ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής ολοκληρωμένων υπηρεσιών Mobile Banking

Γεώργιος Δ. Παπαχρήστου

Εισηγητής: Δρ Κωνσταντίνος Κουκουλέτσος, Καθηγητής

ΑΘΗΝΑ
ΙΟΥΝΙΟΣ 2018

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής ολοκληρωμένων υπηρεσιών Mobile Banking

Γεώργιος Δ. Παπαχρήστου
Α.Μ. ais0119

Εισηγητής:

Δρ Κωνσταντίνος Κουκουλέτσος, Καθηγητής

Εξεταστική Επιτροπή:

…….

…….

Ημερομηνία εξέτασης …….

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος Παπαχρήστου Γεώργιος, του Διογένη, με αριθμό μητρώου AIS-0119 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της ανάπτυξης mobile εφαρμογής σε πλατφόρμα Android για την πληροφόρηση του χρήστη για τις τραπεζικές του κινήσεις. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, ο κύριος Κουκουλέτσος Κωνσταντίνος, τον οποίο θα ήθελα να ευχαριστήσω.

Ακόμα θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη που μου παρείχαν καθ' όλη τη διάρκεια του Μεταπτυχιακού προγράμματος.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη εφαρμογής για κινητή συσκευή, με λειτουργικό σύστημα Android, ώστε οι χρήστες να έχουν πληροφόρηση για τις τραπεζικές τους κινήσεις. Ο χρήστης με το που θα συνδέεται με το email του, θα βλέπει όλους του λογαριασμούς που έχει σε όλες τις τράπεζες. Στην συνέχεια θα μπορεί επιλέγοντας κάποιον συγκεκριμένο λογαριασμό, να βλέπει τις κινήσεις του λογαριασμού αυτού.

Η ανάπτυξη της εφαρμογής αυτής αναλύεται σταδιακά, κάνοντας πρώτα μια αναφορά σε μια ιστορική αναδρομή στο Android και στην συνέχεια επικεντρώνεται στην ανάπτυξη της εφαρμογής.

ABSTRACT

This thesis presents the development of an application for mobile device, with Android operating system, so that users can have information about their bank transactions. The user will have the ability to log into application, using his email address. He then can see his bank accounts in every bank he is customer. Moreover, by selecting a specific account, he will see all the transactions of this specific account.

The development of this application is presented with a high level of detail, starting with a sort history of the Android operating system and afterwards focusing on the details of the application.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη Εφαρμογής σε Λειτουργικό Android
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: app, Android, Τραπεζικοί Λογαριασμοί, Google Maps, Firebase

Πίνακας περιεχομένων

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

1. ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας, γίνεται σύντομη αναφορά στις έξυπνες συσκευές (smart phones) και κάνουμε μια ιστορική αναδρομή στο λειτουργικό Android και στην αρχιτεκτονική του.

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη εφαρμογής Android, για την πληροφόρηση των χρηστών της εφαρμογής για την κίνηση των τραπεζικών τους λογαριασμών. Ο στόχος είναι να δώσουμε την ευκολία στον χρήστη της εφαρμογής να μπορέσει να έχει κάπου συγκεντρωμένα τα δεδομένα από τις κινήσεις που κάνει (αγορές, μεταφορές, πληρωμές, κ.α.), ώστε να μην είναι αναγκασμένος να μπαίνει στην αντίστοιχη εφαρμογή της κάθε τράπεζας. Έτσι θα έχουμε συγκεντρωμένη την πληροφορία από όλες τις τράπεζες για όλους τους λογαριασμούς του χρήστη και θα μπορεί να ενημερώνεται ανά πάσα στιγμή για τις συναλλαγές του.

1.2 Έξυπνες Συσκευές (Smart Phones)

Η έννοια smart phone εισήχθη πρώτη φορά στο λεξιλόγιο της τεχνολογίας το 1997 από την Ericsson, με την παραγωγή του μοντέλου GS 88 “Penelope”. Περιλάμβανε λειτουργικό σύστημα Geoworks, GEOS, και περιελάμβανε POP3 email, SMS, παγκόσμιο ρολόι, browser, modem[7].

Η ραγδαία εξελικτική πορεία που είχε παραγωγή των έξυπνων κινητών συσκευών, μας οδήγησε μέσα στα επόμενα 20 χρόνια, στο να γίνουν οι συσκευές αυτές προσιτές στο ευρύ κοινό, με τεράστιες δυνατότητες σε υλικό (Hardware), αλλά και πληθώρα εφαρμογών που βοηθάνε τον χρήστη με το πάτημα ενός μόνο κουμπιού.

Σήμερα οι κάτοχοι τέτοιων συσκευών, έχουν την δυνατότητα να κάνουν πράγματα που έπρεπε παλιότερα να τα κάνουν μόνο στον ηλεκτρονικό υπολογιστή τους, όπως επεξεργασία κειμένου, ήχου, περιήγησης στον παγκόσμιο ιστό, κλπ. Επίσης το κινητό πλέον είναι ένας πλοηγός (με χρήση GPS), ή

περιλαμβάνει πολύ εξελιγμένη κάμερα για φωτογραφίες και βίντεο καθώς και πολλά άλλα εργαλεία.

Εικόνα 1-1: Το πρώτο ‘Έξυπνο Κινητό’

1.3 Τι είναι το Android

Το Android είναι ένα λειτουργικό σύστημα ανοικτού κώδικα το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Η ανάπτυξή του υλοποιείται από την Google για συσκευές Smart Phones, Tablets, αλλά και για τηλεοράσεις (Android TV), έξυπνα ρολόγια (smart watches), αυτοκίνητα (Android Auto). Η γλώσσα προγραμματισμού που χρησιμοποιείται, για ανάπτυξη εφαρμογών είναι η Java [1].

1.4 Ιστορικά στοιχεία

Το λειτουργικό Android αναπτύχθηκε από την εταιρία Android Inc. όπου εξαγοράστηκε από την Google το 2005, αφού υπήρχε η έντονη επιθυμία της Google να μπει στον χώρο της κινητής τηλεφωνία [8, 2].

Το 2007 ανακοινώθηκε ο συνεταιρισμός Open Handset Alliance, πολλές εταιρείες από τον χώρο της τεχνολογίας όπως οι: Google, HTC, Sony, Samsung, Qualcomm, T-Mobile κτλ., με σκοπό την εξέλιξη των ανοιχτών προτύπων σε συσκευές κινητής τηλεφωνίας και παρουσιάστηκε το νέο λειτουργικό για έξυπνες συσκευές «Android» το οποίο ήταν βασισμένο στον πυρήνα του Linux και είχε σαν

λογότυπο το γνωστό πλέον σε όλους ρομπότ σε χρώμα πράσινο μήλο. Τον Οκτώβριο του 2008 το κινητό HTC Dream παρουσιάστηκε ως η πρώτη συσκευή με λειτουργικό Android.

Η Google παρουσίασε την πρώτη συσκευή της με το λειτουργικό της σύστημα Android τον Ιανουάριο του 2010. Η συσκευή είχε την ονομασία Nexus One και κατασκευάστηκε από την HTC Corporation.

1.5 Εκδόσεις Android

Η αρχική έκδοση του Android ήταν τον Νοέμβριο του 2007 με την beta έκδοση, και στη συνέχεια κυκλοφόρησε η πρώτη εμπορική του έκδοση Android 1.0 τον Σεπτέμβριο του 2008. Η τελευταία έκδοση του παρουσιάστηκε τον Σεπτέμβρη του 2017 με την κωδική ονομασία Oreo. Όλες οι εκδόσεις του, έχουν το όνομα από κάποιο γλυκό εκτός από τις δύο πρώτες (εκδόσεις 1.0, 1.1) που δεν είχαν κάποια ονομασία.

Εικόνα 1-2: Το σήμα του Android

1.5.1 Android 1.0 και Android 1.1 (API Level 1 και 2)

Η έκδοση 1.0 κυκλοφόρησε το Σεπτέμβριο του 2008 με το HTC Dream. Τον Φεβρουάριο του 2009 κυκλοφόρησε και η έκδοση 1.1. Περιελάμβανε την εφαρμογή Android Market για λήψη και αναβάθμιση εφαρμογών, εφαρμογή Web Browser, GPS, υποστήριξη Wi-Fi, χάρτες (Google maps), κάμερα, email, αριθμομηχανή, κλπ.

1.5.2 Android 1.5 Cupcake (API Level 3)

Η έκδοση αυτή κυκλοφόρησε τον Απρίλιο του 2008 και ήταν το πρώτο που είχε όνομα από γλυκό. Περιελάμβανε πολλά νέα χαρακτηριστικά και αλλαγές στο γραφικό περιβάλλον. Το Cupcake ήταν βασισμένο στο Linux Kernel 2.6.27.

Παρείχε νέες δυνατότητες όπως εγγραφή – αναπαραγωγή βίντεο σε μορφή MPEG-4 καθώς και άμεση μεταφόρτωση βίντεο – φωτογραφιών σε YouTube και Picasa αντίστοιχα απευθείας από το τηλέφωνο. Φτιάχτηκε εικονικό πληκτρολόγιο με πρόβλεψη λέξεων και προστέθηκαν νέες μικρές εφαρμογές που προστίθενται στην αρχική οθόνη (widgets) που μπορούν να ανανεώνονται τακτικά ώστε να προβάλουν ενημερωμένο περιεχόμενο.

1.5.3 Android 1.5 Donut (API Level 4)

Τον Σεπτέμβριο του 2009 κυκλοφόρησε η έκδοση Donut η οποία αναβάθμησε την εικόνα και την ταχύτητα απόκρισης. Βασίστηκε στο Linux Kernel 2.6.27.

Περιελάμβανε την βελτίωση της προβολής και αναζήτησης στο Android Market, την εφαρμογή Gallery για προβολή φωτογραφιών και βίντεο, ταχύτερη πρόσβαση στην κάμερα. Επιλογή πολλαπλών αρχείων, αναζήτηση σελιδοδεικτών και ιστορικού στο διαδίκτυο. Υποστήριξη για ανάλυση οθόνης WVGA και μηχανή μετατροπής κειμένου σε ομιλία.

1.5.4 Android 2.0 – 2.1 Eclair (API Level 5 – 7)

Η έκδοση Eclair κυκλοφόρησε τον Οκτώβριο του 2009, μόλις ένα μήνα μετά την έκδοση Donut. Υποστήριζε Bluetooth 2.1, βελτιωμένη ταχύτητα απόκρισης και υποστήριζε περισσότερες αναλύσεις οθονών. Είχε ανανεωμένο Browser για υποστήριξη HTML5, και παρείχε δυνατότητα χρήσης φλας και ψηφιακό zoom.

Το Eclair βασιζόταν στο Linux Kernel 2.6.29 με πλέον υποστήριξη για την τεχνολογία NFC “Near Field Comminication” όπως και για την τεχνολογία SIP “Session Initiation Protocol” που χρησιμοποιείται στην διαδικτυακή τηλεφωνία “VoIP Internet Telephony”.

1.5.5 Android 2.2 – 2.2.3 Froyo (API Level 8)

Τον Μάιο του 2010 κυκλοφόρησε η έκδοση Froyo (από την φράση “Frozen Yogurt”) βασισμένη στο Linux Kernel 2.6.32. Ήταν η πρώτη έκδοση Android που υποστήριζε Adobe Flash, εγκατάσταση εφαρμογών στην κάρτα μνήμης και δυνατότητα μετατροπής του κινητού σε Wi-Fi hotspot. Βελτίωση της ταχύτητας, με καλύτερη διαχείριση μνήμης. Τέλος το Android Market έδινε την δυνατότητα αυτόματων ενημερώσεων και ενσωμάτωση του JacaScript του Chrome V8 στον browser.

1.5.6 Android 2.3 – 2.3.7 Gingerbread (API Level 9 – 10)

Ένα εξάμηνο μετά, τον Δεκέμβριο του 2010, κυκλοφόρησε η έκδοση Gingerbread βασισμένη στο Linux Kernel 2.6.35. Με αυτή την έκδοση υποστηρίζεται το Session Initiation Protocol (SIP) με το οποίο επιτυγχάνεται η κλήση μέσω internet (VoIP). Το User Interface είναι πλέον πιο γρήγορο και εύχρηστο για τον χρήστη καθώς και πιο αποδοτικό ενεργειακά. Η έξυπνη συσκευή Nexus S που κυκλοφόρησε το 2010, ήταν το πρώτο τηλέφωνο της σειράς Google Nexus που περιελάμβανε το Gingerbread και η πρώτη της σειράς με ενσωματωμένη την λειτουργία NFC (Near Field Communication).

1.5.7 Android 3.0 – 3.2.6 Honeycomb (API Level 11 – 13)

Μόλις περίπου 3 μήνες μετά, τέλη Φεβρουαρίου του 2011, κυκλοφόρησε η έκδοση Honeycomb, η οποία ήταν βασισμένη στο Linux Kernel 2.6.32. Η έκδοση αυτή προοριζόταν κυρίως για συσκευές με μεγάλο μέγεθος οθόνης. Στόχος ήταν να υπάρξουν βελτιώσεις σε συσκευές Tablet. Οι κύριες αλλαγές του ήταν στο User Interface καθώς επίσης και η υποστήριξη σε διπύρηνους και τετραπύρηνους επεξεργαστές. Το Honeycomb εισήγαγε ένα μοντέλο αλληλεπίδρασης που χτίστηκε πάνω στα κύρια χαρακτηριστικά του Android, όπως το multitasking, τις κοινοποιήσεις και τα widgets.

1.5.8 Android 4.0 – 4.0.4 Ice Cream Sandwich (API Level 14 – 15)

Τον Οκτώβριο του 2011έκανε την εμφάνιση της η επόμενη έκδοση, το Ice Cream Sandwich το οποίο βασίστηκε στο Linux Kernel 3.0.1. Ανάμεσα στις

πολλές αλλαγές στο User Interface ήταν και το ότι προστέθηκε η δυνατότητα χρήσης κουμπιών πάνω στην οθόνη (πίσω, αρχική, κλπ), επίσης το Face Unlock και η καλύτερη χρήση φωνητικών εντολών. Κάτι ακόμα καινοτόμο ήταν και η δυνατότητα λήψης βίντεο 1080p και το γεγονός ότι επιτρεπόταν ο browser να ανοίγει μέχρι και 16 καρτέλες. Τέλος με την εφαρμογή Android Beam επιτρεπόταν η αποστολή δεδομένων από την συσκευή σε όσες βρίσκονταν μέσα σε μια μικρή ακτίνα.

1.5.9 Android 4.1 – 4.3.1 Jelly Bean (API Level 16 – 18)

Τον Ιούλιο του 2012 κυκλοφόρησε το Jelly Bean 4.1, το οποία βασίστηκε στο Linux Kernel 3.0.31. Οι βελτιώσεις που έγιναν στην έκδοση αυτή εστίαζαν σε πιο φιλικό User Interface καθώς επίσης και στην ταχύτητα. Τον Οκτώβριο το ίδιου χρόνο καθώς και τον Ιούλιο του 2013 εμφανίστηκαν οι εκδόσεις 4.2 και 4.3 αντίστοιχα. Τα κύρια χαρακτηριστικά της 4.2 ήταν η υποστήριξη πολλαπλών χρηστών για tablet, προφυλάξεις οθόνης και widgets στην οθόνη κλειδώματος. Η 4.3 περιείχε βελτιώσεις και ενημερώσεις στην υποκείμενη πλατφόρμα Android.

1.5.10 Android 4.4 – 4.4.4 Kit Kat (API Level 19 – 20)

Η επόμενη έκδοση, που κυκλοφόρησε τον Σεπτέμβριο του 2013, ήταν το Kit Kat το οποίο επικεντρώθηκε κυρίως σε βελτιστοποιήσεις του λειτουργικού συστήματος, όπως επίσης και σε βελτίωση των επιδόσεων σε entry – level συσκευές που διαθέτουν περιορισμένους πόρους. Υποστηρίζεται η επιλογή Εκτύπωσης, προσφέρει δυνατότητα καλύτερης εστίασης στην κάμερα και βελτίωση στο Wi-Fi και το GPS.

1.5.11 Android 5.0 – 5.1.1 Lollipop (API Level 21 – 22)

Το Νοέμβριο του 2014 κυκλοφορεί η έκδοση Lollipop, η οποία εκτός από τις αλλαγές στο User Interface έριξε το βάρος της και στην ασφάλεια. Επίσης έγιναν αλλαγές στην πλατφόρμα, με το Android Runtime (ART) να είναι επίσημο και να αντικαθιστά το Dalvik για βελτίωση στην απόδοση των εφαρμογών, καθώς και αλλαγές για την βελτιστοποίηση της χρήσης της μπαταρίας. Τέλος εμφανίζεται και η υπηρεσία Google TV.

1.5.12 Android 6.0 – 6.0.1 Marshmallow (API Level 23)

Το Οκτώβριο του 2015 κυκλοφορεί η έκδοση Marshmallow όπου είναι η έκτη κύρια έκδοση του Android. Η παρουσίαση της έγινε από τον Μάιο του 2015 αλλά η επίσημη κυκλοφορία ήταν τον Οκτώβριο του ίδιου έτους. Παρουσιάστηκε ένα νέο σύστημα διαχείρισης ενέργειας όπου όταν μια συσκευή είναι σε αδράνεια τότε μειώνει την δραστηριότητα παρασκηνίου. Επίσης χρησιμοποιείται για πρώτη φορά το Now On Tap που παρέχει το πλαίσιο με τα ευαίσθητα αποτελέσματα αναζήτησης. Τέλος παρέχει την δυνατότητα μεταφοράς των δεδομένων και των εφαρμογών σε εξωτερική κάρτα microSD.

1.5.13 Android 7.0 – 7.1.2 Nougat (API Level 24)

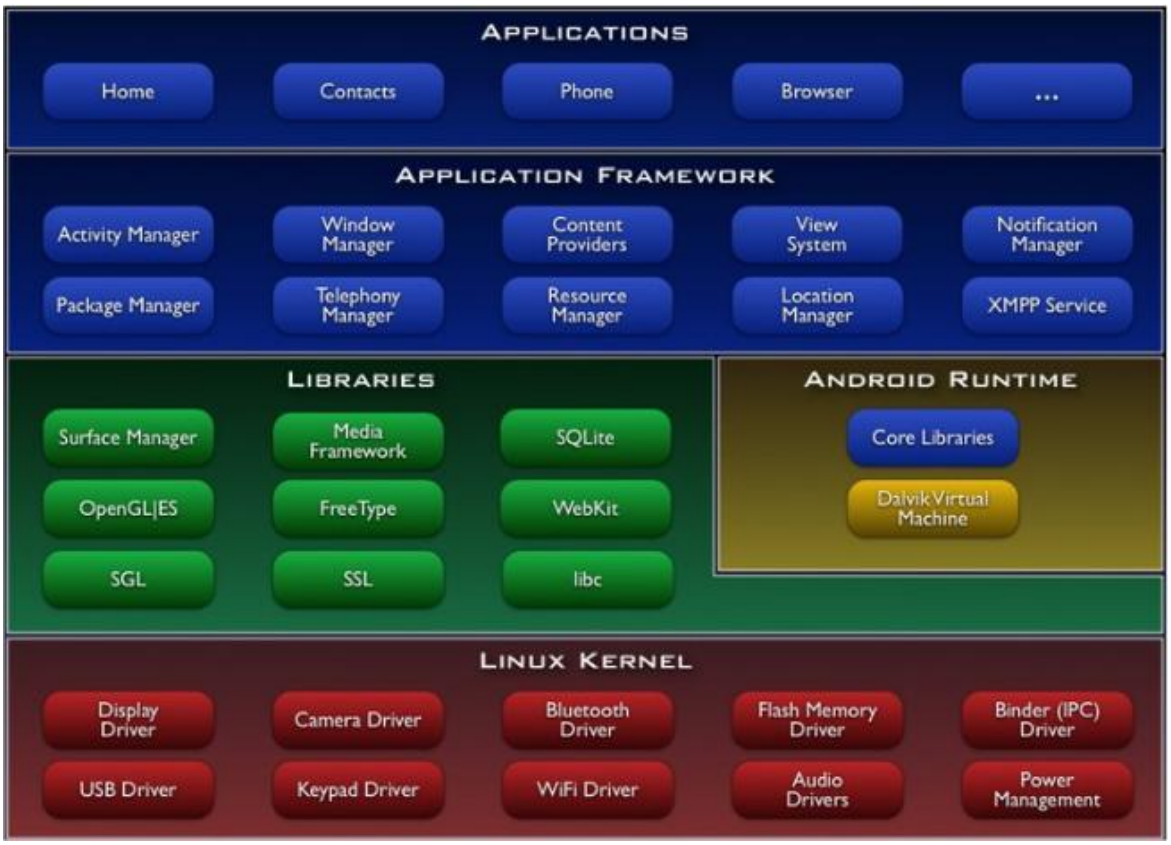
Τον Αύγουστο του 2016 κυκλοφόρησε η έκδοση Nougat με την κωδική ονομασία Android N. Εισήγαγε μερικές σημαντικές αλλαγές στο λειτουργικό σύστημα και στην πλατφόρμα ανάπτυξης του. Μια νέα δυνατότητα είναι η εμφάνιση πολλαπλών εφαρμογών στην οθόνη με προβολή διαίρεσης της οθόνης. Παρείχε περιβάλλον Java βασισμένο στο OpenJDK και υποστήριξη απόδοσης γραφικών Vulkan API.

