Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Αθήνα, Νοέμβριος 2017

Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Σχεδίαση και ανάπτυξη Android εφαρμογής για φόρτιση κινητού

Αναστάσιος - Μιλτιάδης Ταπτάς

ΑΜ: ΜΠΣΠ14086

Επιβλέπων: Ευθύμιος Αλέπης

Αθήνα, Φεβρουάριος 2017

Ευχαριστίες

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω την συμφοιτήτριά μου Αριστέα Κοντογιάννη για την άψογη συνεργασία της και την βασική συμβολή της, στα πλαίσια της εκπόνηση αυτής της μεταπτυχιακής διατριβής.

Ακόμα, θα ήθελα να εκφράσω απέραντες ευχαριστίες μου στους γονείς μου Ξενοφών και Ευαγγελία καθώς και στον αδερφό μου Επαμεινώνδα για την στήριξή τους σε όλο το διάστημα ολοκλήρωσης των μεταπτυχιακών σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω τον κ.Ευθύμη Αλέπη για την εμπιστοσύνη που μας έδειξε όσον αφορά τη θεματολογία της μεταπτυχιακής διατριβής και την βοήθεια που μας παρείχε με σκοπό την ολοκλήρωσή της.

Περίληψη

Στα πλαίσια της παρούσης μεταπτυχιακής διατριβής υλοποιήθηκε μια εφαρμογή για Android λογισμικό κινητών τηλεφώνων και tablets, έχοντας σκοπό την κοινωνική δικτύωση των χρηστών με βάση την τοποθεσία τους έτσι ώστε να επιτευχθεί φόρτιση της μπαταρίας της συσκευής με κάθε διαθέσιμο μέσο. Η εφαρμογή (client) επικοινωνεί με έναν εξυπηρετητή (server) μέσω υπηρεσιών διαδικτύου (Web Services), όπου γίνεται η διασύνδεση με την απομακρυσμένη βάση δεδομένων.

Στο παρόν κείμενο παρουσιάζεται η λογική υλοποίησης της εφαρμογής καθώς επίσης και των υπηρεσιών διαδικτύου και της βάσης δεδομένων. Η ανάλυση-παρουσίαση των προαναφερθέντων έχει προγραμματιστική προσέγγιση δίνοντας έμφαση στις τεχνικές υλοποίησης αλλά και στα αποτελέσματα αυτών.

Abstract

Within this master thesis we implemented an Android software application for mobile phones and tablets, having the purpose the social networking of users based on their location in order to achieve charging of the device battery by any means available. The application (client) communicates with a server via web services, which is the interface to the remote database.

This present paper demonstrates the implementation of the application as well as web services and database. The analysis - presentation of the above has programming approach focusing on implementation techniques and the results thereof.

Περιεχόμενα

Ευχαριστίες ... 3
Περίληψη ... 3
Abstract ... 3
Περιεχόμενα ... 4
Εισαγωγή ... 6
Κεφάλαιο 1 Αρχιτεκτονική συστήματος ... 7
1.1 Σχεδίαση Εφαρμογής ... 8
1.2 Αρχείο AndroidManifest ... 8
1.3 Αρχικοποίηση εφαρμογής ... 12
1.4 Βασική οθόνη εφαρμογής ... 14
1.5 Είσοδος / Εγγραφή Χρήστη ... 18
1.6 Χάρτης χρηστών ... 23
1.7 Προφίλ χρήστη ... 30
1.8 Επαφές χρηστών ... 34
1.9 Λεπτομέριες επαφής ... 35
1.10 Αρχείο Singleton ... 37
Κεφάλαιο 2 Web Services ... 39
2.1 Σύνδεση με τη βάση ... 39
2.2 Δημιουργία χρήστη ... 39
2.3 Επαλήθευση χρήστη ... 40
2.4 Χρήστες ... 41
2.5 Προφίλ χρήστη ... 42
2.6 Ενημέρωση πληροφοριών χρήστη ... 42
2.7 Ενημέρωση εικόνας χρήστη ... 42
2.8 Εμφάνιση εικόνας χρήστη ... 43
2.9 Βαθμολογία χρήστη ... 43
2.10 Αλλαγή στοιχείων εισόδου χρήστη ... 43
2.11 Αποσύνδεση χρήστη ... 43
2.12 Επαφές χρήστη ... 44
2.13 Διαγραφή επαφής ... 46
2.14 Αποδοχή αιτήματος ... 46
2.15 Αποστολή ειδοποίησης ... 46
2.16 Αποστολή μηνύματος ... 47
2.17 Μπαταρία ... 47
2.18 Ποσοστό μπαταρίας ... 47
2.19 Αποστολή αιτήματος ... 48

2.20 Φόρτιση 48
2.21 Μη φόρτιση 48
2.22 Τοποθεσία χρηστών 48
2.23 Ενημέρωση τοποθεσίας χρήστη 48
Κεφάλαιο 3 Η βάση δεδομένων 49
3.1 Πίνακας user 49
3.2 Πίνακας phone 50
3.3 Πίνακας rating 50
3.4 Πίνακας charging_options 51
3.5 Πίνακας token 51
3.6 Πίνακας contacts 52
5. Διάγραμμα UML 53
6. Συμπεράσματα 53
7. Βιβλιογραφία 54

Εισαγωγή

Η τελευταία δεκαετία χαρακτηρίζεται αναμφισβήτητα από την έντονη παρουσία των έξυπνων κινητών συσκευών (smartphones – tablets) στη ζωή μας. Οι δυνατότητες που παρουσιάζονται μέσα από την χρήση αυτών ξεφεύγουν κατα πολύ από την απλή χρήση ενός τηλεφώνου ως μέσο επικοινωνίας. Διαφόρων ειδών εφαρμογές χρησιμοποιούνται καθημερινά από εκατομμύρια χρήστες έξυπνων κινητών συσκευών, εξελίσσοντας την επικοινωνία μεταξύ ανθρώπου μηχανής αλλά και με την αντικατάσταση πολλών ανθρωπίνων συνηθειών – συμπεριφορών έχουν αναγάγει τις έξυπνες κινητές συσκευές ως ένα αναπόσπαστο κομμάτι της ανθρώπινης πλεον ζωής.

Παρόλο που η ανάπτυξη των εφαρμογών έχει φτάσει σε αρκετά υψηλό επίπεδο όσον αφορά τις δυνατότητες που προσφέρουν αλλά και τον τρόπο υλοποίησης αυτών, δεν παύουν να καταναλώνουν τους πόρους της συσκευής σε αρκετά μεγάλο βαθμό. Η τεχνολογία έχει καταφέρει να επιλύσει κάποια ζητήματα όσον αφορά την διαχείριση της μνήμης, το μέγεθος του αποθηκευτικού χώρου κ.α αφ ενός με την εξέλιξη του υλικού (hardware) και αφ ετέρου με την ανάπτυξη λογισμικού (software) το οποίο χρησιμοποιεί τους πόρους του υλικού με βέλτιστο τρόπο.

Το βασικό όμως πρόβλημα που παρουσιάζεται ακόμα και στις μέρες μας είναι η διαχείριση της μπαταρίας της συσκευής. Όπως είναι γνωστό οι έξυπνες αυτές συσκευές έχουν περιορισμένους πόρους όσον αφορά το κομμάτι της τροφοδότησης, όπου με την καθημερινή χρήση η αποδοτικότητα και η διάρκεια ζωής της μπαταρίας συνεχώς μειώνεται. Η χρήση διαδικτύου μέσω Wi-Fi αλλά και δεδομένων του παρόχου κινητής τηλεφωνίας, η χρήση του αισθητήρα τοποθεσίας, η χρήση του γυροσκοπικού αισθητήρα, η φωτεινότητα της οθόνης, οι εφαρμογές που λειτουργούν ακόμα και όταν η συσκευή είναι αδρανής είναι μερικοί από τους λόγους που επιβάλλουν την φόρτιση της συσκευής τουλάχιστον μια φορά καθημερινά.

Στα πλαίσια αυτής της μεταπτυχιακής διατριβής προσπαθήσαμε μέσα από την τάση των κοινωνικών δικτύων να δώσουμε μια λύση στο πρόβλημα αυτό. Δημιουργήσαμε μια εφαρμογή με τα γνωστά χαρακτηριστικά ενός κοινωνικού δικτύου όπου οι χρήστες μπορούν να επικοινωνούν μεταξύ τους με σκοπό την φόρτιση της συσκευής τους με όλους τους πιθανούς τρόπους. Η εν λόγο εφαρμογή χρησιμοποιεί την τοποθεσία του χρήστη ανα συγκεκριμένο χρονικό διάστημα, έτσι ώστε να τον ενημερώσει για τους διαθέσιμους χρήστες σε μια περιορισμένη εμβέλεια, τους τρόπους με τους οποιους μπορεί να φορτίσει την συσκευή του αλλά και να επικοινωνήσει μαζί με τους υπόλοιπους χρήστες μέσω μιας chat πλατφόρμας.

Στη συνέχεια θα παρουσιαστεί ο τρόπος ανάπτυξης και σχεδίασης της εφαρμογής αυτής μέσω προγραμματιστικής προσέγγισης, αλλά και των υπηρεσιών διαδικτύου και της βάσης δεδομένων που είναι απαραίτητα για την πλήρη λειτουργικότητα αυτής.

Κεφάλαιο Αρχιτεκτονική συστήματος

Σε αυτό το κεφάλαιο θα γίνει ανάλυση της αρχιτεκτονικής του συστήματος, με επεξήγηση των βασικών κλάσεων της εφαρμογής καθώς και αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν.

Η εφαρμογή υλοποίηθηκε σε γλώσσα προγραμματισμού Java για Android με αρχιτεκτονική σχεδίασης singleton software design pattern και σε περιβάλλον Android Studio το οποίο και αποτελεί το επίσημο περιβάλλον ανάπτυξης λογισμικού Android από την Google.

