μηνυμάτων και φόρτωση παλαιότερων.

3.3.5. Τροποποίηση του προφίλ

Το σενάριο αυτό περιγράφει τη διαδικασία που ακολουθεί ένας χρήστης για να αλλάξει την εικόνα προφίλ και την κατάστασή του.

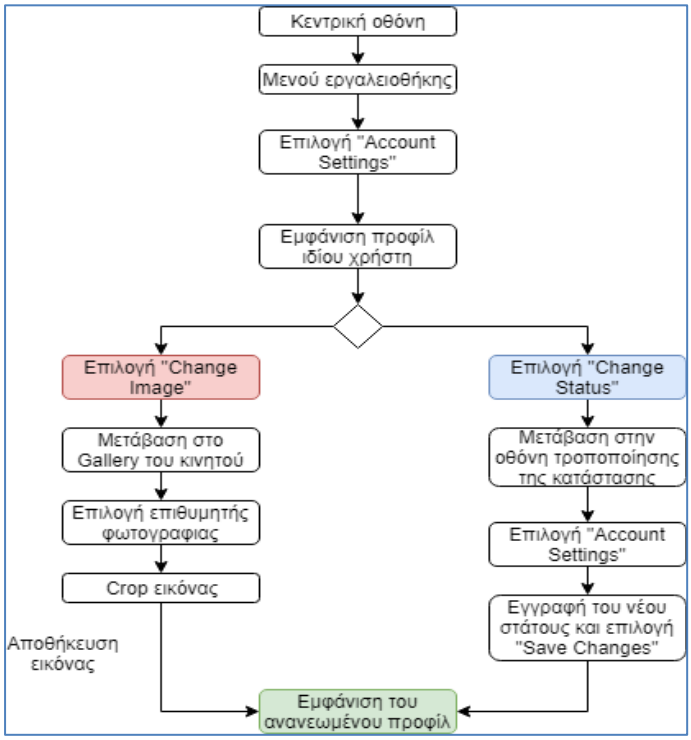
3.3.5.1. Αλλαγή εικόνας προφίλ

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
1. Κεντρική οθόνη. 2. Επιλογή μενού εργαλειοθήκης. 3. Επιλογή “Account Settings”.		
	4. Μετάβαση στο προφίλ του χρήστη.	
5. Επιλογή “Change Image”.		
	6. Μετάβαση στο Gallery του κινητού.	
7. Επιλογή επιθυμητής εικόνας. 8. Crop εικόνας όπως επιθυμεί.		.
		9. Αποθήκευση εικόνας στο Storage. 10.Δημιουργία εγγραφής στη βάση δεδομένων.
	11.Εμφάνιση προφίλ χρήστη με τη νέα εικόνα προφίλ.	

3.3.5.2. Αλλαγή κατάστασης χρήστη

ΧΡΗΣΤΗΣ	ΕΦΑΡΜΟΓΗ	SERVER
Ομοίως η αρχή, με αλλαγή εικόνας προφίλ.		
5. Επιλογή “Change Status”.		
	6. Μετάβαση στην οθόνη τροποποίησης της κατάστασης χρήστη.	
6. Εγγραφή της νέας κατάστασης. 7. Επιλογή “Save Changes”.		
		8. Δημιουργία εγγραφής στη βάση δεδομένων.

3.3.5.3. Διάγραμμα δραστηριοτήτων



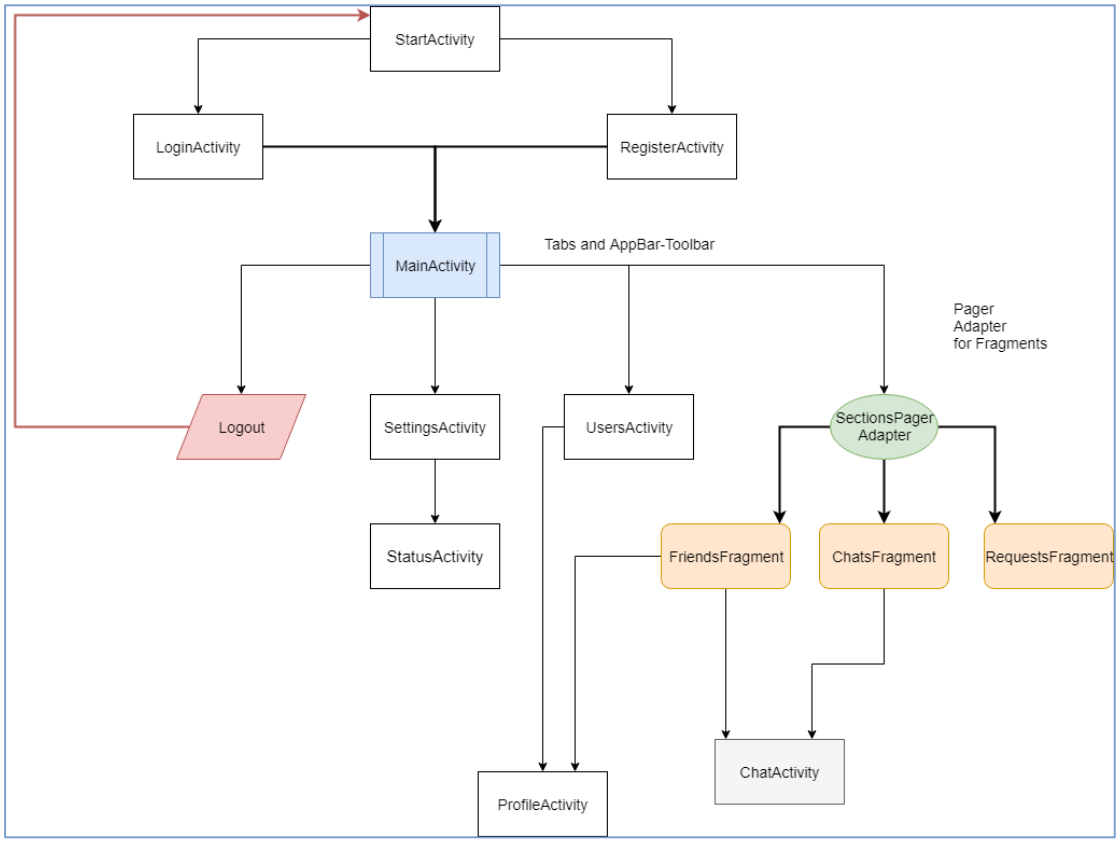
Εικόνα 22 Τροποποίηση προφίλ χρήστη**.**

ΚΕΦΑΛΑΙΟ 4

Σχεδίαση

4.1. Η εφαρμογή “Social Network” στο Android

Προκειμένου να μπορέσουν να υλοποιηθούν τα προαναφερθέντα σενάρια χρήσης του Social Network, θα πρέπει να έχουν δημιουργηθεί οι κατάλληλες οθόνες, ώστε ο χρήστης να αλληλεπιδρά με την εφαρμογή. Οι οθόνες αυτές παρουσιάζονται στο παρακάτω σχήμα (κάθε Activity αντιστοιχεί σε μία οθόνη):



Εικόνα 23 Οι οθόνες (Activities) της εφαρμογής.

4.1.1. Οι οθόνες της εφαρμογής

Οι οθόνες αυτές αναλύονται παρακάτω μία προς μία και είναι:

- ***Εναρκτήρια οθόνη (StartActivity):*** στην οποία ο χρήστης έχει τη δυνατότητα να πραγματοποιήσει, σύνδεση ή εγγραφή, ανάλογα με το αν έχει ξανασυνδεθεί στην εφαρμογή.
- ***Οθόνη σύνδεσης (LoginActivity):*** στην οποία ο χρήστης χρειάζεται να συμπληρώσει τη διεύθυνση ηλεκτρονικού ταχυδρομείο (email) και τον κωδικό του, προκειμένου να αποκτήσει πρόσβαση στην εφαρμογή.
- ***Οθόνη εγγραφής (RegisterActivity)*:** στην οποία ο χρήστης πρέπει να συμπληρώσει το όνομα, το email και τον κωδικό του, προκειμένου να δημιουργήσει λογαριασμό και να συνδεθεί στην εφαρμογή για πρώτη φορά.
- ***Κεντρική οθόνη (MainActivity):*** η οθόνη αυτή αποτελείται από τρεις επιμέρους καρτέλες και ένα μενού εργαλειοθήκης. Οι καρτέλες είναι:
    - ***Αιτήματα φιλίας (RequestsFragment):*** περιλαμβάνει τη λίστα με τα αιτήματα φιλίας που έχει δεχθεί ο χρήστης. Δίνει τη δυνατότητα στο χρήστη κάνοντας κλικ σε ένα από τα αιτήματα, να το αποδεχθεί ή να το απορρίψει.
    - ***Συνομιλίες (ChatsFragment):*** περιλαμβάνει τη λίστα με τις συνομιλίες του χρήστη με τους φίλους του (και ποιοι από αυτούς είναι συνδεδεμένοι). Δίνει τη δυνατότητα στο χρήστη κάνοντας κλικ σε μία από αυτές τις συνομιλίες, να μεταβεί σε αυτή.
    - ***Φίλοι (FriendsFragment):*** περιλαμβάνει τη λίστα με τους φίλους του χρήστη (και ποιοι από αυτούς είναι συνδεδεμένοι). Δίνει τη δυνατότητα στο χρήστη κάνοντας κλικ σε έναν από τους φίλους του, να δει το προφίλ του ή να του στείλει μήνυμα.

  Επίσης, το μενού εργαλειοθήκης, εφόσον επιλεγεί, δίνει τη δυνατότητα στο χρήστη είτε να δει τις ρυθμίσεις του λογαριασμού του, είτε να δει τη λίστα με όλους τους χρήστες της εφαρμογής, είτε να αποσυνδεθεί.
- ***Οθόνη ρυθμίσεων λογαριασμού (SettingsActivity):*** στην οποία ο χρήστης μπορεί είτε να αλλάξει τη φωτογραφία του προφίλ του, επιλέγοντας την εικόνα που επιθυμεί από το Gallery του κινητού (και αφού την κάνει crop να την ανεβάσει), είτε να αλλάξει την κατάστασή/στάτους του.

- ***Οθόνη εμφάνισης όλων των χρηστών (UsersActivity):*** στην οποία βρίσκεται η λίστα με όλους τους χρήστες που έχουν λογαριασμό στην εφαρμογή. Κάνοντας κλικ σε έναν από αυτούς, ο χρήστης μπορεί να μεταβεί στο προφίλ του.
- ***Οθόνη προφίλ χρήστη (ProfileActivity):*** στην οποία ο χρήστης μπορεί να δει το προφίλ κάποιου άλλου χρήστη της εφαρμογής (δηλαδή τη φωτογραφία προφίλ και το στάτους του), καθώς και να αποδεχτεί, απορρίψει, στείλει, ακυρώσει κάποιο αίτημα φιλίας ή να διαγράψει κάποιο φίλο του.
- ***Οθόνη ανταλλαγής μηνυμάτων (ChatActivity):*** στην οποία ο χρήστης μπορεί να στείλει κάποιο νέο μήνυμα (εικόνα/κείμενο) ή να δει τα παλαιότερα. Μπορεί επίσης να παρατηρήσει πριν πόση ώρα έχει συνδεθεί ο χρήστης.
- ***Οθόνη αλλαγής της κατάστασης χρήστη (StatusActivity*)**: στην οποία ο χρήστης μπορεί να αλλάξει το στάτους του και να το αποθηκεύσει.

4.1.2. Οι κλάσεις της εφαρμογής και η αλληλεπίδραση με το UI

Παρακάτω αναλύουμε όλες τις κλάσεις της εφαρμογής, επισημαίνοντας τις σημαντικότερες μεταβλητές και μεθόδους τους.

4.1.2.1. Java κλάσεις που αντιστοιχούν σε οθόνες

Όνομα κλάσης	**StartActivity**	Activity (`AppCompatActivity`)
Μεταβλητές	➢ mRegBtn: UI button ➢ mLoginBtn: UI button	
Μέθοδοι	▪ onCreate: Δημιουργία της StartActivity. ▪ findViewById: Αντιστοίχιση του UI με τα object της κλάσης. ▪ onClick: Κάνει ένα κουμπί “clickable”. ▪ startActivity: Πηγαίνει τον χρήστη σε άλλη Activity. Εδώ είτε στην Login, είτε στην Register.	

Όνομα κλάσης	**RegisterActivity**	Activity (`AppCompatActivity`)
Μεταβλητές	➢ mDisplayName: TextInputLayout ➢ mEmail: TextInputLayout	

	➢ mPassword: TextInputLayout ➢ mCreateBtn: UI Button ➢ mToolbar: UI Toolbar ➢ mRegProgress: ProgressDialog ➢ mDatabase: Firebase DatabaseReference ➢ mAuth: Firebase Authentication ➢ userMap: HashMap
Μέθοδοι	▪ onCreate: Δημιουργία της RegisterActivity. ▪ getInstance: Ενημέρωση του Authentication. ▪ getCurrentUser: Δίνει/επιστρέφει τον «τωρινό» χρήστη. ▪ getReference: Δίνει σημείο αναφοράς την αρχή της βάσης δεδομένων. ▪ getUid: Δίνει το UUID του χρήστη. ▪ setSupportActionBar: «Στήσιμο» του mToolbar. ▪ findViewById, onClick, startActivity ▪ register_user: Εγγραφή του χρήστη στην εφαρμογή μέσω της μεθόδου createUserWithEmailAndPassword. ▪ addOnCompleteListener: «Ακούει» αν ολοκληρώθηκε η εγγραφή του hashMap στο Database, μέσω της μεθόδου onComplete.

Όνομα κλάσης	**LoginActivity**	Activity (AppCompatActivity)
Μεταβλητές	➢ mLoginEmail: TextInputLayout ➢ mLoginPassword: TextInputLayout ➢ mLogin_btn: UI Button ➢ mToolbar: UI Toolbar ➢ mLoginProgress: ProgressDialog ➢ mUserDatabase: Firebase DatabaseReference ➢ mAuth: Firebase Authentication	
Μέθοδοι	▪ onCreate: Δημιουργία της LoginActivity. ▪ getInstance, getCurrentUser, getReference, setSupportActionBar, findViewById, onClick, startActivity,	

	getUid, onComplete ▪ loginUser: Σύνδεση του χρήστη στην εφαρμογή μέσω της μεθόδου signInWithEmailAndPassword. ▪ addOnSuccessListener: «Ακούει» αν έγινε επιτυχώς η εγγραφή στο database, μέσω της μεθόδου onSuccess. ▪ getException: Δίνει το είδος του προβλήματος που παρουσιάστηκε.

Όνομα κλάσης	**MainActivity**	Activity (AppCompatActivity)
Μεταβλητές	➢ mToolbar: UI Toolbar ➢ mUserRef: Firebase DatabaseReference ➢ mAuth: Firebase Authentication ➢ mViewPager: ViewPager ➢ mSectionsPagerAdapter: Custom PagerAdapter ➢ mTabLayout: ViewPagerTabs	
Μέθοδοι	▪ onCreate: Δημιουργία της MainActivity. ▪ getInstance, getCurrentUser, getReference, setSupportActionBar, findViewById, startActivity, getUid ▪ sendToStart: Επιστρέφει στην εναρκτήρια οθόνη. ▪ setAdapter: Αντιστοιχεί τον mSectionsPagerAdapter στο mViewPager. ▪ setupWithViewPager: Αντιστοιχεί το mTabLayout στο mViewPager. ▪ onStart: Το στάδιο της Activity, στο οποίο είναι ορατή από το χρήστη. ▪ onStop: Όταν η Activity παύσει. ▪ onCreateOptionMenu: Δημιουργεί το μενού εργαλειοθήκης. ▪ onOptionsItemSelected: Δημιουργεί τις επιλογές που έχει ο χρήστης όταν ανοίξει το μενού εργαλειοθήκης. ▪ signOut: Αποδέσμευση από το τωρινό Authentication.	