1.5.14 Android 8.0 – 8.1 Oreo (API Level 26)

Τον Αύγουστο του 2017 εμφανίστηκε η 8^{η} και τελευταία μέχρι στιγμής έκδοση του λειτουργικού Android. Εμφάνισε αλλαγές στο User Interface, όπως και στο λειτουργικό σύστημα. Η παρουσίαση του έγινε κοντά στο εργοστάσιο Nabisco που δημιούργησε το πρώτο μπισκότο Oreo.

1.6 Αρχιτεκτονική Android

Στο ακόλουθο διάγραμμα φαίνεται η αρχιτεκτονική του Android[3, 8].



Εικόνα 1-3: Αρχιτεκτονική Android

Τα 5 βασικά επίπεδα που φαίνονται και παραπάνω είναι αυτά που αποτελούν και την αρχιτεκτονική του Android.

Ξεκινώντας από το υψηλότερο, συναντάμε το επίπεδο των εφαρμογών (Applications), το οποίο αποτελείται από βασικές εφαρμογές. Αυτά είναι η επαφές, ο περιηγητής ιστού (browser), πρόγραμμα στα αποστολή SMS, χάρτες κ.α. Οι εφαρμογές αυτές είναι γραμμένες σε κώδικα Java.

Το επίπεδο που ακολουθεί είναι το Επίπεδο Πλαισίου Εφαρμογών (Applications Framework) το οποίο δίνει την δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν το hardware της συσκευής για να έχουν πρόσβαση στην κάμερα, είτε στις υπηρεσίες εντοπισμού θέσης ή ακόμα να εμφανίσουν ειδοποιήσεις. Τέλος επιτρέπει την διαχείριση των πόρων της συσκευής είτε τον ορισμό του κύκλου ζωής των εφαρμογών.

Το Επίπεδο Βιβλιοθηκών (Libraries) που είναι και το επόμενο, περιλαμβάνει βιβλιοθήκες οι οποίες είναι γραμμένες σε γλώσσα C και προσφέρουν στους προγραμματιστές προσβασιμότητα μέσου του επιπέδου πλαισίου εφαρμογών.

Το Επίπεδο Εκτέλεσης (Android Runtime) αποτελείται από ένα σύνολο από βασικές βιβλιοθήκες και το Dalvik Virtual Machine.

Τέλος το χαμηλότερο επίπεδο είναι ο πυρήνας του Linux (Linux Kernel) ο οποίος διαχειρίζεται την μνήμη, τις διεργασίες την ασφάλεια. Λειτουργεί ως ένα ενδιάμεσο επίπεδο αφαίρεσης μεταξύ της στοίβας λογισμικού και υλικού.

2. Περιβάλλον ανάπτυξης Android Studio

Για την υλοποίηση της εφαρμογής, χρησιμοποιήθηκε το περιβάλλον ανάπτυξης Android Studio. Το Android Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE – Integrated Development Environment) το οποίο είναι βασισμένο στο λογισμικό Intellij IDEA της JetBrains' και είναι σχεδιασμένο ώστε να γίνεται ανάπτυξη εφαρμογών Android. Το Android Studio ανακοινώθηκε τον Μάιο του 2013, ενώ η πρώτη σταθερή έκδοση εμφανίστηκε τον Δεκέμβριο του 2014. Μέχρι τότε η ανάπτυξη εφαρμογών Android γινόταν στο περιβάλλον Eclipse, κατεβάζοντας το αντίστοιχο Android SDK [6].

2.1 Χαρακτηριστικά Android Studio

Μερικά από τα βασικά χαρακτηριστικά του Android Studio, είναι τα ακόλουθα:

- Powerful Code Editing: Το οποίο είναι ένα έξυπνο σύστημα συμπλήρωσης του κώδικα ώστε να διευκολύνει τον χρήστη όταν πληκτρολογεί τον κώδικα.
- Διάφορες ευκολίες στην επεξεργασία, όπως όταν αλλάζεις το όνομα μιας κλάσης, τότε αλλάζει σε όλα τα σημεία που καλείται αυτή η κλάση αυτόματα.
- On the fly Code Analysis: Όπου κατά την διάρκεια της σύνταξης του κώδικα υπάρχει ειδοποίηση ότι αυτό που πληκτρολογείται είναι λάθος και προτείνει τι μπορεί να επιλεγεί σαν εναλλακτική.
- Για το Interface παρέχει την ευκολία του Drag n drop, όπου επιλέγοντας ποιο component θέλουμε να προσθέσουμε στην οθόνη μας (π.χ. TextView, ListView, Button, κα.), το μόνο που έχουμε να κάνουμε είναι να το επιλέξουμε και να το σύρουμε στην οθόνη μας, χωρίς να γράψουμε γραμμή στο αρχείο xml.
- Προσφέρει την δυνατότητα για αποσφαλμάτωση (debug) είτε σε εικονική συσκευή, είτε σε πραγματική συσκευή.

2.2 Android Debug Bridge (ADB)

Για την σωστή λειτουργία των εφαρμογών μας, το περιβάλλον ανάπτυξης μας παρέχει μια λειτουργία, ώστε να συνδέει τον κώδικα που γράφουμε για την εφαρμογή μας με κάποια εικονική εφαρμογή, είτε με κάποια φυσική συσκευή

Android. Τον ρόλο αυτό τον αναλαμβάνει το Android Debug Bridge [4] το οποίο αποτελείται από τα ακόλουθα μέρη:

- Έναν client που εκτελείται στον υπολογιστή και μπορούμε να τον εκκινήσουμε είτε χειροκίνητα είτε αυτόματα (DDMS, ADT Plug-in)
- Έναν server, ο οποίος αναλαμβάνει την σωστή επικοινωνία μεταξύ του client και του 'δαίμονα' (daemon) που τρέχει στην συσκευή.
- Τον 'δαίμονα' (daemon) που εκτελείται είτε στην εικονική είτε στην φυσική συσκευή που χρησιμοποιείται για την εξομοίωση.

2.3 Εργαλείο Καταγραφής Συμβάντων (LogCat)

Το Android Studio προσφέρει το εργαλείο καταγραφής συμβάντων (LogCat) το οποίο είναι υπεύθυνο για τη συλλογή και την προβολή των αρχείων αποσφαλμάτωσης της εφαρμογής (debug). Αυτά τα στοιχεία συγκεντρώνονται σε μια σειρά από buffers, τα οποία φιλτράρονται και προβάλλονται με την εντολή LogCat.

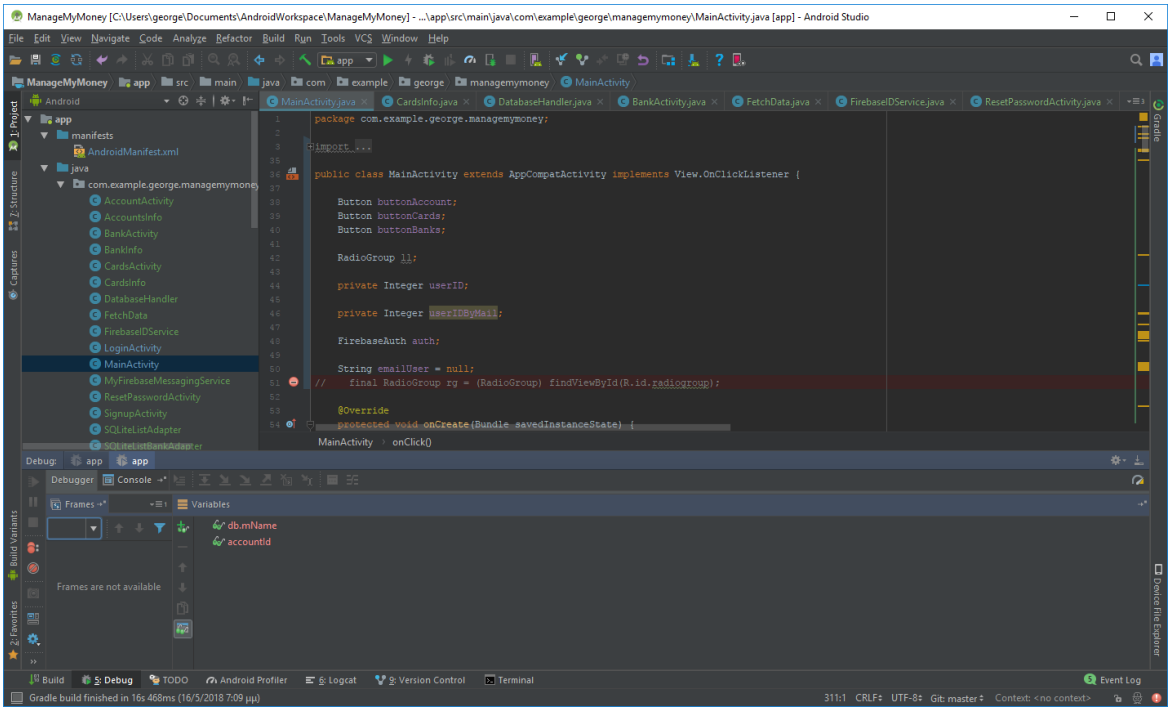
Οι προγραμματιστές, είθισται να χρησιμοποιούν εξαιρέσεις (exceptions) στον κώδικα τους, ώστε να προλαμβάνουν σημεία της εφαρμογής που κάτι δεν γίνεται σωστά. Όταν δεν έχει γίνει σωστός χειρισμός μιας εξαίρεσης τότε η εφαρμογή διακόπτει την λειτουργία της, με άμεσο τερματισμό της εφαρμογής και ένα μήνυμα λάθους 'Force Close'.

Έτσι το LogCat χρησιμοποιείται σε αυτό ακριβώς το σημείο μέσω του ADB για να μας ενημερώσει για τα debug logs της συσκευής που εκτελέσαμε την εφαρμογή. Βεβαίως, υπάρχουν και σφάλματα στην εφαρμογή που δεν προκαλούν αναγκαστικό κλείσιμο της, αλλά εμποδίζουν στην σωστή λειτουργία της εφαρμογής.

Στην περίπτωση τέτοιου σφάλματος χρησιμοποιούμε το σύστημα καταγραφής του Android, για να ενημερωθούμε για τα λάθη αυτά. Για τον λόγο αυτό υπάρχει η κλάση Log. Περιέχει διάφορες μεθόδους, ανάλογα με το λάθος που ψάχνουμε. Η πιο κοινή είναι η Log.d(), όπου d σημαίνει Debug.

2.4 Βασικά στοιχεία του Android Studio

Το κυρίως παράθυρο του Android Studio όπως φαίνεται στην παρακάτω εικόνα περιλαμβάνει τα ακόλουθα βασικά στοιχεία [6]:

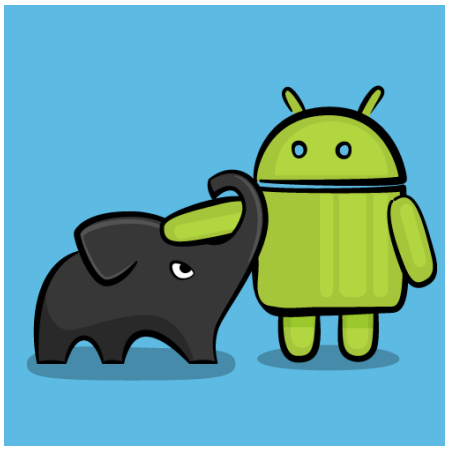


Εικόνα 2-1: Οθόνη Android Studio

- Η γραμμή εργαλείων (toolbar) περιλαμβάνει πολλές επιλογές, όπως να αποθηκεύσουμε ή να ανοίξουμε ένα project, αλλά και να το εκτελέσουμε ή να το κάνουμε debug.
- Η γραμμή πλοήγησης (navigation bar) περιλαμβάνει την διαδρομή (path) της εφαρμογής μας και δίνει την δυνατότητα περιήγησης σε αυτή.
- Το παράθυρο του προγράμματος επεξεργασίας (editor window) είναι το σημείο όπου γίνεται η ανάπτυξη του κώδικα. Εκεί ανάλογα με το αρχείο που δημιουργείται ή ανοίγει κάθε φορά, παίρνει την αντίστοιχη μορφή. Για παράδειγμα όταν είναι αρχείο java, τότε εμφανίζει απλά τον κώδικα όπως φαίνεται στην εικόνα 2-1. Όταν είναι αρχείο οθόνης (.xml) τότε εμφανίζει το σχεδιαστικό κομμάτι της οθόνης όπου μπορούν να προστεθούν τα διάφορα components.
- Τα παράθυρα εργαλείων (tool bar) δίνουν πρόσβαση σε συγκεκριμένες εργασίες όπως την διαχείριση του project, αναζήτηση, έλεγχο εκδόσεων και άλλα.
- Η γραμμή κατάστασης (status bar) εμφανίζει την κατάσταση του έργου και του IDE, καθώς και ειδοποιήσεις ή μηνύματα.

2.5 Gradle

Το Gradle [10], είναι ένα σύστημα το οποίο χτίζει αυτόματα την εφαρμογή, κατεβάζοντας όλα τα απαραίτητα αρχεία, αλλά και βιβλιοθήκες που απαιτούνται ώστε να μπορεί να λειτουργήσει σωστά η εφαρμογή. Το Gradle έκανε την εμφάνιση του το 2013, μαζί με το Android Studio. Η προσθήκη αυτή βοήθησε πολύ στον προγραμματισμό μιας εφαρμογής, αφού αυτοματοποίησε ένα σημαντικό κομμάτι, το οποίο είναι η προσθήκη βιβλιοθηκών που απαιτούνται σε μια εφαρμογή. Τέτοια λειτουργικότητα δεν υπήρχε στο περιβάλλον Eclipse, που ήταν ο προκάτοχος του Android Studio.



Εικόνα 2-2: Gradle και Android

Τα αρχεία του Gradle στην κάθε εφαρμογή είναι δύο. Το ένα είναι για τους ορισμούς στο επίπεδο της εφαρμογής (Project-level build.gradle), ενώ το δεύτερο είναι για τις βιβλιοθήκες που χρησιμοποιούνται (Module-level build.gradle).

Παρακάτω βλέπουμε το αρχείο build.gradle(Module: app) της εφαρμογής μας.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.george.managemymoney"
        minSdkVersion 14
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
```