Ακόμα χρησιμοποιήθηκαν τα παρακάτω 3rd party libraries όπως φαίνεται από τα dependencies του αρχείου gradle της εφαρμογής.

```
dependencies {
    compile files('libs/nineoldandroids-2.4.0.jar')
    compile files('libs/org.apache.http.legacy.jar')
    compile files('libs/volley.jar')
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile 'com.android.support:support-v4:24.2.1'
    compile 'com.google.android.gms:play-services:9.6.1'
    compile 'com.google.firebase:firebase-core:9.6.1'
    compile 'com.google.firebase:firebase-crash:9.6.1'
    compile 'com.google.firebase:firebase-messaging:9.6.1'
    compile 'com.android.support:multidex:1.0.1'
    compile 'com.squareup.picasso:picasso:2.5.2'
    compile 'com.google.android.gms:play-services-ads:9.6.1'
    compile 'com.google.android.gms:play-services-auth:9.6.1'
    compile 'com.google.android.gms:play-services-gcm:9.6.1'
    compile 'com.google.code.gson:gson:2.5'
    compile 'com.google.firebase:firebase-config:9.6.1'
    compile 'com.google.firebase:firebase-database:9.4.0'
    compile 'de.hdodenhof:circleimageview:1.3.0'
    compile 'com.android.support:design:24.2.1'
    compile 'com.firebaseui:firebase-ui-database:0.4.0'
    compile 'com.github.bumptech.glide:glide:3.6.1'
}
```

Τα παραπάνω αφορούν λειτουργίες όπως networking, design, image loading / caching κ.α

Ιδιαίτερη αναφορά πρέπει να γίνει στην πλατφόρμα Firebase η οποία χρησιμοποιήθηκε για crash reporting, push notifications, realtime database και remote configuration.

1.1 Σχεδίαση εφαρμογής

Για την σχεδίαση της εφαρμογής χρησιμοποιήθηκε η τεχνική σχεδίασης singleton όπου ένα αρχείο (κλάση) επιστρέφει το instance του εαυτού του κάθε φορά που καλείται έτσι ώστε διάφορες μεταβλητές και μέθοδοι να είναι διαθέσιμα και προσπελάσιμα από όλο το project. Έτσι μας δίνεται αφ ενός η δυνατότητα για συντήρηση της εφαρμογής σε ένα αρχείο καθώς και σωστή διαχείριση της διαθέσιμης μνήμης όπου δίνεται στο instance της εφαρμογής κατά το ξεκίνημά της.

Περισσότερα για το εν λόγο αρχείο θα αναλυθούν στη συνέχεια.

Ακόμα, η εφαρμογή αποτελείται από 3 Activities και πολλά Fragments. Πιο συγκεκριμένα, από την MainActivity που είναι απαραίτητη για την εκκίνηση της εφαρμογής, την Home όπου πάνω σε εκείνη βρίσκονται τα διάφορα Fragments καθώς και οι απαραίτητοι Adapters για τις λίστες. Επίσης, υπάρχει και η τρίτη Activity, που αφορά την διεπαφή για το Chat.

Τέλος υπάρχει και ένας broadcast receiver όπου παρακολουθεί και στέλνει δεδομένα για το αν το κινητό φορτίζει ή όχι.

1.2 AndroidManifest.xml

Στη συνέχεια παρατίθεται το AndroidManifest.xml αρχείο της παρούσας εφαρμογής, το οποίο βρίσκεται στον manifests/ φάκελο του project.

Το AndroidManifest.xml αποτελεί το ένα και μοναδικό αρχείο που χρειάζεται κάθε Android εφαρμογή για να ξεκινήσει και είναι αυτό που διαβάζεται αρχικά από το Android λειτουργικό έτσι ώστε να ερμηνευθεί ο τρόπος που πρέπει να εκτελεσθεί η εφαρμογή. Παρακάτω παρουσιάζονται εν συντομία οι υπηρεσίες που προσφέρονται μέσω του αρχείου αυτού.

- Ονομάζονται και αναγνωρίζονται τα java πακέτα τις εφαρμογής μέσα στα οποία βρίσκονται οι java κλάσεις.
- Περιγράφονται όλα τα στοιχεία από τα οποία αποτελείται μια εφαρμογή όπως activities, services και broadcast receivers.
- Προσδιορίζονται τα processes που θα χρειαστούν για την υποστήριξη της εφαρμογής.
- Προσδιορίζονται τα δικαιώματα τα οποία πρέπει να έχει η τρέχουσα εφαρμογή έτσι ώστε να είναι εφικτή η αλληλοεπίδραση με άλλες εφαρμογές.
- Ορίζονται τα δικαιώματα που άλλες εφαρμογές πρέπει να έχουν για να χρησιμοποιήσουν την παρούσα εφαρμογή.
- Δηλώνεται το ελάχιστο επίπεδο του Android API που απαιτεί η εφαρμογή.
- Παρατίθενται οι απαραίτητες για την εφαρμογή βιβλιοθήκες.

Δυο απαραίτητα elements τα οποία πρέπει να εμφανίζονται μια μόνο φορά μέσα στο AndroidManifest.xml αρχείο, είναι το manifest και το application. Το application element εμπεριέχεται μέσα στο root element manifest.

Το manifest element του αρχείου AndroidManifest.xml της παρούσας εφαρμογής είναι το εξής:

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

```
    android:versionCode="1"
    android:versionName="1.0" >
```

Εδώ ορίζονται δύο απαραίτητα attributes: το xmlns:android και το package. Το xmlns:android ορίζει το Android namespace και πρέπει πάντα να δείχνει στο «http://schemas.android.com/apk/res/android" ενώ το package attribute ορίζει το java πακέτο μέσα στο οποίο βρίσκεται η εκτελέσιμη java κλάση της εφαρμογής. Στη συγκεκριμένη περίπτωση το : "com.androidunipi" Επιπλέον, εδώ ορίζεται το android:versionCode, ένας εσωτερικός αριθμός που δείχνει την παρούσα έκδοση της εφαρμογής. Το android:versionCode δεν εμφανίζεται στους χρήστες, ενώ το android:versionName είναι ορατό σε αυτούς, και έχει ως στόχο να τους δείχνει την έκδοση της εφαρμογής που τρέχουν.

Στη συνέχεια, στο AndroidManifest.xml χρειάζεται ο καθορισμός στην εφαρμογή δικαιωμάτων πρόσβασης σε λειτουργείες του συστήματος. Ο καθορισμός ενός δικαιώματος γίνεται με την προσθήκη ενός στοιχείου <uses-permission> στο προαναφερθέν αρχείο της εφαρμογής. Ορισμένα από τα permissions που δίνονται είναι τα εξής:

- "android.permission.INTERNET" : Δικαίωμα χρήσης του διαδικτύου.
- "android.permission.ACCESS_NETWORK_STATE": Δικαίωμα ελέγχου της κατάστασης των συνδέσεων της συσκευής, ώστε καθορίσει αν είναι εφικτή η λήψη δεδομένων.
- " com.google.android.providers.gsf.permission.READ_GSERVICES" : Δικαίωμα πρόσβασης στις διαδικτυακές υπηρεσίες της Google.
- "android.permission.WRITE_EXTERNAL_STORAGE" : Δικαίωμα καταχώρισης δεδομένων στον αποθηκευτικό χώρο της συσκευής.
- "android.permission.ACCESS_COARSE_LOCATION" "android.permission.ACCESS_FINE_LOCATION" : Δικαίωμα πρόσβασης στην τοποθεσία.

Παρακάτω παρατίθεται ο κώδικας από τα δικαιώματα που δίνονται στην εφαρμογή.

```
<permission
android:name="com.androidunipi.permission.MAPS_RECEIVE"
android:protectionLevel="signature" />

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
<uses-permission android:name="com.androidunipi.permission.MAPS_RECEIVE"
/>
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"
/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
<uses-permission android:name="android.permission.CALL_PHONE" />

<!-- Required to show current location -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>
```

Ένα άλλο element που τοποθετείται στο AndroidManifest αρχείο είναι το uses-sdk. Ο ρόλος αυτού του element είναι να τηρεί την συμβατότητα της εφαρμογής με μία ή περισσότερες εκδόσεις της πλατφόρμας του Android. Το «android:minSdkVersion» θέτει τον ακέραιο αριθμό που ορίζει το

ελάχιστο επίπεδο API που απαιτείται για να τρέξει η εφαρμογή. Το λειτουργικό Android θα απαγορέψει το χρήστη να εγκαταστήσει οποιαδήποτε εφαρμογή αν το επίπεδο του API του συστήματος είναι χαμηλότερο από την τιμή που καθορίζεται στην «android:minSdkVersion» παράμετρο. Η άλλη παράμετρος που συναντάται εδώ, το «android:targetSdkVersion» ορίζει το επίπεδο API στο οποίο στοχεύει η εφαρμογή. Στη παρούσα εφαρμογή τα παραπάνω πεδία ορίζονται ως εξής:

```
<uses-sdk
    android:minSdkVersion="16"
    android:targetSdkVersion="25" />
```

Το application αποτελεί ένα ακόμη xml element του AndroidManifest αρχείου. Αυτό το element περιέχει σημαντικά στοιχεία που επηρεάζουν όλα τα κύρια χαρακτηριστικά της Android εφαρμογής όπως το εικονίδιο της εφαρμογής, την άδεια, την ασφάλεια λειτουργίας της, και πολλές άλλες προκαθορισμένες τιμές. Πιο συγκεκριμένα, το «android:allowBackup», έχοντας την τιμή “True” επιτρέπει στον χρήστη της εφαρμογής να κάνει backup την ίδια την εφαρμογή για να μπορέσει να την ανακτήσει ξανά. Στο πεδίο «android:icon» ορίζεται το path το εικονίδιο της εφαρμογής ενώ το όνομα της εφαρμογής ορίζεται στο «android:label» attribute. Τέλος, το «android:theme» Αναφέρεται στο default theme της εφαρμογής. Παρακάτω παρουσιάζεται το application element της παρούσας εφαρμογής.

```
<application
    android:name="com.androidunipi.model.AppController"
    android:allowBackup="true"
    android:icon="@drawable/app_bat"
    android:label="@string/app_name"
    android:theme="@style/AppTheme"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|u
iMode|screenSize" >
```

Το παρακάτω element είναι απαραίτητο καθώς ορίζει ότι η εφαρμογή απαιτεί OpenGL ES έκδοση 2. Εξωτερικές υπηρεσίες μπορούν να ανιχνεύσουν αυτήν την ειδοποίηση και να ενεργήσουν αναλόγως, όπως η υπηρεσία χαρτών της Google.

```
<!-- Required OpenGL ES 2.0. for Maps V2 -->
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />
```

Ένα άλλο element του αρχείου αυτού, είναι το activity element. Σε αυτό ορίζεται η java activity κλάση που είναι υπεύθυνη για την λειτουργία της εφαρμογής και για την δημιουργία του visual interface που θα εμφανιστεί στην οθόνη της συσκευής.