Οι 3 καρτέλες (Tabs) που δημιουργήσαμε με τον ViewPager παίρνουν μορφή από τα 3 παρακάτω Fragments:

Όνομα κλάσης	**RequestsFragment**	Fragment
Μεταβλητές	➢ mRequestList: RecyclerView ➢ mRequestDatabase: Firebase DatabaseReference ➢ mRootRef: Firebase DatabaseReference ➢ mAuth: Firebase Authentication ➢ mCurrent_user_id: UID String ➢ mMainView: FragmentView ➢ builder: AlertDialog Builder ➢ friendsMap: HashMap	
Μέθοδοι	▪ onCreateView: Δημιουργία της όψης του Fragment. ▪ getInstance, getCurrentUser, getReference, onClick, findViewById, getUid, onStart ▪ setLayoutManager: Ορίζει τη μορφή του mRequestList και, συγκεκριμένα εδώ, του δίνει τη μορφή του LinearLayout. ▪ populateViewHolder: Δημιουργεί τον mRequestList, στελεχώνοντάς τον με τα ViewHolders. ▪ getRef(i).getKey: Δίνει το UUID του εκάστοτε χρήστη της λίστας από την οποία παίρνει τα δεδομένα. ▪ addValueEventListener: «Ακούει» στο path του database που τον βάζουμε, για πιθανές αλλαγές μέσω των μεθόδων onDataChange, onCancelled. ▪ updateChildren: Ενημερώνει το database και γράφει τα νέα δεδομένα μέσω της μεθόδου onComplete. ▪ setName, setStatus, setUserImage: Βάζουν το όνομα, το στάτους και την εικόνα του χρήστη αντίστοιχα, στην κατάλληλη θέση του ViewHolder.	

Όνομα κλάσης	**ChatsFragment**	Fragment
Μεταβλητές	➢ mConvList: RecyclerView ➢ mMessageDatabase: Firebase DatabaseReference	

	➢ mConvDatabase: Firebase DatabaseReference ➢ mUsersDatabase: Firebase DatabaseReference ➢ mAuth: Firebase Authentication ➢ mCurrent_user_id: UID String ➢ builder: AlertDialog Builder ➢ mMainView: FragmentView
Μέθοδοι	▪ onCreateView: Δημιουργία της όψης του Fragment. ▪ getInstance, getCurrentUser, getReference, onClick, findViewById, getUid, onStart, setLayoutManager, addValueEventListener, startActivity, setName, setUserImage, getRef(i).getKey ▪ populateViewHolder: Δημιουργεί τον mConvList, στελεχώνοντάς τον με τα ViewHolders. ▪ addChildEventListener: «Ακούει» τα παιδιά του path (του database) που τον έχουμε βάλει, για πιθανές αλλαγές, με τις μεθόδους onChildAdded, onChildChanged, onChildRemoved, onChildMoved, onCancelled. ▪ setMessage: Βάζει το τελευταίο μήνυμα (κείμενο ή φωτογραφία, με bold ή χωρίς) που έχει σταλεί, ώστε αυτό να εμφανίζεται στη λίστα συνομιλιών με τον κάθε χρήστη. ▪ setUserOnline: Εμφανίζει πράσινο λαμπάκι, αν ο χρήστης είναι συνδεδεμένος.

Όνομα κλάσης	**FriendsFragment**	Fragment
Μεταβλητές	➢ mFriendsList: RecyclerView ➢ mFriendsDatabase: Firebase DatabaseReference ➢ mUsersDatabase: Firebase DatabaseReference ➢ mAuth: Firebase Authentication ➢ mCurrent_user_id: UID String ➢ mMainView: FragmentView	
Μέθοδοι	▪ onCreateView: Δημιουργία της όψης του Fragment. ▪ getInstance, getCurrentUser, getReference, onClick, findViewById, getUid, onStart, getRef(i).getKey,	

	setLayoutManager, addValueEventListener, startActivity, setName, setUserImage, setUserOnline ▪ populateViewHolder: Δημιουργεί τον mFriendsList, στελεχώνοντάς τον με τα ViewHolders. ▪ setDate: Βάζει την ημερομηνία που γίνανε φίλοι.

Όνομα κλάσης	**SettingsActivity**	Activity (AppCompatActivity)
Μεταβλητές	➢ mUserDatabase: Firebase DatabaseReference ➢ mCurrentUser: Firebase User ➢ mDisplayImage: Circle ImageView ➢ mName: TextView ➢ mStatus: TextView ➢ mStatusBtn: UI Button ➢ mImageBtn: UI Button ➢ mImageStorage: Firebase StorageReference ➢ mProgressDialog: ProgressDialog ➢ filepath: Firebase StorageReference ➢ baos: ByteArray OutputStream ➢ update_hashMap: HashMap	
Μέθοδοι	▪ onCreate: Δημιουργία της SettingsActivity. ▪ getInstance, getCurrentUser, getReference, onClick, findViewById, getUid, addValueEventListener, startActivity, addOnCompleteListener, updateChildren ▪ onSuccess: Εκτελείται, αν επιτύχει η φόρτωση της φωτογραφίας προφίλ του χρήστη. ▪ onError: Εκτελείται, αν αποτύχει η φόρτωση της φωτογραφίας προφίλ του χρήστη. ▪ startActivityForResult: Ενεργοποιεί την onActivityResult. ▪ onActivityResult: Εκτελείται για να φέρει πίσω ένα αποτέλεσμα (εδώ την εικόνα από το Gallery). ▪ CropImage: Κάνει crop την εικόνα προφίλ, όπως επιθυμεί ο χρήστης.	

	▪ compress: Σπάει, η εικόνα, σε έναν συρμό από bytes, προκειμένου να «ανέβει».

Όνομα κλάσης	**UsersActivity**	Activity (AppCompatActivity)
Μεταβλητές	➢ mUserDatabase: Firebase DatabaseReference ➢ mToolbar: UI Toolbar ➢ mUsersList: RecyclerView ➢ mLayoutManager: Linear LayoutManager	
Μέθοδοι	▪ onCreate: Δημιουργία της UsersActivity. ▪ getInstance, getCurrentUser, getReference, onClick, findViewById, setSupportActionBar, startActivity, onStart, setUserImage, getRef(i).getKey, setLayoutManager ▪ populateViewHolder: Δημιουργεί τον mUsersList, στελεχώνοντάς τον με τα ViewHolders. ▪ setDisplayName, setUserStatus: Βάζουν το όνομα και το στάτους του χρήστη αντίστοιχα, στην κατάλληλη θέση του ViewHolder. ▪ putExtra: Μεταφέρει ένα ζευγάρι μεταβλητής/τιμής στην επόμενη Activity που οδηγεί η startActivity.	

Όνομα κλάσης	**ProfileActivity**	Activity (AppCompatActivity)
Μεταβλητές	➢ mUserDatabase: Firebase DatabaseReference ➢ mFriendDatabase: Firebase DatabaseReference ➢ mFriendReqDatabase: Firebase DatabaseReference ➢ mRootRef: Firebase DatabaseReference ➢ mCurrentUser: Firebase User ➢ mProfileName: TextView ➢ mProfileStatus: TextView ➢ mProfileImage: ImageView ➢ mProfileSenReqBtn: UI Button ➢ mDeclineBtn: UI Button ➢ mProgressDialog: ProgressDialog	

	➢ mCurrent_state: String ➢ requestMap, friendsMap, unfriendMap: HashMap
Μέθοδοι	▪ onCreate: Δημιουργία της ProfileActivity. ▪ getInstance, getCurrentUser, getReference, onClick, findViewById, getUid, addValueEventListener, startActivity, updateChildren ▪ setVisibility: Καθορίζει αν το button θα είναι ορατό. ▪ setEnabled: Καθορίζει αν το button θα είναι clickable. ▪ addListenerForSingleValueEvent: : «Ακούει» στο path (του database) που τον έχουμε βάλει, για πιθανές αλλαγές μέσω των μεθόδων onDataChange, onCancelled, μόνο μια φορά. ▪ hasChild: Ελέγχει αν υπάρχει «παιδί» στο συγκεκριμένο path, με το ζητούμενο όνομα.

Όνομα κλάσης	**ChatActivity**	Activity (AppCompatActivity)
Μεταβλητές	➢ mRootRef: Firebase DatabaseReference ➢ mAuth: Firebase Authentication ➢ mTitleView: TextView ➢ mLastSeenView: TextView ➢ mProfileImage: Circle ImageView ➢ mChatAddBtn: UI Button ➢ mChatSendBtn: UI Button ➢ mChatMessageView: EditText ➢ mChatUser, mCurrentUserId: String ➢ mMessageList: RecyclerView ➢ mRefreshLayout: Swipe RefreshLayout ➢ messageList: Array List ➢ mAdapter: Custom MessageAdapter ➢ mChatToolBar: UI Toolbar ➢ mImageStorage: Firebase StorageReference ➢ chatAddMap, chatUserMap, messageMap, messageUserMap: HashMap	
Μέθοδοι	▪ onCreate: Δημιουργία της ChatActivity.	

	▪ getInstance, getCurrentUser, getReference, onClick, findViewById, getUid, addValueEventListener, setSupportActionBar, setLayoutManager, setAdapter, hasChild, updateChildren, startActivityForResult, onActivityResult, addOnCompleteListener, addChildEventListener ▪ setOnRefreshListener: Παρατεταμένο scrolling που δείχνει ότι φορτώνει μηνύματα, μέσω της μεθόδου onRefresh. ▪ loadMoreMessages: Φορτώνει τα παλαιότερα μηνύματα. ▪ notifyDataSetChanged: Ενημερώνει το αντάπτορα για αλλαγές στην λίστα. ▪ loadMessages: Φορτώνει τα πιο πρόσφατα 10 μηνύματα. ▪ scrollToPosition: Το σημείο της συνομιλίας το οποίο βλέπει ο χρήστης (εδώ το πιο πρόσφατο μήνυμα). ▪ sendMessage: Αποστολή μηνύματος κειμένου και εγγραφή στο database.

Όνομα κλάσης	**StatusActivity**	Activity (AppCompatActivity)
Μεταβλητές	➢ mToolbar: UI Toolbar ➢ mStatus: TextInputLayout ➢ mStatusDatabase: Firebase DatabaseReferencee ➢ mCurrentUser:FirebaseUser ➢ mProgress: ProgressDialog	
Μέθοδοι	▪ onCreate: Δημιουργία της StatusActivity. ▪ getInstance, getCurrentUser, getReference, onClick, findViewById, getUid, setSupportActionBar, addOnCompleteListener	

4.1.2.2. Βοηθητικές Java κλάσεις

Όνομα κλάσης	**SocialNetwork**	Application
Μεταβλητές	➢ mUserDatabase: Firebase DatabaseReferencee ➢ mAuth: Firebase Authentication	
Μέθοδοι	▪ onCreate: Δημιουργία της κλάσης. ▪ getInstance, getCurrentUser, getReference, getUid, addValueEventListener ▪ setPersistenceEnabled: Αποθηκεύει τα δεδομένα τοπικά στη συσκευή, στην περίπτωση που είναι offline. ▪ Picasso block of instructions: Διαχείριση προσωρινής μνήμης (για τις φωτογραφίες).	

Όνομα κλάσης	**SectionsPagerAdapter**	FragmentPagerAdapter
Μεταβλητές		
Μέθοδοι	▪ getItem: Ορίζει το περιεχόμενο της κάθε καρτέλας του ViewPager καθορίζοντας το κατάλληλο fragment. ▪ getCount: Επιστρέφει τον αριθμό των καρτελών του ViewPager. ▪ getPageTitle: Δίνει τίτλο σε κάθε καρτέλα.	

Όνομα κλάσης	**MessageAdapter**	RecyclerView.Adapter
Μεταβλητές	➢ mMessageList: ArrayList ➢ mUserDatabase: Firebase DatabaseReference	
Μέθοδοι	▪ onCreateViewHolder: Δημιουργία του ViewHolder. ▪ getInstance, getReference, addValueEventListener, setVisibility ▪ onBindViewHolder: Γεμίζει τα πεδία του ViewHolder με τα σωστά στοιχεία του κάθε χρήστη. ▪ getItemCount: Επιστρέφει το μέγεθος της συνδεδεμένης λίστας.	

Όνομα κλάσης	**GetTimeAgo**	Application
Μεταβλητές		
Μέθοδοι	▪ getTimeAgo: Μετατρέπει το χρόνο από milliseconds σε λεπτά και ώρες, υπό μορφή string. ▪ System.currentTimeMillis: Δίνει την τωρινή χρονική στιγμή σε milliseconds.	

4.1.2.3. Java model κλάσεις

Όνομα κλάσης	**Conv**	Object Constructor
Μεταβλητές	➢ seen: boolean ➢ timestamp: long	
Μέθοδοι	▪ isSeen: Επιστρέφει “true” αν ο χρήστης έχει ανοίξει τα τελευταία μηνύματα που έχουν σταλεί.	

Όνομα κλάσης	**Friends**	Object Constructor
Μεταβλητές	➢ date: string	
Μέθοδοι	▪ getDate: Επιστρέφει την ημερομηνία κατά την οποία δύο χρήστες γίνανε φίλοι.	

Όνομα κλάσης	**Messages**	Object Constructor
Μεταβλητές	➢ message, type, from: string ➢ seen: boolean ➢ time: long	
Μέθοδοι	▪ getFrom: Επιστρέφει το ποιος έστειλε το μήνυμα. ▪ setFrom: Θέτει το ποιος έστειλε το μήνυμα. ▪ getMessage: Επιστρέφει το περιεχόμενο του μηνύματος. ▪ getType: Επιστρέφει το είδος του μηνύματος. ▪ getTime: Επιστρέφει τη χρονική στιγμή που στάλθηκε. ▪ setTime: Θέτει τη χρονική στιγμή που στάλθηκε.	

Όνομα κλάσης	**Request**	Object Constructor
Μεταβλητές	➢ request_type: string	
Μέθοδοι		

Όνομα κλάσης	**Users**	Object Constructor
Μεταβλητές	➢ name, image, status, thumb_image: string	
Μέθοδοι	▪ getName: Επιστρέφει το όνομα του χρήστη. ▪ setName: Θέτει το όνομα του χρήστη. ▪ getImage: Επιστρέφει την εικόνα του χρήστη. ▪ setImage: Θέτει την εικόνα του χρήστη. ▪ getStatus: Επιστρέφει το στάτους του χρήστη. ▪ setStatus: Θέτει το στάτους του χρήστη. ▪ getThumb_image: Επιστρέφει τη μικρή εικόνα του χρήστη που χρησιμοποιείται στο chat.	

4.2. Ο Firebase server

Στη συγκεκριμένη εφαρμογή, το ρόλο του εξυπηρετητή (server) παίζει το Firebase, καθώς παρέχει όλες τις απαραίτητες υπηρεσίες, χωρίς να χρειάζεται να φιλοξενήσουμε τον κώδικα σε κάποιο server και να διαχειριστούμε τις βάσεις δεδομένων μας.