```
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.0.2'
    implementation 'com.google.android.gms:play-services-
maps:15.0.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation
'com.android.support.test:runner:1.0.1'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.1'
    //this added for database...
    implementation 'android.arch.lifecycle:extensions:1.0.0-rc1'
    implementation 'android.arch.persistence.room:runtime:1.0.0-
rc1'
    annotationProcessor 'android.arch.lifecycle:compiler:1.0.0-
rc1'
    annotationProcessor
'android.arch.persistence.room:compiler:1.0.0-rc1'
    //firebase setup
    implementation 'com.google.firebase:firebase-core:15.0.2'
    implementation 'com.google.firebase:firebase-messaging:15.0.2'
    //authentication
    implementation 'com.google.firebase:firebase-auth:15.1.0'
    implementation 'com.google.firebase:firebase-database:15.0.0'
    implementation 'com.android.volley:volley:1.0.0'
}
//firebase setup
apply plugin: 'com.google.gms.google-services'
```

3. Απαιτήσεις εφαρμογής και τεχνολογίες

Ο σκοπός της εφαρμογής αυτής, είναι να παρέχει πληροφορίες για τις κινήσεις που κάνει ο χρήστης, στους λογαριασμούς που διαθέτει σε διάφορες τράπεζες. Για την επίτευξη αυτού του στόχου, είναι απαραίτητη η συνεργασία με τις τράπεζες, για την άντληση των δεδομένων των κινήσεων των χρηστών. Επειδή η εφαρμογή είναι μόνο πληροφοριακή και δεν μπορεί να κάνει τραπεζικές κινήσεις, όπως πληρωμές και μεταφορές δεν μπορεί να χαρακτηριστεί επικίνδυνη, αφού δεν περιέχει ευαίσθητα προσωπικά δεδομένα.

Οπότε έχοντας αυτή την πληροφορία, όπου θα προέρχεται από έναν Server, η εφαρμογή (Manage My Money) θα μπορεί να προσφέρει την ενημέρωση στους χρήστες της.

Για την εφαρμογή αυτή χρειάζεται ένα Log In για να μπαίνει ο χρήστης και να μπορεί να βλέπει τα δικά του στοιχεία. Επίσης τα δεδομένα όλα που χρειάζονται θα έρχονται με την μορφή Json από το API του Server. Τα δεδομένα αυτά θα σώζονται τοπικά σε μία βάση SQLite, όπου συντηρείται μέσα στην εφαρμογή. Στην συνέχεια και αφού έχει ήδη συνδεθεί ο χρήστης και έχουμε τα δεδομένα του μέσα στην τοπική βάση δεδομένων, θα έχει την επιλογή ο χρήστης ώστε να πάρει τις πληροφορίες που θέλει για τις κινήσεις των λογαριασμών του, τις κάρτες που διατηρεί, τις τράπεζες που υποστηρίζει η εφαρμογή, αλλά και έναν χάρτη όπου θα εμφανίζεται η θέση του, αλλά και τα κοντινά καταστήματα ή ATM σε περίπτωση που τα χρειαστεί.

Φυσικά όταν υπάρχει κάποια νέα κίνηση, ο χρήστης θα ειδοποιείται μέσω ειδοποίησης της εφαρμογής (με push notification) στο κινητό του.

Όλα τα παραπάνω που αναφέρθηκαν χρησιμοποιούν διάφορες τεχνολογίες όπου θα αναλυθούν πιο κάτω.

3.1 Firebase Real-time Database

Το Firebase [11] (σχεδιασμένο από την Google) είναι μια Cloud cross-platform NoSQL βάση δεδομένων που αποθηκεύει τα δεδομένα σε JSON μορφή. Οι αλλαγές που γίνονται στα δεδομένα της βάσης ενημερώνουν τους χρήστες σε πραγματικό χρόνο. Επίσης οι εφαρμογές που χρησιμοποιούν το Firebase έχουν την δυνατότητα για πρόσβαση στα δεδομένα όταν βρίσκονται σε offline

κατάσταση. Με την επανασύνδεση στο δίκτυο οι εφαρμογές ενημερώνονται αυτόματα με τις τυχών αλλαγές που υπήρξαν αυτό το διάστημα. Είναι σχεδιασμένο ώστε να υποστηρίζει Android, iOS και Web. Τέλος μέσω του Rest API μπορεί να υποστηρίξει και desktop εφαρμογές με γλώσσες Java, Python κ.α.

Μερικές καινούργιες λειτουργίες που παρέχει είναι το Authentication (ταυτοποίηση) του χρήστη. Έτσι βοηθάει μια εφαρμογή να αποθηκεύει με ασφάλεια τα δεδομένα του χρήστη στο Cloud (σύννεφο). Υποστηρίζει έλεγχο ταυτοποίησης με το Google account, το Facebook, το Twitter κ.α. Η πιστοποίηση του χρήστη βασίζεται στα πλέον σύγχρονα πρότυπα ασφάλειας, όπως το Auth 2.0 και το OpenID Connect, ώστε να μπορεί εύκολα να ενσωματωθεί εύκολα σε ένα custom backend.

Επίσης παρέχει λειτουργίες όπως Αποστολή Ειδοποιήσεων (notifications), Hosting, πληροφόρηση για σκασίματα της εφαρμογής (Crashlytics) και αρκετά άλλα.

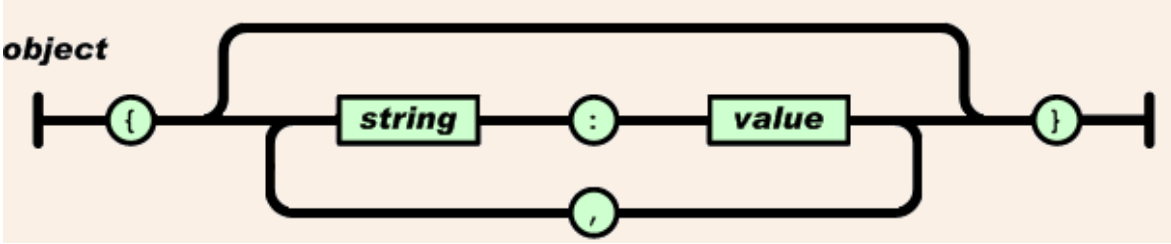
3.2 JSON

Το JSON [12] (JavaScript Object Notation) είναι ένα πρότυπο ανταλλαγής δεδομένων. Το χαρακτηριστικό του είναι ότι είναι πολύ απλό στο να διαβαστεί και να γραφτεί. Οι μηχανές μπορούν εύκολα να το αναλύσουν (parse) αλλά και να το παράγουν (generate). Είναι βασισμένο πάνω στην γλώσσα προγραμματισμού JavaScript. Το πρότυπο JSON είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί αρκετές πρακτικές που είναι γνωστές από τις γλώσσες C, C#, Java, Python αλλά και άλλες.

Το JSON βασίζεται στις ακόλουθες δομές:

- Μια συλλογή από ζευγάρια ονομάτων / τιμών.
- Μια ταξινομημένη λίστα τιμών.

Μια απλή μορφή JSON είναι η ακόλουθη που φαίνεται στην εικόνα 3-1:



Εικόνα 3-1: Απλή μορφή JSON

Ένα αντικείμενο (object) είναι ένα σύνολο από ονόματα / τιμές. Ξεκινάει με αριστερό άγκιστρο ({) και τελειώνει με δεξιό άγκιστρο (}). Κάθε όνομα ακολουθείται από άνω κάτω τελεία (:) και κάθε ζευγάρι ονόματος / τιμής ακολουθείται από ένα κόμμα (,) .

3.3 SQLite

Η SQLite [13], είναι μια ενσωματωμένη μηχανή SQL βάσης δεδομένων. Σε αντίθεση με άλλες βάσεις δεδομένων, η SQLite δεν έχει ξεχωριστό Server. Το βασικό της χαρακτηριστικό είναι ότι γράφει και διαβάζει σε απλά αρχεία δίσκου. Όπως και οι βασικές βάσεις δεδομένων, περιέχει πίνακες, triggers και views.

Η SQLite είναι μια συμπαγείς βιβλιοθήκη. Με όλες τις δυνατότητες της ενεργοποιημένες, το μέγεθος της μπορεί να φτάσει μέχρι τα 500 kilobyte ανάλογα με την πλατφόρμα και τις ρυθμίσεις του μεταγλωττιστή (Compiler). Η SQLite τρέχει πιο γρήγορα όσο περισσότερη μνήμη της δίνεται. Παρόλα αυτά, είναι αρκετά αποδοτική και σε συστήματα με αρκετά χαμηλότερη μνήμη.

Σε κάθε της έκδοση ελέγχεται πολύ προσεκτικά, και έχει την φήμη ότι είναι πολύ αξιόπιστη. Σχεδόν όλος ο πηγαίος κώδικας της SQLite βασίζεται στην δοκιμή και την επαναχρησιμοποίηση.

Είναι η πιο δημοφιλής ενσωματωμένη βάση δεδομένων για τοπική αποθήκευση αλλά και σε χρήση από φυλλομετρητές (browsers). Χρησιμοποιείται επίσης από λειτουργικά συστήματα και από ενσωματωμένα συστήματα, καθώς επίσης έχει συνδέσεις με πολλές γλώσσες προγραμματισμού.

3.4 Google Maps API

Το Google Maps API [9] δημιουργήθηκε από την Google, έτσι ώστε να δίνει την δυνατότητα να ενσωματώνονται στις εφαρμογές που χρειάζεται, οι χάρτες της Google. Με την βοήθεια αυτού του API, δίνεται πλέον η δυνατότητα να περιλαμβάνονται σε μια εφαρμογή όλες σχεδόν οι δυνατότητες του Google Maps, όπως το Street View, το Satellite View, ο υπολογισμός αποστάσεων, η κίνηση όπου αυτό υποστηρίζεται και πολλά άλλα.

Επίσης δίνει την δυνατότητα εκτός από τα διάφορα σημεία ενδιαφέροντος που παρέχει η ίδια η εφαρμογή των χαρτών, να προστίθενται και διάφορα σημεία που τα ορίζει ο προγραμματιστής. Φυσικά όλες αυτές οι λειτουργίες, ενεργοποιούνται από τις διάφορες μεθόδους που παρέχει το API.

Πλέον, πολλές εφαρμογές έχουν σαν βασικό χαρακτηριστικό την χρήση των χαρτών, οπότε είναι ένα εργαλείο πολύ χρήσιμο, το οποίο δίνει τροφή για την υλοποίηση νέων ιδεών.

4. Βασική Δομή και Υλοποίηση εφαρμογής

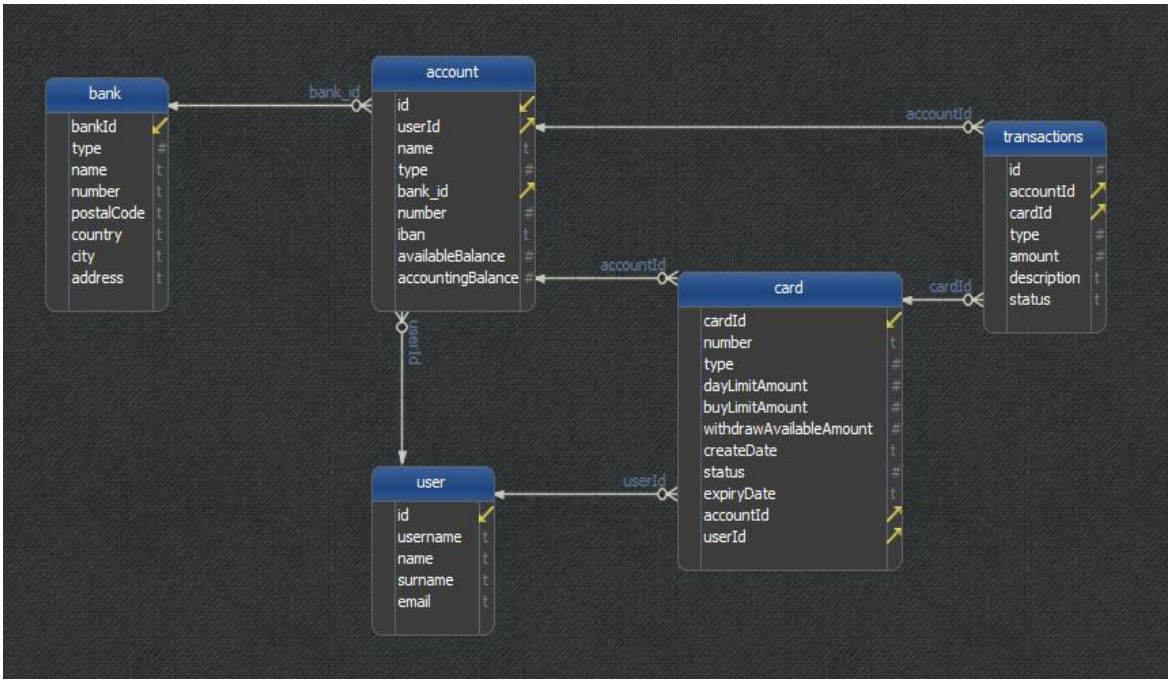
Στο κεφάλαιο αυτό αναπτύσσεται ο τρόπος που υλοποιήθηκε η εφαρμογή και η δομή του συστήματος μας.

4.1 Η βάση δεδομένων (SQLite)

Για τις ανάγκες της εφαρμογής, χρησιμοποιήθηκαν οι ακόλουθοι 5 πίνακες. Ο πίνακας των Χρηστών (Users), όπου περιλαμβάνει τις πληροφορίες του χρήστη, όπως το ονοματεπώνυμο, και το email του που είναι απαραίτητο για την διαδικασία της σύνδεσης στην εφαρμογή (Login). Ο πίνακας των λογαριασμών, όπου περιέχει την πληροφορία για τα στοιχεία των λογαριασμών που ανήκουν στον χρήστη, αλλά και το διαθέσιμο ποσό του κάθε λογαριασμού.

Ένας σημαντικός ακόμα πίνακας είναι ο πίνακας των κινήσεων, όπου περιέχει το είδος της κίνησης (π.χ. Ανάληψη, Κατάθεση, Πληρωμή κ.α.). Οι επόμενοι δύο πίνακες είναι ο πίνακας των Καρτών και των Τραπεζών, όπου περιέχει τα στοιχεία από τις κάρτες του χρήστη, αλλά και τις τράπεζες που παρέχουν τα στοιχεία τους στην εφαρμογή.

Το σχήμα της βάσης μας, φαίνεται και στην παρακάτω εικόνα.



Εικόνα 4-1: Το σχήμα της SQLite βάσης

Για τον σχεδιασμό της βάσης μας στην εφαρμογή, προστέθηκε η κλάση DatabaseHandler.java, όπου περιλαμβάνει την δημιουργία των πινάκων, αλλά και τις μεθόδους που χρειάζονται ώστε να προστίθενται τα δεδομένα στους πίνακες, αλλά και για να αντλούμε τα δεδομένα που χρειαζόμαστε κάθε φορά από τους πίνακες μας.

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {

    //Create user table

    String CREATE_USER_TABLE = "CREATE TABLE " + TABLE_USER + "("
    + USER_KEY_ID + " INTEGER PRIMARY KEY, " + USER_KEY_USERNAME +
" TEXT, "
    + USER_NAME + " TEXT, " + USER_SURNAME + " TEXT, "
    + USER_EMAIL + " TEXT " + ")";

    sqLiteDatabase.execSQL(CREATE_USER_TABLE);

    String CREATE_BANK_TABLE = "CREATE TABLE " + TABLE_BANK + " ("
            + BANK_KEY_ID + " INTEGER PRIMARY KEY, "
            + BANK_TYPE + " INTEGER, "
            + BANK_NAME + " TEXT, "
            + BANK_NUMBER + " TEXT, "
            + BANK_POSTAL_CODE + " TEXT, "
            + BANK_COUNTRY + " TEXT, "
            + BANK_CITY + " TEXT, "
            + BANK_ADDRESS + " TEXT " + ")";

    sqLiteDatabase.execSQL(CREATE_BANK_TABLE);

    String CREATE_ACCOUNT_TABLE = "CREATE TABLE " + TABLE_ACCOUNT
+ " ("
            + ACCOUNT_KEY_ID + " INTEGER PRIMARY KEY, "
            + ACCOUNT_USER_ID + " INTEGER, "
            + ACCOUNT_NAME + " TEXT, "
            + ACCOUNT_TYPE + " INTEGER, "
            + ACCOUNT_BANK_ID + " INTEGER, "
            + ACCOUNT_NUMBER + " INTEGER, "
            + ACCOUNT_IBAN + " TEXT, "
            + ACCOUNT_AVAILABLE_BALANCE + " REAL, "
            + ACCOUNT_ACCOUNTING_BALANCE + " REAL, "

            + "FOREIGN KEY(" + ACCOUNT_USER_ID + ") REFERENCES " +
TABLE_USER + "(" + USER_KEY_ID + "), "
            + "FOREIGN KEY(" + ACCOUNT_BANK_ID + ") REFERENCES " +
TABLE_BANK + "(" + BANK_KEY_ID + "))";

    sqLiteDatabase.execSQL(CREATE_ACCOUNT_TABLE);

    String CREATE_CARD_TABLE = "CREATE TABLE " + TABLE_CARD + " ("
            + CARD_KEY_ID + " INTEGER PRIMARY KEY, "
            + CARD_NUMBER + " TEXT, "
```

```
            + CARD_TYPE + " INTEGER, "
            + CARD_DAY_LIMIT_AMOUNT + " REAL, "
            + CARD_BUY_LIMIT_AMOUNT + " REAL, "
            + CARD_WITHDRAW_AVAILABLE_AMOUNT + " REAL, "
            + CARD_CREATE_DATE + " TEXT, "
            + CARD_STATUS + " INTEGER, "
            + CARD_EXPIRY_DATE + " TEXT, "
            + CARD_ACCOUNT_ID + " INTEGER, "
            + CARD_USER_ID + " INTEGER, "

            + "FOREIGN KEY(" + CARD_USER_ID + ") REFERENCES " +
TABLE_USER + "(" + USER_KEY_ID + "), "
            + "FOREIGN KEY(" + CARD_ACCOUNT_ID + ") REFERENCES " +
TABLE_ACCOUNT + "(" + ACCOUNT_KEY_ID + "))";

    sqLiteDatabase.execSQL(CREATE_CARD_TABLE);

    String CREATE_TRANSACTION_TABLE = "CREATE TABLE " +
TABLE_TRANSACTION + " ("
            + TRANSACTION_KEY_ID + " INTEGER PRIMARY KEY, "
            + TRANSACTION_ACCOUNT + " INTEGER, "
            + TRANSACTION_CARD_ID + " INTEGER, "
            + TRANSACTION_TYPE + " INTEGER, "
            + TRANSACTION_AMOUNT + " REAL, "
            + TRANSACTION_DESC + " TEXT, "
            + TRANSACTION_STATUS + " TEXT, "
            + "FOREIGN KEY(" + TRANSACTION_ACCOUNT + ") REFERENCES
" + TABLE_ACCOUNT + "(" + ACCOUNT_KEY_ID + "), "
            + "FOREIGN KEY(" + TRANSACTION_CARD_ID + ") REFERENCES
" + TABLE_CARD + "(" + CARD_KEY_ID + "))";

    sqLiteDatabase.execSQL(CREATE_TRANSACTION_TABLE);
}
```

```
// Adding new User
public void addUser(User user) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(USER_KEY_USERNAME, user.getUsername()); // User
username
    values.put(USER_NAME, user.getName()); // User Name
    values.put(USER_SURNAME, user.getSurname()); // User Surname
    values.put(USER_EMAIL, user.getEmail()); // User Email
// Inserting Row
    db.insert(TABLE_USER, null, values);
    db.close(); // Closing database connection
}
// Adding new Account
public void addAccount(AccountsInfo account) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(ACCOUNT_USER_ID, account.getUserId()); // Account
User Id
    values.put(ACCOUNT_NAME, account.getName()); // Account Name
    values.put(ACCOUNT_IBAN, account.getIban()); // Iban
```

```
    values.put(ACCOUNT_AVAILABLE_BALANCE,
account.getAvailableBalance()); // Available Balance
    values.put(ACCOUNT_ACCOUNTING_BALANCE,
account.getAccountingBalance()); // Account Balance
    values.put(ACCOUNT_TYPE, account.getType()); // Account Type
    values.put(ACCOUNT_BANK_ID, account.getBankId()); // Account
Bank Id
    values.put(ACCOUNT_NUMBER, account.getNumber()); // Account
Number
    // Inserting Row
    db.insert(TABLE_ACCOUNT, null, values);
    db.close(); // Closing database connection
}

// Adding new Transaction
public void addTransaction(Transactions transaction) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(TRANSACTION_ACCOUNT, transaction.getAccountId());
// Transaction Account Id
    values.put(TRANSACTION_CARD_ID, transaction.getCardId()); //
Transaction Card Id
    values.put(TRANSACTION_TYPE, transaction.getType()); //
Transaction Type
    values.put(TRANSACTION_AMOUNT, transaction.getAmount()); //
Transaction Amount
    values.put(TRANSACTION_DESC, transaction.getDescription()); //
Transaction description
    values.put(TRANSACTION_STATUS, transaction.getStatus()); //
Transaction Status
    // Inserting Row
    db.insert(TABLE_TRANSACTION, null, values);
    db.close(); // Closing database connection
}

// Adding new Card
public void addCard(CardsInfo cardsInfo) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(CARD_NUMBER, cardsInfo.getCardNumber());
    values.put(CARD_TYPE, cardsInfo.getType());
    values.put(CARD_DAY_LIMIT_AMOUNT,
cardsInfo.getDayLimitAmount());
    values.put(CARD_BUY_LIMIT_AMOUNT, cardsInfo.getBuyLimit());
    values.put(CARD_WITHDRAW_AVAILABLE_AMOUNT,
cardsInfo.getWithdrawalAvailableAmount());
    values.put(CARD_CREATE_DATE,
cardsInfo.getDateCreate().toString());
    values.put(CARD_STATUS, cardsInfo.getStatus());
    values.put(CARD_EXPIRY_DATE,
cardsInfo.getExpiryDate().toString());
    values.put(CARD_ACCOUNT_ID, cardsInfo.getAccountId());
    values.put(CARD_USER_ID, cardsInfo.getUserId());
    // Inserting Row
    db.insert(TABLE_CARD, null, values);
    db.close(); // Closing database connection
```

```
}

// Adding new Bank
public void addBank(BankInfo bankInfo) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(BANK_TYPE, bankInfo.getType());
    values.put(BANK_NAME, bankInfo.getName());
    values.put(BANK_NUMBER, bankInfo.getNumber());
    values.put(BANK_POSTAL_CODE, bankInfo.getPostalCode());
    values.put(BANK_COUNTRY, bankInfo.getCountry());
    values.put(BANK_CITY, bankInfo.getCity());
    values.put(BANK_ADDRESS, bankInfo.getAddress());
    // Inserting Row
    db.insert(TABLE_BANK, null, values);
    db.close(); // Closing database connection
}
```

```
// Getting user
    public User getUser(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.query(TABLE_USER, new String[] {
USER_KEY_ID,
                        USER_KEY_USERNAME, USER_NAME,
USER_SURNAME, USER_EMAIL }, USER_KEY_ID + "=?",
                new String[] { String.valueOf(id) }, null, null,
null, null);
        if (cursor != null)
            cursor.moveToFirst();
        User user = new
User(Integer.parseInt(cursor.getString(0)),
                cursor.getString(1), cursor.getString(2),
cursor.getString(3), cursor.getString(4));
        // return shop
        return user;
    }

    public User getUserIdByEmail(String email) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.query(TABLE_USER, new String[] {
USER_KEY_ID,
                USER_KEY_USERNAME, USER_NAME, USER_SURNAME,
USER_EMAIL }, USER_EMAIL + "=?",
                new String[] { String.valueOf(email) }, null,
null, null, null);
        if (cursor != null)
            cursor.moveToFirst();
        if (cursor.getCount() > 0) {
            User user = new
User(Integer.parseInt(cursor.getString(0)),
                    cursor.getString(1), cursor.getString(2),
cursor.getString(3), cursor.getString(4));
            // return shop
```

```
            return user;
        }
        else
        {
            return null;
        }
    }
    // Getting Account
    public AccountsInfo getAccount(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.query(TABLE_ACCOUNT, new String[] {
ACCOUNT_KEY_ID, ACCOUNT_USER_ID,
                        ACCOUNT_NAME, ACCOUNT_TYPE,
ACCOUNT_BANK_ID, ACCOUNT_NUMBER, ACCOUNT_IBAN,
ACCOUNT_AVAILABLE_BALANCE, ACCOUNT_ACCOUNTING_BALANCE },
ACCOUNT_KEY_ID + "=?",
                new String[] { String.valueOf(id) }, null, null,
null, null);
        if (cursor != null)
            cursor.moveToFirst();
        AccountsInfo account = new
AccountsInfo(Integer.parseInt(cursor.getString(0)),
                Integer.parseInt(cursor.getString(1)),
cursor.getString(2), Integer.parseInt(cursor.getString(3)),
Integer.parseInt(cursor.getString(4)),
                Integer.parseInt(cursor.getString(5)),
cursor.getString(6), Double.parseDouble(cursor.getString(7)),
Double.parseDouble(cursor.getString(8)));
        // return shop
        return account;
    }

    // Getting Account by User_ID
    public List<AccountsInfo> getAccountByUser(int i) {
        List<AccountsInfo> accountList = new
ArrayList<AccountsInfo>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_ACCOUNT + "
WHERE " + ACCOUNT_USER_ID + " = ?";

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery,  new String[] {
String.valueOf(i) });

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                AccountsInfo account = new AccountsInfo();

account.setAccountId(Integer.parseInt(cursor.getString(0)));

account.setUserId(Integer.parseInt(cursor.getString(1)));
                account.setName(cursor.getString(2));

account.setType(Integer.parseInt(cursor.getString(3)));
```

```
account.setBankId(Integer.parseInt(cursor.getString(4)));

account.setNumber(Integer.parseInt(cursor.getString(5)));
                account.setIban(cursor.getString(6));

account.setAccountingBalance(Double.parseDouble(cursor.getString(7
)));

account.setAvailableBalance(Double.parseDouble(cursor.getString(8)
));
                // Adding contact to list
                accountList.add(account);
            } while (cursor.moveToNext());
        }
        // return contact list
        return accountList;
    }
    // Getting Account by Bank Account
    public int getAccountByBankAccount(int i) {
        int accountId = 0;
        // Select All Query
        String selectQuery = "SELECT " + ACCOUNT_KEY_ID + " FROM "
+ TABLE_ACCOUNT + " WHERE " + ACCOUNT_NUMBER + " = ?";

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery,  new String[] {
String.valueOf(i) });

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                AccountsInfo account = new AccountsInfo();

account.setAccountId(Integer.parseInt(cursor.getString(0)));
                // Adding contact to list
                accountId = Integer.parseInt(cursor.getString(0));
            } while (cursor.moveToNext());
        }

        // return contact list
        return accountId;
    }
    // Getting Account Number by Account Id
    public int getAccountNumberByAccountId(int i) {
        int accountNumber = 0;
        // Select All Query
        String selectQuery = "SELECT " + ACCOUNT_NUMBER + " FROM "
+ TABLE_ACCOUNT + " WHERE " + ACCOUNT_KEY_ID + " = ?";

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery,  new String[] {
String.valueOf(i) });

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
```

```
                AccountsInfo account = new AccountsInfo();

account.setNumber(Integer.parseInt(cursor.getString(0)));
                // Adding contact to list
                accountNumber =
Integer.parseInt(cursor.getString(0));
            } while (cursor.moveToNext());
        }

        // return contact list
        return accountNumber;
    }

    // Getting Transactions
    public Transactions getTransactions(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.query(TABLE_TRANSACTION, new String[] {
TRANSACTION_KEY_ID, TRANSACTION_ACCOUNT,
                        TRANSACTION_CARD_ID, TRANSACTION_TYPE,
TRANSACTION_AMOUNT, TRANSACTION_DESC, TRANSACTION_STATUS },
TRANSACTION_KEY_ID + "=?",
                new String[] { String.valueOf(id) }, null, null,
null, null);
        if (cursor != null)
            cursor.moveToFirst();
        Transactions trans = new
Transactions(Integer.parseInt(cursor.getString(0)),
                Integer.parseInt(cursor.getString(1)),
Integer.parseInt(cursor.getString(2)),
Integer.parseInt(cursor.getString(4)),
                Double.parseDouble(cursor.getString(5)),
cursor.getString(6), cursor.getString(7));
        // return shop
        return trans;
    }
    // Getting Account by User_ID
    public List<Transactions> getTransactionByUserAndAccount(int
user, int account) {
        List<Transactions> transactionsList = new
ArrayList<Transactions>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_TRANSACTION
+ " WHERE " + TRANSACTION_ACCOUNT + " = ?" ;

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery,  new String[] {
String.valueOf(account) });

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                Transactions transactions = new Transactions();

transactions.setAccountId(Integer.parseInt(cursor.getString(1)));

transactions.setCardId(Integer.parseInt(cursor.getString(2)));
```

```
transactions.setType(Integer.parseInt(cursor.getString(3)));

transactions.setAmount(Double.parseDouble(cursor.getString(4)));
                transactions.setDescription(cursor.getString(5));
                transactions.setStatus(cursor.getString(6));
                // Adding contact to list
                transactionsList.add(transactions);
            } while (cursor.moveToNext());
        }
        // return contact list
        return transactionsList;
    }
    // Getting Banks
    public List<BankInfo> getAllBanks() {
        List<BankInfo> bankInfoList = new ArrayList<BankInfo>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_BANK ;

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery,  new String[]
{});

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                BankInfo bankInfo = new BankInfo();

bankInfo.setType(Integer.parseInt(cursor.getString(1)));
                bankInfo.setName(cursor.getString(2));
                bankInfo.setNumber(cursor.getString(3));
                bankInfo.setPostalCode(cursor.getString(4));
                bankInfo.setCountry(cursor.getString(5));
                bankInfo.setCity(cursor.getString(6));
                bankInfo.setAddress(cursor.getString(7));

                // Adding contact to list
                bankInfoList.add(bankInfo);
            } while (cursor.moveToNext());
        }
        // return bank list
        return bankInfoList;
    }

    // Getting Banks
    public String getBankNameByID(int i) {
//        List<BankInfo> bankInfoList = new ArrayList<BankInfo>();
        // Select All Query
        String selectQuery = "SELECT " + BANK_NAME + " FROM " +
TABLE_BANK + " WHERE " + BANK_KEY_ID + " = ?";

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery,  new String[] {
String.valueOf(i) });

        String bankName = "";
```

```
        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                bankName = cursor.getString(0);

            } while (cursor.moveToNext());
        }
        // return bank list
        return bankName;
    }
    // Getting Cards by User_ID
    public List<CardsInfo> getCardsByUser(int i) {
        List<CardsInfo> cardsInfoList = new
ArrayList<CardsInfo>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_CARD + "
WHERE " + CARD_USER_ID + " = ?";

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery,  new String[] {
String.valueOf(i) });

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                CardsInfo cardsInfo = new CardsInfo();
                cardsInfo.setCardNumber(cursor.getString(1));

cardsInfo.setType(Integer.parseInt(cursor.getString(2)));

cardsInfo.setDayLimitAmount(Double.parseDouble(cursor.getString(3)
));

cardsInfo.setBuyLimit(Double.parseDouble(cursor.getString(4)));

cardsInfo.setWithdrawalAvailableAmount(Double.parseDouble(cursor.g
etString(5)));
                cardsInfo.setDateCreate(cursor.getString(6));

cardsInfo.setStatus(Integer.parseInt(cursor.getString(7)));
                cardsInfo.setExpiryDate(cursor.getString(8));

cardsInfo.setAccountId(Integer.parseInt(cursor.getString(9)));

cardsInfo.setUserId(Integer.parseInt(cursor.getString(10)));
                // Adding cards to list
                cardsInfoList.add(cardsInfo);
            } while (cursor.moveToNext());
        }
        // return cards list
        return cardsInfoList;
    }
```