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Material.Light"
    android:screenOrientation="portrait"
    tools:ignore="NewApi">
    <intent-filter>
```

```
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Επιπλέον, στην εφαρμογή γίνεται έλεγχος για το αν το κινητό φορτίζει ή όχι. Σε περίπτωση που κάποιος χρήστης φορτίζει, δεν είναι διαθέσιμος να προσφέρει μπαταρία σε άλλους χρήστες. Για να επιτευχθεί αυτός ο έλεγχος, ορίζεται στο αρχείο αυτό ένας receiver που παρουσιάζεται παρακάτω.

```
<receiver android:name=".PowerConnectionReceiver">
    <intent-filter>
        <action
android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action
android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
    </intent-filter>
</receiver>
```

Τέλος, στο παρακάτω element ορίζεται το api key, δηλαδή το κλειδί το οποίο έχει αποκτηθεί από το Google APIs Console και είναι απαραίτητο για την χρήση του χάρτη.

```
      <!-- Google API Key -->
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyDWKW3JxFCQhoMLieNsSAAnzQrX7WE9EDw" />
```

1.3 Αρχικοποίηση εφαρμογής

Η αρχική κλάση της εφαρμογής, αυτή που καλείται κατά την εκκίνηση, είναι η MainActivity, η οποία παρατίθεται και παρακάτω. Με τη μέθοδο onCreate(), η οποία χρησιμοποιείται στην αρχή κάθε Activity, ορίζεται μέσω του setContentView() το xml αρχείο που αντιστοιχεί στην κλάση αυτή. Συγκεκριμένα στην MainActivity εμφανίζεται στον χρήστη το γραφικό της λεγόμενης Splash Screen αλλά παράλληλα εκτελλούνται διάφοροι έλεγχοι σχετικά με την σωστή λειτουργίας της εφαρμογής.

Αρχικά γίνεται έλεγχος για το αν η εφαρμογή μέσα από τον αισθητήρα τοποθεσίας μπορεί να πάρει το γεωγραφικό πλάτος / ύψος της συσκευής καθώς και περίπτωση όπου ο χρήστης δεν έχει ενεργοποιήσει τον συγκεκριμένο αισθητήρα, εμφανίζεται σχετικό μήνυμα και γίνεται παραπομπή του χρήστη για την ενεργοποίηση αυτού.

```
if(gps.canGetLocation()){
     mylocation=  gps.getLocation();

     latitude = gps.getLatitude();
     longitude = gps.getLongitude();

     common.latitude = String.valueOf(latitude);
     common.longitude = String.valueOf(longitude);

     checkGPS = 1;
```

Στη συνέχεια γίνεται αντίστοιχος έλεγχος και για την σύνδεση με το διαδίκτυο με παρόμοιο τρόπο όπως παραπάνω. Αν οι 2 αυτοί έλεγχοι περάσουν τότε η εφαρμογή χρησιμοποιόντας ένα animation μεταφέρεται στην Home όπου και αποτελεί την βασική κλάση πάνω στην οποία πατάνε όλα τα διαθέσιμα Fragments. Σε αυτό το σημείο η MainActivity τερματίζεται.

```
      //network check
if(common.NetworkExists(MainActivity.this)){

          new Handler().postDelayed(new Runnable() {
              @Override
              public void run() {
                  final Intent mainIntent = new Intent(MainActivity.this,
Home.class);
                  startActivity(mainIntent);
                  overridePendingTransition(R.animator.maximize,
R.animator.minimize);
                  finish();
              }
          }, 3000);
}
else{
   @SuppressWarnings("deprecation")
         AlertDialog.Builder builder = new AlertDialog.Builder(this,
```

```
AlertDialog.THEME_HOLO_LIGHT);
         builder.setTitle(R.string.app_name).setMessage("No network.")
           .setCancelable(false)
           .setPositiveButton("The app will close.", new
DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                    dialog.cancel();
                    MainActivity.this.finish();
               }
           })

          .setNegativeButton("Connect me.", new
DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int id) {
                  startActivityForResult(new
Intent(Settings.ACTION_WIFI_SETTINGS), 1);
               }
          });
           alert = builder.create();
           alert.show();
           dialog.ChangeDialogUI(alert);
}
```

1.4 Βασική οθόνη της εφαρμογής

Η Home, αποτελεί ένα Activity στο οποίο εκτελούνται πολύ βασικές λειτουργίες της εφαρμογής, και στο οποίο φορτώνονται τα εκάστοτε Fragments, ανάλογα με τις ενέργειες του χρήστη. Αρχικά στο Activity αυτό, γίνεται έλεγχος για το αν ο χρήστης είναι συνδεδεμένος. Σε περίπτωση που είναι, εμφανίζεται το κύριο μενού της εφαρμογής, το οποίο αποτελείται από τρεις επιλογές πλοήγισης:

- Profile
- Map
- Contacts

οι οποίες αναπαρίστανται γραφικά, και φορτώνεται ένα Fragment το οποίο by default είναι το Map fragment. Επιλέγοντας μια από τις παραπάνω επιλογές, ο χρήστης μπορεί να πλοηγηθεί στα τρία βασικά Fragments, το οποία θα παρουσιαστούν πιο αναλυτικά στη συνέχεια.

Σε περίπτωση που δεν είναι συνδεδεμένος κάποιος χρήστης στην εφαρμογή, το προαναφερθέν μενού γίνεται αόρατο και φορτώνεται το LoginFragment, από όπου ο χρήστης έχει τη δυνατότητα είτε να συνδεθεί, αν έχει ήδη λογαριασμό, είτε να δημιουργήσει καινούργιο account.

```
if(common.is_logged){
   //load home fragment and make bottom bar visible

   bottom_bar.setVisibility(View.VISIBLE);
   tv_icon_home.setBackgroundResource(R.drawable.prof_inactive);
   tv_map.setBackgroundResource(R.drawable.map_act);
   tv_icon_contacts.setBackgroundResource(R.drawable.contacts_inactive);
   rel_map.setEnabled(false);
   Map fragment = new Map();
   FragmentTransaction ft = getFragmentManager().beginTransaction();
   ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
   ft.replace(R.id.frame_container, fragment).commit();
}
else{
   //load login/register fragment and make bottom bar gone
   bottom_bar.setVisibility(View.GONE);

   fragmentManager.popBackStack();
   fragmentManager.popBackStack(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);

   LoginFragment fragment = new LoginFragment();
   FragmentTransaction ft = getFragmentManager().beginTransaction();
   ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
   ft.replace(R.id.frame_container, fragment).commit();

}
```

Μία ακόμη σημαντική λειτουργία που λαμβάνει χώρα σε αυτό το Activity, είναι ο έλεγχος της μπαταρίας του χρήστη σε σχέση με αυτή της βάσης. Κάθε ένα συγκεκριμένο χρονικό διάστημα «delay», γίνεται ένας έλεγχος με την τρέχουσα μπαταρία του χρήστη, και με αυτή που είναι αποθηκευμένη στη βάση και, ως εκ τούτου, ορατή από τους άλλους χρήστες. Αν η σύγκριση δώσει αποτέλεσμα μεγαλύτερο από το 5%, η βάση ενημερώνεται. Με τον τρόπο αυτό, και η εφαρμογή δεν επικοινωνεί συνέχεια με τον server με αποτέλεσμα να μην καταναλώνεται συνεχώς η μπαταρία σε network requests αλλά και η βάση δεν έχει μεγάλη απόκλιση από την αλήθεια στις εγγραφές της. Παρόμοια λειτουργία υπάρχει και για τη γεωγραφική τοποθεσία του χρήστη. Ένας έλεγχος που συμβαίνει ανά ένα μικρό χρονικό διάστημα, προσδιορίζει το αν έχει αλλάξει η τοποθεσία του χρήστη σε σχέση με αυτή της βάσης και το αν θα πρέπει η τελευταία να ενημερωθεί.

```
h = new Handler();
delay = Integer.parseInt(remoteConfig.getString("delay"));

final Handler handler = new Handler();
Runnable runnable = new Runnable() {
   public void run() {
      if(delay != 0 && common.is_logged) {
         new CheckBatteryOn().execute();
         new CheckLocationOn().execute();
      }
      handler.postDelayed(this, delay);
   }
};
```

Εδώ, έχει γίνει χρήση του Firebase, έτσι ώστε να είναι εφικτή η τροποποίηση του χρόνου που γίνονται οι προαναφερθέντες έλεγχοι, χωρίς να πρέπει να τροποποιηθεί ο κώδικας της εφαρμογής.

```
FirebaseRemoteConfigSettings remoteConfigSettings = new
FirebaseRemoteConfigSettings.Builder()
      .setDeveloperModeEnabled(BuildConfig.DEBUG)
      .build();

remoteConfig = FirebaseRemoteConfig.getInstance();
remoteConfig.setConfigSettings(remoteConfigSettings);
remoteConfig.setDefaults(R.xml.remote_config_defaults);

fetch();
```

Η μέθοδος fetch() είναι υπεύθυνη για την επικοινωνία με την πλατφόρμα της Firebase όπου και επιστρέφεται η τιμή για την μεταβλητή delay που αφορά τον χρόνο όπου η εφαρμογή κοιτάζει για αλλαγές στην μπαταρία αλλά και στην τοποθεσία του χρήστη.