4.2.1. Οι λειτουργίες του Firebase server

Ο server του Firebase αποτελεί ζωτικό κομμάτι της εφαρμογής, διότι χωρίς αυτόν η αλληλεπίδραση χρήστη-εφαρμογής και χρηστών μεταξύ τους, θα ήταν αδύνατη. Έτσι, παρέχονται κάποιες λειτουργίες ταυτοποίησης στοιχείων, πραγματικού χρόνου βάσης δεδομένων και αποθήκευσης (από τον server), οι οποίες αναλυτικότερα είναι:

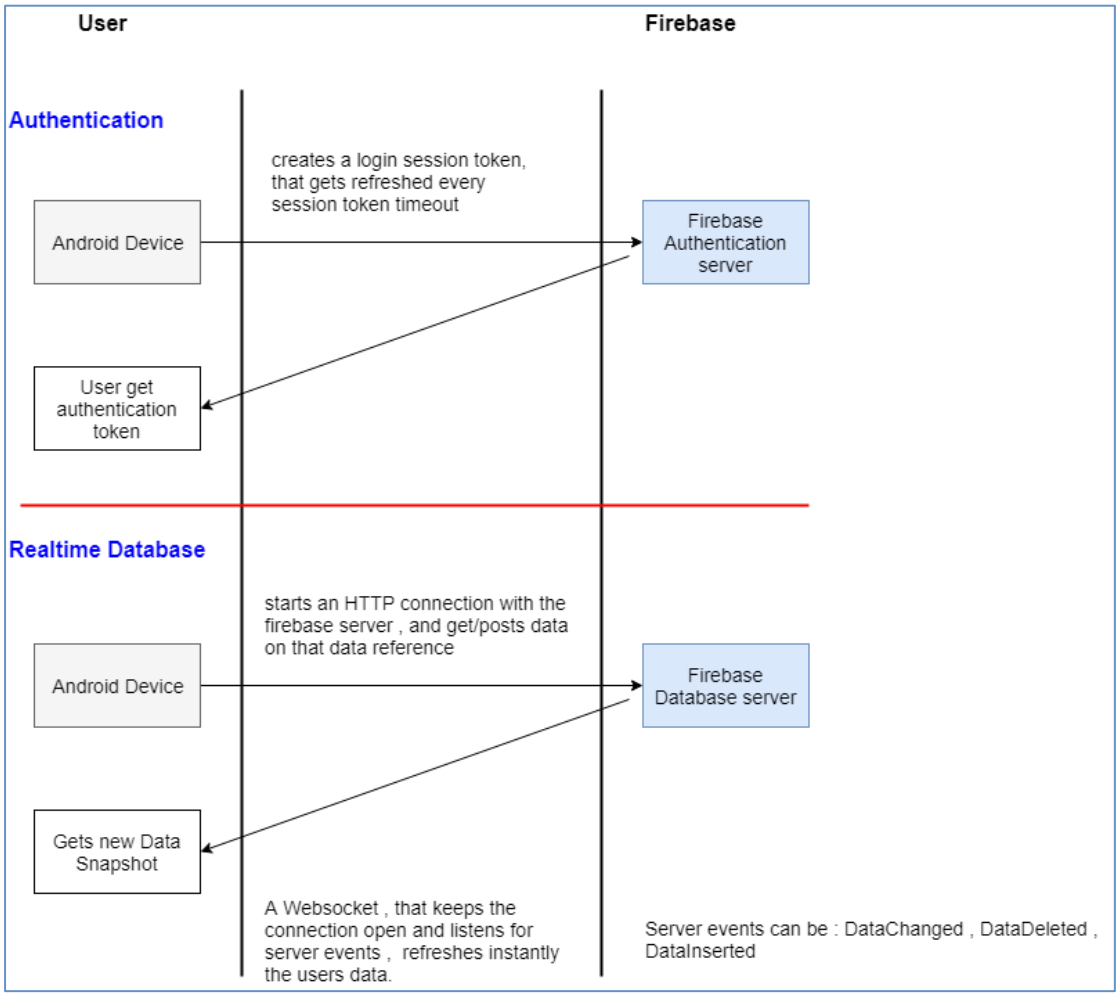
- Όταν ο χρήστης κάνει εγγραφή για πρώτη φορά στην εφαρμογή συμπληρώνοντας τα στοιχεία του, στην ουσία κάνει μια αίτηση στον server να δημιουργήσει το κατάλληλο πιστοποιητικό (credential). Το πιστοποιητικό

αυτό περιέχει το UUID του χρήστη που του επιτρέπει στη συνέχεια να συνδέεται στην εφαρμογή.

- Όταν ο χρήστης συνδεθεί στην εφαρμογή, έχοντας πλέον λογαριασμό, συμπληρώνοντας το ζευγάρι email/password στέλνει τα στοιχεία του στον server, προκειμένου να ελέγξει αν είναι έγκυρα για να του επιτρέψει ή όχι τη σύνδεση.
- Όταν ο χρήστης στείλει/ακυρώσει/αποδεχτεί/απορρίψει κάποιο αίτημα φιλίας ή διαγράψει κάποιο φίλο, ενημερώνεται η βάση δεδομένων, ώστε δύο χρήστες να θεωρηθούν από την εφαρμογή φίλοι ή όχι.
- Όταν ο χρήστης στείλει μήνυμα κειμένου ή φωτογραφίας, αυτό καταγράφεται στην βάση δεδομένων του server.
- Όταν ο χρήστης αλλάξει φωτογραφία προφίλ ή στάτους, αυτό καταγράφεται στη βάση δεδομένων του server.
- Όταν ο χρήστης είναι συνδεδεμένος στην εφαρμογή, αυτό καταγράφεται στη βάση δεδομένων και φαίνεται online στους υπόλοιπους, με ένα πράσινο λαμπάκι. Διαφορετικά, αποθηκεύεται το timestamp και εμφανίζεται η τελευταία φορά που έχει συνδεθεί ο χρήστης.
- Όταν ο χρήστης στείλει φωτογραφία ή ανεβάσει φωτογραφία προφίλ, αυτή αποθηκεύεται στο storage του server, προκειμένου να τη φορτώνει κάθε φορά που συνδέεται.
- Γενικότερα, οτιδήποτε εμφανίζεται στις οθόνες της εφαρμογής, όπως η λίστα φίλων, η λίστα συνομιλιών, η λίστα αιτημάτων φιλίας, η λίστα όλων των χρηστών, το αν κάποιο μήνυμα έχει προβληθεί ή όχι, κ.λπ., είναι απόρροια των πληροφοριών που έχουν εγγραφεί στην βάση δεδομένων του server.

4.2.2. Το σύστημα client-server

Παρακάτω αναπαρίσταται το σχήμα λειτουργίας του συστήματος client-server πιο αναλυτικά:



Εικόνα 24 Το σύστημα χρήστης-server.

Αναλυτικά, κατά το Authentication ο χρήστης (client) στέλνει ένα HTTP request με περιεχόμενο το ζευγάρι email/password προκειμένου το Firebase (server) να ελέγξει αν τα στοιχεία είναι έγκυρα και να δημιουργήσει το κατάλληλο πιστοποιητικό (credential). Ο server από την πλευρά του, αν όλα πάνε καλά, δημιουργεί το προαναφερθέν πιστοποιητικό και απαντάει με ένα HTTP response το οποίο περιλαμβάνει το login session token. Αυτό διατηρείται για συγκεκριμένο χρονικό διάστημα και όταν λήξει ανανεώνεται, έτσι ώστε η σύνδεση client-server να παραμένει ανοιχτή για όσο χρειαστεί.

Από την άλλη, το Real-Time Database βασισμένο στο επιτυχές Authentication που έχει γίνει, μας δίνει την δυνατότητα να στέλνουμε HTTP requests με μεθόδους GET/POST στη βάση δεδομένων. Αυτή η επικοινωνία μένει ανοιχτή προκειμένου να μας ενημερώνει άμεσα για τυχόν συμβάντα που παράγονται από την εγγραφή/διαγραφή δεδομένων στο database.

4.2.3. Μορφή βάσης δεδομένων

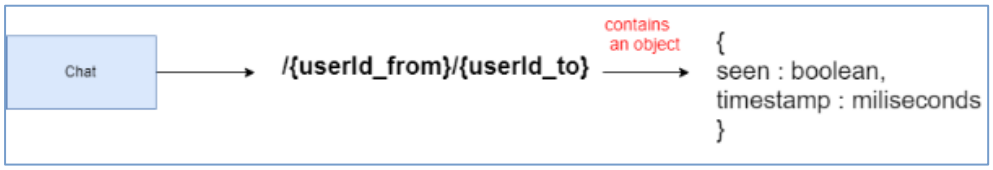
Η βάση δεδομένων του Firebase περιέχει τις εξής κατηγορίες:

1. Chat
2. Friends
3. Users
4. Messages
5. Notifications
6. Friend_req

Παρακάτω παρουσιάζονται αναλυτικά αυτές οι οντότητες, μαζί με τα δεδομένα που η κάθε μία από αυτές περιέχει καθώς και τα path που οδηγούν σε αυτά:

1) ***Chats***

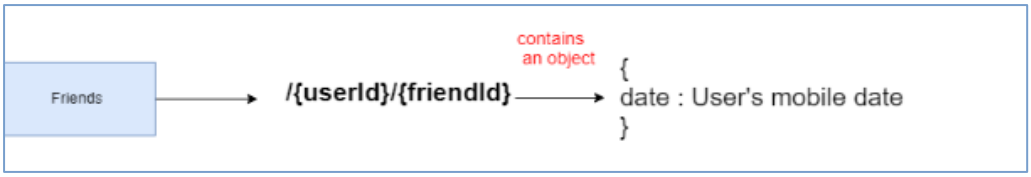
Τα χαρακτηριστικά μιας συνομιλίας είναι αν ο χρήστης έχει ανοίξει το τελευταίο μήνυμα (με την λογική μεταβλητή seen, η οποία είναι true/false) και η χρονική στιγμή που το μήνυμα αυτό στάλθηκε (με την μεταβλητή timestamps σε milliseconds).



Εικόνα 25 Η οντότητα της βάσης δεδομένων “Chats”.

2) ***Friends***

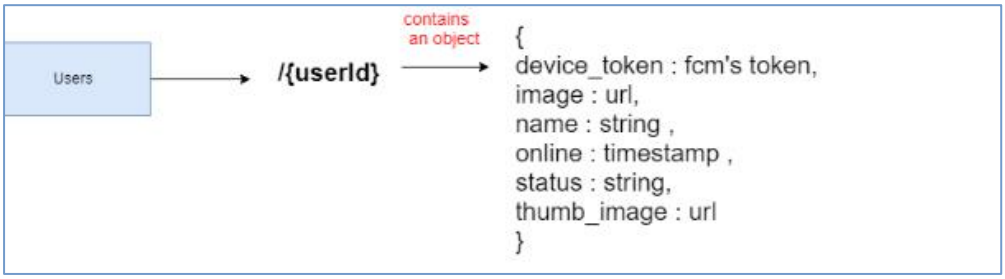
Τα χαρακτηριστικά ενός φίλου είναι μόνο η ημερομηνία που έγινε φίλος με το χρήστη της εφαρμογής (μέσω της μεταβλητής date που παίρνει τιμή την ημερομηνία από τη συσκευή του χρήστη).



Εικόνα 26 Η οντότητα της βάσης δεδομένων “Friends”.

3) ***Users***

Τα χαρακτηριστικά ενός χρήστη (ο οποίος προσδιορίζεται από ένα UUID) είναι το device token (με τη μεταβλητή device_token που παίρνει τιμή ένα αλφαριθμητικό κλειδί, προσδιοριστικό της συσκευής από την οποία συνδέθηκε τελευταία ο χρήστης), την εικόνα προφίλ (με τη μεταβλητή image που περιέχει ένα url), το όνομα του χρήστη (με τη μεταβλητή name, τύπου string), τη χρονική στιγμή που συνδέθηκε τελευταία φορά ο χρήστης (με τη μεταβλητή online σε timestamps (ms)), το στάτους του χρήστη (με τη μεταβλητή status, τύπου string) και τη μικρή εικόνα του χρήστη που χρησιμοποιείται στο chat (με την μεταβλητή thumb_image που περιέχει ένα url).



Εικόνα 27 Η οντότητα της βάσης δεδομένων “Users”.

4) ***Messages***

Τα χαρακτηριστικά ενός μηνύματος (το οποίο προσδιορίζεται από ένα αλφαριθμητικό κλειδί) είναι το ποιος το έστειλε (με τη μεταβλητή from, που περιέχει το UUID του αποστολέα), το περιεχόμενο του μηνύματος (με τη μεταβλητή message, τύπου string), το αν το μήνυμα έχει προβληθεί (με την λογική μεταβλητή seen), τη χρονική στιγμή που στάλθηκε (με την μεταβλητή time σε timestamps) και τον τύπο του μηνύματος, κείμενο/φωτογραφία (με την μεταβλητή type που παίρνει τιμή text/image).

Εικόνα 28 Η οντότητα της βάσης δεδομένων “Messages”.

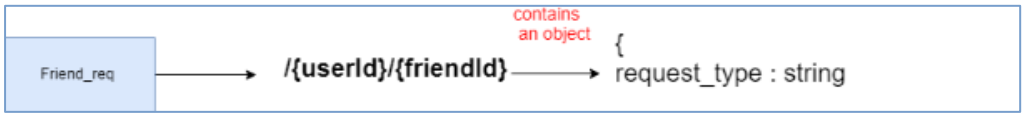
5) ***Notifications***

Τα χαρακτηριστικά μιας ειδοποίησης είναι το ποιος έστειλε το αίτημα φιλίας (με τη μεταβλητή from που παίρνει το UUID του αιτώντος και ο τύπος της ειδοποίησης που είναι πάντα αίτημα φιλίας (με την μεταβλητή type που παίρνει τιμή πάντα request)).

Εικόνα 29 Η οντότητα της βάσης δεδομένων “Notifications”.

6) ***Friend_req***

Τα χαρακτηριστικά ενός αιτήματος φιλίας είναι μόνο το ποιος είναι ο αποστολέας και ποιος ο παραλήπτης (με την μεταβλητή request_type που παίρνει τιμή sender/receiver).



Εικόνα 30 Η οντότητα της βάσης δεδομένων “Friend_req”.

Οι οντότητες αυτές είναι ανεξάρτητες μεταξύ τους και καταγράφονται στη βάση δεδομένων σε μορφή JSON, όπως έχουμε δείξει στις εικόνες 12, 13, 14 της παρούσας διπλωματικής. Προκειμένου να ενοποιηθούν και να εκφραστούν σαν μια ενιαία οντότητα, έχουμε ως ρίζα (root) όλων αυτών των διακλαδώσεων το social-network-d3278 και βασιζόμαστε στα UUID χρήστη. Για παράδειγμα, η οντότητα “Chat” για να υλοποιηθεί, χρησιμοποιεί τα UUID των δυο συνομιλητών, τα οποία καθορίζουν την ταυτότητα της συνομιλίας. Ομοίως λειτουργούν οι οντότητες “Friends”, “Users”, “Friend_req”, που με βάση τα UUID των χρηστών, ορίζουν τα βασικά στοιχεία του κάθε χρήστη, το ποιοί είναι φίλοι με ποιούς και το ποιός έχει στείλει αίτημα φιλίας σε ποιόν. Τέλος, οι οντότητες “messages” και “notifications”, σε πρώτο επίπεδο καθορίζονται και πάλι από τα UUID του κάθε χρήστη, αλλά καθώς προχωράμε σε βάθος στις διακλαδώσεις υπάρχουν νέα unique identifiers, τα οποία αυτή τη φορά, δίνουν ταυτότητα στα μηνύματα και στα αιτήματα φιλίας αντίστοιχα.

ΚΕΦΑΛΑΙΟ 5

Υλοποίηση

5.1. Τα σημαντικότερα σημεία του κώδικα της εφαρμογής

Προκειμένου να αναπτυχθεί η εφαρμογή που αναλύθηκε και σχεδιάστηκε στα προηγούμενα κεφάλαια, ερχόμαστε σε επαφή με τα frameworks του Android SDK και τις ανάγκες του Firebase. Στις επόμενες παραγράφους θα αναλυθούν τα σημαντικότερα σημεία του κώδικα των κλάσεων, τα οποία περιλαμβάνουν μεθόδους και πρακτικές που έπαιξαν καθοριστικό ρόλο στην επίτευξη της λειτουργικότητας της εφαρμογής και, οι οποίες, είναι απαραίτητες για την υλοποίηση οποιασδήποτε παρόμοιας Android εφαρμογής.