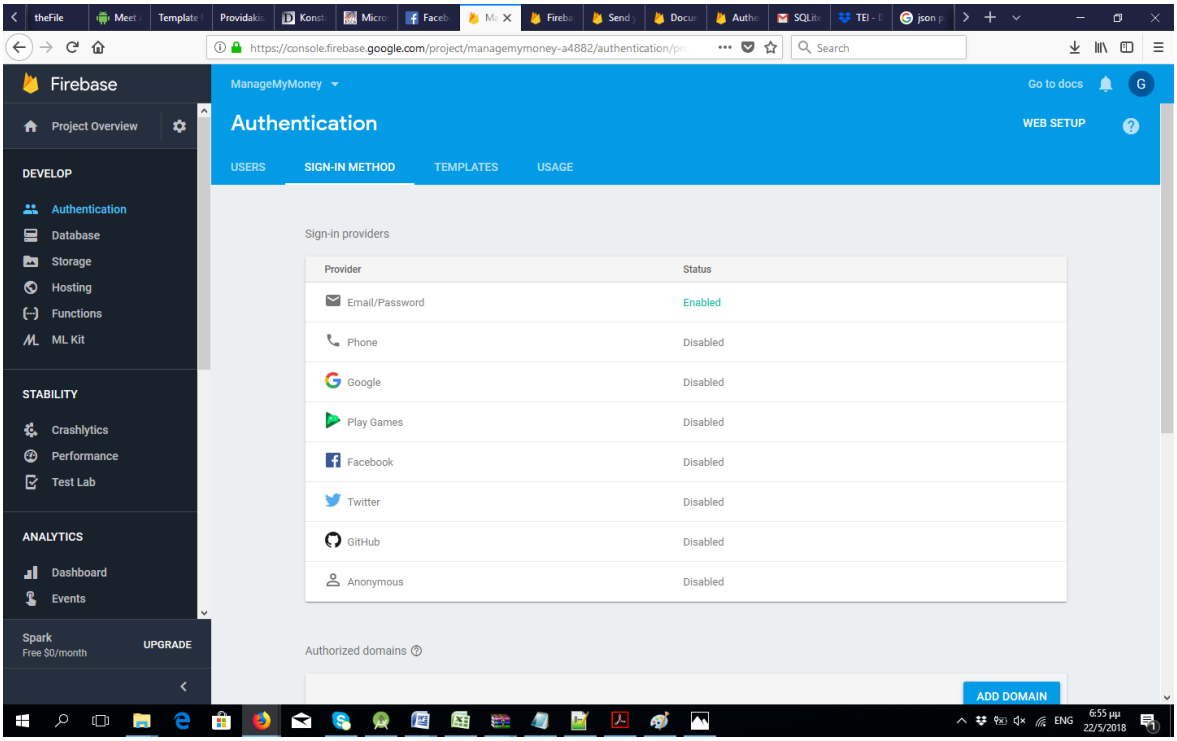
Κάθε πίνακας έχει και μια αντίστοιχη κλάση όπου περιλαμβάνει τα πεδία της βάσης μας. Στην πραγματικότητα, κάνει αντιστοίχιση του κώδικα Java με τον αντίστοιχο πίνακα από την βάση, ώστε να μπορέσει η εφαρμογή να χειριστεί τα στοιχεία του πίνακα. Έτσι για τον πίνακα USER έχουμε την User.java κλάση, για τον Transactions έχουμε Transactions.Java κλάση. Ενδεικτικά δίνεται ο κώδικας από την κλάση AccountsInfo, όπου αναφέρεται στον πίνακα Account. Αντίστοιχα είναι και για τους υπόλοιπους πίνακες.

```
@Entity
public class AccountsInfo {

    @PrimaryKey
    public Integer accountId;
    public Integer userId;
    public String name;
    public Integer type;
    public Integer bankId;
    public Integer number;
    public String iban;
    public Double availableBalance;
    public Double accountingBalance;

    public AccountsInfo() {
    }

    public AccountsInfo(Integer accountId, Integer userId, String
name, Integer type, Integer bankId, Integer number, String iban,
Double availableBalance, Double accountingBalance) {
        this.accountId = accountId;
        this.userId = userId;
        this.name = name;
        this.type = type;
        this.bankId = bankId;
        this.number = number;
        this.iban = iban;
        this.availableBalance = availableBalance;
        this.accountingBalance = accountingBalance;
    }

    public Integer getAccountId() {
        return accountId;
    }
    public void setAccountId(Integer accountId) {
        this.accountId = accountId;
    }
    public Integer getUserId() {
        return userId;
    }
    public void setUserId(Integer userId) {
        this.userId = userId;
```

```
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getType() {
        return type;
    }
    public void setType(Integer type) {
        this.type = type;
    }

    public Integer getBankId() {
        return bankId;
    }

    public void setBankId(Integer bankId) {
        this.bankId = bankId;
    }

    public Integer getNumber() {
        return number;
    }

    public void setNumber(Integer number) {
        this.number = number;
    }

    public String getIban() {
        return iban;
    }

    public void setIban(String iban) {
        this.iban = iban;
    }

    public Double getAvailableBalance() {
        return availableBalance;
    }

    public void setAvailableBalance(Double availableBalance) {
        this.availableBalance = availableBalance;
    }

    public Double getAccountingBalance() {
        return accountingBalance;
    }

    public void setAccountingBalance(Double accountingBalance) {
        this.accountingBalance = accountingBalance;
    }
}
```

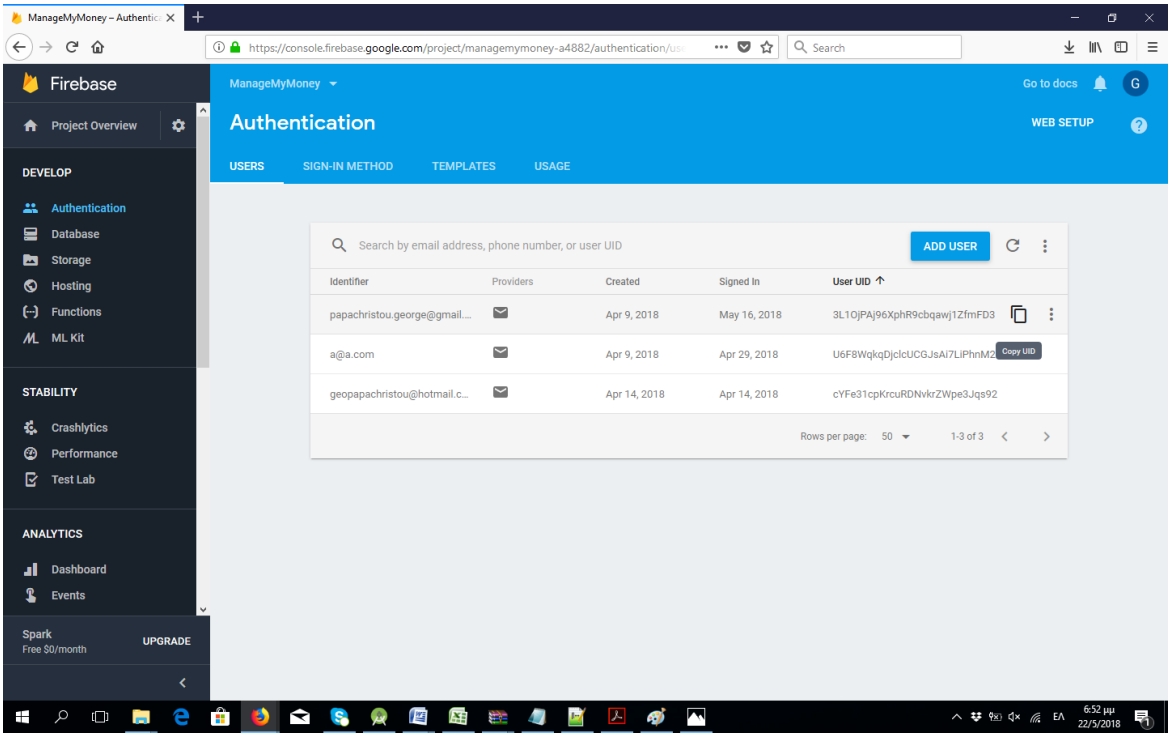
4.2 Firebase Authentication

Όπως αναφέραμε και στο προηγούμενο κεφάλαιο, η ταυτοποίηση του χρήστη έγινε με το Authentication σύστημα του Firebase. Αφού δημιουργήθηκε νέο project στο firebase και προστέθηκε μέσα στην εφαρμογή (ManageMyMoney), στο menu Authentication επιλέχθηκε να γίνεται sign-in με το Email/Password του χρήστη, όπως φαίνεται και στην εικόνα 4-2. Εδώ παρατηρούμε ότι μας δίνονται και οι επιλογές σύνδεσης μέσω Facebook, Twitter, λογαριασμού Google, κ.α.



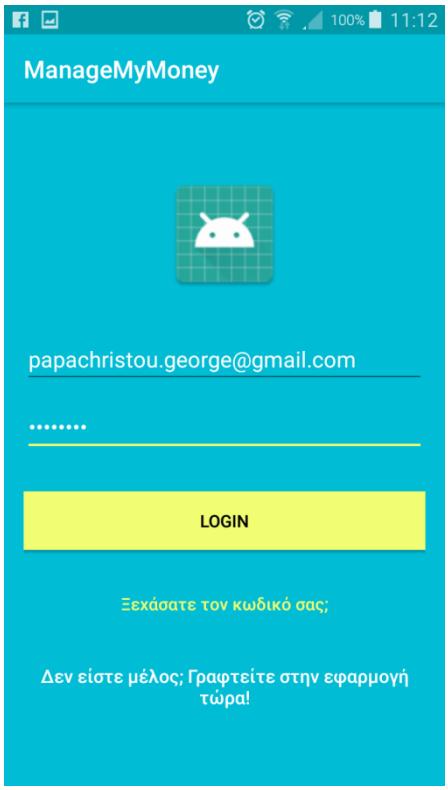
Εικόνα 4-2: Επιλογή μεθόδου Authentication

Στην συνέχεια στο tab Users, ορίζονται οι χρήστες όπου θα έχουν το δικαίωμα να συνδέονται στην συσκευή καθώς και ο κωδικός τους. Στο σημείο αυτό πρέπει να τονιστεί, ότι την διαδικασία αυτή θα την διαχειρίζεται η API εφαρμογή όπου θα μιλάει με την mobile εφαρμογή. Οπότε αφού ορίστηκαν και οι χρήστες που θα έχουν πρόσβαση, όπως φαίνεται και στην εικόνα 4-3, συνεχίζουμε με την ανάπτυξη του κώδικα για την σύνδεση της εφαρμογής με το firebase.



Εικόνα 4-3: Ορισμός χρηστών εφαρμογής

Η οθόνη του Login της εφαρμογής είναι η ακόλουθη όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 4-4):



Εικόνα 4-4: Οθόνη Login

Εδώ ο χρήστης θα έχει την δυνατότητα να επιλέγει το email του με το οποίο έχει γραφτεί στην εφαρμογή και τον κωδικό του, ώστε να συνδεθεί επιτυχώς και να μπορέσει να ενημερωθεί για τους λογαριασμούς του. Η οθόνη αυτή αντιστοιχεί στο αρχείο activity_login.xml του project.