Ακόμα η μέθοδος appControl() είναι υπεύθυνη για το κλείσιμο ή όχι της εφαρμογής κατά το πάτημα του back κουμπιού της συσκευής. Όπως αναφέρθηκε και παραπάνω η εφαρμογή χρησιμοποιεί κυρίως Fragments όπου χρείαζεται ιδιαίτερος χειρισμός αυτών κατά το πάτημα του back button. Έτσι λοιπόν υλοποιήθηκε μια δικλίδα ασφαλείας όπου ο χρήστης όταν πατήση 2 συνεχόμενες φορές το back button εμφανίζεται παράθυρο επιβεβαίωσης για έξοδο από την εφαρμογή.

```
public void appControl(){
   common.back++;

   if(common.back == 2){
      @SuppressWarnings("deprecation")
      AlertDialog.Builder builder = new AlertDialog.Builder(this,
AlertDialog.THEME_HOLO_LIGHT);
      builder.setTitle(R.string.app_name).setMessage("Are you sure you
what to exit the app?")
            .setCancelable(false)
            .setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                  Home.this.finish();
               }
            })
            .setNegativeButton("No", new
DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                  common.back = 0;
                  dialog.cancel();
               }
            });
      AlertDialog alert = builder.create();
      alert.show();
      dialog.ChangeDialogUI(alert);
   }
   else{
      if(common.current_fragment.equals("Register")){
         common.back = 0;
         super.onBackPressed();
      }
   }
}
```

Τέλος, η Home ως βασική Activity είναι υπεύθυνη και για τη διαχείριση των push notifications που έρχονται από την πλατφόρμα της Firebase και αφορούν το Chat. Αν υπάρξει τέτοια περίπτωση ο χρήστης μεταφέρεται αυτομάτως στην ChatActivity.

```
Bundle extras = getIntent().getExtras();

if (extras != null) {
   for (String key : getIntent().getExtras().keySet()) {
      if(key.equals("ID")){
         common.myNotification = (NotificationObj)
getIntent().getExtras().get(key);
         if(common.myNotification.isChat()) {
            final Intent mainIntent = new Intent(this,
ChatActivity.class);
            mainIntent.putExtra("data", common.myNotification);
            startActivity(mainIntent);
         }
      }
   }
}
```

Σχετικά για την γραφική απεικόνιση της κλάσης, όλα τα στοιχεία του αρχείου αυτού, περικλείονται σε ένα «RelativeLayout» και η διάταξή τους είναι κάθετη. Αρχικά, ένα «FrameLayout» χρησιμοποιείται για να περιέχει το fragment ου φορτώνεται κάθε φορά ανάλογα με τις ενέργειες του χρήστη.

Στη συνέχεια δημιουργείται το bottom navigation bar, το οποίο αποτελεί το κύριο μενού της εφαρμογής. Εδώ χρησιμοποιείται ένα «LinearLayout» με «orientation: Horizontal» , καθώς η διάταξη των στοιχείων στο μενού είναι οριζόντια. Επιπλέον, για την σχεδίαση του κάθε στοιχείου του μενού, χρησιμοποιείται ένα View και ένα Textview, τοποθετημένα μέσα σε ένα «RelativeLayout». Με τα παραπάνω επιτυγχάνεται ο κύριος σχεδιασμός των οθονών της παρούσας εφαρμογής.

1.5 Είσοδος / Εγγραφή Χρήστη

Σε περίπτωση που ο χρήστης δεν έχει συνδεθεί στην εφαρμογή μέσω της συσκευής στην οποία «τρέχει» η εφαρμογή, τότε φορτώνεται το LoginFragment.

Ένα Fragment αποτελεί ένα τμήμα της διεπαφής χρήστη σε ένα Activity. Μπορεί να θεωρηθεί ως ένα κομμάτι από ένα Activity, το οποίο έχει τον δικό του κύκλο ζωής, έχει τα δικά του inputs, τις δικές του λειτουργίες και το οποίο μπορεί να προστεθεί ή να αφαιρεθεί ενώ το activity βρίσκεται σε λειτουργία.

Ένα Fragment είναι πάντα ενσωματωμένο σε ένα Activity και ο κύκλος ζωής του επηρεάζεται άμεσα από τον κύκλο ζωής του Activity αυτού . Για παράδειγμα, όταν ένα Activity είναι σε παύση, έτσι είναι και όλα τα Fragment σε αυτό, ενώ όταν το Activity καταστρέφεται, το ίδιο συμβαίνει και σε όλα το Fragments που περιέχει.

Η κλάση LoginFragment ορίζει στην αρχή της ότι αποτελεί ένα Fragment κάνοντας extend το Fragment:

```
public class LoginFragment extends Fragment {
```

Στην κλάση αυτή, ο χρήστης έχει την δυνατότητα ή να συνδεθεί, σε περίπτωση που έχει ήδη λογαριασμό στην εφαρμογή, ή να πατήσει το κουμπί «Register» για να μεταβεί στη κλάση RegisterFragment.java.

Για να συνδεθεί στην εφαρμογή, έχοντας ήδη λογαριασμό, πρέπει να εισάγει το email και το password και στη συνέχεια να πατήσει το κουμπί Login. Όταν ο χρήστης πληκτρολογεί το email και, γίνονται οι παρακάτω ενέργειες στην κλάση. Ίδιος τρόπος χρησιμοποιείται και για το password.

```
et_mail.addTextChangedListener(new TextWatcher() {

   @Override
   public void onTextChanged(CharSequence s, int start, int before, int
count) {
      email = et_mail.getText().toString();
   }

   @Override
   public void beforeTextChanged(CharSequence s, int start, int count,
                           int after) {
      // TODO Auto-generated method stub
   }

   @Override
   public void afterTextChanged(Editable s) {
      // TODO Auto-generated method stub

   }
});
```

Με το που πατηθεί το κουμπί Login, γίνεται έλεγχος έτσι ώστε να επιβεβαιωθεί ότι και τα δύο προαναφερθέντα πεδία έχουν συμπληρωθεί, και σε περίπτωση που αυτό έχει γίνει καλείται και εκτελείται η Login που επεκτείνει την AsyncTask κλάση . Σε περίπτωση που δεν έχουν συμπληρωθεί όλα ή κάποιο από τα πεδία, εμφανίζεται το ανάλογο μήνυμα στον χρήστη. Παρακάτω παρατίθεται ο προαναφερθέν κώδικας.

```
btn_login.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      // TODO Auto-generated method stub
      //regex for email
      if(email.equals("")){
         if(password.equals("")){
            Toast.makeText(myActivity, "Please fill all
feilds",Toast.LENGTH_SHORT).show();
         }else {
            Toast.makeText(myActivity, "Email field cannot be
empty",Toast.LENGTH_SHORT).show();
         }
      }else if(password.equals("")){
         Toast.makeText(myActivity, "Password field cannot be
empty",Toast.LENGTH_SHORT).show();
      } else{
         //execute async
         new Login().execute();
      }
   }
});
```

Στη παρούσα εφαρμογή η επικοινωνία με τον server εκτελείται με τη χρήση ασύγχρονων κλάσεων, οι οποίες κληρονομούν από την κλάση AsyncTask. Με τον τρόπο αυτό, επιτυγχάνεται η εκτέλεση ενός παράλληλου νήματος για την λειτουργία της επικοινωνίας, χωρίς να επηρεάζεται η εφαρμογή.

Η Login αποτελεί μια τέτοια ασύγχρονη κλάση. Κατά την εκτέλεση της κλάσης αυτής, ένα μήνυμα ανάδρασης εμφανίζεται στον χρήστη που τον ενημερώνει ότι εκτελείται μια διαδικασία στο background. Η διαδικασία αυτή, λαμβάνει ορισμένα στοιχεία του χρήστη:

```
protected String doInBackground(String... args) {
   List<NameValuePair> params = new ArrayList<NameValuePair>();

   params.add(new BasicNameValuePair("email", email));
   params.add(new BasicNameValuePair("password", password));
   params.add(new BasicNameValuePair("longitude", common.longitude));
   params.add(new BasicNameValuePair("latitude", common.latitude));
   params.add(new BasicNameValuePair("token",
FirebaseInstanceId.getInstance().getToken()));

   JSONObject jObj = jsonParser.makeHttpRequest(url_check_user, "POST",
params, "OBJECT");
```

και τα αποστέλλει στον server. Πέρα από το email και τον κωδικό πρόσβασης, αποστέλλεται το token που παράγεται μέσω του Firebase και είναι απαραίτητο για τις ειδοποιήσεις και τα μηνύματα, καθώς και η τοποθεσία του χρήστη. Το php αρχείο που στη συγκεκριμένη περίπτωση βρίσκεται στην διεύθυνση:

```
private static String url_check_user =
"http://androidun.eu.pn/android/verify_user.php";
```

Λαμβάνει τα στοιχεία με τη μέθοδο «Post», ελέγχει αν υπάρχει πράγματι χρήστης στη βάση με αυτό το email και password και στέλνει ένα json response πίσω στην εφαρμογή. Στην ερχόμενη ενότητα θα παρουσιαστούν όλα τα php αρχεία και οι λειτουργίες που επιτελούν αναλυτικά.

Το πεδίο «user_code» που επιστρέφεται στο json response ορίζει αν υπάρχει χρήστης με αυτά τα credentials. Σε περίπτωση που user_code =1 , υπάρχει χρήστης με αυτά τα στοιχεία, οπότε ορισμένα στοιχεία του αποθηκεύονται στην εφαρμογή για μελλοντική χρήση και φορτώνεται το Map Fragment. Σε κάθε άλλη περίπτωση, η διαδικασία σύνδεσης δεν ήταν επιτυχής και ο χρήστης λαμβάνει το αντίστοιχο μήνυμα ανάδρασης.

Στο LoginFragment, αντιστοιχεί ένα xml αρχείο, το οποίο αποτελείται από 2 buttons, ένα για την εκτέλεση του Login και ένα για την μεταφορά του χρήστη στο Register Fragment, 2 editTexts για το email και για το password καθώς και κάποια textviews για τους τίτλους και διάφορα views.

Σε περίπτωση που ο χρήστης επιλέξει το κουμπί «Register», φορτώνεται «RegisterFragment» όπου ο χρήστης καλείται να δημιουργήσει ένα νέο λογαριασμό. Πιο συγκεκριμένα, καλείται να εισάγει τα παρακάτω προσωπικά στοιχεία:

- Όνομα
- Επώνυμο
- Email
- Password
- Όνομα Χρήστη
- Ηλικία
- Φύλλο
- Επιλογές φόρτισης που προσφέρει

Είναι υποχρεωτική η εισαγωγή των παραπάνω πεδίων, καθώς, σε περίπτωση που κάποιο πεδίο μείνει κενό, εμφανίζεται στον χρήστη το ανάλογο μήνυμα. Το κομμάτι του κώδικα της κλάσης αυτής που διεξάγει τον προαναφερθέν έλεγχο, είναι το εξής

Αν ο παραπάνω έλεγχος δε δείξει κάποιο πρόβλημα στα στοιχεία που εισήγαγε ο χρήστης καλείται η ασύγχρονη κλάση. Ενα μήνυμα ανάδρασης εμφανίζεται στον χρήστη που τον ενημερώνει ότι εκτελείται μια διαδικασία στο background, όπως και προηγουμένως, και τα στοιχεία που εισήγαγε ο χρήστης αποστέλλονται με τη μέθοδο GET στον server έτσι ώστε να εισαχθούν τα απαραίτητα πεδία στη βάση, αφού γίνει έλεγχος για το αν ήδη υπάρχει χρήστης με αυτό το email. Πρέπει να τονιστεί εδώ ότι το email του χρήστη πρέπει να είναι πάντα μοναδικό.