---Εγγραφή χρήστη---

Κλάση	*RegisterActivity*

```
- private void register_user(final String display_name, String email, String
password) {

    //Firebase Auth Call to create user
    mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {

            //callback is valid
            if(task.isSuccessful()){

                //Get user uid
                FirebaseUser current_user =
FirebaseAuth.getInstance().getCurrentUser();
                String uid = current_user.getUid();

                //Set up Database Ref
                mDatabase =
FirebaseDatabase.getInstance().getReference().child("Users").child(uid);

                //Calling FCM to take Device Token
                String device_token =
FirebaseInstanceId.getInstance().getToken();

                //Set HashMap
```

```
                HashMap<String, String> userMap = new HashMap<>();
                userMap.put("name", display_name);
                userMap.put("status", "About me");
                userMap.put("image", "default");
                userMap.put("thumb_image", "default");
                userMap.put("device_token", device_token);

                mDatabase.setValue(userMap).addOnCompleteListener(new
OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {

                        if(task.isSuccessful()){

                            //Dismiss Loading
                            mRegProgress.dismiss();

                            //Intent set up and flags definition
                            Intent mainIntent = new
Intent(RegisterActivity.this, MainActivity.class);

mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                            startActivity(mainIntent);
                            finish();
                        }
                    }
                });

            } else {

                mRegProgress.hide();
                Toast.makeText(RegisterActivity.this, "Cannot Sign in.
Please check the form and try again.", Toast.LENGTH_LONG).show();

            }

        }
    });

}
```

Η συνάρτηση register_user δημιουργεί ένα χρήστη με βάση το ζεύγος email/password και τον εγγράφει μαζί με τα στοιχεία του στη βάση δεδομένων του Firebase. Συγκεκριμένα, μέσω της ***createUserWithEmailAndPassword***, δίνοντας της για ορίσματα το ζεύγος email/password, παίρνουμε σαν αποτέλεσμα ένα ***OnComplete*** συμβάν. Το όρισμα της onComplete είναι ένα Task, το οποίο αν ολοκληρωθεί επιτυχώς, γίνεται και η εγγραφή των στοιχείων του χρήστη στη βάση δεδομένων. Διαφορετικά, αν αποτύχει (onComplete), εμφανίζει μηνύματα σφάλματος. Εμφωλευμένα μέσα στην αρχική onComplete, υπάρχει και δεύτερη (onComplete), η οποία περιέχει το block κώδικα που μας οδηγεί στην MainActivity (αν ολοκληρωθούν και οι δύο onComplete επιτυχώς).

---Σύνδεση χρήστη---

Κλάση	*LoginActivity*

```
- private void loginUser(String email, String password) {


    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
         if(task.isSuccessful()){

                mLoginProgress.dismiss();
                //get user uid and FCM token
                String current_user_id = mAuth.getCurrentUser().getUid();
                String deviceToken =
FirebaseInstanceId.getInstance().getToken();


mUserDatabase.child(current_user_id).child("device_token").setValue(deviceTo
ken).addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {

                        Intent mainIntent = new Intent(LoginActivity.this,
MainActivity.class);
                        mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        startActivity(mainIntent);
                        finish();


                    }
                });

            } else {

                mLoginProgress.hide();

                String task_result =
task.getException().getMessage().toString();
                Toast.makeText(LoginActivity.this, "Error : " + task_result,
Toast.LENGTH_LONG).show();

            }
        }
    });
}
```

Η συνάρτηση loginUser ελέγχει αν τα στοιχεία που έδωσε ο χρήστης είναι έγκυρα και, σε περίπτωση που είναι, συνδέει το χρήστη στην εφαρμογή. Συγκεκριμένα, μέσω της ***signInWithEmailAndPassword***, δίνοντας της για ορίσματα το ζεύγος email/password, παίρνουμε σαν αποτέλεσμα ένα ***OnComplete*** συμβάν (ένα Task) που υποδεικνύει αν η μέθοδος ολοκληρώθηκε επιτυχώς. Σε περίπτωση επιτυχίας, αυτό καλεί την ***onSuccess***, η οποία είναι το block κώδικα που μας οδηγεί στην

MainActivity. Διαφορετικά, αν αποτύχει (onComplete), εμφανίζει μηνύματα σφάλματος.

---Σύνδεση του ViewPager της MainActivity με τα Tabs/Fragments---

Κλάση	*SectionsPagerAdapter*

```
-public Fragment getItem(int position) {

    switch(position) {
        case 0:
            RequestsFragment requestsFragment = new RequestsFragment();
            return requestsFragment;

        case 1:
            ChatsFragment chatsFragment = new ChatsFragment();
            return  chatsFragment;

        case 2:
            FriendsFragment friendsFragment = new FriendsFragment();
            return friendsFragment;

        default:
            return  null;
    }
}
```

```
-public CharSequence getPageTitle(int position){

    switch (position) {
        case 0:
            return "REQUESTS";

        case 1:
            return "CHATS";

        case 2:
            return "FRIENDS";

        default:
            return null;
    }
}
```

Ένα Fragment αντιπροσωπεύει μια συμπεριφορά ή ένα τμήμα μιας διεπαφής χρήστη (UI) σε μια δραστηριότητα (Activity). Πολλαπλά Fragments μπορούν να συνδυαστούν σε μία Activity ώστε να δημιουργηθεί ένα UI πολλαπλών παραθύρων, ή, εναλλακτικά, ένα Fragment να ξαναχρησιμοποιηθεί σε πολλαπλές Activities. Ένα Fragment μπορεί να θεωρηθεί ως ένα αρθρωτό κομμάτι μιας Activity, το οποίο έχει το δικό του κύκλο ζωής, λαμβάνει τα δικά του γεγονότα εισόδου και το οποίο μπορεί να προστεθεί ή να αφαιρεθεί από μια δραστηριότητα όταν αυτή εκτελείται

(είναι κάτι σαν μια υπο-Activity, η οποία μπορεί να χρησιμοποιηθεί σε διάφορες Activities) [24].

Ένα ViewPager είναι ένας διαχειριστής διάταξης που επιτρέπει στο χρήστη να μεταπηδήσει αριστερά και δεξιά σε σελίδες δεδομένων. Πολύ συχνά χρησιμοποιείται σε σύζευξη με Fragments, όπως στην προκειμένη περίπτωση, όπου ο συνδυασμός τους μας δίνει τη δυνατότητα να μεταβαίνουμε από Fragment σε Fragment (καρτέλα σε καρτέλα) σαν να διαχειριζόμαστε slides [25].

Η συνάρτηση getItem, με βάση τη θέση (position) που βρίσκεται ο χρήστης, επιστρέφει το κατάλληλο fragment. Για παράδειγμα, αν ο χρήστης έχει πάει στην MainActivity στο πιο αριστερό slide, αυτό θα αντιστοιχηθεί στην θέση 0 και θα επιστραφεί το requestsFragment. Η συνάρτηση getPageTitle δίνει σε κάθε καρτέλα (Tab), που αντιστοιχεί σε ένα fragment, τον κατάλληλο τίτλο.

---Το Fragment της λίστας συνομιλιών---

Κλάση	*ChatsFragment*

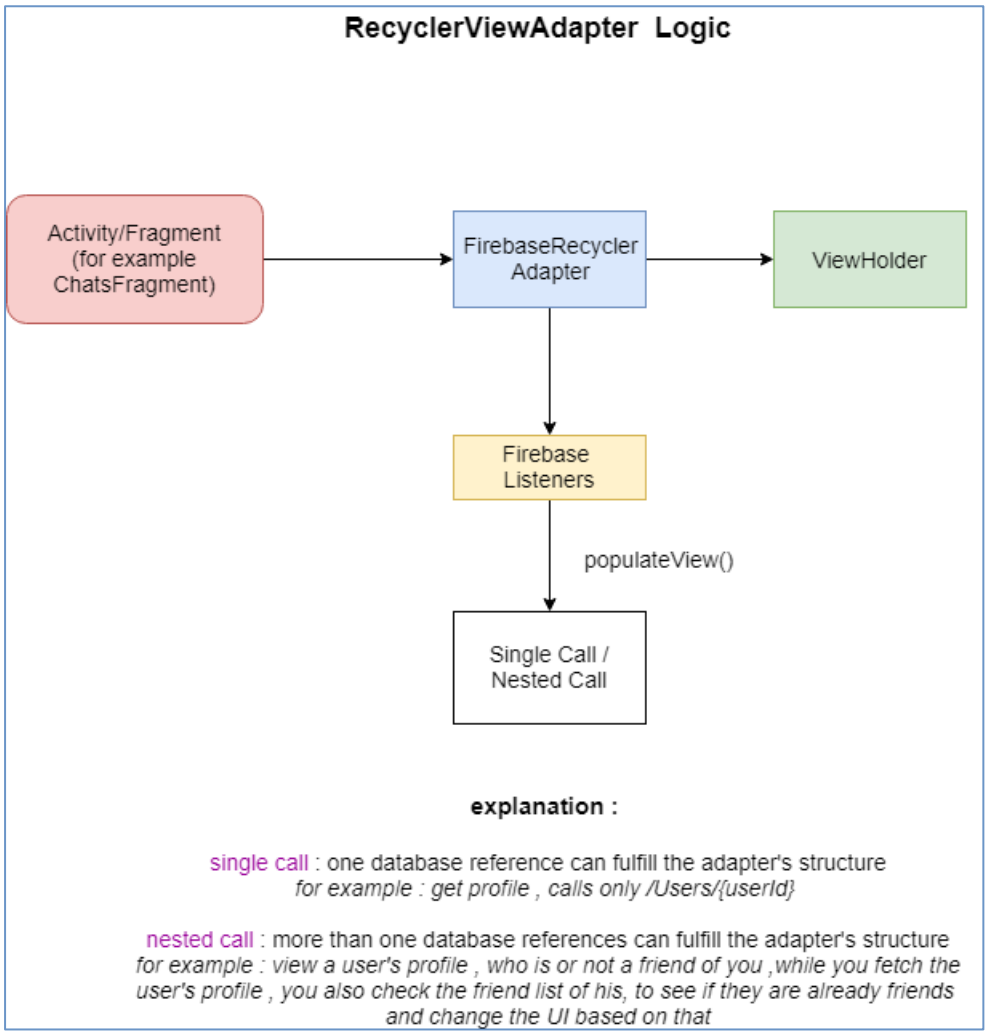
```
-public void onStart() {
    super.onStart();

    //Query based on timestamp
    Query conversationQuery = mConvDatabase.orderByChild("timestamp");

    //Firebase Recycler Adapter set up
    FirebaseRecyclerAdapter<Conv, ConvViewHolder> firebaseConvAdapter = new
FirebaseRecyclerAdapter<Conv, ConvViewHolder>(
            Conv.class,
            app.apognar.socialnet.R.layout.users_single_layout,
            ConvViewHolder.class,
            conversationQuery
    ) {

        @Override
        protected void populateViewHolder(final ConvViewHolder
convViewHolder, final Conv conv, int i) {


            final String list_user_id = getRef(i).getKey();
            Query lastMessageQuery =
mMessageDatabase.child(list_user_id).limitToLast(1);

            //Child event listener for the last message to refresh data
            lastMessageQuery.addChildEventListener(new ChildEventListener()
{
                @Override
                public void onChildAdded(DataSnapshot dataSnapshot, String
s) {
                    String data =
dataSnapshot.child("message").getValue().toString();
                    convViewHolder.setMessage(data, conv.isSeen());

                }

                @Override
                public void onChildChanged(DataSnapshot dataSnapshot, String
```

```
s) {

                }

                @Override
                public void onChildRemoved(DataSnapshot dataSnapshot) {

                }

                @Override
                public void onChildMoved(DataSnapshot dataSnapshot, String
s) {

                }

                @Override
                public void onCancelled(DatabaseError databaseError) {

                }
            });

            //Value Event Listener for Data change
            mUsersDatabase.child(list_user_id).addValueEventListener(new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {

                    //take name and thumbnail
                    final String userName =
dataSnapshot.child("name").getValue().toString();
                    String userThumb =
dataSnapshot.child("thumb_image").getValue().toString();

                    //if he is online set round green drawble
                    if(dataSnapshot.hasChild("online")) {

                        String userOnline =
dataSnapshot.child("online").getValue().toString();
                        convViewHolder.setUserOnline(userOnline);

                    }

                    convViewHolder.setName(userName);
                    convViewHolder.setUserImage(userThumb, getContext());

                    //Open Chat Activity with the specific user , put string
on intent
                    convViewHolder.mView.setOnClickListener(new
View.OnClickListener() {
                        @Override
                        public void onClick(View view) {


                            Intent chatIntent = new Intent(getContext(),
ChatActivity.class);
                            chatIntent.putExtra("user_id", list_user_id);
                            chatIntent.putExtra("user_name", userName);
                            startActivity(chatIntent);

                        }
                    });
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {

                }
            });
```

```
        }
    };

    mConvList.setAdapter(firebaseConvAdapter);}
```

```
-public static class ConvViewHolder extends RecyclerView.ViewHolder {

    View mView;

    public ConvViewHolder(View itemView) {
        super(itemView);

        mView = itemView;
    }

    public void setMessage(String message, boolean isSeen){
        TextView userStatusView = (TextView)
mView.findViewById(app.apognar.socialnet.R.id.user_single_status);

        if(message.contains("http")){
            userStatusView.setText("Sent a photo");

        }else{
            userStatusView.setText(message);
        }

        if(!isSeen){
            userStatusView.setTypeface(userStatusView.getTypeface(),
Typeface.BOLD);
        } else {
            userStatusView.setTypeface(userStatusView.getTypeface(),
Typeface.NORMAL);
        }
      }

    public void setName(String name){

        TextView userNameView = (TextView)
mView.findViewById(app.apognar.socialnet.R.id.user_single_name);
        userNameView.setText(name);

    }

    public void setUserImage(String thumb_image, Context ctx){

        CircleImageView userImageView = (CircleImageView)
mView.findViewById(app.apognar.socialnet.R.id.user_single_image);

Picasso.with(ctx).load(thumb_image).placeholder(app.apognar.socialnet.R.draw
able.default_avatar).into(userImageView);}
       public void setUserOnline(String online_status) {

        ImageView userOnlineView = (ImageView)
mView.findViewById(app.apognar.socialnet.R.id.user_single_online_icon);

        if(online_status.equals("true")){

            userOnlineView.setVisibility(View.VISIBLE);

        } else {

            userOnlineView.setVisibility(View.INVISIBLE);

        }
     }
}
```

Το παρακάτω σχήμα απεικονίζει τη λογική με την οποία συνδέονται μεταξύ τους το Fragment, ο FirebaseRecyclerAdapter, ο ViewHolder, οι Firebase Listeners και η μέθοδος populateViewHolder. Τα στοιχεία αυτά συνδυάζονται προκειμένου να υλοποιήσουμε την πρακτική του RecyclerView. Ο RecyclerView είναι μια πιο προχωρημένη και ευέλικτη έκδοση της ListView και χρησιμοποιείται για να εμφανίσει μια κυλιόμενη (scrolling) λίστα στοιχείων, τα οποία είτε περιέχουν μεγάλα σύνολα δεδομένων, είτε αλλάζουν συχνά. Κάθε στοιχείο της λίστας αντιπροσωπεύει έναν ViewHolder, μία όψη δηλαδή την οποία μπορούμε να διαμορφώσουμε όπως εμείς θέλουμε. Η διαχείριση του ViewHolder και πώς αυτός θα συμπληρωθεί, γίνεται με τον RecyclerViewAdapter (στην προκειμένη περίπτωση, τον FirebaseRecyclerAdapter) [27].



Εικόνα 31 Η λογική του Firebase Recycler Adapter.