Αρχικά στο αρχείο AndroidManifest.xml, προστέθηκαν οι εξής γραμμές για την σύνδεση με την firebase:

```
<service android:name=".FirebaseIDService">
    <intent-filter>
        <action
android:name="com.google.firebase.INSTANCE_ID_EVENT" />
    </intent-filter>
</service>
```

Επίσης πολύ σημαντικό, είναι το να ορίσουμε την πρόσβαση στο δίκτυο με το permission:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Ο κώδικας του LoginActivity.java, όπου κάνει τους απαραίτητους ελέγχους ώστε είτε να συνεχίσει επιτυχώς είτε να ενημερώσει τον χρήστη με μήνυμα αποτυχίας φαίνεται παρακάτω:

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //Get Firebase auth instance
        auth = FirebaseAuth.getInstance();

        if (auth.getCurrentUser() != null) {
            startActivity(new Intent(LoginActivity.this,
MainActivity.class));
            finish();
        }
        // set the view now
        setContentView(R.layout.activity_login);

        inputEmail = (EditText) findViewById(R.id.email);
        inputPassword = (EditText) findViewById(R.id.password);
        progressBar = (ProgressBar)
findViewById(R.id.progressBar);
        btnSignup = (Button) findViewById(R.id.btn_signup);
        btnLogin = (Button) findViewById(R.id.btn_login);
        btnReset = (Button) findViewById(R.id.btn_reset_password);

        //Get Firebase auth instance
```

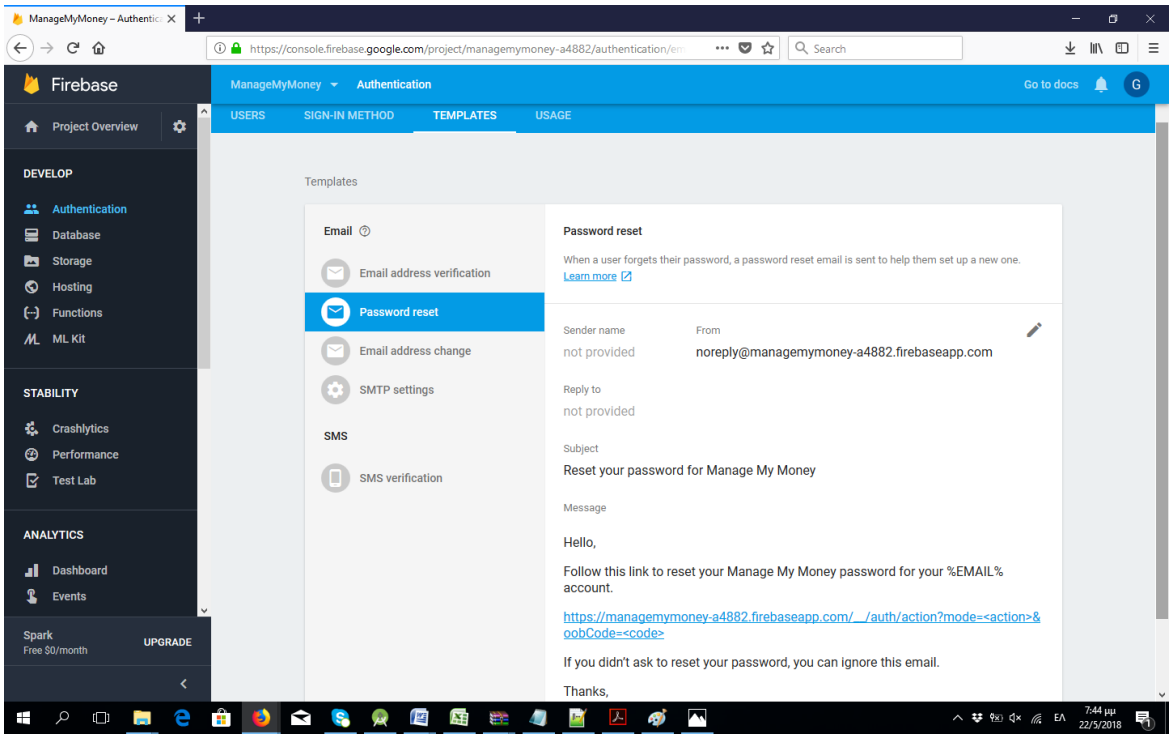
```
        auth = FirebaseAuth.getInstance();

        btnSignup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(LoginActivity.this,
SignupActivity.class);
                startActivity(new Intent(intent));
            }
        });

        btnReset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent (LoginActivity.this,
ResetPasswordActivity.class));
            }
        });

        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = inputEmail.getText().toString();
                final String password =
inputPassword.getText().toString();

                if (TextUtils.isEmpty(email)) {
                    Toast.makeText(getApplicationContext(),
"Εισάγετε email!", Toast.LENGTH_SHORT).show();
                    return;
                }

                if (TextUtils.isEmpty(password)) {
                    Toast.makeText(getApplicationContext(),
"Εισάγετε Κεδικό!", Toast.LENGTH_SHORT).show();
                    return;
                }

                progressBar.setVisibility(View.VISIBLE);

                //authenticate user
                auth.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(LoginActivity.this,
new OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull
Task<AuthResult> task) {

progressBar.setVisibility(View.GONE);
                                if (!task.isSuccessful()) {
                                    // there was an error
                                    if (password.length() < 6) {

inputPassword.setError(getString(R.string.minimum_password));
                                    } else {
```

```
Toast.makeText(LoginActivity.this,
getString(R.string.auth_failed), Toast.LENGTH_LONG).show();
                                    }
                                } else {
                                    Intent intent = new
Intent(LoginActivity.this, MainActivity.class);
                                    startActivity(intent);
                                    finish();
                                }
                            }
                        });
                }
            });
    }
```

Στην συνέχεια δίνεται η δυνατότητα να επαναφερθεί ο κωδικός του χρήστη σε περίπτωση που τον έχει ξεχάσει. Αυτό γίνεται σε συνδυασμό με την firebase όπου με την βοήθεια της μεθόδου sendPasswordResetEmail ενεργοποιεί την υπηρεσία του firebase, όπου στέλνει ένα link στο email του χρήστη όπου μπορεί να το πατήσει για να βάλει καινούργιο κωδικό (Εικόνα 4-5).

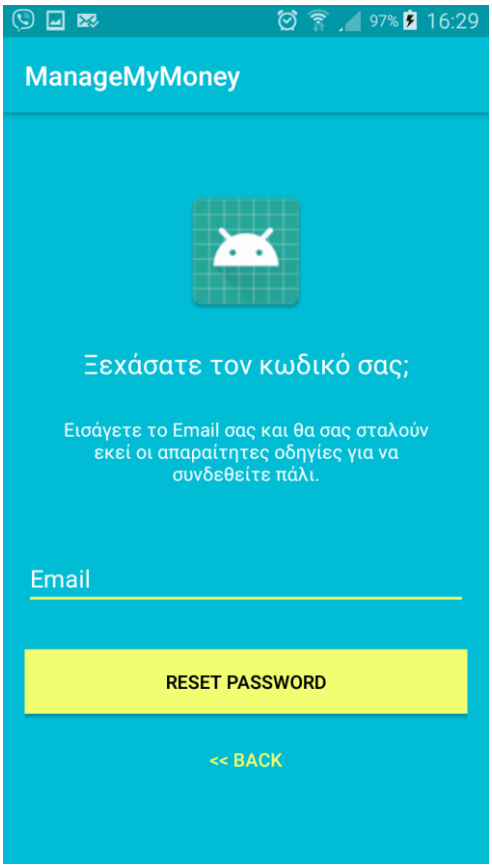
```
private EditText inputEmail;
private Button btnReset, btnBack;
private FirebaseAuth auth;
private ProgressBar progressBar;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_reset_password);

    inputEmail = (EditText) findViewById(R.id.email);
    btnReset = (Button) findViewById(R.id.btn_reset_password);
    btnBack = (Button) findViewById(R.id.btn_back);
    progressBar = (ProgressBar) findViewById(R.id.progressBar);

    auth = FirebaseAuth.getInstance();

    btnBack.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();
        }
    });

    btnReset.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            String email = inputEmail.getText().toString().trim();
```

```
            if (TextUtils.isEmpty(email)) {
                Toast.makeText(getApplication(), "Εισάγετε το
email σας.", Toast.LENGTH_SHORT).show();
                return;
            }

            progressBar.setVisibility(View.VISIBLE);
            auth.sendPasswordResetEmail(email)
                    .addOnCompleteListener(new
OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void>
task) {
                            if (task.isSuccessful()) {

Toast.makeText(ResetPasswordActivity.this, "We have sent you
instructions to reset your password!", Toast.LENGTH_SHORT).show();
                            } else {

Toast.makeText(ResetPasswordActivity.this, "Failed to send reset
email!", Toast.LENGTH_SHORT).show();
                            }

                            progressBar.setVisibility(View.GONE);
                        }
                    });
        }
    });
}
```



Εικόνα 4-5: Επαναφορά κωδικού Χρήστη

Στην εικόνα 4-6 φαίνεται η οθόνη της επαναφοράς κωδικού χρήστη.



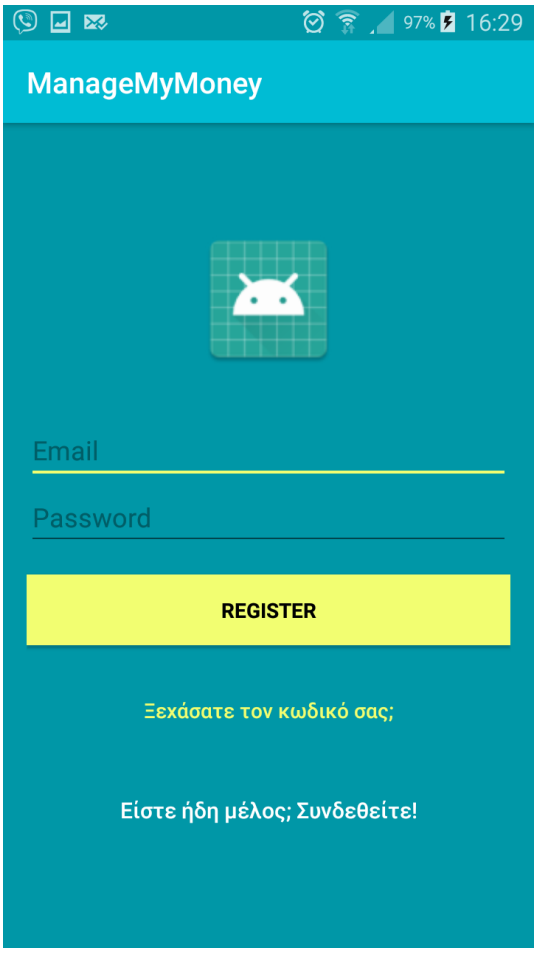
Εικόνα 4-6: Οθόνη Επαναφοράς Κωδικού Χρήστη

Τέλος παρακάτω φαίνεται ο κώδικας της κλάσης SignupActivity, όπου ο χρήστης μπορεί να γραφτεί στην εφαρμογή. Στην εικόνα 4-7 φαίνεται και η αντίστοιχη οθόνη.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);

        //Get Firebase auth instance
        auth = FirebaseAuth.getInstance();

        btnSignIn = (Button) findViewById(R.id.sign_in_button);
        btnSignUp = (Button) findViewById(R.id.sign_up_button);
        inputEmail = (EditText) findViewById(R.id.email);
        inputPassword = (EditText) findViewById(R.id.password);
        progressBar = (ProgressBar)
findViewById(R.id.progressBar);
        btnResetPassword = (Button)
findViewById(R.id.btn_reset_password);

        btnResetPassword.setOnClickListener(new
View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                startActivity(new Intent(SignupActivity.this,
ResetPasswordActivity.class));
            }
        });
        btnSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        btnSignUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String email =
inputEmail.getText().toString().trim();
                String password =
inputPassword.getText().toString().trim();

                if (TextUtils.isEmpty(email)) {
                    Toast.makeText(getApplicationContext(),
"Εισάγετε email!", Toast.LENGTH_SHORT).show();
                    return;
                }

                if (TextUtils.isEmpty(password)) {
                    Toast.makeText(getApplicationContext(),
"Εισάγετε κωδικό!", Toast.LENGTH_SHORT).show();
                    return;
                }
                if (password.length() < 6) {
                    Toast.makeText(getApplicationContext(), "Ο
κωδικός πρέπει να είναι τουλάχιστον 6 χαρακτήρες!",
Toast.LENGTH_SHORT).show();
                    return;
                }
                progressBar.setVisibility(View.VISIBLE);
                //create user
                auth.createUserWithEmailAndPassword(email,
password)

.addOnCompleteListener(SignupActivity.this, new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull
Task<AuthResult> task) {

Toast.makeText(SignupActivity.this,
"createUserWithEmail:onComplete:" + task.isSuccessful(),
Toast.LENGTH_SHORT).show();

progressBar.setVisibility(View.GONE);

                                if (!task.isSuccessful()) {
```

```
Toast.makeText(SignupActivity.this, "Authentication failed." +
task.getException(),

Toast.LENGTH_SHORT).show();
                                    } else {

Toast.makeText(SignupActivity.this, "Successful sign up. Please
Login." + task.getException(),

Toast.LENGTH_SHORT).show();
//                                      startActivity(new
Intent(SignupActivity.this, LoginActivity.class));
                                                finish();
                                    }
                                }
                            });
                }
            });
    }
```

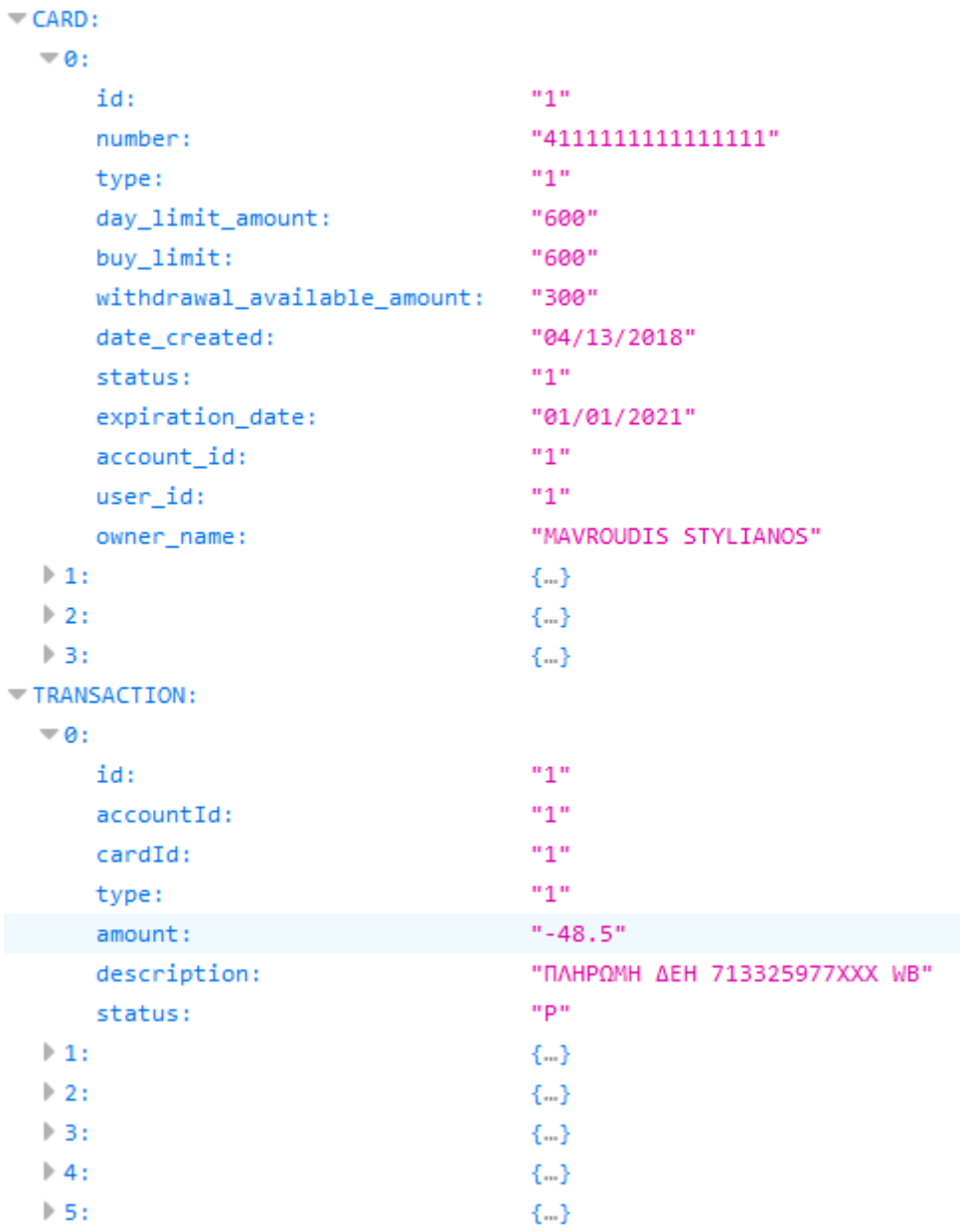


Εικόνα 4-7: Οθόνη εγγραφής νέου χρήστη

4.3 Η μορφή του JSON με τα δεδομένα

Σειρά έχει ο τρόπος με τον οποίο θα γεμίζει με δεδομένα η τοπική βάση (SQLite) της εφαρμογής. Η δομή του JSON που χρησιμοποιείται στην εφαρμογή είναι (όπως φαίνεται στην εικόνα 4-8) η ακόλουθη


```
USER:
  0:
    id:                 "1"
    username:           "georpapa"
    name:               "George"
    surname:            "Papachristou"
    email:              "papachristou.george@gmail.com"
  1:                    {…}
ACCOUNT:
  0:
    accountId:          "1"
    userId:             "1"
    name:               "ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ"
    type:               "1"
    bankId:             "1"
    number:             "256094720"
    iban:               "GR 54ABNA0256094720"
    availableBalance:   "300.15"
    accountingBalance:  "300.15"
  1:                    {…}
  2:                    {…}
  3:                    {…}
  4:                    {…}
  5:                    {…}
BANK:
  0:
    id:                 "1"
    type:               "1"
    name:               "Πειραιώς"
    number:             "19"
    postal_code:        "185 35"
    country:            "Αθήνα"
    city:               "Ελλάδα"
    address:            "Καραΐσκου 143"
  1:
    id:                 "2"
    type:               "1"
```



Εικόνα 4-8: Δομή JSON

Για να διαβαστούν τα δεδομένα από το JSON και να αποθηκευτούν στην SQLite βάση, ακολουθήθηκε η παρακάτω διαδικασία που φαίνεται στον κώδικα. (Ενδεικτικά παρουσιάζονται οι πίνακες User, Bank και Transactions)

```
String url = "https://api.myjson.com/bins/97fk6  ";


        StringRequest request = new StringRequest(url, new
Response.Listener<String>() {
            @Override
            public void onResponse(String string) {
                try {
                    Vector userList = parseJsonUsersData(string);
```

```
                    Vector bankList = parseJsonBankData(string);
                    Vector transactionList =
parseJsonTransactionData(string);
                    Vector accountList =
parseJsonAccountsData(string);
                    Vector cardList = parseJsonCardsData(string);

                    for (int i=0; i<userList.size(); i++) {
                        HashMap userValue = (HashMap)
bankList.get(i);
                        int id =
Integer.parseInt(userValue.get("id").toString());

                        db.addUser(new User(id,
userValue.get("username").toString(),
userValue.get("name").toString(),
userValue.get("surname").toString(),
userValue.get("email").toString()));
                    }

                    for (int i=0; i<bankList.size(); i++) {
                        HashMap bankValue = (HashMap)
bankList.get(i);
                        int id =
Integer.parseInt(bankValue.get("id").toString());
                        int bankType =
Integer.parseInt(bankValue.get("type").toString());

                        db.addBank(new BankInfo(id, bankType,
bankValue.get("name").toString(),
bankValue.get("number").toString(),
bankValue.get("postal_code").toString(),
bankValue.get("country").toString(),
bankValue.get("city").toString(),
bankValue.get("address").toString()));
                    }

                    for (int i=0; i<transactionList.size(); i++) {
                        HashMap transValue = (HashMap)
transactionList.get(i);
                        int id =
Integer.parseInt(transValue.get("id").toString());
                        int accountId =
Integer.parseInt(transValue.get("accountId").toString());
                        int cardId =
Integer.parseInt(transValue.get("cardId").toString());
                        int transType =
Integer.parseInt(transValue.get("type").toString());
                        double amount =
Double.parseDouble(transValue.get("amount").toString());

                        db.addTransaction(new Transactions(id,
accountId, cardId, transType, amount,
transValue.get("description").toString(),
transValue.get("status").toString()));
                    }
```

```
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError volleyError) {
                Toast.makeText(getApplicationContext(), "Some
error occurred!!", Toast.LENGTH_SHORT).show();
//                dialog.dismiss();
            }
        });
```

Ουσιαστικά γεμίζει μία λίστα με την μορφή που περιμένει η SQLite, με τα δεδομένα. Η λίστα αυτή αρχικοποιείται με τις τιμές που διαβάστηκαν από το JSON με την ακόλουθη μέθοδο (παράδειγμα για την οντότητα bank).

```
public Vector parseJsonBankData(String jsonString) throws
UnsupportedEncodingException {
    Vector bankList = null;
    try {
        JSONObject object = new JSONObject(jsonString);
        JSONArray jsonArray = object.getJSONArray("BANK");

        bankList = new Vector();

        ArrayList bank = new ArrayList();

        for (int i = 0; i < jsonArray.length(); ++i) {

            HashMap values = new HashMap();

            JSONObject jsonObject = (JSONObject) jsonArray.get(i);

            values.put("id", jsonObject.get("id"));
            values.put("type", jsonObject.get("type"));
            values.put("name", new
String(jsonObject.getString("name").getBytes("ISO-8859-1"), "UTF-
8"));
            values.put("number", jsonObject.get("number"));
            values.put("postal_code",
jsonObject.get("postal_code"));
            values.put("country", new
String(jsonObject.getString("country").getBytes("ISO-8859-1"),
"UTF-8"));
            values.put("city", new
String(jsonObject.getString("city").getBytes("ISO-8859-1"), "UTF-
8"));
            values.put("address", new
String(jsonObject.getString("address").getBytes("ISO-8859-1"),
"UTF-8"));

            bankList.add(values);
```

```
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }

    return bankList;
}
```

4.4 Αποστολή ειδοποιήσεων (Push Notification)

Πάλι με την βοήθεια του firebase και κάνοντας χρήση της λειτουργικότητας που παρέχει για την αποστολή ειδοποιήσεων (με το Cloud Messaging) στην εφαρμογή όταν εκτελείτε στο παρασκήνιο (background) υλοποιήθηκε στην εφαρμογή Manage My Money η διαδικασία ώστε να λαμβάνει αυτές τις ειδοποιήσεις και πατώντας πάνω στην ειδοποίηση να μας πηγαίνει στην κεντρική οθόνη της εφαρμογής.