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

   @Override
   public void onClick(View v) {
      // TODO Auto-generated method stub
      if(charger.isChecked())
      {
         ch="1";
      }else{
         ch="0";
      }
      if(powerbag.isChecked())
      {
         p="1";
      }else{
         p="0";
      }
      if(usb.isChecked())
      {
         u="1";
      }else{
         u="0";
      }
      if(usb_otg.isChecked())
      {
         uo="1";
      }else{
         uo="0";
      }


      if(name.equals("")){

if(surname.equals("")||email.equals("")||username.equals("")||password.eq
uals("")||gender.equals("")||age.equals("")){
            Toast.makeText(myActivity, "All feilds are
mandatory",Toast.LENGTH_SHORT).show();
         }else{
            Log.e("error", "no name inserted ");
            Toast.makeText(myActivity, "Name field cannot be
empty",Toast.LENGTH_SHORT).show();
         }
      }else if(surname.equals("")){

if(email.equals("")||username.equals("")||password.equals("")||gender.equ
als("")||age.equals("")){
            Toast.makeText(myActivity, "All feilds are
mandatory",Toast.LENGTH_SHORT).show();
         }else{
            Toast.makeText(myActivity, "Surname field cannot be
empty",Toast.LENGTH_SHORT).show();
         }
      }else if(email.equals("")){
```

```
if(username.equals("")||password.equals("")||gender.equals("")||age.equal
s("")){
            Toast.makeText(myActivity, "All feilds are
mandatory",Toast.LENGTH_SHORT).show();
         }else{
            Toast.makeText(myActivity, "Email field cannot be
empty",Toast.LENGTH_SHORT).show();
         }
      }else if(username.equals("")){
         if(password.equals("")||gender.equals("")||age.equals("")){
            Toast.makeText(myActivity, "All feilds are
mandatory",Toast.LENGTH_SHORT).show();
         }else{
            Toast.makeText(myActivity, "Username field cannot be
empty",Toast.LENGTH_SHORT).show();
         }
      }else if(password.equals("")){
         if(gender.equals("")||age.equals("")){
            Toast.makeText(myActivity, "All feilds are
mandatory",Toast.LENGTH_SHORT).show();
         }else{
            Toast.makeText(myActivity, "Password field cannot be
empty",Toast.LENGTH_SHORT).show();
         }
      }else if(gender.equals("")){
         if(age.equals("")){
            Toast.makeText(myActivity, "All feilds are
mandatory",Toast.LENGTH_SHORT).show();
         }else{
            Toast.makeText(myActivity, "Gender field cannot be
empty",Toast.LENGTH_SHORT).show();
         }
      }else if(age.equals("")){
         Toast.makeText(myActivity, "Age field cannot be
empty",Toast.LENGTH_SHORT).show();
      }else{
         new Register().execute();
      }
```

Στη συνέχεια, το web service επιστρέφει ένα json response ώστε να επιβεβαιωθεί ή όχι η επιτυχία της δημιουργίας του νέου χρήστη. Οι πιθανές περιπτώσεις είναι η επιτυχία, η αποτυχία λόγω κάποιου λάθους από το service ή να υπάρχει ήδη χρήστης με αυτό το email. Αν η εγγραφή είναι επιτυχής, αποθηκεύονται ορισμένα στοιχεία του χρήστη στην εφαρμογή για μελλοντική χρήση και φορτώνεται το Map Fragment, όπως ακριβώς συμβαίνει και στο LoginFragment όταν ο χρήστης συνδέεται επιτυχώς.

```
Map fragment = new Map();
FragmentTransaction ft = getFragmentManager().beginTransaction();
ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
ft.replace(R.id.frame_container, fragment).commit();
```


Εικόνα 1: Διεπαφή Register

1.6 Χάρτης χρηστών

Το Map Fragment αποτελεί τη κύρια οθόνη της εφαρμογής, αυτή που θα κατευθυνθεί ο χρήστης με το που συνδεθεί ή μόλις δημιουργήσει λογαριασμό. Στη κλάση αυτή της εφαρμογής, προβάλεται ο χάρτης του Google Maps προκειμένου να παρουσιαστεί η θέση του χρήστη στο χάρτη και, με το πάτημα ενός κουμπιού, να φορτώσει τους χρήστες γύρω του.

Το Google Maps API αποτελεί μια διαδικτυακή εφαρμογή υπηρεσιών, η οποία παρέχεται από την Google, όπως δηλώνει και το όνομά της, και προσφέρει οδικούς χάρτες και υπηρεσίες πλοήγησης τόσο σε δικτυακούς τόπους όσο και σε εφαρμογές Android. Ορισμένες από τις υπηρεσίες που προσφέρονται είναι οι εξής:

- οδικοί χάρτες
- πλοήγηση διαδρομής με μια πληθώρα μέσων μεταφοράς
- εντοπισμός επιχειρήσεων, υπηρεσιών, πανεπιστήμια κλπ

Επιπλέον, αποτελεί μια δωρεάν για εμπορική χρήση διαδικτυακή εφαρμογή με ορισμένους όρους, όπως για παράδειγμα το site ή η εφαρμογή στην οποία χρησιμοποιείται είναι προσβάσιμο στο κοινό χωρίς χρέωση για κάθε πρόσβαση.

Για τις εφαρμογές Android χρησιμοποιείται το Google Maps Android API v2, μια ειδικά σχεδιασμένη για έκδοση του API.

```
if (googleMap == null) {
   googleMap = getMapFragment();
   googleMap.getMapAsync(new OnMapReadyCallback() {
      @Override
      public void onMapReady(GoogleMap googleMap) {
      googleMap.setMapType(common.MAP_TYPE);
      CameraPosition cameraPosition = new
CameraPosition.Builder().target(new LatLng(latitude,
longitude)).zoom(17).build();

googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosit
ion));
      if (ActivityCompat.checkSelfPermission(mActivity,
android.Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(mActivity,
android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
         return;
      }
      else{
         googleMap.setMyLocationEnabled(true);
      }
      myGoogleMap = googleMap;
      myGoogleMap1 = googleMap;
      }
   });
```

Στην οθόνη αυτή, όπως προαναφέρθηκε, υπάρχει ένα κουμπί το οποίο μόλις πατηθεί φορτώνει όλους τους χρήστες σε μια συγκεκριμένη χιλιομετρική απόσταση από τον χρήστη. Η απόσταση αυτή είναι, by default, ένα χιλιόμετρο αλλά μπορεί να αλλάξει από τα settings του χρήστη. Μόλις πατηθεί το κουμπί, καλείται η ασύγχρονη κλάση LoadUsersOnMap. Σε ένα web service αποστέλλεται με τη μέθοδο POST το email και η χιλιομετρική απόσταση στην οποία θα εντάσσονται οι χρήστες.

```
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("email", common.token_email));
params.add(new BasicNameValuePair("target", common.target_distance));
JSONObject json = jsonParser.makeHttpRequest(
"http://androidun.eu.pn/android/return_users.php", "POST", params,
"OBJECT");
```

Το json response που λαμβάνεται, περιέχει τους χρήστες που φορτώθηκαν από το πάτημα του κουμπιού χωρισμένους σε τέσσερις κατηγορίες, ανάλογα με τον αν είναι υπάρχουσες επαφές, αν είναι άγνωστοι ή αν είναι αιτήματα φιλίας που έχει λάβει ή έχει αποστείλει ο χρήστης. Με τον τρόπο αυτό, επιτυγχάνεται η διαφορετική χρωματική απεικόνιση των δεικτών στο χάρτη που αντιπροσωπεύουν τις τοποθεσίες των χρηστών.

```
for( int i = 0 ; i < common.friends_on_map.size() ; i++){
   Double lat =
Double.valueOf(common.friends_on_map.get(i).getLatitude());
   Double lon =
Double.valueOf(common.friends_on_map.get(i).getLongitude());

   String full_name = common.friends_on_map.get(i).getName()+"
"+common.friends_on_map.get(i).getSurname();

   MarkerOptions markerfriend = new MarkerOptions().position(new
LatLng(lat, lon)).title(full_name);

markerfriend.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorF
actory.HUE_GREEN));
   myGoogleMap1.addMarker(markerfriend).showInfoWindow();
}
```

Επιπλέον, οι markers στον χάρτη που αντιστοιχούν στους χρήστες, όταν κλικαριστούν, εμφανίζουν το όνομα και ο επώνυμο του χρήστη. Στη συνέχεια, αν αυτό πατηθεί, φορτώνεται το Fragment με το προφίλ του χρήστη. Πιο συγκεκριμένα, φορτώνεται ένα διαφορετικό Fragment ανάλογα με την ιδιότητα του χρήστη.

```
myGoogleMap.setOnInfoWindowClickListener(new
GoogleMap.OnInfoWindowClickListener() {

   @Override
   public void onInfoWindowClick(Marker arg0) {
      String titleToCheck = arg0.getTitle();
      for(int i = 0 ; i< common.unknown_on_map.size() ; i++){

         String name_to_find = common.unknown_on_map.get(i).getName()+"
```

```
"+common.unknown_on_map.get(i).getSurname();
         if(titleToCheck.equals(name_to_find)){

            if(titleToCheck.equals(name_to_find)){

               String email = common.unknown_on_map.get(i).getEmail();
               common.contact_email = email;
               UnknownUserFragment fragment = new UnknownUserFragment();
               FragmentTransaction ft =
getFragmentManager().beginTransaction();
               ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
               ft.replace(R.id.frame_container, fragment).commit();

            }

         }

      }
```

Στη διεπαφή του Map, όπως προαναφέρθηκε, εμφανίζεται και ο αριθμός των αδιάβαστων ειδοποιήσεων του χρήστη. Όταν το γραφικό που τις παρουσιάζει πατηθεί, τότε φορτώνεται το NotificationsFragment. Το Fragment περιέχει μια λίστα στην οποία φορτώνονται όλες οι ειδοποιήσεις.

```
//check the notification list for unread notifications
SharedPreferences sharedPrefs =
PreferenceManager.getDefaultSharedPreferences(mActivity);
Gson gson = new Gson();
String json = sharedPrefs.getString("Notifications", null);
if(json != null) {
   Type type = new TypeToken<ArrayList<NotificationObj>>() {}.getType();
   ArrayList<NotificationObj> arrayList = gson.fromJson(json, type);
   int counter = 0;
   for (int i = 0 ; i < arrayList.size() ; i++) {
      if(!arrayList.get(i).isRead()) {
         counter++;
      }
   }

   common.tvBadge.setText(counter != 0 ? ""+counter : "");

}
```

Εδώ δημιουργείται ένα Arraylist από ένα object το NotificationObj, το οποίο έχει δημιουργηθεί ώστε να γίνεται set και get όλων των απαραίτητων δεδομένων για το Notification.