Αρχικά δημιουργούμε ένα query, τέτοιο ώστε να περιέχει μία λίστα με τις συνομιλίες του κάθε χρήστη που έχουν καταγραφεί στη βάση δεδομένων, ταξινομημένες με βάση τα timestamps. Με άλλα λόγια, με την εντολή ***"Query conversationQuery = mConvDatabase.orderByChild("timestamp");"*** δημιουργούμε για κάθε χρήστη τη λίστα συνομιλιών του, με τις πιο πρόσφατες από αυτές να εμφανίζονται υψηλότερα [26]. Στη συνέχεια, δημιουργούμε έναν FirebaseRecyclerAdapter, δίνοντας του για ορίσματα το προαναφερθέν query και έναν ViewHolder («κρατάει την όψη» που θέλουμε για κάθε συνομιλία). Ο συγκεκριμένος ViewHolder είναι προσαρμοσμένος στις ανάγκες μας και διαθέτει τις μεθόδους setMessage, setName, setUserImage, setUserOnline τις οποίες έχουμε εξηγήσει στο προηγούμενο κεφάλαιο. Στη συνέχεια, η populateViewHolder αναπαράγει, όσες φορές χρειαστεί, την όψη που «κρατάει» ο ViewHolder, μία για κάθε αντικείμενο τύπου Conv, δημιουργώντας για τον κάθε χρήστη τη λίστα με τις συνομιλίες του. Αναλυτικότερα, αρχικά δημιουργούμε άλλο ένα query, τέτοιο ώστε για κάθε συνομιλία του χρήστη με κάποιο φίλο του, να δίνει μόνο το τελευταίο/πιο πρόσφατο μήνυμα που έχουν ανταλλάξει (***Query lastMessageQuery = mMessageDatabase.child(list_user_id).limitToLast(1);***) και σε αυτό (το query) προσθέτουμε έναν ***addChildEventListener***. Ο addChildEventListener είναι μια υπηρεσία (ένας Listener) που παρέχει το Firebase, προκειμένου να «ακούει» συμβάντα που ενεργοποιούνται σε απόκριση συγκεκριμένων λειτουργιών στα παιδιά ενός κόμβου της βάσης δεδομένων. Παραδείγματα τέτοιων λειτουργιών είναι όταν ένα παιδί προστίθεται (onChildAdded) ή όταν ένα παιδί ανανεώνεται (updateChildren). Σε αυτό το κομμάτι της εφαρμογής ο συγκεκριμένος Listener χρησιμοποιεί στο lastMeassageQuery τη μέθοδο ***onChildAdded*** [26], έτσι ώστε η τελευταία να καλείται μία φορά στην αρχή για κάθε υπάρχον παιδί και, στη συνέχεια, κάθε φορά που ένα νέο παιδί προστίθεται στο συγκεκριμένο path. Αυτό έχει σαν αποτέλεσμα να μας επιστρέφεται συνεχώς το ανανεωμένο τελευταίο μήνυμα της κάθε συνομιλίας του χρήστη, με το οποίο γεμίζει το επιθυμητό πεδίο του ViewHolder. Στη συνέχεια, ορίζουμε άλλον έναν Listener, αυτή τη φορά τύπου ***addValueEventListener***[28] και τον προσθέτουμε στο path Users/UUID_φίλου της βάσης δεδομένων, προκειμένου να «ακούει» οποιαδήποτε αλλαγή συμβαίνει στα δεδομένα αυτού του φίλου και αντιστοίχως να ενημερώνει τα πεδία του ViewHolder. Ο addValueEventListener χρησιμοποιεί τη μέθοδο ***onDataChange***, για να ανακτήσει τα δεδομένα του φίλου μία φορά στην αρχή και, ακολούθως, κάθε φορά που κάποιο

δεδομένο σε αυτό το path αλλάζει, έτσι ώστε τα πεδία του ονόματος, της εικόνας προφίλ και το πράσινο λαμπάκι (για το αν είναι συνδεδεμένος) να παραμένουν ενημερωμένα διαρκώς.

Παρόμοια λογική ακολουθούν και τα υπόλοιπα fragments, καθώς και η UsersActivity. Για το λόγο αυτό, δεν θα γίνει ξεχωριστή αναφορά στον κώδικά τους.

---Η δραστηριότητα ανταλλαγής μηνυμάτων---

Κλάση	*ChatActivity*

```
-protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);

    if(requestCode == GALLERY_PICK && resultCode == RESULT_OK){

        Uri imageUri = data.getData();

        final String current_user_ref = "messages/" + mCurrentUserId + "/" +
mChatUser;
        final String chat_user_ref = "messages/" + mChatUser + "/" +
mCurrentUserId;

        DatabaseReference user_message_push = mRootRef.child("messages")
                .child(mCurrentUserId).child(mChatUser).push();

        // to uid tou message
        final String push_id = user_message_push.getKey();


        //Connect with firebase storage and set the path reference
        StorageReference filepath =
mImageStorage.child("message_images").child( push_id + ".jpg");

        //put the file to the reference and then check if its successful
        filepath.putFile(imageUri).addOnCompleteListener(new
OnCompleteListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<UploadTask.TaskSnapshot>
task) {

                if(task.isSuccessful()){

                    //retrive url
                    String download_url =
task.getResult().getDownloadUrl().toString();

                    //construct the message
                    Map messageMap = new HashMap();
                    messageMap.put("message", download_url);
                    messageMap.put("seen", false);
                    messageMap.put("type", "image");
                    messageMap.put("time", ServerValue.TIMESTAMP);
                    messageMap.put("from", mCurrentUserId);
                    Map messageUserMap = new HashMap();
                    messageUserMap.put(current_user_ref + "/" + push_id,
messageMap);
                    messageUserMap.put(chat_user_ref + "/" + push_id,
messageMap);

                    mChatMessageView.setText("");
```

```
                    //Update reference
                    mRootRef.updateChildren(messageUserMap, new
DatabaseReference.CompletionListener() {
                        @Override
                        public void onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {

                            if(databaseError != null){

                                Log.d("CHAT_LOG",
databaseError.getMessage().toString());

                            }

                        }
                    });
                }
            }
        });
    }
}
```

```
-private void loadMoreMessages() {

        DatabaseReference messageRef =
mRootRef.child("messages").child(mCurrentUserId).child(mChatUser);

        Query messageQuery =
messageRef.orderByKey().endAt(mLastKey).limitToLast(TOTAL_ITEMS_TO_LOAD);

        messageQuery.addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {


                Messages message = dataSnapshot.getValue(Messages.class);
                String messageKey = dataSnapshot.getKey();

                if(!mPrevKey.equals(messageKey)){

                    messagesList.add(itemPos++, message);

                } else {

                    mPrevKey = mLastKey;

                }


                if(itemPos == 1) {

                    mLastKey = messageKey;

                }

                Log.d("TOTALKEYS", "Last Key : " + mLastKey + " | Prev Key :
" + mPrevKey + " | Message Key : " + messageKey);

                mAdapter.notifyDataSetChanged();


                mRefreshLayout.setRefreshing(false);

                mLinearLayout.scrollToPositionWithOffset(10, 0);

            }

            @Override
```

```
            public void onChildChanged(DataSnapshot dataSnapshot, String s)
{

            }

            @Override
            public void onChildRemoved(DataSnapshot dataSnapshot) {

            }

            @Override
            public void onChildMoved(DataSnapshot dataSnapshot, String s) {

            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
}
```

```
-private void loadMessages() {

    DatabaseReference messageRef =
mRootRef.child("messages").child(mCurrentUserId).child(mChatUser);

    //begins with loadMessages() which queries and filters the last
TOTAL_ITEMS = 10 * 1 values ,
    Query messageQuery = messageRef.limitToLast(mCurrentPage *
TOTAL_ITEMS_TO_LOAD);


    //akouei ta teletuaia auta 11 mhnuamta otan prosti8ontai
    messageQuery.addChildEventListener(new ChildEventListener() {
        @Override
        //Retrieve lists of items or listen for additions to a list of items
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {

            Messages message = dataSnapshot.getValue(Messages.class);

            itemPos++;

            if(itemPos == 1){

                String messageKey = dataSnapshot.getKey();

                mLastKey = messageKey;
                mPrevKey = messageKey;

            }

            messagesList.add(message);
            mAdapter.notifyDataSetChanged();

            // se paei sto katw katw mhnuma
            mMessagesList.scrollToPosition(messagesList.size() - 1);

            //den xreiazetai refresh
            mRefreshLayout.setRefreshing(false);
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) {

        }

        @Override
```

```
        public void onChildRemoved(DataSnapshot dataSnapshot) {

        }

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });}
```

Η εκκίνηση μιας δραστηριότητας δεν χρειάζεται να είναι αποκλειστικά μονόδρομη, αλλά μπορεί να ξεκινήσει και με σκοπό να επιστρέψει πίσω ένα αποτέλεσμα. Για να γίνει αυτό, καλούμε την ***startActivityForResult*** και μεταβαίνουμε στην onActivityResult, η οποία είναι η δραστηριότητα που θα μας επιστρέψει το επιθυμητό αντικείμενο [29]. Εδώ η onActivityResult θα μας φέρει πίσω μια φωτογραφία από το Gallery του κινητού, την οποία θα στείλει ο χρήστης στο φίλο του και θα αποθηκευτεί στο Storage του Firebase. Επιπλέον, τα δεδομένα του μηνύματος φωτογραφίας θα εγγραφούν στη βάση δεδομένων, μέσω της δημιουργίας ενός hashMap, ο οποίος θα περάσει σαν όρισμα στην ***updateChildren***.

Η loadMoreMessages ενεργοποιείται όταν κάποιος κάνει παρατεταμένο scrolling (***setOnRefreshListener***) και έχει ως στόχο τη φόρτωση παλαιότερων μηνυμάτων. Αρχικά, δημιουργούμε ένα query με τα δέκα πιο πρόσφατα μηνύματα, εκτός αυτών που ήδη βλέπουμε στην οθόνη (***Query messageQuery= messageRef.orderByKey().endAt(mLastKey).limitToLast(10);***)[26] και του προσθέτουμε έναν addChildEventListener, ώστε να είμαστε συνεχώς ενήμεροι του τι έχει φορτωθεί και αν υπάρχει καινούριο μήνυμα. Στη συνέχεια, ενημερώνουμε τον custom ***MessageAdapter*** που έχουμε δημιουργήσει, ώστε όλες οι αλλαγές να περνάνε στην οθόνη του χρήστη χωρίς καθυστέρηση.

Η loadMessages ενεργοποιείται αμέσως, μόλις ο χρήστης μπει στην ChatActivity και φορτώνει τα έντεκα τελευταία/πιο πρόσφατα μηνύματα, που έχουν ανταλλαχθεί μεταξύ του χρήστη και του φίλου του. Η πρακτική που χρησιμοποιείται είναι η ίδια με της loadMoreMessages, γι' αυτό και δεν θα επεκταθούμε περαιτέρω.

Στο συγκεκριμένο κομμάτι της εφαρμογής, παρόλο που χρησιμοποιούμε την πρακτική του RecyclerView προκειμένου να φτιάξουμε UI του chat, δεν χρησιμοποιούμε τον FirebaseRecyclerAdapter, αλλά δημιουργούμε το δικό μας MessageAdapter, ο οποίος είναι προσαρμοσμένος στις απαιτήσεις μας.

---Ο custom MessageAdapter---

Κλάση	*MessageAdapter*

```
-public class MessageAdapter extends
RecyclerView.Adapter<MessageAdapter.MessageViewHolder>{


    private List<Messages> mMessageList;
    private DatabaseReference mUserDatabase;

    public MessageAdapter(List<Messages> mMessageList) {

        this.mMessageList = mMessageList;

    }

    @Override
    public MessageViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {

        View v = LayoutInflater.from(parent.getContext())

.inflate(app.apognar.socialnet.R.layout.message_single_layout ,parent,
false);

        return new MessageViewHolder(v);

    }

    public class MessageViewHolder extends RecyclerView.ViewHolder {

        public TextView messageText;
        public CircleImageView profileImage;
        public TextView displayName;
        public TextView timestamp;
        public ImageView messageImage;

        public MessageViewHolder(View view) {
            super(view);

            messageText = (TextView)
view.findViewById(app.apognar.socialnet.R.id.message_text_layout);
            timestamp = (TextView)
view.findViewById(app.apognar.socialnet.R.id.time_text_layout);
            profileImage = (CircleImageView)
view.findViewById(app.apognar.socialnet.R.id.message_profile_layout);
            displayName = (TextView)
view.findViewById(app.apognar.socialnet.R.id.name_text_layout);
            messageImage = (ImageView)
view.findViewById(app.apognar.socialnet.R.id.message_image_layout);

        }
    }

    @Override
    public void onBindViewHolder(final MessageViewHolder viewHolder, int i)
{

        Messages c = mMessageList.get(i);

        String from_user = c.getFrom();
        String message_type = c.getType();

        Long timestamp = c.getTime();

        Date date = new Date(timestamp);
```

```
        Format format = new SimpleDateFormat("HH:mm");
        String finalized_time = format.format(date);

        viewHolder.timestamp.setText(finalized_time);

        mUserDatabase =
FirebaseDatabase.getInstance().getReference().child("Users").child(from_user
);

        mUserDatabase.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

                String name =
dataSnapshot.child("name").getValue().toString();
                String image =
dataSnapshot.child("thumb_image").getValue().toString();

                viewHolder.displayName.setText(name);


Picasso.with(viewHolder.profileImage.getContext()).load(image)

.placeholder(app.apognar.socialnet.R.drawable.default_avatar).into(viewHolde
r.profileImage);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });

        if(message_type.equals("text")) {

            viewHolder.messageText.setText(c.getMessage());
            viewHolder.messageImage.setVisibility(View.INVISIBLE);

        } else {

            viewHolder.messageText.setVisibility(View.INVISIBLE);

Picasso.with(viewHolder.profileImage.getContext()).load(c.getMessage())

.placeholder(app.apognar.socialnet.R.drawable.default_avatar).into(viewHolde
r.messageImage);

        }
    }

    @Override
    public int getItemCount() {
        return mMessageList.size();
    }
```

Σε αυτό το σημείο του κώδικα αρχικά δημιουργούμε τον ViewHolder, δίνοντας του τη μορφή του layout message_single_layout.xml που έχουμε σχεδιάσει οι ίδιοι στο Android Studio. Στη συνέχεια, τα διάφορα πεδία του γεμίζουν και μπορούμε να τα διαχειριστούμε μέσω της μεθόδου ***onBindViewHolder***, η οποία χρησιμοποιεί τους γνωστούς Listeners για αυτή τη δουλειά.