Η υλοποίηση αυτή έγινε με την κλάση MyFirebaseMessagingService

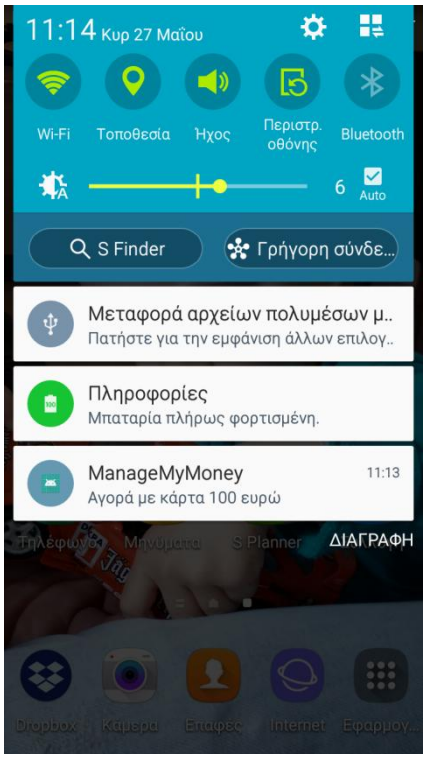
```
public class MyFirebaseMessagingService extends
FirebaseMessagingService {
    private static final String TAG = "FCM Service";

    private static final int REQUEST_CODE = 1;
    private static final int NOTIFICATION_ID = 6578;

    public MyFirebaseMessagingService() {
        super();
    }

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        // TODO: Handle FCM messages here.
        // If the application is in the foreground handle both
data and notification messages here.
        // Also if you intend on generating your own notifications
as a result of a received FCM
        // message, here is where that should be initiated.

        super.onMessageReceived(remoteMessage);

        Log.d(TAG, "From: " + remoteMessage.getFrom());
        Log.d(TAG, "Notification Message Body: " +
remoteMessage.getNotification().getBody());

        final String title =
remoteMessage.getNotification().getTitle();
        final String message =
remoteMessage.getNotification().getBody();

        showNotifications(title, message);
    }
```

```
    private void showNotifications(String title, String msg) {
        Intent i = new Intent(this, MainActivity.class);
        PendingIntent pendingIntent =
PendingIntent.getActivity(this, REQUEST_CODE,
                i, PendingIntent.FLAG_UPDATE_CURRENT);

        Notification notification = new
NotificationCompat.Builder(this)
                .setContentText(msg)
                .setContentTitle(title)
                .setContentIntent(pendingIntent)
                .setSmallIcon(R.mipmap.ic_launcher_round)
                .build();

        NotificationManager manager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        manager.notify(NOTIFICATION_ID, notification);
    }
}
```

Επίσης στο αρχείο AndroidManifest.xml προστέθηκε και το ακόλουθο service:

```
<service android:name=".MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING EVENT"
/>
    </intent-filter>
</service>
```

Στην επόμενη εικόνα φαίνεται και η ειδοποίηση που λαμβάνει ο χρήστης στην συσκευή του. (Εικόνα 4-9)



Εικόνα 4-9: Push Notification

4.5 Αποσύνδεση (LogOut)

Μια ακόμα λειτουργία που παρέχει η εφαρμογή, είναι όταν τελειώσει με την διαδικασία του Login και συνδεθεί ο χρήστης, ώστε να μπορεί να μπει στην κεντρική οθόνη και να συνεχίσει με την ενημέρωση για τους λογαριασμούς του, είναι και το κουμπί της αποσύνδεσης. Η επιλογή αυτή έχει μπει στην Κεντρική Οθόνη, αλλά και στην οθόνη με τους Λογαριασμούς που θα δούμε και στο επόμενο κεφάλαιο. Η αποσύνδεση στην εφαρμογή, είναι πολύ σημαντική, για τον λόγο ότι όταν θέλει κάποιος άλλος χρήστης να συνδεθεί για να πληροφορηθεί για τους δικούς του λογαριασμούς, πρέπει να βρεθεί πάλι στην αρχική οθόνη του Login. Ο κώδικας της αποσύνδεσης, Logout είναι ο ακόλουθος:

```
public void logOut(View view)
{

    auth.getInstance().signOut();

    startActivity(new Intent(MainActivity.this,
LoginActivity.class));
    finish();
}
```

4.6 Το αρχείο AndroidManifest.xml

Οι εφαρμογές του Android χρησιμοποιούν ένα αρχείο, το AndroidManifest.xml, το οποίο καθορίζει ποιες λειτουργίες (υλικού αλλά και λογισμικού) θα χρησιμοποιήσει η εφαρμογή. Κάποιες από αυτές είναι η κάμερα, το GPS, αλλά και το Bluetooth. Ουσιαστικά ορίζει τα δικαιώματα που θα έχει η εφαρμογή στα διάφορα συστατικά της συσκευής. Επίσης περιλαμβάνει και την έκδοση του API που μπορεί να υποστηριχθεί η εφαρμογή.

Παρακάτω βλέπουμε και το αρχείο AndroidManifest της εφαρμογής:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.george.managemymoney">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <application
```

```
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <service android:name=".MyFirebaseMessagingService">
            <intent-filter>
                <action
android:name="com.google.firebase.MESSAGING_EVENT" />
            </intent-filter>
        </service>
        <service android:name=".FirebaseIDService">
            <intent-filter>
                <action
android:name="com.google.firebase.INSTANCE_ID_EVENT" />
            </intent-filter>
        </service>
        <activity android:name=".LoginActivity"
              android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:screenOrientation="portrait"
            android:name=".SignupActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>
                <category
android:name="android.intent.category.ALTERNATIVE" />
            </intent-filter>
        </activity>
        <activity android:screenOrientation="portrait"
            android:name=".ResetPasswordActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>
                <category
android:name="android.intent.category.ALTERNATIVE" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity"
                      android:screenOrientation="portrait"/>
        <activity android:name=".AccountActivity"
              android:screenOrientation="portrait"
              android:parentActivityName=".MainActivity" />
        <activity android:name=".TransactionActivity"
              android:screenOrientation="portrait"
              android:parentActivityName=". MainActivity" />
        <activity android:name=".BankActivity"
```

```
            android:screenOrientation="portrait"
            android:parentActivityName=".MainActivity" />
        <activity android:name=".CardsActivity"
            android:screenOrientation="portrait"
            android:parentActivityName=".MainActivity" />
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />
        <activity
            android:name=".MapsActivity"
            android:screenOrientation="portrait"
       android:label="@string/title_activity_maps"></activity>
    </application>
```

5. Βασική Δομή και Υλοποίηση εφαρμογής

5.1 Η κεντρική οθόνη (MainActivity)

Προχωράμε με την κεντρική οθόνη της εφαρμογής (Εικόνα 5-1). Η οθόνη αυτή, αφού έχει ολοκληρωθεί η διαδικασία του Log-In θα γεμίζει την βάση μας με τα δεδομένα από την διαδικασία που εξηγήθηκε πιο πάνω με το JSON και επιλέγοντας το κουμπί που επιθυμούμε θα βλέπουμε την αντίστοιχη πληροφορία.

Πατώντας το κουμπί Λογαριασμοί, θα μας πηγαίνει στην οθόνη με την πληροφορία για τους λογαριασμούς του χρήστη. Επίσης με το κουμπί Κάρτες, θα έρχεται η πληροφορία για τις διαθέσιμες κάρτες του χρήστη. Το κουμπί Τράπεζες, παρέχει λίγες πληροφορίες για τις τράπεζες που θα είναι συνδεδεμένες με την εφαρμογή, ενώ με το Κουμπί Βρείτε Τράπεζα, θα ανοίγει έναν χάρτη όπου δείχνει το στίγμα του χρήστη και τα διαθέσιμα ATM ή καταστήματα. Τέλος θα έχει και την επιλογή να αποσυνδεθεί από την εφαρμογή για να επιστρέψει στην αρχική οθόνη.

Εικόνα 5-1: Η Αρχική οθόνη

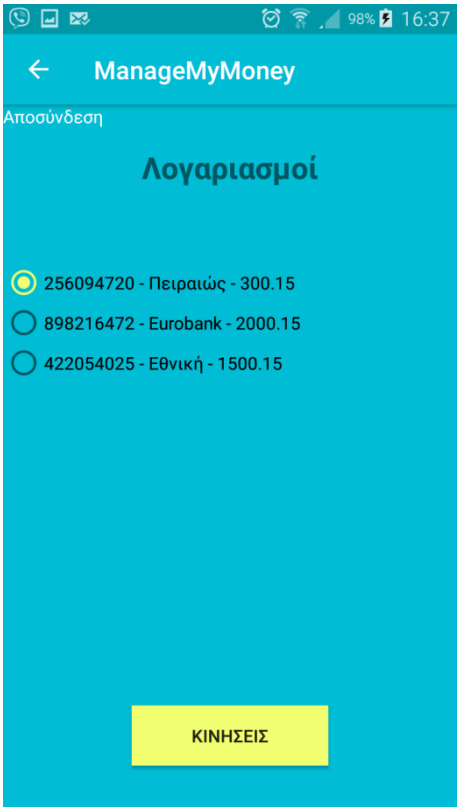
Παρακάτω βλέπουμε και σε κώδικα πως υλοποιήθηκε η λειτουργικότητα που αναφέρθηκε:

```
@Override
    public void onClick(View view) {

//        int selectedRadioButtonID =
ll.getCheckedRadioButtonId();

        if (view.getId() == buttonAccount.getId()) {
            Intent intent = new Intent(this,
AccountActivity.class);

            intent.putExtra("User", userID);
            startActivity(intent);
        }
        else if (view.getId() == buttonCards.getId()) {
            Intent intent = new Intent(this, CardsActivity.class);

            intent.putExtra("User", userID);
            startActivity(intent);
        }
        else if (view.getId() == buttonBanks.getId()) {
            Intent intent = new Intent(this, BankActivity.class);

            startActivity(intent);
        }
        else if (view.getId() == buttonMaps.getId()) {
            Intent intent = new Intent(this, MapsActivity.class);

            startActivity(intent);
        }

    }
```

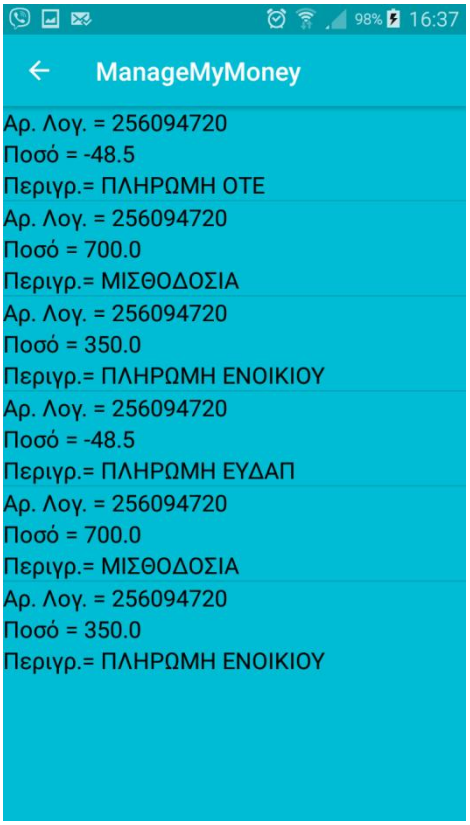
5.2 Οθόνες Λογαριασμοί και κινήσεις

Από την κεντρική οθόνη της εφαρμογής αν επιλέξουμε το κουμπί Λογαριασμοί, τότε μας οδηγεί στην οθόνη λογαριασμοί (Εικόνα 5-2), όπου ο χρήστης πληροφορείται για τους λογαριασμούς που διαθέτει. Εκεί σε μια ομάδα από Radio Buttons, βλέπει τον αριθμό λογαριασμού, το όνομα της τράπεζας που διατηρεί τον λογαριασμό, αλλά και το διαθέσιμο ποσό που έχει στον κάθε λογαριασμό.

Στην συνέχεια, επιλέγοντας τον επιθυμητό λογαριασμό και πατώντας το κουμπί Κινήσεις, τότε κατευθύνεται στην επόμενη οθόνη (Εικόνα 5-3), όπου υπάρχουν οι αναλυτικές κινήσεις για αυτό τον λογαριασμό. Στην οθόνη αυτή, ο χρήστης βλέπει το ποσό της κίνησης, αλλά και μια περιγραφή.



Εικόνα 5-2: Οθόνη Λογαριασμοί



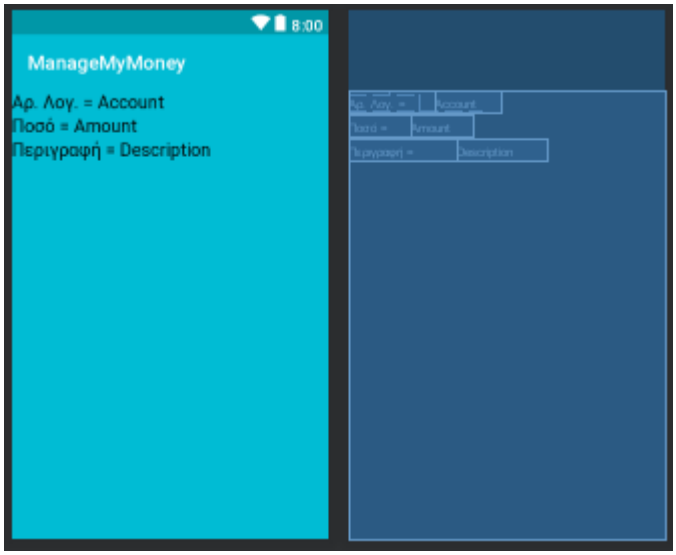
Εικόνα 5-3: Οθόνη Κινήσεις

Παρακάτω παρουσιάζεται ο κώδικας του AccountActivity.java, της οθόνης Λογαριασμοί, όπου φαίνεται το δυναμικό γέμισμα των Radio Buttons με τα στοιχεία των λογαριασμών, ανάλογα με τα στοιχεία του χρήστη.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_account);

        button = (Button) findViewById(R.id.button);

        Intent intent = getIntent();
        userID = intent.getIntExtra("User", 0);

        TextView t2 = (TextView) findViewById(R.id.text2);
        t2.setMovementMethod(LinkMovementMethod.getInstance());

        button.setOnClickListener(this);

        DatabaseHandler db = new DatabaseHandler(this);

        List<AccountsInfo> accountList = new ArrayList<>();

        accountList = db.getAccountByUser(userID);

        addRadioButtons(accountList.size(), accountList, db);
    }

    //dynamic fill radio buttons
    public void addRadioButtons(int number, List<AccountsInfo>
accountList, DatabaseHandler db) {

        for (int row = 0; row < 1; row++) {
            ll = new RadioGroup(this);
            ll.setOrientation(LinearLayout.VERTICAL);

            for (int i = 1; i <= number; i++) {
                RadioButton rdbtn = new RadioButton(this);
                rdbtn.setId((row * 2) + i);
                String account = accountList.get(i-
1).getNumber().toString();
                Double amount = accountList.get(i-
1).getAccountingBalance();

                String bankName = "";

                bankName = db.getBankNameByID(accountList.get(i-
1).getBankId());

                rdbtn.setText(account + " - " + bankName + " - " +
amount.toString());
                ll.addView(rdbtn);
            }
```

```
            ((ViewGroup)
findViewById(R.id.radiogroup)).addView(ll);
        }

    }

    @Override
    public void onClick(View view) {

        int selectedRadioButtonID = ll.getCheckedRadioButtonId();

        if (view.getId() == button.getId()) {
            Intent intent = new Intent(this,
TransactionActivity.class);

            String selectedRadioButtonText = null;

            if (selectedRadioButtonID != -1) {
                RadioButton selectedRadioButton = (RadioButton)
findViewById(selectedRadioButtonID);
                selectedRadioButtonText =
selectedRadioButton.getText().toString();

                intent.putExtra("User", userID);
                intent.putExtra("Account",
selectedRadioButtonText.substring(0,
selectedRadioButtonText.indexOf(" ")));
                startActivity(intent);
            }
            else
            {
                Context context = getApplicationContext();
                CharSequence text = "Δεν έχετε επιλέξει
λογαριασμό";
                int duration = Toast.LENGTH_SHORT;

                Toast toast = Toast.makeText(context, text,
duration);
                toast.show();
            }
        }
    }
```

Τέλος, παρατηρούμε ότι στην οθόνη της εικόνας 5-3, όπου φαίνονται οι κινήσεις, το List View που εμφανίζεται, περιέχει περισσότερες από μία γραμμές. Για να επιτευχθεί αυτό, δημιουργήθηκε η οθόνη listviewdatalayout.xml εικόνα 5-4,



Εικόνα 5-4: listviewdatalayout.xml

στην οποία σχεδιάστηκε η επιθυμητή μορφή του List View. Ο κώδικας της JAVA κλάσης SQLiteListAdapter είναι ο ακόλουθος:

```
public class SQLiteListAdapter extends BaseAdapter {

    Context context;
    ArrayList<String> userAccount;
    ArrayList<Double> UserAmount;
    ArrayList<String> UserDescription;

    public SQLiteListAdapter(
            Context context2,
            ArrayList<String> account,
            ArrayList<Double> amount,
            ArrayList<String> description
    )
    {
        this.context = context2;
        this.userAccount = account;
        this.UserAmount = amount;
        this.UserDescription = description;
    }
    public int getCount() {
        // TODO Auto-generated method stub
        return userAccount.size();
    }

    public Object getItem(int position) {
        // TODO Auto-generated method stub
        return null;
    }

    public long getItemId(int position) {
        // TODO Auto-generated method stub
        return 0;
    }
```

```
    public View getView(int position, View child, ViewGroup
parent) {

        Holder holder;

        LayoutInflater layoutInflater;

        if (child == null) {
            layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            child =
layoutInflater.inflate(R.layout.listviewdatalayout, null);

            holder = new Holder();

            holder.textviewAccount = (TextView)
child.findViewById(R.id.textViewBank);
            holder.textviewAmount = (TextView)
child.findViewById(R.id.textViewAmount);
            holder.textviewDescription = (TextView)
child.findViewById(R.id.textViewDescription);

            child.setTag(holder);

        } else {

            holder = (Holder) child.getTag();
        }
        holder.textviewAccount.setText(userAccount.get(position));

holder.textviewAmount.setText(UserAmount.get(position).toString())
;

holder.textviewDescription.setText(UserDescription.get(position));

        return child;
    }

    public class Holder {
        TextView textviewAccount;
        TextView textviewAmount;
        TextView textviewDescription;
    }

}
```