```
notifications.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      NotificationsFragment fragment = new NotificationsFragment();
      FragmentTransaction ft = getFragmentManager().beginTransaction();
      ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
      ft.replace(R.id.frame_container, fragment).commit();

   }
});
```

Επιπλέον, έχει δημιουργηθεί ένας adapter, ο οποίος κληρονομεί από τον BaseAdapter για την εμφάνιση κάθε εγγραφής στην λίστα. Σε περίπτωση που δεν υπάρχει καμία ειδοποίηση, τότε η λίστα γίνεται αόρατη ενώ εμφανίζεται ένα μήνυμα στη διεπαφή : «There are no notifications at the moment». Ειδάλλως, παρουσιάζονται τα notification στη λίστα με χρήση του adapter.

```
NotificationAdapter.ViewHolder holder = (NotificationAdapter.ViewHolder)
myView.getTag();
final NotificationObj notification = list.get(i);

holder.message.setTypeface(notification.isRead() ? Typeface.DEFAULT :
Typeface.DEFAULT_BOLD);

String my_message= notification.getMessage()+"
"+notification.getUserName()+" "+notification.getsurName();
holder.message.setText(my_message);
holder.date.setText(notification.getDate());
holder.time.setText(notification.getTime());
if(notification.getUserPhoto()==null) {
    Picasso.with(myActivity).load(R.drawable.user).into(holder.photo);
}else {

Picasso.with(myActivity).load(notification.getUserPhoto()).into(holder.ph
oto);
}
```

Οι ειδοποιήσεις που στέλνονται από χρήστη σε χρήστη, γίνονται μέσω του Firebase, πιο συγκεκριμένα με χρήση της κλάσης MessagingService που κληρονομεί το FirebaseMessagingService, και μπορεί να είναι δύο ειδών. Είτε battery request, όπως φαίνεται παραπάνω, είτε chat request. Παρακάτω παρατίθεται το κομμάτι του κώδικα της κλάσης Messaging Service που είναι υπεύθυνο για τη δημιουργία ειδοποιήσεων.

Δημιουργία Notification

```
//make the notification
NotificationObj currNotification = new NotificationObj();
currNotification.setId(arrayList.size() == 0 ? 0 : arrayList.size()+1);
currNotification.setUserId(0);
currNotification.setEmail(remoteMessage.getData().get("email"));
currNotification.setDate(remoteMessage.getData().get("date"));
currNotification.setTime(remoteMessage.getData().get("time"));
currNotification.setUserName(remoteMessage.getData().get("name"));
currNotification.setSurName(remoteMessage.getData().get("surname"));
currNotification.setUserPhoto(remoteMessage.getData().get("img"));
currNotification.setMessage(remoteMessage.getNotification().getBody());
currNotification.setRead(false);
if(remoteMessage.getData().get("chat_token") != null){

currNotification.setChat_token(remoteMessage.getData().get("chat_token"))
;
    currNotification.setChat(true);
}
else{
    currNotification.setChat(false);
}
```

Εμφάνιση Notification

```
Intent intent = new Intent(this, Home.class);
intent.putExtra("ID", currNotification);
PendingIntent pIntent = PendingIntent.getActivity(this, (int)
System.currentTimeMillis(), intent, 0);

NotificationCompat.Builder builder = new
NotificationCompat.Builder(getApplicationContext());
Notification notification = builder.setContentTitle("BattBook")
        .setContentText(remoteMessage.getNotification().getBody())
        .setSmallIcon(R.drawable.app_bat_icon)
        .setContentIntent(pIntent).build();

notification.defaults |= Notification.DEFAULT_SOUND;
notification.defaults |= Notification.DEFAULT_VIBRATE;
notification.flags |= Notification.FLAG_SHOW_LIGHTS;
notification.flags |= Notification.FLAG_AUTO_CANCEL;

NotificationManager notificationManager = (NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(0, notification);
```

Επιπλέον, όταν ο χρήστης λάβει μια ειδοποίηση για battery request, έχει τη δυνατότητα είτε να την αγνοήσει, είτε να τη διαγράψει, είτε να ξεκινήσει μια συνομιλία με τον χρήστη που του απέστειλε την ειδοποίηση. Αν επιλεχθεί να ξεκινήσει μια συνομιλία, τότε σε περίπτωση που η εφαρμογή είναι κλειστή,μια ειδοποίηση αποστέλλεται στον χρήστη που είχε στείλει το αρχικό αίτημα για μπαταρία, αλλιώς απλά φορτώνεται μια διεπαφή για το chat.

Η κλάση ChatActivity, η οποία κληρονομεί από την κλάση AppCompatActivity υλοποιεί το chat. Ο κώδικας της κλάσης αυτής είναι δημιουργημένος με βάση τα πρότυπα που προσφέρει το Firebase. Καθώς η συνομιλία περιορίζεται ανάμεσα σε δύο χρήστες, δημιουργείται κατά την εκκίνηση μιας νέας συνομιλίας ένα chat_token που κάνει τη συνομιλία ανάμεσα στους δύο αυτούς χρήστες μοναδική.

Χρησιμοποιώντας το χαρακτηριστικό RealTime Database της πλατφόρμας Firebase έχουμε με έλαχιστες γραμμές κώδικα το ιστορικό της συνομιλίας μεταξύ των δυο χρηστών αλλά και άμεση ενημέρωση της απομακρισμένης βάσης.

```
mFirebaseDatabaseReference =
FirebaseDatabase.getInstance().getReference();
Query items =
mFirebaseDatabaseReference.child(MESSAGES_CHILD).orderByChild("authIDs").
equalTo(dataObj.getChat_token());
```

```
new SendNotification().execute();
MessageObj friendlyMessage = new
MessageObj(mMessageEditText.getText().toString(), mUsername,
           mPhotoUrl, dataObj.getChat_token());
mFirebaseDatabaseReference.child(MESSAGES_CHILD).push().setValue(friendly
Message);
mMessageEditText.setText("");
```

Για την απεικόνιση χρησιμοποιήθηκε αντί για ListView και BaseAdapter, RecyclerView και RecyclerViewAdapter.

```
mFirebaseAdapter = new FirebaseRecyclerAdapter<MessageObj,
MessageViewHolder>(
        MessageObj.class,
        R.layout.item_message,
        MessageViewHolder.class,
        items) {

    @Override
    protected void populateViewHolder(MessageViewHolder viewHolder,
MessageObj friendlyMessage, int position) {
        mProgressBar.setVisibility(ProgressBar.INVISIBLE);
        viewHolder.messageTextView.setText(friendlyMessage.getText());
        viewHolder.messengerTextView.setText(friendlyMessage.getName());
        if (friendlyMessage.getPhotoUrl() == null) {

viewHolder.messengerImageView.setImageDrawable(ContextCompat.getDrawable(
ChatActivity.this,
                    R.drawable.ic_account_circle_black_36dp));
        } else {
            Glide.with(ChatActivity.this)
                    .load(friendlyMessage.getPhotoUrl())
                    .into(viewHolder.messengerImageView);
```

```
        }
    }
};

mFirebaseAdapter.registerAdapterDataObserver(new
RecyclerView.AdapterDataObserver() {
    @Override
    public void onItemRangeInserted(int positionStart, int itemCount) {
        super.onItemRangeInserted(positionStart, itemCount);
        int friendlyMessageCount = mFirebaseAdapter.getItemCount();
        int lastVisiblePosition =
mLinearLayoutManager.findLastCompletelyVisibleItemPosition();
        // If the recycler view is initially being loaded or the user is
at the bottom of the list, scroll
        // to the bottom of the list to show the newly added message.
        if (lastVisiblePosition == -1 ||
                (positionStart >= (friendlyMessageCount - 1) &&
lastVisiblePosition == (positionStart - 1))) {
            mMessageRecyclerView.scrollToPosition(positionStart);
        }
    }
});
```

1.7 Προφίλ χρήστη

Στη κλάση Profile που, επίσης, κληρονομεί από την κλάση Fragment, γίνονται οι βασικές ενέργειες που αφορούν το προσωπικό προφίλ του χρήστη και τις προσωπικές του ρυθμίσεις στην εφαρμογή.

Η ασύγχρονη κλάση LoadProfile αποστέλλει με τη μέθοδο GET σε ένα web service το email του χρήστη και λαμβάνει ένα json response με όλα τα χαρακτηριστικά του προφίλ του. Στη συνέχεια, στη διεπαφή χρήστη αυτά παρουσιάζονται. Στο αρχείο profile_fragment.xml, το οποίο αντιστοιχεί στην java κλάση Profile, ορίζεται τα πεδία όπου θα φορτωθούν τα στοιχεία του χρήστη ως EditText. Με τον τρόπο αυτό, είναι επεξεργάσιμα από τον χρήστη. Παρακάτω δίνεται ένα μέρος του xml κώδικα ως παράδειγμα.

```
<RelativeLayout
android:id="@+id/rel_main3"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@+id/view2"
android:orientation="horizontal" >
 <TextView
     android:id="@+id/title_username"
     android:layout_width="120dp"
        android:layout_height="35dp"
     android:paddingTop="6dip"
     android:textStyle="bold"
     android:textSize="17sp"
     android:textColor="#125688"
     android:layout_marginLeft="5dp"
     android:text="@string/Username"/>

 <EditText
     android:id="@+id/username"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:layout_alignParentTop="true"
     android:layout_toRightOf="@+id/title_username"
     android:paddingTop="6dip"
     android:textSize="17sp"
     android:background="#FFFFFF"
     android:textColor="#000000"
     android:textStyle="bold" />

</RelativeLayout>
```

Επιπλέον, υπάρχει ένα button: “Save Changes” το οποίο όταν επιλεχθεί καλείται η ασύγχρονη κλάση SaveUsersDetails. Στη κλάση αυτή, αποστέλλονται τα τροποποιημένα στοιχεία του χρήστη σε ένα web service με τη μέθοδο «POST» έτσι ώστε να τροποποιηθεί η βάση και να εισαχθούν τα νέα χαρακτηριστικά του χρήστη.