---Η βοηθητική και απαραίτητη εφαρμογή SocialNetwork---

Κλάση	*SocialNetwork*

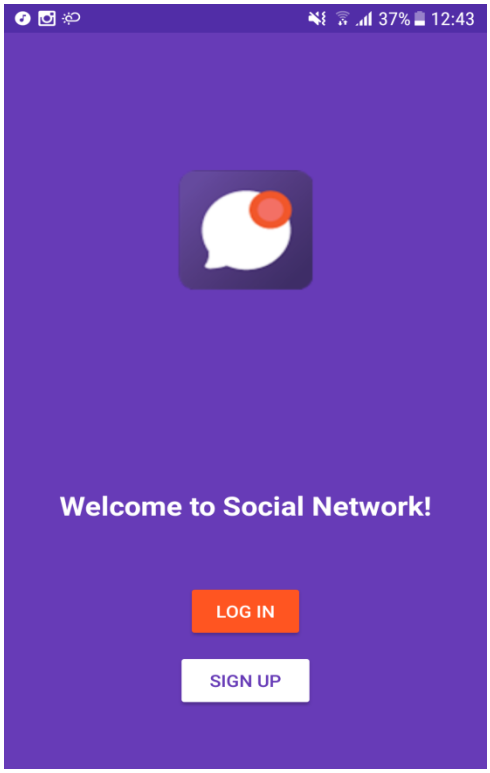
```
-public class SocialNetwork extends Application{

    private DatabaseReference mUserDatabase;
    private FirebaseAuth mAuth;

    @Override
    public void onCreate() {
        super.onCreate();

        FirebaseDatabase.getInstance().setPersistenceEnabled(true);
        Picasso.Builder builder = new Picasso.Builder(this);
        builder.downloader(new OkHttpDownloader(this, Integer.MAX_VALUE));
        Picasso built = builder.build();
        built.setIndicatorsEnabled(true);
        built.setLoggingEnabled(true);
        Picasso.setSingletonInstance(built);
        mAuth = FirebaseAuth.getInstance();

        if(mAuth.getCurrentUser() != null) {
            mUserDatabase = FirebaseDatabase.getInstance()

.getReference().child("Users").child(mAuth.getCurrentUser().getUid());

            mUserDatabase.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {

                    if (dataSnapshot != null) {

mUserDatabase.child("online").onDisconnect().setValue(ServerValue.TIMESTAMP)
;
                    }
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {

                }
            });
        }
    }

}
```

Η κλάση SocialNetwork είναι στην ουσία μία δεύτερη εφαρμογή μέσα στην αρχική, η οποία, λειτουργώντας με τρόπο ασύγχρονο, πραγματοποιεί κάποιες βασικές διεργασίες, ώστε να μην «πέσει» ποτέ η αρχική εφαρμογή από αυτές. Αναλυτικά, χρησιμοποιείται για να αποθηκεύει τα δεδομένα τοπικά στη συσκευή, στην περίπτωση που αυτή είναι offline (setPersistenceEnabled), να ρυθμίζει το Picasso για να έχει τις κατάλληλες βιβλιοθήκες για τις εικόνες της εφαρμογής και να φορτώνει το timestamp στη βάση δεδομένων για κάθε χρήστη, όταν αυτός συνδέεται.

5.2. Παράδειγμα σεναρίου χρήσης

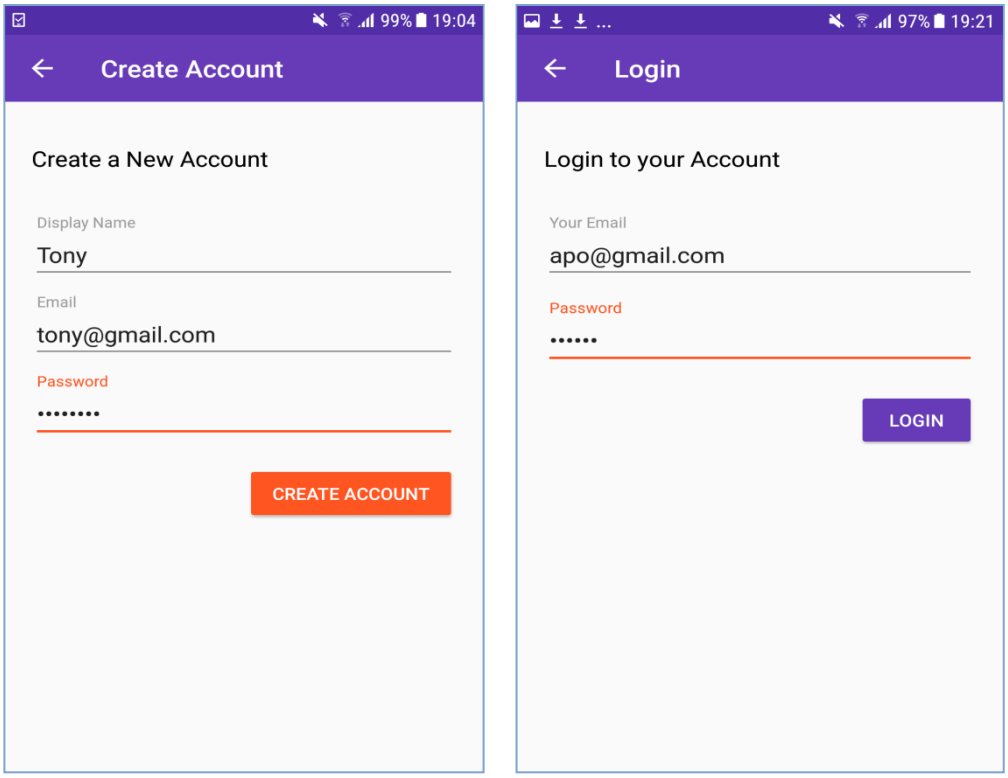
Στην παράγραφο αυτή θα περιγραφεί ένα σενάριο χρήσης της εφαρμογής άμεσης ανταλλαγής μηνυμάτων. Στο σενάριο θα συμπεριληφθούν δύο χρήστες, ο Tony και ο Apo. Μέσω αυτών των δυο χρηστών θα παρουσιαστούν όλες οι ενέργειες που μπορεί κάποιος να πραγματοποιήσει. Τα αρχικά δεδομένα του σεναρίου είναι στην ουσία οι χρήστες που ήδη έχουν εγγραφεί και έχουν αλληλεπιδράσει μεταξύ τους, είτε με αιτήματα φιλίας, είτε με μηνύματα. Εκτελώντας αυτό το σενάριο χρήσης, θα γίνει μια προσπάθεια καλύτερης κατανόησης των λειτουργιών της εφαρμογής.

Αρχικά, οι δυο χρήστες θα πρέπει να εγκαταστήσουν την εφαρμογή στο κινητό τους. Μόλις το βήμα αυτό ολοκληρωθεί, επιλέγοντας την εφαρμογή, θα μεταβούν στην εναρκτήρια οθόνη.



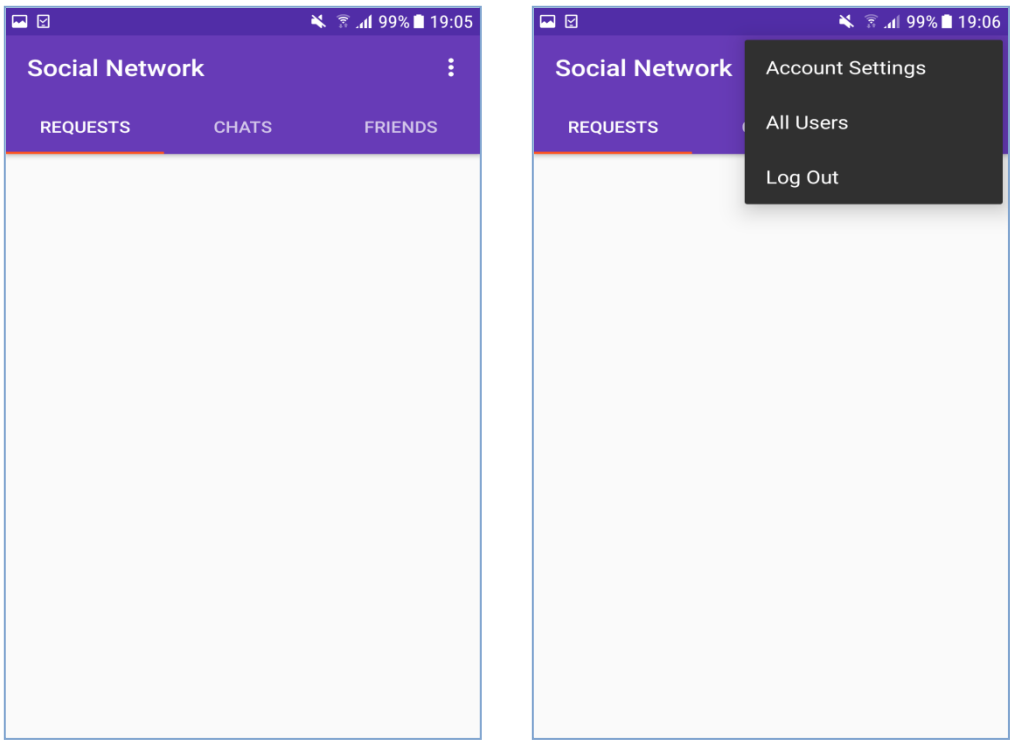
Εικόνα 32 Εναρκτήρια οθόνη εφαρμογής.

Στη συνέχεια, ο χρήστης Apo που ήδη έχει εγγραφεί στην εφαρμογή παλαιότερα, θα πατήσει Log in, ενώ ο χρήστης Tony που συνδέεται για πρώτη φορά θα πατήσει Sign up. Οι δύο χρήστες μεταβαίνουν στις οθόνες συμπλήρωσης των στοιχείων τους.

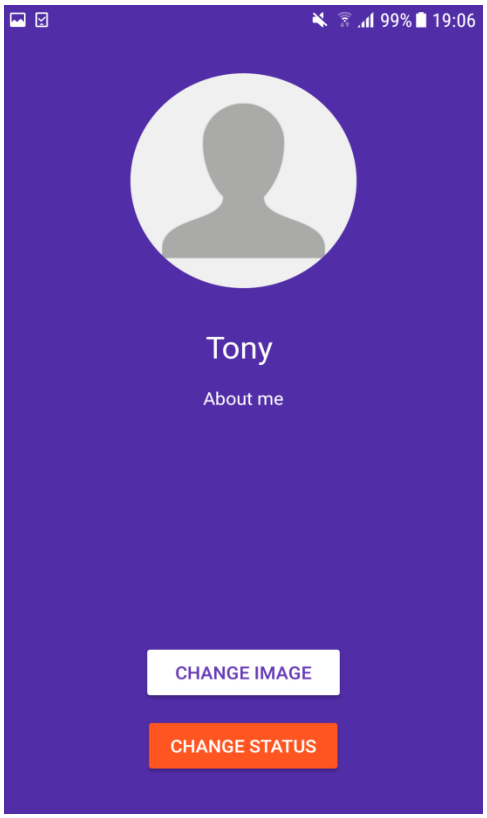


Εικόνα 33 Η οθόνη εγγραφής (αριστερά) και η οθόνη σύνδεσης (δεξιά).

Τα επόμενα βήματα αφορούν αποκλειστικά το χρήστη Tony. Μόλις αυτός εισάγει για πρώτη φορά τα στοιχεία του και γίνει έλεγχος ότι αυτά δεν υπάρχουν από άλλο χρήστη, θα μεταβεί στην κεντρική οθόνη της εφαρμογής. Η κεντρική οθόνη θα είναι άδεια από αιτήματα φιλίας, συνομιλίες και φίλους, εφόσον μπαίνει για πρώτη φορά. Επιλέγοντας το μενού πάνω δεξιά, του εμφανίζονται τρείς επιλογές: “Account Settings”, “All Users” και “Log Out”. Ο Tony αρχικά επιλέγει την πρώτη (Account Settings) προκειμένου να ρυθμίσει το προφίλ του. Έτσι, θα μεταβεί στο προφίλ του προκειμένου να ρυθμίσει την εικόνα και το στάτους του (που μέχρι εκείνη τη στιγμή είναι τα default).

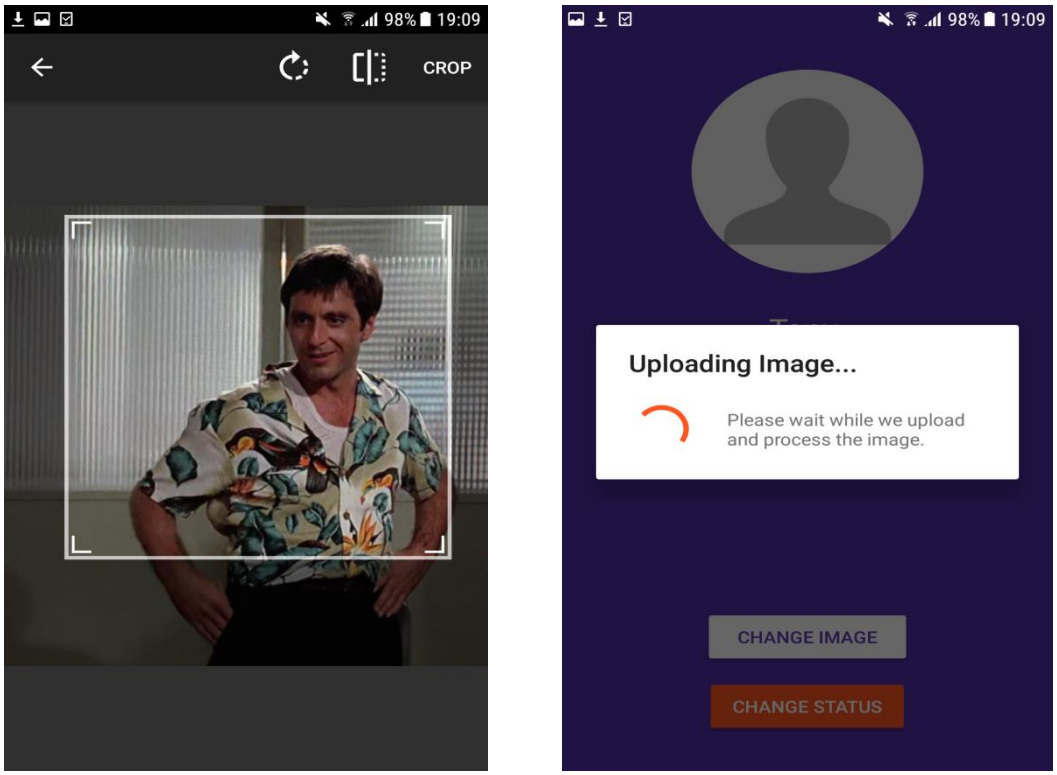


Εικόνα 34 Η κεντρική οθόνη (με τις καρτέλες της) και οι επιλογές του μενού.



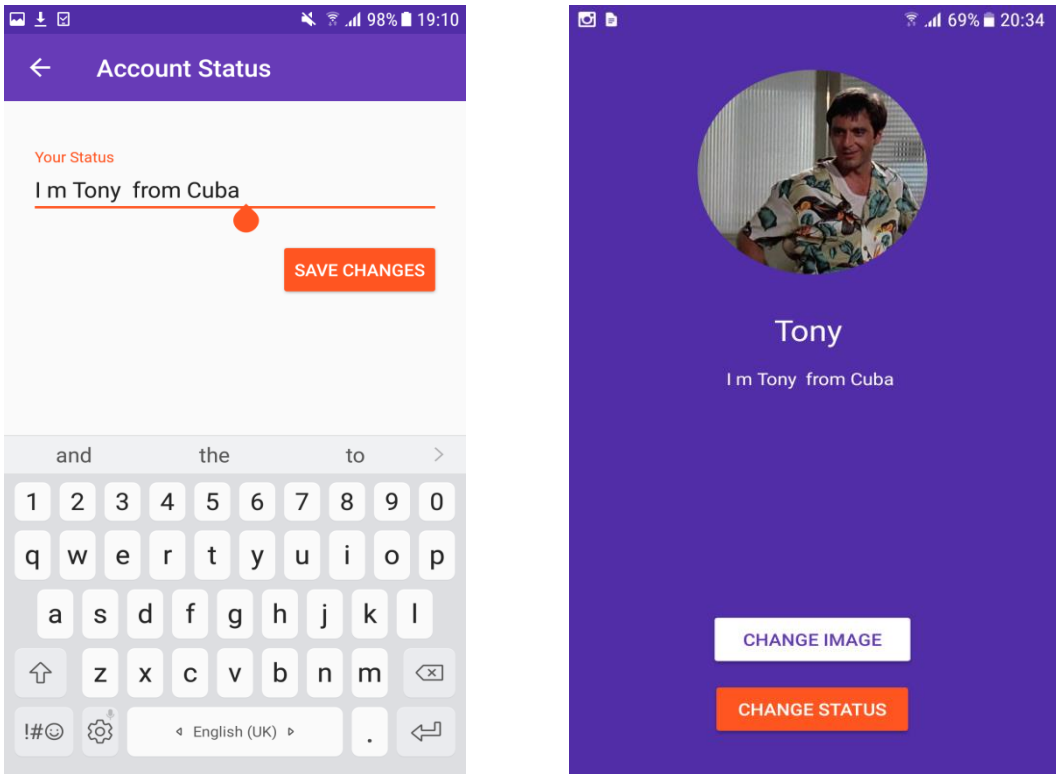
Εικόνα 35 Η οθόνη ρυθμίσεων λογαριασμού του χρήστη με τα default στοιχεία.