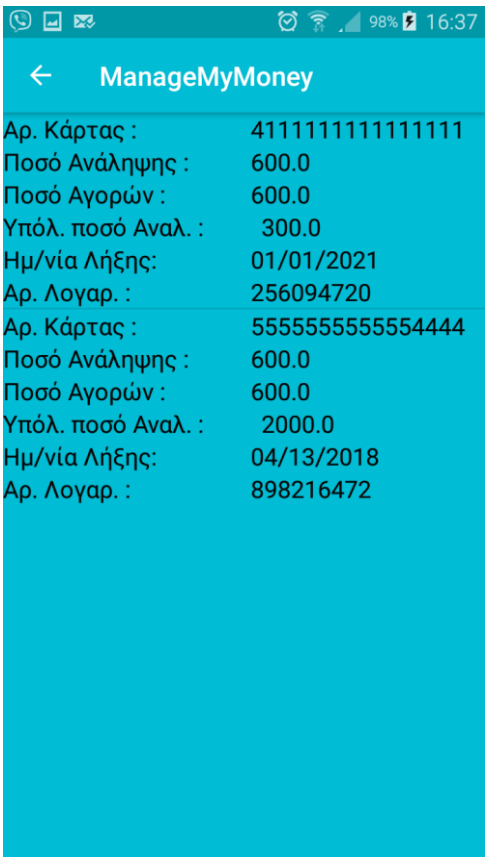
Έτσι ο κώδικας της οθόνης Κινήσεις όπου καλεί εσωτερικά την παραπάνω κλάση, είναι ο ακόλουθος (TransactionActivity):

```
public class TransactionActivity extends AppCompatActivity {

    ListView listView;
    Cursor cursor;
```

```
    SQLiteListAdapter ListAdapter;

    ArrayList<String> Account_ArrayList = new ArrayList<String>();
    ArrayList<Double> Amount_ArrayList = new ArrayList<Double>();
    ArrayList<String> Description_ArrayList = new
ArrayList<String>();

    DatabaseHandler db = new DatabaseHandler(this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_transaction);

        listView = (ListView)findViewById(R.id.listView);

        DatabaseHandler db = new DatabaseHandler(this);

        Intent intent = getIntent();
        Integer user = intent.getIntExtra("User", 0);
        String accountNo = intent.getStringExtra("Account");

        List<Transactions> transactionsList = new ArrayList<>();
        int accountId =
db.getAccountByBankAccount(Integer.parseInt(accountNo));
        transactionsList = db.getTransactionByUserAndAccount(user,
accountId);

        Account_ArrayList.clear();
        Amount_ArrayList.clear();
        Description_ArrayList.clear();

        for(int i=0; i<transactionsList.size(); i++) {
            System.out.println("------" +
transactionsList.get(i).description);

            Account_ArrayList.add(accountNo);
            Amount_ArrayList.add(transactionsList.get(i).amount);

Description_ArrayList.add(transactionsList.get(i).description);
        }
        ListAdapter = new
SQLiteListAdapter(TransactionActivity.this,

                Account_ArrayList,
                Amount_ArrayList,
                Description_ArrayList

        );
        listView.setAdapter(ListAdapter);

    }
}
```

5.3 Οθόνη Κάρτες

Αν η επιλογή που κάνει ο χρήστης είναι το κουμπί κάρτες, τότε ανοίγει η οθόνη των Καρτών (Εικόνα 5-5). Η οθόνη αυτή περιέχει την πληροφορία των καρτών του χρήστη: πόσες κάρτες έχει και σε ποια τράπεζα είναι η κάθε μία. Επίσης μπορεί να δει το ποσό ανάληψης όπου έχει το δικαίωμα να κάνει η κάρτα αυτή, αντίστοιχα το επιτρεπόμενο ποσό για αγορές, αλλά και το υπολειπόμενο ποσό για ανάληψη.



Εικόνα 5-5: Οθόνη Κάρτες

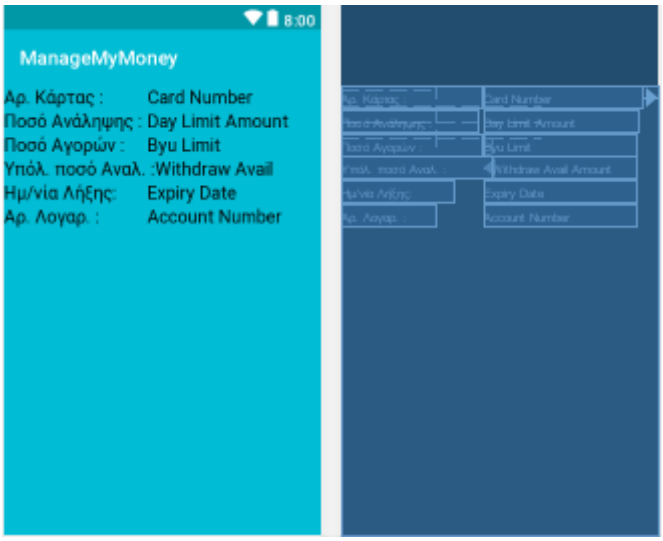
Πιο κάτω φαίνεται και ο κώδικα της κλάσης CardsActivity όπου περιλαμβάνει όπως και στην προηγούμενη οθόνη, ένα custom ListView.

```
ListView listView;

SQLiteListCardAdapter listCardAdapter;

ArrayList<String> CardNumber_ArrayList = new ArrayList<String>();
ArrayList<Double> DayLimitAmount_ArrayList = new
ArrayList<Double>();

ArrayList<Double> BuyLimit_ArrayList = new ArrayList<Double>();
```

```
ArrayList<Double> WithdrawAvailableAmount_ArrayList = new
ArrayList<Double>();
ArrayList<String> ExpiryDate_ArrayList = new ArrayList<String>();
ArrayList<Integer> AccountNo_ArrayList = new ArrayList<Integer>();


DatabaseHandler db = new DatabaseHandler(this);


@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cards);

    listView = (ListView)findViewById(R.id.listView);

    // Get the Intent that started this activity and extract the
string
    Intent intent = getIntent();
    Integer user = intent.getIntExtra("User", 0);

    List<CardsInfo> cardsInfoList = new ArrayList<>();
    cardsInfoList = db.getCardsByUser(user);


    CardNumber_ArrayList.clear();
    DayLimitAmount_ArrayList.clear();
    BuyLimit_ArrayList.clear();
    WithdrawAvailableAmount_ArrayList.clear();
    ExpiryDate_ArrayList.clear();
    AccountNo_ArrayList.clear();

    for(int i=0; i<cardsInfoList.size(); i++) {

        CardNumber_ArrayList.add(cardsInfoList.get(i).cardNumber);

DayLimitAmount_ArrayList.add(cardsInfoList.get(i).dayLimitAmount);
        BuyLimit_ArrayList.add(cardsInfoList.get(i).buyLimit);

WithdrawAvailableAmount_ArrayList.add(cardsInfoList.get(i).withdra
walAvailableAmount);

        ExpiryDate_ArrayList.add(cardsInfoList.get(i).expiryDate);

        int accounNo =
db.getAccountNumberByAccountId(cardsInfoList.get(i).getAccountId()
);
        AccountNo_ArrayList.add(accounNo);

    }

    listCardAdapter = new
SQLiteListCardAdapter(CardsActivity.this,
            CardNumber_ArrayList,
            DayLimitAmount_ArrayList,
```

```
            BuyLimit_ArrayList,
            WithdrawAvailableAmount_ArrayList,
            ExpiryDate_ArrayList,
            AccountNo_ArrayList);

    listView.setAdapter(listCardAdapter);
}
```

Εδώ φαίνεται η οθόνη (Εικόνα 5-6) και ο κώδικας του Custom ListView το οποίο έχει το όνομα SQLiteListCardAdapter.



Εικόνα 5-6: listviewcardsdatalayout.xml

```
public class SQLiteListCardAdapter extends BaseAdapter {

    Context context;
    ArrayList<String> cardNumber;
    ArrayList<Double> cardDayLimitAmount;
    ArrayList<Double> cardByuLimit;
    ArrayList<Double> cardWithdrawAlailableAmount;
    ArrayList<String> cardExpiryDate;
    ArrayList<Integer> cardAccountNumber;

    public SQLiteListCardAdapter(
            Context context2,
            ArrayList<String> number,
            ArrayList<Double> dayLimitAmount,
            ArrayList<Double> byuLimitAmount,
            ArrayList<Double> withdrawAvailableAmount,
            ArrayList<String> expiryDate,
            ArrayList<Integer> accountNumber
    )
    {
        this.context = context2;
        this.cardNumber = number;
        this.cardDayLimitAmount= dayLimitAmount;
```

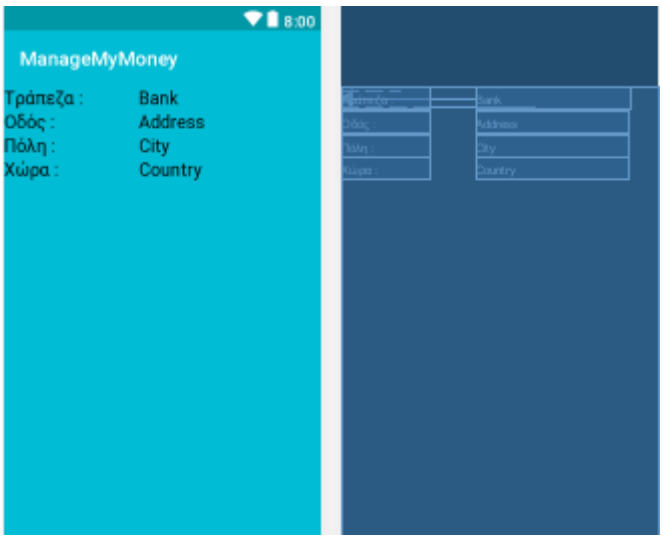
```
        this.cardByuLimit = byuLimitAmount;
        this.cardWithdrawAlailableAmount =
withdrawAvailableAmount;
        this.cardExpiryDate = expiryDate;
        this.cardAccountNumber = accountNumber;
    }
    public int getCount() {
        // TODO Auto-generated method stub
        return cardNumber.size();
    }
    public Object getItem(int position) {
        // TODO Auto-generated method stub
        return null;
    }
    public long getItemId(int position) {
        // TODO Auto-generated method stub
        return 0;
    }

    public View getView(int position, View child, ViewGroup
parent) {

        Holder holder;

        LayoutInflater layoutInflater;

        if (child == null) {
            layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            child =
layoutInflater.inflate(R.layout.listviewcardsdatalayout, null);

            holder = new Holder();

            holder.textViewCardNumber = (TextView)
child.findViewById(R.id.textViewCardNumber);
            holder.textViewDayLimitAmount = (TextView)
child.findViewById(R.id.textViewDayLimitAmount);
            holder.textViewByuLimit = (TextView)
child.findViewById(R.id.textViewByuLimit);
            holder.textViewWithdrawAvailableAmount = (TextView)
child.findViewById(R.id.textViewWithdrawAvailableAmount);
            holder.textViewExpiryDate= (TextView)
child.findViewById(R.id.textViewExpiryDate);
            holder.textViewAccountNo = (TextView)
child.findViewById(R.id.textViewAccountNo);

            child.setTag(holder);

        } else {
            holder = (Holder) child.getTag();
        }

holder.textViewCardNumber.setText(cardNumber.get(position));

holder.textViewDayLimitAmount.setText(cardDayLimitAmount.get(posit
```

```
ion).toString());

holder.textViewByuLimit.setText(cardByuLimit.get(position).toStrin
g());

holder.textViewWithdrawAvailableAmount.setText(cardWithdrawAlailab
leAmount.get(position).toString());

holder.textViewExpiryDate.setText(cardExpiryDate.get(position));

holder.textViewAccountNo.setText(Integer.toString(cardAccountNumbe
r.get(position)));

        return child;
    }
    public class Holder {
        TextView textViewCardNumber;
        TextView textViewDayLimitAmount;
        TextView textViewByuLimit;
        TextView textViewWithdrawAvailableAmount;
        TextView textViewExpiryDate;
        TextView textViewAccountNo;
    }

}
```

5.4 Οθόνη Τράπεζες

Το επόμενο κουμπί, ονομάζεται Τράπεζες και περιλαμβάνει τις τράπεζες οι οποίες έχουν συνεργαστεί με την εφαρμογή, ώστε να δώσουν τα απαιτούμενα δεδομένα, για την ενημέρωση των χρηστών. Περιέχει στοιχεία, όπως το όνομα, η διεύθυνση, η Πόλη και η χώρα.

Και σε αυτή την οθόνη πάλι χρησιμοποιούμε ένα Custom ListView το οποίο φαίνεται και στην εικόνα 5-7.



Εικόνα 5-7: listviewbankdatalayout.xml

```
public class SQLiteListBankAdapter extends BaseAdapter {

    Context context;
    ArrayList<String> bankName;
    ArrayList<String> bankAddress;
    ArrayList<String> bankCity;
    ArrayList<String> bankCountry ;

    public SQLiteListBankAdapter(
            Context context2,
            ArrayList<String> name,
            ArrayList<String> address,
            ArrayList<String> city,
            ArrayList<String> country
    )
    {
        this.context = context2;
        this.bankName = name;
        this.bankAddress= address;
        this.bankCity = city;
        this.bankCountry = country;
    }
    public int getCount() { // TODO Auto-generated method stub
        return bankName.size();
    }
    public Object getItem(int position){//TODO Auto-generated
method stub
        return null;
    }
    public long getItemId(int position){//TODO Auto-generated
method stub
        return 0;
    }

    public View getView(int position, View child, ViewGroup
parent) {

        Holder holder;

        LayoutInflater layoutInflater;

        if (child == null) {
            layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            child =
layoutInflater.inflate(R.layout.listviewbankdatalayout, null);

            holder = new Holder();

            holder.textviewBankName = (TextView)
child.findViewById(R.id.textViewBankName);
            holder.textViewAddress = (TextView)
child.findViewById(R.id.textViewAddress);
            holder.textViewCity = (TextView)
```

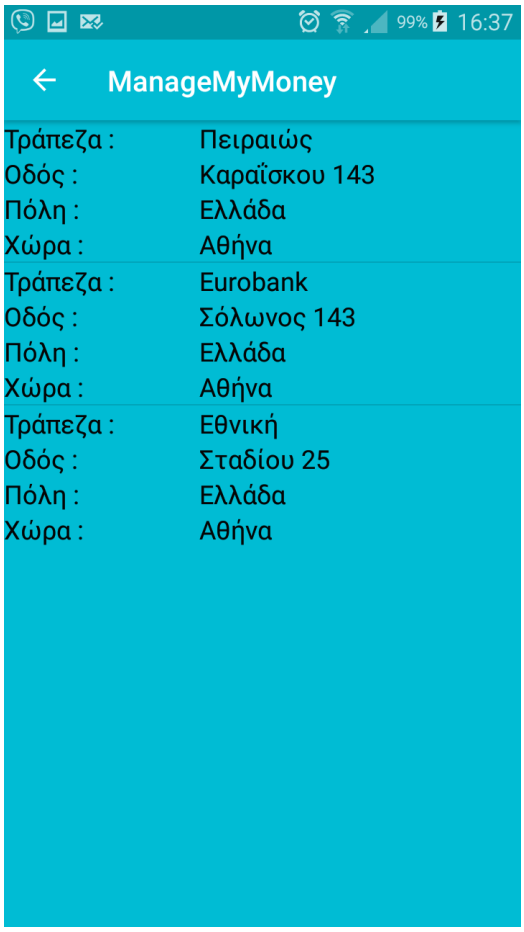
```
child.findViewById(R.id.textViewCity);
            holder.textViewCountry = (TextView)
child.findViewById(R.id.textViewCountry);

            child.setTag(holder);

        } else {

            holder = (Holder) child.getTag();
        }
        holder.textviewBankName.setText(bankName.get(position));
        holder.textViewAddress.setText(bankAddress.get(position));
        holder.textViewCity.setText(bankCity.get(position));
        holder.textViewCountry.setText(bankCountry.get(position));

        return child;
    }
    public class Holder {
        TextView textviewBankName;
        TextView textViewAddress;
        TextView textViewCity;
        TextView textViewCountry;
    }

}
```

Τέλος πιο κάτω είναι ο κώδικας την οθόνης Κάρτες, όπου καλεί το πιο πάνω SQLiteListBankAdapter, ώστε να το γεμίσει δυναμικά με τις τιμές από την τοπική βάση (SQLite).

```
ListView listView;
SQLiteListBankAdapter listBankAdapter;
ArrayList<String> BankName_ArrayList = new ArrayList<String>();
ArrayList<String> Address_ArrayList = new ArrayList<String>();
ArrayList<String> City_ArrayList = new ArrayList<String>();
ArrayList<String> Country_ArrayList = new ArrayList<String>();
DatabaseHandler db = new DatabaseHandler(this);
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_bank);

    listView = (ListView)findViewById(R.id.listView);
    List<BankInfo> banksList = new ArrayList<>();
    banksList = db.getAllBanks();

    BankName_ArrayList.clear();
    Address_ArrayList.clear();
    City_ArrayList.clear();
    Country_ArrayList.clear();

    for(int i=0; i<banksList.size(); i++) {
```

```
            BankName_ArrayList.add(banksList.get(i).name);
            Address_ArrayList.add(banksList.get(i).address);
            City_ArrayList.add(banksList.get(i).city);
            Country_ArrayList.add(banksList.get(i).country);
        }

        listBankAdapter = new SQLiteListBankAdapter(BankActivity.this,
                BankName_ArrayList,
                Address_ArrayList,
                City_ArrayList,
                Country_ArrayList);

        listView.setAdapter(listBankAdapter);
}
```

Η οθόνη των τραπεζών, φαίνεται παρακάτω (Εικόνα 5-8):



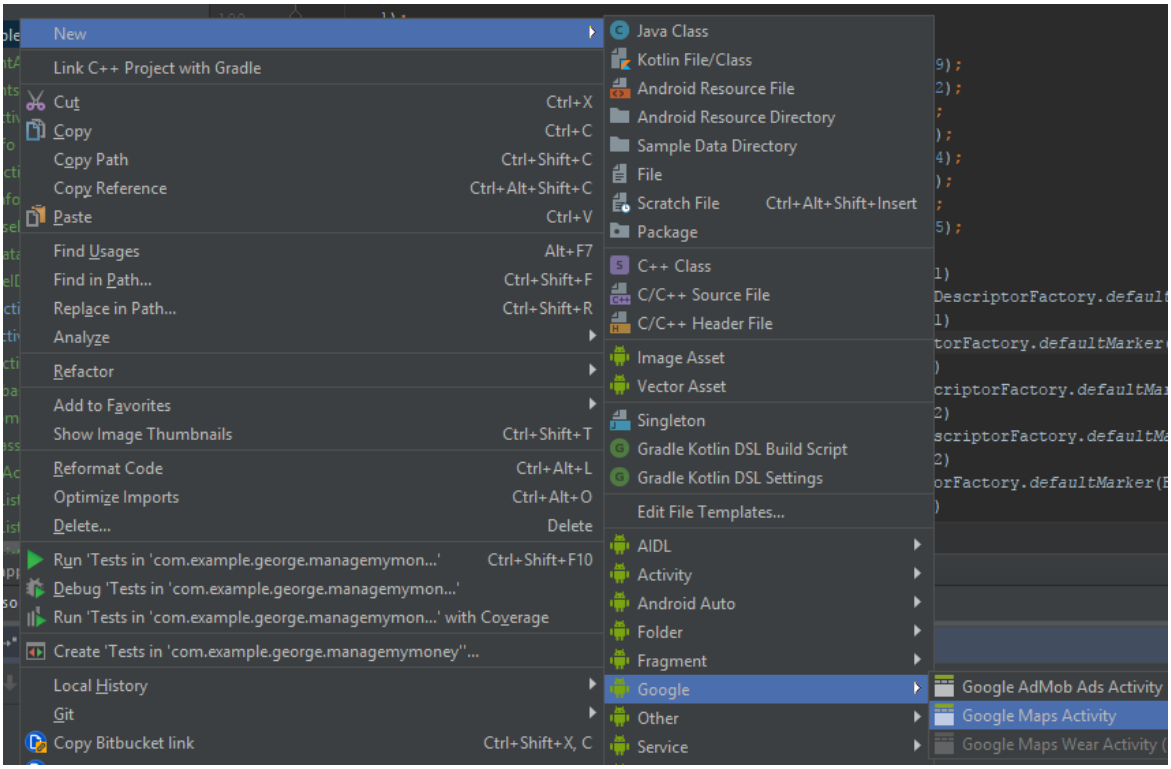
Εικόνα 5-8: Οθόνη Τράπεζες

5.5 Οθόνη Βρείτε Τράπεζα

Η τελευταία επιλογή που μας παρέχει η κεντρική οθόνη, είναι το κουμπί Βρείτε Τράπεζα. Πατώντας αυτό το κουμπί, ανοίγει μια οθόνη με τους χάρτες της Google (Google Maps). Σε αυτόν τον χάρτη μας δείχνει σε πιο σημείο βρίσκονται οι διάφορες τράπεζες ή κάποιο ATM. Έτσι στην περίπτωση που θα χρειαστεί ο χρήστης της εφαρμογής κάποιο αυτόματο μηχάνημα ανάληψης χρημάτων (ATM) ή κάποια τράπεζα ώστε να κάνει κάποια δουλειά που χρειάζεται, τότε θα βρίσκει με την βοήθεια του χάρτη τι είναι πιο κοντά του που μπορεί να τον εξυπηρετεί.