```
// Building Parameters
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("name", user_name));
params.add(new BasicNameValuePair("surname", user_surname));
params.add(new BasicNameValuePair("username", user_username));
params.add(new BasicNameValuePair("gender", user_gender));
params.add(new BasicNameValuePair("age", user_age));
params.add(new BasicNameValuePair("email", common.token email));

JSONObject json = jsonParser.makeHttpRequest(update_user_info,
        "POST", params, "OBJECT");
```

Ας τονιστεί εδώ, ότι τα στοιχεία που μπορεί να τροποποιήσει σε αυτή την οθόνη δε συμπεριλαμβάνουν ούτε τους τρόπους φόρτισης που προσφέρει ούτε τα διαπιστευτήριά του. Αυτά αλλάζουν μέσω των ρυθμίσεων του χρήστη που θα παρουσιαστούν στη συνέχεια.

Στο Fragment αυτό, ο χρήστης έχει επίσης τη δυνατότητα να αλλάξει την εικόνα του προφίλ του είτε διαλέγοντας μια εικόνα αποθηκευμένη στο κινητό του, είτε βγάζοντας μια νέα φωτογραφία. Παρακάτω παρατίθεται ο κώδικας που το καθιστά αυτό δυνατό.

Ανάλογα με την επιλογή του χρήστη, όταν ο χρήστης επιλέξει να αλλάξει φωτογραφία προφίλ, η μεταβλητή «requestCode» μπορεί να έχει δύο τιμές: requestCode == REQUEST_CAMERA ή requestCode == GET_FROM_GALLERY.

```
if (requestCode == REQUEST_CAMERA) {

    thumbnail = (Bitmap) data.getExtras().get("data");
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    thumbnail.compress(Bitmap.CompressFormat.JPEG, 90, bytes);

    File destination = new
File(Environment.getExternalStorageDirectory(),
System.currentTimeMillis() + ".jpg");
    FileOutputStream fo;

    byte[] byte_arr = bytes.toByteArray();

    image_str = Base64.encodeToString(byte_arr, Base64.DEFAULT);
    //img.setImageBitmap(thumbnail);
    Picasso.with(myActivity).load(user_image).into(img);
    new SaveUsersImage().execute();


    try {
        destination.createNewFile();
        fo = new FileOutputStream(destination);
        fo.write(bytes.toByteArray());
        fo.close();
```

```
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

} else if (requestCode == GET_FROM_GALLERY) {

    Uri selectedImageUri = data.getData();
    String[] projection = {MediaColumns.DATA};
    Cursor cursor = myActivity.managedQuery(selectedImageUri, projection,
null, null,
            null);
    int column_index = cursor.getColumnIndexOrThrow(MediaColumns.DATA);
    cursor.moveToFirst();
    String selectedImagePath = cursor.getString(column_index);
    Bitmap bm;
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(selectedImagePath, options);
    final int REQUIRED_SIZE = 200;
    int scale = 1;
    while (options.outWidth / scale / 2 >= REQUIRED_SIZE
            && options.outHeight / scale / 2 >= REQUIRED_SIZE)
        scale *= 2;
    options.inSampleSize = scale;
    options.inJustDecodeBounds = false;
    bm = BitmapFactory.decodeFile(selectedImagePath, options);
    // img.setImageBitmap(bm);
    Picasso.with(myActivity).load(user_image).into(img);

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bm.compress(Bitmap.CompressFormat.JPEG, 90, baos);
    byte[] imageBytes = baos.toByteArray();
    image_str = Base64.encodeToString(imageBytes, Base64.DEFAULT);
    new SaveUsersImage().execute();

}
```

Στο Profile Fragment, επιπλέον, ο χρήστης έχει την δυνατότητα να αλλάζει τις ρυθμίσεις του. Στο profile_fragment.xml ορίζεται η διεπαφή των ρυθμίσεων με τέτοιο τρόπο έτσι ώστε να μην είναι ορατή στην οθόνη με το που φορτωθεί το Profile. Οι ρυθμίσεις γίνονται ορατές μόνο όταν ο χρήστης σύρει το δάχτυλό του στην οθόνη από το αριστερά προς τα δεξιά, ή όταν πατήσει το εικονίδιο των ρυθμίσεων που βρίσκεται στη διεπαφή του προφιλ.

Στη Profile.java, στην κλάση onCreateView() ορίζεται η λειτουργικότητα των settings και πιο συγκεκριμένα στον παρακάτω κώδικα. Όπως φαίνεται, οι ρυθμίσεις εμφανίζονται στη διεπαφή με τους δύο τρόπους που αναφέρθηκαν προηγουμένως.

```
mDrawerLayout = (DrawerLayout) rootView.findViewById(R.id.drawerLayout);

// Populate the Navigtion Drawer with options
mDrawerPane = (RelativeLayout) rootView.findViewById(R.id.drawerPane);
mDrawerLayout.setDrawerListener(mDrawerToggle);
// initializing navigation menu
mDrawerToggle = new ActionBarDrawerToggle(myActivity, mDrawerLayout,
R.string.drawer_open, R.string.drawer_close) {
    @Override
    public void onDrawerOpened(View drawerView) {
        super.onDrawerOpened(drawerView);

        invalidateOptionsMenu();
    }

    @Override
    public void onDrawerClosed(View drawerView) {
        super.onDrawerClosed(drawerView);
        //Log.d( "onDrawerClosed: " + getTitle());

        invalidateOptionsMenu();
    }
};
```

1.8 Επαφές χρηστών

Η κλάση Contacts.java κληρονομεί επίσης από την κλάση Fragment. Στη κλάση αυτή φορτώνονται τρεις λίστες που αντιστοιχούν στις κατηγορίες των χρηστών : φίλους, απεσταλμένα στον χρήστη αιτήματα φιλίας καθώς και απεσταλμένα από τον χρήστη αιτήματα φιλίας.

```
      listView    = (ListView) rootView.findViewById(R.id.list);
      list_req    = (ListView) rootView.findViewById(R.id.list_req);
      list_my_req =(ListView) rootView.findViewById(R.id.list_my_req);
```

Για την εμφάνιση των χρηστών στην εκάστοτε λίστα, χρησιμοποιούνται τρεις adapters:

- ContactsAdapter: Αφορά τις ήδη υπάρχουσες επαφές του χρήστη.
- RequestsAdapter: Για τη λίστα με τα απεσταλμένα στον χρήστη αιτήματα φιλίας
- MyRequestsAdapter: Για τα απεσταλμένα από τον χρήστη αιτήματα φιλίας.

Πατώντας σε κάθε εγγραφή στη λίστα, ο χρήστης μεταβαίνει στο προφίλ του χρήστη στον οποίο αντιστοιχεί η εγγραφή. Ανάλογα με την κατηγορία στην οποία ανήκει ο χρήστης ένα διαφορετικό Fragment φορτώνεται. Για την επικοινωνια των Adapters με το Fragment έχουν υλοποιηθεί αντίστοιχα interfaces και μέθοδοι, όπου παρουσιάζονται παρακάτω.

ContactsAdapter

```
interface onContactClickedListener {
    void onContactClicked(String email);
}
```

MyRequestsAdapter

```
interface onRequestClickedListener {
    void onRequestClicked(String email);
}
```

RequestsAdapter

```
interface onMyRequestClickedListener {
    void onMyRequestClicked(String email);
}
```

Contacts

```
@Override
public void onContactClicked(String email) {
   common.contact_email = email;
   ContactsDetailsFragment fragment = new ContactsDetailsFragment();
   FragmentTransaction ft = getFragmentManager().beginTransaction();
   ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
   ft.replace(R.id.frame_container, fragment).commit();
}
```

```
@Override
public void onRequestClicked(String email) {
   common.contact_email = email;
   ContactRequestFragment fragment = new ContactRequestFragment();
   FragmentTransaction ft = getFragmentManager().beginTransaction();
   ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
   ft.replace(R.id.frame_container, fragment).commit();
}

@Override
public void onMyRequestClicked(String email) {
   common.contact_email = email;
   MyRequestFragment fragment = new MyRequestFragment();
   FragmentTransaction ft = getFragmentManager().beginTransaction();
   ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
   ft.replace(R.id.frame_container, fragment).commit();
}
```

Για την φόρτωση των χρηστών χρησιμοποιείται η ασύγχρονη κλάση «AllUsers». Το email του χρήστη αποστέλλεται σε ένα web service με τη μέθοδο GET, και λαμβάνεται ένα json response με τους χρήστες χωρισμένους στις τρεις προαναφερθείς κατηγορίες. Αξίζει να σημειωθεί εδώ ότι η λίστα των επαφών ταξινομείται με βάση τα πόσα αστεράκια έχει ο κάθε χρήστης. Με άλλα λόγια, οι επαφές με υψηλότερη βαθμολογία τοποθετούνται πιο ψηλά από αυτές με χαμηλότερη. Έτσι επιτυγχάνεται μια ιεράρχηση των χρηστών.

1.9 Λεπτομέρειες επαφής

Το ContactsDetailsFragment υλοποιεί την εμφάνιση όλων τον χαρακτηριστικών του χρήστη, καθώς και τη δυνατότητα αποστολής αιτήματος για μπαταρία.

Πιο συγκεκριμένα, μια ασύγχρονη κλάση, LoadProfile, αποστέλλει με τη μέθοδο GET σε δυο web services το email του χρήστη, και λαμβάνει σε ένα json response όλα τα προσωπικά του στοιχεία καθώς και την βαθμολογία του και τα εμφανίζει μέσω του contactsdetails_fragment.xml στον χρήστη.

Επιπλέον, εδώ δίνεται η δυνατότητα βαθμολόγησης του χρήστη. Απλά κλικάροντας από ένα έως πέντε αστέρια, η βαθμολογία αποστέλλεται σε ένα web service, αποθηκεύεται στη βάση και η νέα βαθμολογία γίνεται διαθέσιμη μέσω ενός json response στον χρήστη.