Από κει και πέρα, υπάρχουν δύο επιλογές. Αρχικά, αλλάζει την εικόνα του προφίλ του, επιλέγοντας το “Change Image” και, μεταβαίνοντας στο Gallery του κινητού, επιλέγει τη φωτογραφία που επιθυμεί, την κάνει crop και την ανεβάζει.



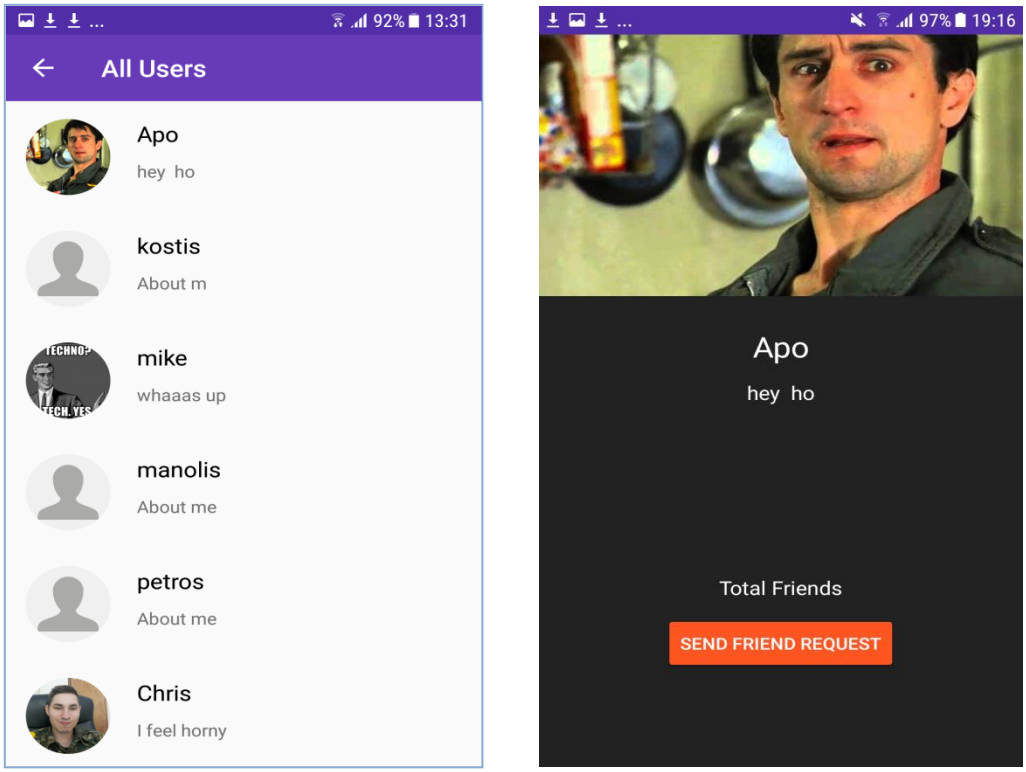
Εικόνα 36 Το crop της επιθυμητής φωτογραφίας και το «ανέβασμα» της.

Στη συνέχεια, επιλέγει το “Change Status”, μεταβαίνει στην οθόνη αλλαγής κατάστασης και πληκτρολογεί το νέο του στάτους. Αφού το πληκτρολογήσει, πατάει αποθήκευση (“Save Changes”) και επιστρέφει στην οθόνη ρυθμίσεων του λογαριασμού, όπου πλέον εμφανίζεται η νέα φωτογραφία και η νέα κατάσταση του χρήστη.

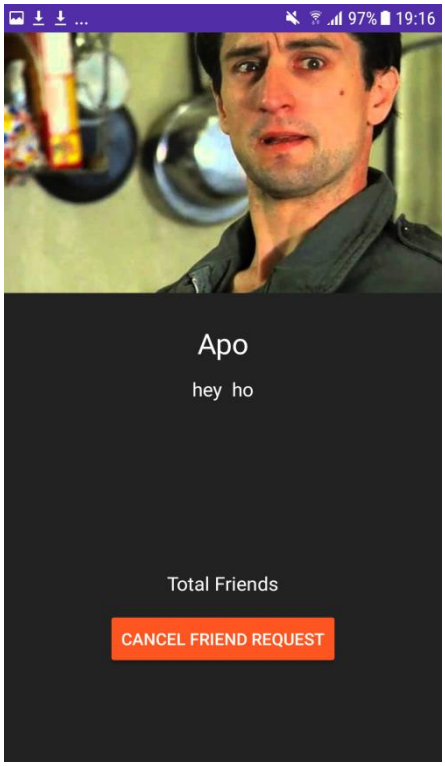


Εικόνα 37 Η οθόνη αλλαγής κατάστασης (αριστερά) και το νέο ανανεωμένο προφίλ (δεξιά).

Στη συνέχεια, ο Tony, επιστρέφει στην κεντρική οθόνη, ανοίγει το προαναφερθέν μενού και αυτή τη φορά επιλέγει το "All Users". Η επιλογή αυτή τον οδηγεί στη λίστα με όλους του χρήστες που έχουν δημιουργήσει λογαριασμό στο "Social Network". Από τους χρήστες επιλέγει τον Apo και μεταβαίνει στο προφίλ του. Εκεί μπορεί να δει την εικόνα προφίλ και το στάτους (του Apo), καθώς και να του στείλει ένα αίτημα φιλίας, πατώντας το κουμπί "Send Friend Request". Αφού πατήσει την αποστολή αιτήματος φιλίας, του εμφανίζεται πλέον η επιλογή ακύρωσης του αιτήματος φιλίας που έστειλε ("Cancel Friend Request").

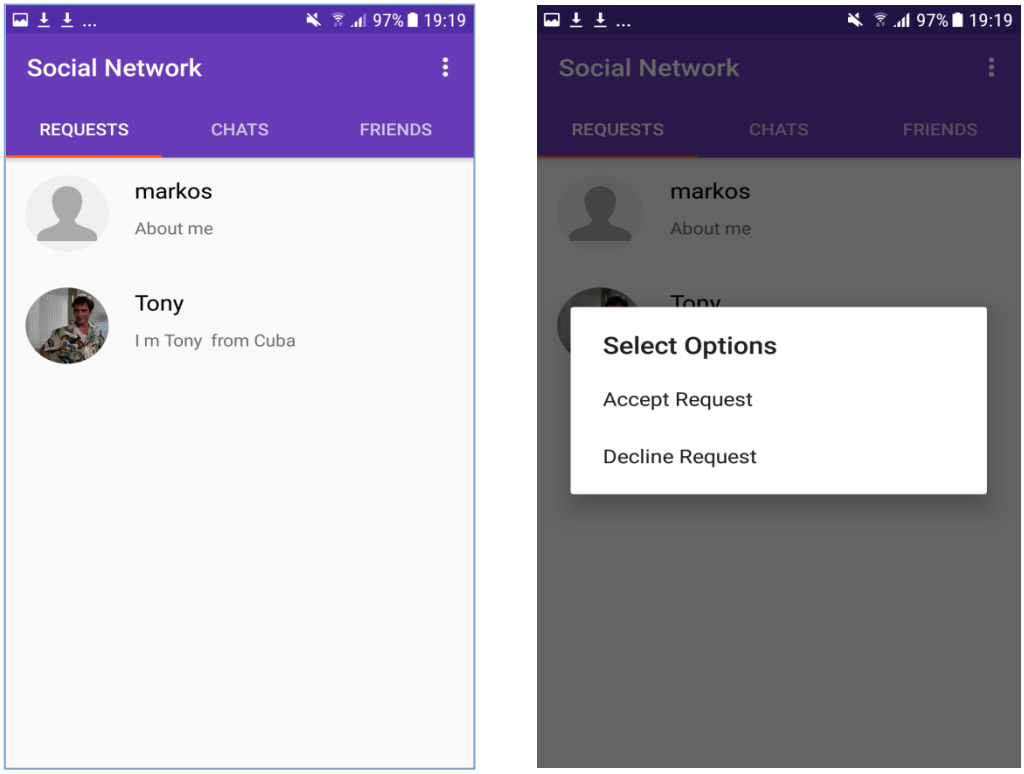


Εικόνα 38 Η λίστα όλων των χρηστών (αριστερά) και το προφίλ του χρήστη Apo (δεξιά).



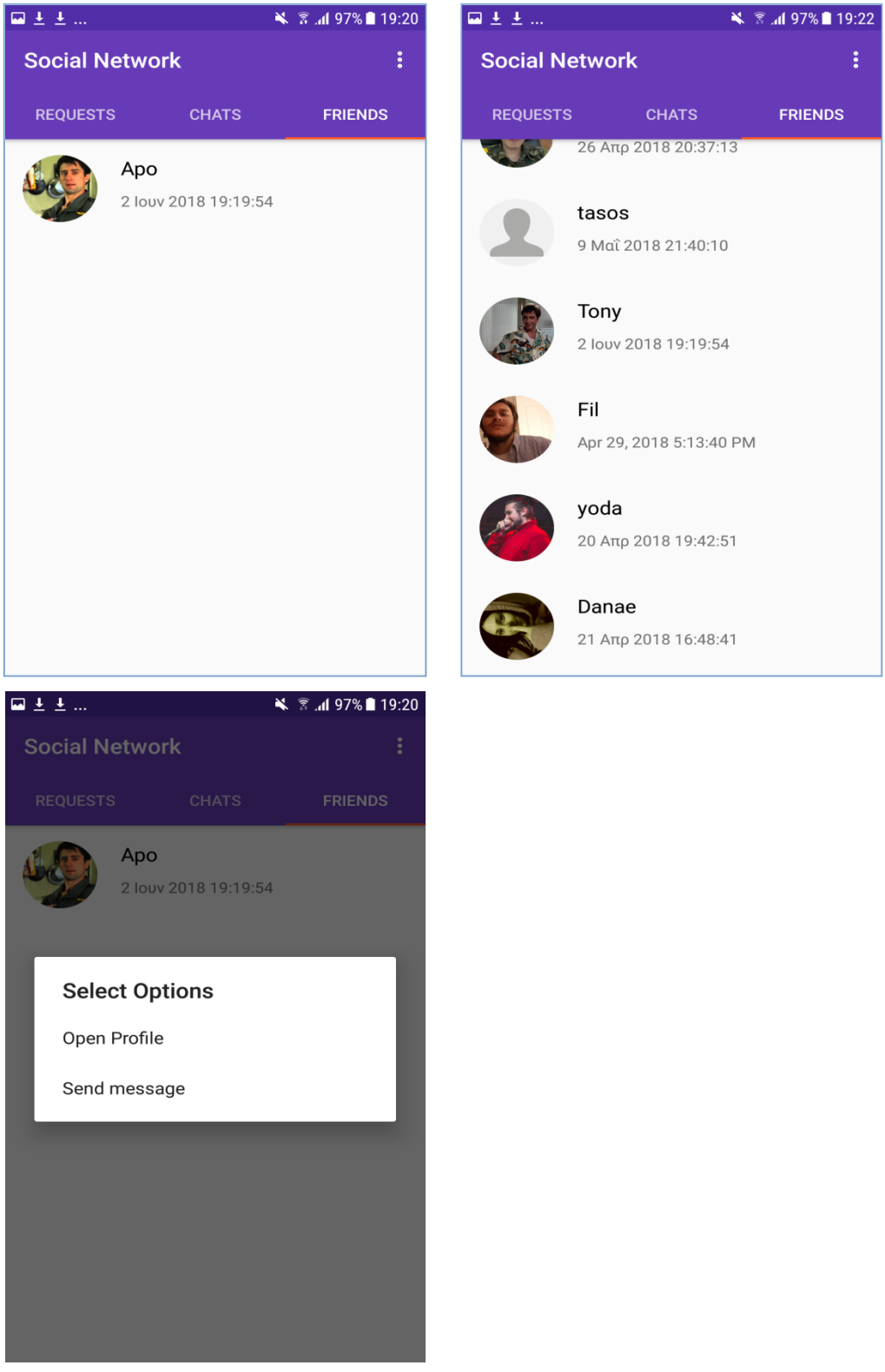
Εικόνα 39 Η επιλογή ακύρωσης του αιτήματος φιλίας που έστειλε.

Επανερχόμαστε στο χρήστη Apo, ο οποίος έχει συνδεθεί στην εφαρμογή και βρίσκεται στην κεντρική οθόνη. Εκεί, στην πρώτη καρτέλα, εμφανίζονται τα αιτήματα φιλίας που έχει δεχθεί. Ένα από αυτά είναι το αίτημα που έστειλε ο Tony. Το επιλέγει και του ανοίγουν δύο επιλογές: να το αποδεχτεί (“Accept Request”) ή να το απορρίψει (“Decline Request”).



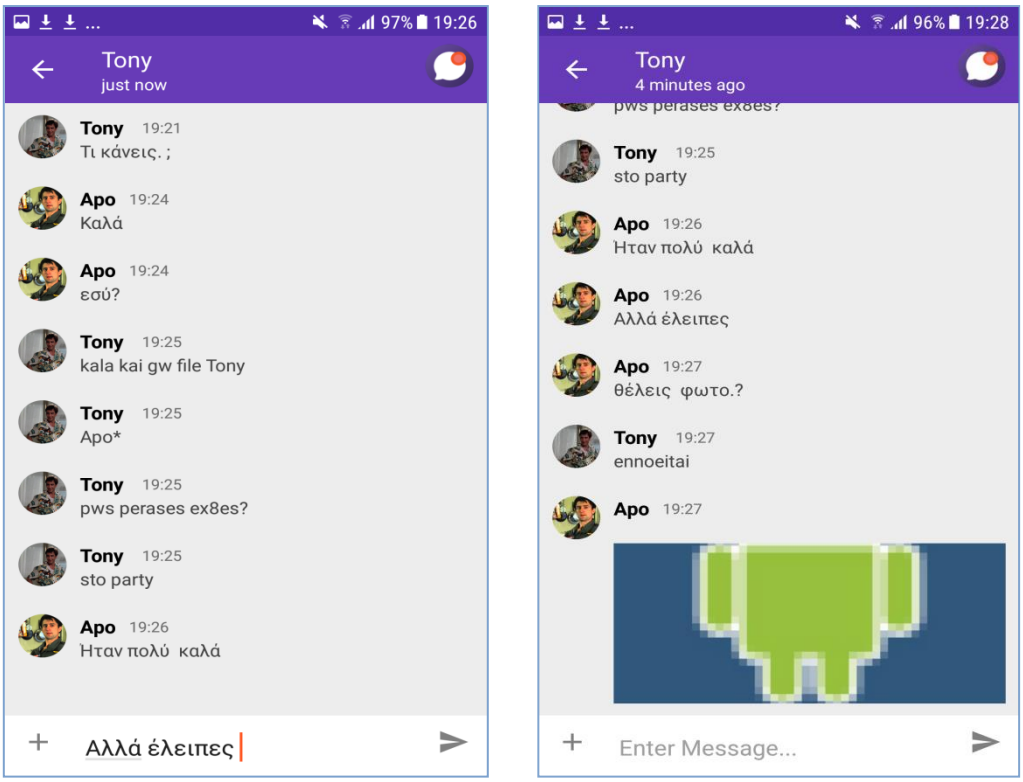
Εικόνα 40 Η λίστα αιτημάτων φιλίας και η αποδοχή/απόρριψη ενός εξ αυτών.