Στον χάρτη με την μπλε κουκίδα φαίνεται το σημείο που βρίσκεται ο χρήστης, ενώ με τα σηματάκια με τα διάφορα χρώματα φαίνονται τα σημεία ενδιαφέροντος, που για την εφαρμογή αυτή είναι οι τράπεζες και τα ATM.

Για να προστεθεί στην εφαρμογή, η οθόνη Βρείτε Τράπεζα, έγινε δημιουργία νέου αρχείου από το μενού (δεξί κλικ στο Android Studio, στην δομή του Project αριστερά) New → Google → Google Maps Activity (Εικόνα 5-9).



Εικόνα 5-9: Δημιουργία νέας οθόνης με υποστήριξη Google Maps

Το επόμενο σημαντικό βήμα, αφού δημιουργηθεί η νέα οθόνη, είναι να προστεθεί στην εφαρμογή το Google Maps API key [7], το οποίο είναι απαραίτητο για να ενεργοποιηθούν και να δουλέψουν σωστά οι χάρτες στην εφαρμογή.

Αυτή η διαδικασία επιτυγχάνεται με την ακόλουθη διαδικασία. Όταν δημιουργείται το νέο Activity με την καινούργια οθόνη, δημιουργείται και ένα νέο αρχείο με το όνομα google_maps_api.xml. Μέσα στο αρχείο αυτό, στα σχόλια υπάρχει ένα URL link, το οποίο αν το ανοίξουμε σε έναν Browser και ακολουθήσουμε τις οδηγίες που δίνει, τότε δημιουργείται ένα αποκλειστικό κλειδί για την εφαρμογή από την οποία άνοιξε αυτό το URL. Το κλειδί αυτό προστίθεται μέσα στο google_maps_api .xml με μια γραμμή όπως η ακόλουθη:

```
<string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">AIzaSyDq6vuGDsJ5PWQV2g1WMvC3Tz4h4kkOBlg</stri
ng>
```

Οπότε η εφαρμογή είναι έτοιμη να χρησιμοποιήσει το Google Maps.

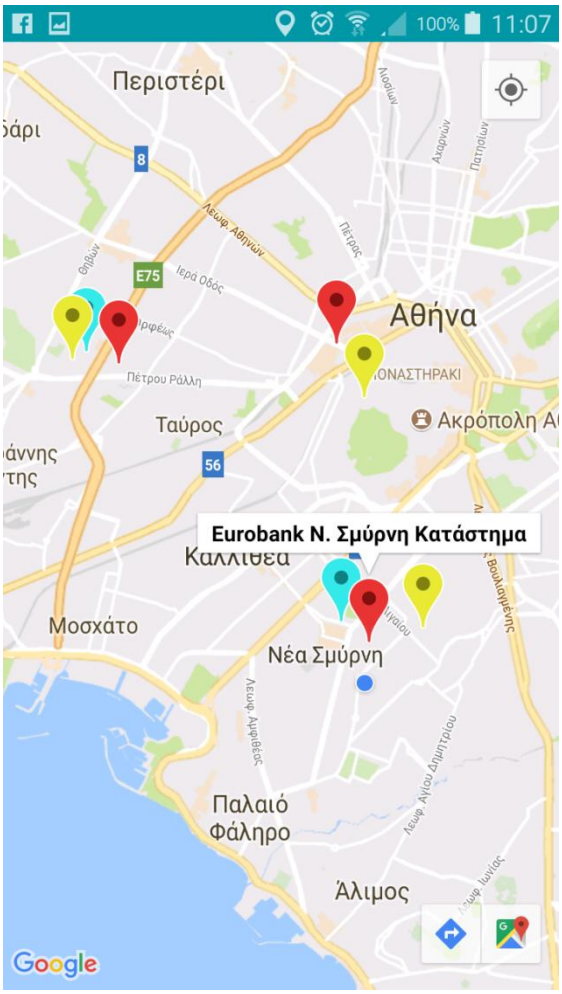
Για τις ανάγκες της εφαρμογής Manage My Money, δημιουργήθηκε η κλάση MapsActivity της οποίας ο κώδικας είναι ο ακόλουθος:

```
public class MapsActivity extends FragmentActivity implements
GoogleMap.OnMyLocationButtonClickListener,
        GoogleMap.OnMyLocationClickListener, OnMapReadyCallback ,
GoogleMap.OnPoiClickListener{

    private GoogleMap mMap;

    String myLat;
    String myLong;
    String name;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);

        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        LatLng eurobank_1 = new LatLng(37.9454424,23.7180259);
```

```
        LatLng peiraios_1 = new LatLng(37.9471122,23.7259542);
        LatLng ethniki_1 = new LatLng(37.947841, 23.713929);
        LatLng eurobank_2 = new LatLng(37.977835,23.6812588);
        LatLng peiraios_2 = new LatLng(37.9783086,23.6745104);
        LatLng ethniki_2 = new LatLng(37.9792219,23.6765757);
        LatLng eurobank_3 = new LatLng(37.98012,23.7132893);
        LatLng peiraios_3 = new LatLng(37.9738874,23.7173775);

        googleMap.addMarker(new
MarkerOptions().position(eurobank_1)
                .title("Eurobank N. Σμύρνη
Κατάστημα").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescr
iptorFactory.HUE_RED)));
        googleMap.addMarker(new
MarkerOptions().position(peiraios_1)
                .title("Πειραιώς Ν.Σμύρνη
ATM").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorF
actory.HUE_YELLOW)));
        googleMap.addMarker(new
MarkerOptions().position(ethniki_1)
                .title("Εθνική Ν.Σμύρνη
Κατάστημα").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescr
iptorFactory.HUE_CYAN)));
        googleMap.addMarker(new
MarkerOptions().position(eurobank_2)
                .title("Eurobank Αιγάλεω
Κατάστημα").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescr
iptorFactory.HUE_RED)));
        googleMap.addMarker(new
MarkerOptions().position(peiraios_2)
                .title("Πειραιώς Αιγάλεω
ATM").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorF
actory.HUE_YELLOW)));
        googleMap.addMarker(new
MarkerOptions().position(ethniki_2)
                .title("Εθνική Αιγάλεω
ATM").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorF
actory.HUE_CYAN)));
        googleMap.addMarker(new
MarkerOptions().position(eurobank_3)
                .title("Eurobank
ATM").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorF
actory.HUE_RED)));
        googleMap.addMarker(new
MarkerOptions().position(peiraios_3)
                .title("Πειραιώς Σύνταγμα
Κατάστημα").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescr
iptorFactory.HUE_YELLOW)));


        // TODO: Before enabling the My Location layer, you must
request
        // location permission from the user. This sample does not
include
        // a request for location permission.
        if (ActivityCompat.checkSelfPermission(this,
```

```
android.Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    Enable My Location
            mMap.setMyLocationEnabled(true);

        }
        else
        {
            Toast.makeText(this,"Please give me the GPS
Permission!",Toast.LENGTH_LONG).show();
        }
        mMap.setOnMyLocationButtonClickListener(this);
        mMap.setOnMyLocationClickListener(this);

    @Override
    public boolean onMyLocationButtonClick() {
        Toast.makeText(this, "MyLocation button clicked",
Toast.LENGTH_SHORT).show();

        return false;
    }

    @Override
    public void onMyLocationClick(@NonNull Location location) {
        Toast.makeText(this, "Current location:\n" + location,
Toast.LENGTH_LONG).show();
    }

    @Override
    public void onPoiClick(PointOfInterest pointOfInterest) {
        Marker poiMarker = mMap.addMarker(new MarkerOptions()
                .position(pointOfInterest.latLng)
                .title(pointOfInterest.name));

        poiMarker.showInfoWindow();
    }
}
```

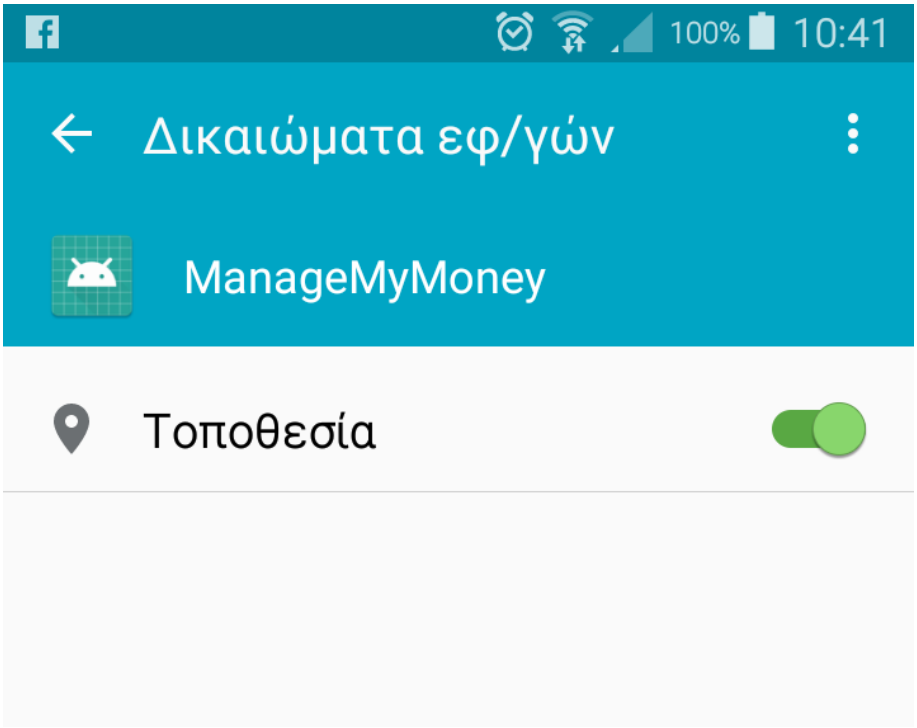
Η οθόνη που δημιουργήθηκε φαίνεται στην εικόνα 5-10.



Εικόνα 5-10: Οθόνη Βρείτε Τράπεζα

Έτσι στην οθόνη αυτή φαίνονται τα κοντινά καταστήματα, ή ATM καθώς επίσης και η περιγραφή του κάθε σημείου όταν πατήσουμε πάνω στην έγχρωμη κουκίδα-στόχο. Επίσης πατώντας τα κουμπιά που εμφανίζονται κάτω δεξιά, μπορούμε να πάρουμε οδηγίες για το πώς θα φτάσουμε στο σημείο αυτό.

Τέλος ένα πολύ σημαντικό σημείο ώστε να δουλέψει σωστά η λειτουργικότητα αυτή, είναι ότι πρέπει να είναι ενεργοποιημένο το GPS στην συσκευή, αλλά και από τις ρυθμίσεις της εφαρμογής είναι απαραίτητο, να είναι ενεργοποιημένη η άδεια (permission) της τοποθεσίας (Εικόνα 5-11).



Εικόνα 5-11: Ρυθμίσεις εφαρμογής

6. Συμπεράσματα - Μελλοντικές Βελτιώσεις

6.1 Συμπεράσματα

Τελειώνοντας αυτή την εφαρμογή, παρατηρήθηκαν κάποια πράγματα τόσο για τον προγραμματισμό μιας εφαρμογής σε περιβάλλον Android, αλλά και για την εξέλιξη των εφαρμογών όπως τις γνωρίζουμε σήμερα.

Για τον σχεδιασμό αλλά και την ανάπτυξη μιας εφαρμογής, απαραίτητη προϋπόθεση είναι η γνώση της γλώσσας προγραμματισμού Java. Σημαντικό ρόλο πλέον στην ανάπτυξη μιας εφαρμογής, παίζει η δυνατότητα που υπάρχει για εύκολη πρόσβαση σε πληθώρα υλικού, το οποίο βοηθάει στην ανάπτυξη αλλά και στην επίλυση τυχών προβλημάτων που αντιμετωπίζονται κατά την υλοποίηση μιας εφαρμογής.

Επίσης δεκάδες νέες εφαρμογές βγαίνουν καθημερινά για κινητές συσκευές, αλλά και για wearable's, οι οποίες αποσκοπούν στο να διευκολύνουν την ζωή των χρηστών τους. Το βασικό συστατικό για να πετύχει μια εφαρμογή είναι να προσφέρει κάτι νέο και καινοτόμο στους χρήστες, ώστε να τους δώσει ένα κίνητρο να την εγκαταστήσουν και να ασχοληθούν μαζί της. Ακόμα ένα βασικό στοιχείο είναι, η διαφήμιση και η προβολή της εφαρμογής, ώστε να γίνει ευρέως γνωστή.

Φυσικά και πάρα πολλές ιδέες υπάρχουν αλλά και πολλές εφαρμογές πλέον για κάθε ιδέα. Το βασικό ζήτημα όμως είναι η κάθε εφαρμογή να φέρει κάτι νέο και καινοτόμο, ώστε να την κάνει ξεχωριστή και χρήσιμη στους χρήστες.

Έτσι και η εφαρμογή Manage My Money, ενώ βρίσκεται ήδη σε έναν τομέα όπου είναι αρκετά κορεσμένος, αφού ήδη οι περισσότερες τράπεζες έχουν την αντίστοιχη εφαρμογή τους για κινητές συσκευές, ώστε να μπορούν οι χρήστες όχι μόνο να ενημερώνονται, αλλά και να κάνουν τις κινήσεις που επιθυμούν μέσω του κινητού τους, φέρνει κάτι νέο και καινοτόμο. Προσπαθεί να ενοποιήσει όλη αυτή την πληροφορία που έχει από όλες τις τράπεζες και να δώσει μια συνολική εικόνα στον χρήστη για όλες τις κινήσεις που έχει κάνει. Έτσι βλέπει συνοπτικά και σχετικά απλά, όλους τους λογαριασμούς του, όλες τις κινήσεις στον κάθε λογαριασμό, ώστε να είναι ενήμερος κάθε στιγμή, και να μην χρειάζεται να συνδέεται σε 3 ή 4 διαφορετικές εφαρμογές με διαφορετικούς κωδικούς, και διακινδυνεύοντας από επιτήδειους, για να ενημερωθεί για κάποιες κινήσεις που έκανε μέσα στην μέρα ή σε οποιοδήποτε χρονικό διάστημα.

6.2 Μελλοντικές Βελτιώσεις

Τέλος η εφαρμογή Manage My Money, έχει αρκετές λειτουργικότητες και αυτοματισμούς όπου χρειάζονται για μία σύγχρονη εφαρμογή. Όμως σίγουρα μπορεί να βελτιωθεί και άλλο. Έτσι κάποιες μελλοντικές βελτιώσεις που θα μπορούσαν να γίνουν αναλύονται στην συνέχεια.

Αρχικά το πρώτο, που θεωρώ ότι είναι και ένα από τα πιο σημαντικά, είναι το οπτικό κομμάτι της εφαρμογής. Δηλαδή η διεπαφή του χρήστη (interface). Για να επιτευχθεί αυτό, είναι πολύ σημαντικό να υπάρχει βοήθεια από κάποιον γραφίστα, όπου γνωρίζει τους συνδυασμούς των χρωμάτων, αλλά και το τι είναι πιο όμορφο για τον χρήστη αλλά και πιο ‘ξεκούραστο’ για τα μάτια του. Επίσης θα μπορούσε να βοηθήσει και στα σχέδια των κουμπιών αλλά και στην γενικότερη δομή της οθόνης.

Μια ακόμα σημαντική βελτίωση, θα ήταν η συνεργασία με ένα API, με το οποίο θα γίνεται όλη η ενημέρωση. Δηλαδή θα στέλνει τα δεδομένα σωστά και θα ενημερώνει την βάση της εφαρμογής. Υπάρχουν φυσικά ήδη αρκετές υποδομές για την σύνδεση αυτή, αλλά σίγουρα χρειάζεται ακόμα δουλεία ώστε να υπάρξει ένα ολοκληρωμένο αποτέλεσμα.

Τέλος μια ακόμα βελτίωση θα μπορούσε να είναι στον χάρτη όπου ανοίγει στην εφαρμογή, όπου πατώντας πάνω θα έβγαζε την απόσταση όπου είναι μακριά το σημείο ενδιαφέροντος (ATM, ή Τράπεζα), ή κάποια παραπάνω στοιχεία στην περιγραφή, όπως ακριβή διεύθυνση.

Βιβλιογραφία

[1] Mark L. Murphy - COMMONSWARE. (2015). *The Busy Coder's Guide to Android Development v6.5.* USA.

[2] Mew, R. B.-K. (2016). *Android Application Development Cookbook.* BIRMINGHAM - MUMBAI: PACKT.

[3] Friesen, J. (2010). Learn Java for Android Development. Apress.

[4] Felker, D. (2011). *Android Application Development For Dummies.* Indianapolis: Wiley Publishing.

[5] *Android Developers*. (2018). Retrieved from https://developer.android.com/

[6] *Android Studio* . (2018). Retrieved from https://developer.android.com/studio/intro/

[7] *Ericssoners*. (2016). Retrieved from https://ericssoners.wordpress.com/2016/06/13/gs88/

[8] *Wikipedia*. (2018, May). Retrieved from Android (operating system): https://en.wikipedia.org/wiki/Android_(operating_system)

[9] Google Maps Platform. (2018). Ανάκτηση από Maps SDK for Android: https://developers.google.com/maps/documentation/android-sdk/start

[10] *raywenderlich.com*. (2018). Ανάκτηση από Gradle Tutorial for Android: Getting Started: https://www.raywenderlich.com/175940/gradle-build-script-tutorial-android-getting-started

[11] *Firebase Documentation*. (2018). Retrieved from https://firebase.google.com/docs/auth/

[12] *About SQLite*. Retrieved from https://www.sqlite.org/about.html

[13] *Εισαγωγή στο JSON*. Retrieved from https://www.json.org/json-el.html

[14] *Stack Overflow*. Retrieved from www.stackoverflow.com