Ο χρήστης Apo επιλέγει να αποδεχτεί το αίτημα φιλίας και να δημιουργήσει μια νέα φιλία. Αυτό φαίνεται στην καρτέλα “Friends”, όπου υπάρχει η λίστα φίλων των χρηστών (μαζί με την ημερομηνία που έγιναν φίλοι). Έτσι, ο Tony, αποκτάει τον πρώτο του φίλο, πατάει κλικ πάνω σε αυτόν και του ανοίγουν δύο επιλογές : να δει το προφίλ του φίλου/χρήστη Apo (“Open Profile”) ή να του στείλει μήνυμα (“Send Message”). Επιλέγει τη δεύτερη και μεταβαίνει στη συνομιλία.



Εικόνα 41 Η λίστα φίλων (και των δυο χρηστών) και οι δυνατές ενέργειες μετά την επιλογή του φίλου Apo από τον Tony.

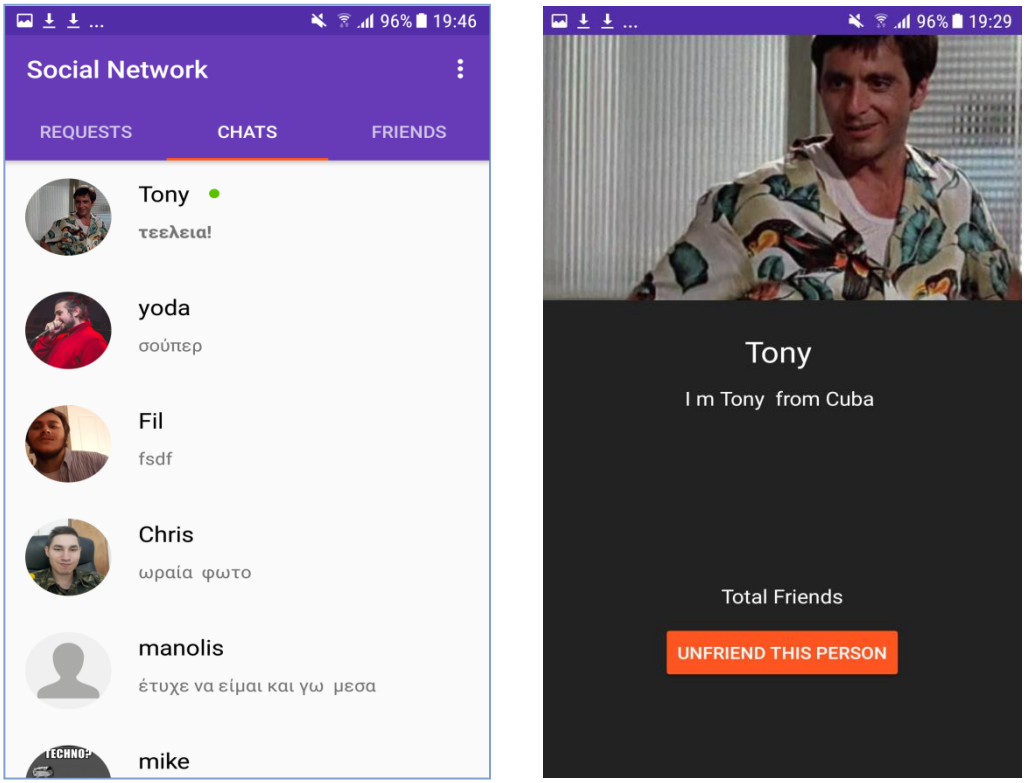
Οι δυο χρήστες αρχίζουν και ανταλλάσουν μηνύματα. Επιλέγοντας τη μπάρα εγγραφής κειμένου, δίνεται η δυνατότητα σε κάθε χρήστη, να γράψει ένα μήνυμα και με το σύμβολο της σαΐτας να το αποστείλει. Επιπλέον, πατώντας το σύμβολο «συν», ο κάθε χρήστης μπορεί να επιλέξει και να στείλει μία φωτογραφία από το Gallery του κινητού του. Τέλος, αν κάποιος από τους δύο χρήστες επιθυμεί να δει παλαιότερα μηνύματα, με παρατεταμένο scrolling, θα πραγματοποιήσει τη φόρτωσή τους. Έτσι, αφού ανταλλάξουν κάποια μηνύματα κειμένου οι δύο χρήστες, ο Apo στέλνει μία φωτογραφία. Μέσα στη συνομιλία με κάθε φίλο, φαίνεται πριν πόση ώρα έχει συνδεθεί ο καθένας ή αν είναι ενεργός τώρα.



Εικόνα 42 Η ανταλλαγή μηνύματος κειμένου και φωτογραφίας.

Τέλος, δείχνουμε τη λίστα συνομιλιών (καρτέλα “Chats”) του χρήστη Apo, στην οποία κάθε μήνυμα που δεν έχει διαβαστεί εμφανίζεται με bold, ενώ ταυτόχρονα όσοι χρήστες είναι εκείνη τη στιγμή συνδεδεμένοι στην εφαρμογή έχουν δίπλα στο όνομα τους ένα πράσινο λαμπάκι.

Αν κάποιος θελήσει να διαγράψει ένα φίλο, μπορεί να πάει στην λίστα φίλων, να επιλέξει τον φίλο, να πατήσει “Open Profile” και εκεί θα υπάρχει η επιλογή διαγραφής φίλου (“Unfriend This Person”). Δείχνουμε την επιλογή που έχει ο Apo να διαγράψει τον Tony από τους φίλους του.



Εικόνα 43 Η λίστα συνομιλιών του χρήστη Apo και η επιλογή διαγραφής φίλου.

ΚΕΦΑΛΑΙΟ 6

Επίλογος

6.1. Συμπεράσματα

Κατά την προσπάθεια σχεδίασης και ανάπτυξης της εφαρμογής άμεσης ανταλλαγής μηνυμάτων, ήρθαμε σε επαφή με ειδικά θέματα ανάπτυξης διαδικτυακών εφαρμογών τα οποία μας οδήγησαν σε ορισμένα χρήσιμα συμπεράσματα.

Η δημιουργία μίας εφαρμογής για κινητά τηλέφωνα απαιτεί την ανάλυση πολλών επιμέρους διαδικασιών. Εκτός από τις βασικές υπηρεσίες που πρέπει αυτή να παρέχει στο χρήστη, θα πρέπει να πληροί επιπλέον κριτήρια χρηστικής φύσεως για να θεωρηθεί ότι ολοκληρώθηκε με επιτυχία. Το πιο σημαντικό ίσως από αυτά, είναι η λειτουργικότητά της. Ένας χρήστης θέλει να εκμεταλλεύεται τις λειτουργίες που του προσφέρονται εύκολα και με τρόπο αποτελεσματικό. Επιπλέον, είναι απαραίτητο το περιεχόμενο της εφαρμογής να εμφανίζεται με καλή σχεδίαση και να ταιριάζει στα ιδιαίτερα χαρακτηριστικά της εκάστοτε συσκευής. Ακόμα και το όνομα της εφαρμογής μπορεί να παίξει ρόλο στην τελική της επιτυχία. Όλα τα προαναφερθέντα απαιτούν να συνεκτιμώνται προσεκτικά και συνιστούν σημαντικό κομμάτι της σχεδίασης μίας εφαρμογής αντίστοιχου τύπου.

Η λειτουργία που προσφέρει μια υπηρεσία άμεσης ανταλλαγής μηνυμάτων είναι ιδιαίτερα χρήσιμη, ειδικά στις μέρες μας, όπου η καθημερινότητα εξελίσσεται με διαρκώς ταχύτερους ρυθμούς. Οι άνθρωποι προκειμένου να οργανώσουν όλες τις επιθυμητές δραστηριότητες, εργασιακές ή κοινωνικές, έχουν ανάγκη να επικοινωνούν με άλλους, στα στενά χρονικά πλαίσια που διαθέτουν. Τη λύση σε αυτό το πρόβλημα, δίνουν με σύγχρονο και αποτελεσματικό τρόπο οι εφαρμογές άμεσης ανταλλαγής μηνυμάτων, παρέχοντας έναν εύκολο τρόπο επικοινωνίας μεταξύ των ανθρώπων.

Για την ολοκλήρωση της εφαρμογής χρησιμοποιήθηκε η πλατφόρμα της Google, Firebase. Το Firebase έπαιξε το ρόλο του εξυπηρετητή (server), παρέχοντας

υπηρεσίες ταυτοποίησης στοιχείων, βάσης δεδομένων πραγματικού χρόνου και αποθήκευσης. Τα ιδιαίτερα πλεονεκτήματα που προσφέρει αυτή η πλατφόρμα είναι η ευκολία στη χρήση της (αποφεύγεται ο backend κώδικας), η δωρεάν πρόσβαση σε αυτή (οι βασικές υπηρεσίες της προσφέρονται δωρεάν στο διαδίκτυο), το εύρος λειτουργιών που παρέχει (Authentication, Real-Time Database, Storage), η απλότητα της ενσωμάτωσής της σε ένα Android Project και η ευελιξία της σε πιθανές αλλαγές.

Τέλος, η ενασχόληση με τις σχετικές τεχνολογίες στο πλαίσιο της παρούσας εργασίας, μας οδήγησε σε ένα ενδιαφέρον συμπέρασμα για την εξέλιξη τους . Μπορεί η καινοτομία και ο στόχος της τεχνολογίας των υπολογιστών να παραμένει η συνεχής βελτίωση της αποτελεσματικότητάς τους και η διείσδυση στην καθημερινότητα των ανθρώπων, αλλά πλέον, στο κέντρο των αναγκών ενός χρήστη έχει εισέλθει η φορητότητα. Οι άνθρωποι, επειδή χρησιμοποιούν τις «έξυπνες» συσκευές για πολλαπλούς σκοπούς (όχι μόνο επαγγελματικούς, όπως παλαιότερα), έλκονται ολοένα και περισσότερο από το συνδυασμό της ευκολίας στη μεταφορά τους αλλά και του εύρους των δυνατοτήτων που αυτές μπορούν να προσφέρουν.

6.2. Μελλοντικές επεκτάσεις

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, η εφαρμογή που αναπτύχθηκε είναι εύκολη στην χρήση και πρακτική, πληρώντας όλα τα απαραίτητα κριτήρια για να διατεθεί στο ηλεκτρονικό κατάστημα εφαρμογών της Google, Google Play Store. Ωστόσο, αυτό δεν σημαίνει ότι δεν υπάρχουν οι δυνατότητες για προσθήκη νέων χαρακτηριστικών και βελτίωση των ήδη υπαρχόντων. Μερικά από αυτά είναι:

- Ειδοποιήσεις ή Push Notifications. Στην τρέχουσα μορφή της εφαρμογής, προκειμένου ο χρήστης να ενημερώνεται για πιθανά μηνύματα ή αιτήματα φιλίας που έχει δεχθεί, πρέπει η εφαρμογή να εκτελείται στο front-end του κινητού. Προσθέτοντας push notifications (τα οποία ουσιαστικά είναι πληροφοριακά μηνύματα που παραδίδονται στους χρήστες μιας εφαρμογής ακόμα και αν δεν τη χρησιμοποιούν), ο χρήστης θα μπορεί να ενημερώνεται διαρκώς για τις εξελίξεις του λογαριασμού του, χωρίς να χρειάζεται να έχει συνεχώς ανοιχτή την εφαρμογή. Το στοιχείο αυτό θα μπορούσε να πραγματοποιηθεί με τις Cloud Functions που προσφέρει το Firebase.

- Κλήσεις. Η προσθήκη εκτέλεσης τηλεφωνικών κλήσεων στην εφαρμογή μας, αποτελεί μία σημαντική προσθήκη για την αναβάθμισή της. Το στοιχείο αυτό, παρέχει ένα σοβαρό πλεονέκτημα στην εφαρμογή, αφού ο χρήστης θα μπορεί να αποφύγει τη χρέωση μιας τηλεφωνικής υπηρεσίας.
- Αποστολή αρχείων διαφόρων τύπων. Στην τρέχουσα μορφή της εφαρμογής, ο χρήστης μπορεί να ανταλλάσσει με τους φίλους του μηνύματα κειμένου και φωτογραφίες. Μία ενδιαφέρουσα βελτίωση, είναι η δυνατότητα αποστολής αρχείων άλλων τύπων, όπως doc, pdf, mp4, zip κλπ.
- Κρυπτογραφημένα μηνύματα. Πολύ σημαντική προσθήκη στην εφαρμογή είναι η κρυπτογράφηση των μηνυμάτων, ειδικά στις μέρες μας, όπου οι καταναλωτές επιζητούν υπηρεσίες που θα τους παρέχουν ασφάλεια. Αυτό απαιτεί τη χρήση ενός αλγορίθμου κρυπτογράφησης, ο οποίος θα κρυπτογραφεί το επιθυμητό μήνυμα πριν αυτό αποσταλεί και θα το αποκρυπτογραφεί όταν φτάνει στον παραλήπτη. Απαραίτητη διαδικασία σε αυτή την κατεύθυνση είναι η προσθήκη μιας γεννήτριας παραγωγής κλειδιών, προκειμένου το μοτίβο της κρυπτογράφησης να εναλλάσσεται.

Βιβλιογραφία

[1] Εμμανουήλ Γ. Καραμανή, Πέτρου Φλώριου Ν. Μπάκαλου, *Σχεδίαση και ανάπτυξη εφαρμογής προσωπικού προγραμματιστή δραστηριοτήτων σε πλατφόρμα iOS*, 2013

[2] Christine, *The Evolution of Communication Technology*, SmarterWare, 2017

[3] Wikipedia, *Dial-up Internet access*, 2017

[4] Guru, *Top 10 Mobile Phones Operating Systems*, Shout Me Loud, 2017

[5] Statista, *Global market share held by smartphone operating systems from 2009 to 2017*, 2018

[6] Statista, *Number of available applications in the Google Play Store from December 2009 to March 2018*, 2018

[7] Editorial Team, *The History of Social Networking: How It All Began*, 1st WebDesigner, 2016

[8] Technopedia, *Distributed Network*, 2017

[9] Jajish Thomas, *Centralized and Distributed Computer Network Model*, Omnisecu

[10] Technopedia, *Distributed Database*, 2017

[11] Wikipedia, *Android Studio*, 2018

[12] Wikipedia, *Android (operating system)*, 2018

[13] Android Developers, *Platform Architecture*, (https://developer.android.com/guide/platform/), 2018

[14] TutorialsPoint, *Android-Architecture*, (https://www.tutorialspoint.com/android/android_architecture.htm), 2017

[15] Android Developers, *Application Fundamentals*, (https://developer.android.com/guide/components/fundamentals), 2018

[16] Βικιπαίδεια, *Java*, 2018

[17] Wikipedia, *Java performance*, 2018

[18] Wikipedia, *Java (programming language)*, 2018

[19] Android Developers, *Download Android Studio and SDK Tools*, (https://developer.android.com/studio/), 2018

[20] Firebase, (https://firebase.google.com/)

[21] JSON, (https://www.json.org/)

[22] Virender Singh, *Client Server Architecture and HTTP Protocol*, ToolsQA, 2017

[23] Πανεπιστήμιο Θεσσαλίας, *Το Πρωτόκολλο TCP/IP*, 1997

[24] Android Developers, *Fragment*, (https://developer.android.com/guide/components/fragments#Design), 2018

[25] Android Developers, *ViewPager*, (https://developer.android.com/reference/android/support/v4/view/ViewPager), 2018

[26] Firebase, *Work with Lists of Data on Android*, (https://firebase.google.com/docs/database/android/lists-of-data), 2018

[27] Android Developers, *Create a List with RecyclerView*, (https://developer.android.com/guide/topics/ui/layout/recyclerview), 2018

[28] Firebase, *Read and Write Data on Android*, (https://firebase.google.com/docs/database/android/read-and-write), 2018

[29] Android Developers, *Getting a Result from an Activity*, (https://developer.android.com/training/basics/intents/result), 2018