```
class LoadProfile extends AsyncTask<String, String, String>
class SendNotification extends AsyncTask<String, String, String>
class RateUser extends AsyncTask<String, String, String>
class DeleteContact extends AsyncTask<String, String, String>
```

Τα στοιχεία που εμφανίζονται στη διεπαφή χρήστη είναι:

- Το όνομα και το επώνυμο
- Η ηλικία
- Το μοντέλο του κινητού με το οποίο συνδέεται
- Το ποσοστό της μπαταρίας που έχει ο χρήστης
- Οι τρόποι φόρτισης που προσφέρει
- Η βαθμολογία του και ο αριθμός των χρηστών που τον έχουν βαθμολογήσει.

Σε περίπτωση που κάποιος χρήστης θέλει να ζητήσει μπαταρία, επιλέγει το κουμπί: «Request Battery». Τότε θα αποσταλεί σε ένα web service το token του χρήστη που ζητάει μπαταρία μαζί με τα στοιχεία του, καθώς και το token του χρήστη από τον οποίο ζητάει. Σε περίπτωση που κάποιος χρήστης φορτίζει, η επιλογή «Request Battery» είναι απενεργοποιημένη. Ο έλεγχος για το αν ο χρήστης φορτίζει ή όχι γίνεται μέσω ενός receiver ο οποίος παρατίθεται στη συνέχεια. Ανάλογα με το αν φορτίζει, ενημερώνεται η βάση δεδομένων μέσω ενός web service έτσι ώστε αυτό να είναι ορατό στους υπόλοιπους χρήστες.

```
public class PowerConnectionReceiver extends BroadcastReceiver {
    CommonData common;
    JSONParser jsonParser = new JSONParser();
    public void onReceive(Context context , Intent intent) {
        common = CommonData.getInstance();
        String action = intent.getAction();

        if(action.equals(Intent.ACTION_POWER_CONNECTED)) {
            new ChangeChargingState().execute();
        }
        else if(action.equals(Intent.ACTION_POWER_DISCONNECTED)) {
            new NoCharging().execute();
        }
    }
    class ChangeChargingState extends AsyncTask<String, String, String> {
        @Override
        protected String doInBackground(String... args) {
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("email",
common.token_email));
            JSONObject json = jsonParser.makeHttpRequest(
                    "http://androidun.eu.pn/android/charging.php", "GET",
params, "OBJECT");
            return null;
        }
    }
    class NoCharging extends AsyncTask<String, String, String> {
        @Override
        protected String doInBackground(String... args) {
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("email",
common.token_email));
            JSONObject json = jsonParser.makeHttpRequest(
                    "http://androidun.eu.pn/android/no_charging.php",
"GET", params, "OBJECT");
            return null;
```

1.10 CommonData.java

Τέλος, θα παρουσιαστεί ένα ιδιαίτερα βασικό αρχείο της παρούσας εφαρμογής, το CommonData.java. Στο αρχείο αυτό ορίζονται όλα τα απαραίτητα για την εφαρμογή δεδομένα, τα οποία αποθηκεύονται και χρησιμοποιούνται από όλες σχεδόν τις κλάσεις. Το αρχείο περιέχει πολλές μεταβλητές που για λόγους χώρου παραλείπονται. Θα παρουσιαστούν όμως οι μέθοδοι.

Singleton Pattern

```
private static CommonData mInstance = null;
//fragments - activities
public static synchronized CommonData getInstance(){

   if(null == mInstance){
      mInstance = new CommonData();
   }
   return mInstance;
}
```

Network Check

```
public Boolean NetworkExists(Activity ctx){
   myActivity = ctx;

   boolean haveConnectedWifi = false;
   boolean haveConnectedMobile = false;
   ConnectivityManager cm = (ConnectivityManager)
ctx.getSystemService(Activity.CONNECTIVITY_SERVICE);
   NetworkInfo[] netInfo = cm.getAllNetworkInfo();

   for (NetworkInfo ni : netInfo) {
      if (ni.getTypeName().equalsIgnoreCase("WIFI"))
         if (ni.isConnected())
            haveConnectedWifi = true;
      if (ni.getTypeName().equalsIgnoreCase("MOBILE"))
         if (ni.isConnected())
            haveConnectedMobile = true;
   }
   return haveConnectedWifi || haveConnectedMobile;
}
```

Get Device Name

```
public String getDeviceName() {
   String manufacturer = Build.MANUFACTURER;
   String model = Build.MODEL;
   if (model.startsWith(manufacturer)) {
      return capitalize(model);
   } else {
      return capitalize(manufacturer) + " " + model;
   }
}
```

String Capitalization

```
private String capitalize(String s) {
   if (s == null || s.length() == 0) {
      return "";
   }
   char first = s.charAt(0);
   if (Character.isUpperCase(first)) {
      return s;
   } else {
      return Character.toUpperCase(first) + s.substring(1);
   }
}
```

Κεφάλαιο 2 Web Services

Όπως προαναφέρθηκε, για την επικοινωνία με τον server χρησιμοποιείται η αρχιτεκτονική REST (Representational State Transfer). Η εφαρμογή στέλνει κωδικοποιημένα δεδομένα μέσω HTTP POST ή GET μεθόδους προς την υπηρεσία διαδικτύου (web service) και, στη συνέχεια, η υπηρεσία διαδικτύου λαμβάνει αυτά τα δεδομένα, εκτελεί sql queries και επιστρέφει τα δεδομένα που διάβασε από την βάση και το αποτέλεσμα από το query που εκτέλεσε με τη μορφή JSON.

Στη συνέχεια, παρουσιάζονται τα php αρχεία που χρησιμοποιούνται στη παρούσα εφαρμογή για την επικοινωνία με τον server.

2.1 Σύνδεση με τη βάση

Σε αυτό το αρχείο php πραγματοποιείται η σύνδεση με τη βάση. Αποτελεί το πιο βασικό αρχείο το οποίο γίνεται «include» σε όλα τα υπόλοιπα αρχεία για να έχουν πρόσβαση στα δεδομένα της βάσης. Η σύνδεση με την MySQL βάση γίνεται με PDO (PHP Data Objects). Σε περίπτωση που αλλάξει η βάση, απλά χρειάζεται η τροποποίηση των στοιχείων στο αρχείο αυτό.

```
<?php
$dbusername="name";
$dbpassword="pass";
try {
   $conn = new PDO('mysql:host=fdb3.freehostingeu.com;dbname=db_android', $dbusername, $dbpassword);
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
   echo 'ERROR: ' . $e->getMessage();
}
?>
```

2.2 Δημιουργία χρήστη

Στο αρχείο αυτό πραγματοποιείται η δημιουργία νέου χρήστη. Τα στοιχεία που εισάγει ο χρήστης κατά την δημιουργία νέου λογαριασμού, αποστέλλονται με τη μέθοδο GET στο παρόν αρχείο, το οποίο εκτελεί ένα query και τα αποθηκεύει στη βάση. Ένα μέρος του κώδικα παρατίθεται στη συνέχεια.

```
try {
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
 //Prepared Statement for inserting new user into user table
 $result = $conn->prepare('INSERT INTO user VALUES(null, :name, :surname, :email, :username, :password, :gender, :age, now(), :longitude, :latitude, :active, :img, :rate, :charging)');

 //Bind  data
 $result->bindParam(':name', $name);
```

```
 $result->bindParam(':surname', $surname);
 $result->bindParam(':email', $email);
 $result->bindParam(':username', $username);
 $result->bindParam(':password', $hashPassword);
 $result->bindParam(':gender', $gender);
 $result->bindParam(':age', $age);
 $result->bindParam(':latitude', $latitude);
 $result->bindParam(':longitude', $longitude);
 $result->bindParam(':active', $active);
 $result->bindParam(':img', $myimage, PDO::PARAM_LOB);
 $result->bindParam(':rate', $rate);
 $result->bindParam(':charging', $charging);

 //Execute the query
 $result->execute();
$insert_id = $conn->lastInsertId();

} catch(PDOException $e) {
  echo $e->getMessage();
}
```

2.3 Επαλήθευση χρήστη

Αυτό το αρχείο χρησιμοποιείται όταν ένας ήδη υπάρχον χρήστης προσπαθεί να συνδεθεί στην εφαρμογή. Μέσω μιας μεθόδου Post, αποστέλλονται στο web service τα απαραίτητα στοιχεία του χρήστη και, ανάλογα με το αν αυτά είναι σωστά και αντιστοιχούν σε κάποιον αποθηκευμένο στη βάση χρήστη λαμβάνεται το ανάλογο json response. Παρακάτω παρατίθεται ένα παράδειγμα από απόκριση από τον server σε περίπτωση που είναι έγκυρα τα στοιχεία του χρήστη.

```
{
  "user_code": 1,
  "user_email": "ak@ak.com",
  "user_name": "Αριστεα",
  "user_surname": "Κοντογιάννη",
  "user_img": "http://androidun.eu.pn/android/showimage.php?id=86"
}
```

2.4 Χρήστες

Αυτό το αρχείο καλείται όταν φορτώνονται οι χρήστες στο χάρτη εντός ορισμένης χιλιομετρικής απόστασης.

Πιο συγκεκριμένα, με μια μέθοδο POST αποστέλλεται το email και η απόσταση στην οποία είναι επιθυμητό να βρίσκονται οι χρήστες, και λαμβάνεται στην εφαρμογή ένα json response με αυτούς. Για την εύρεση μόνο των χρηστών εντός της απόστασης αυτής, χρησιμοποιείται ο παρακάτω κώδικας:

```
//Get target distance in miles

$target_distance = $target*0.621371192;

$find = $conn->prepare('SELECT *, ( 3959 * acos( cos( radians(:base_lat) ) * cos( radians( latitude ) ) * cos( radians( longitude ) - radians(:base_lng) ) + sin( radians(:base_lat) ) * sin( radians( latitude ) ) ) ) AS distance
FROM user
WHERE latitude IS NOT NULL
AND longitude IS NOT NULL
AND id!= :id
HAVING distance < :target_distance
ORDER BY distance ASC');

 $find->bindParam(':base_lat', $base_lat);
 $find->bindParam(':base_lng', $base_lng);
 $find->bindParam(':target_distance', $target_distance);
 $find->bindParam(':id', $id);

 $find->execute();
```

Στη συνέχεια, οι χρήστες κατηγοριοποιούνται ανάλογα άμα είναι φίλοι, άγνωστοι ή αίτημα φιλίας σε τέσσερις κατηγορίες.

2.5 Προφιλ χρήστη

Αυτό το αρχείο είναι υπεύθυνο για την επιστροφή στην εφαρμογή όλων των στοιχείων σχετικών με τον χρήστη. Με βάση το email του χρήστη, εκτελείται ένα query στη βάση για την επιστροφή των στοιχείων. Παρακάτω παρουσιάζεται το json που επιστρέφεται από αυτό το web service.


```
{
  "users": [
    {
      "id": "86",
      "name": "Αριστεα",
      "surname": "Κοντογιάννη",
      "gender": "Female",
      "age": "25",
      "username": "karistea",
      "active": "1",
      "charging": "0",
      "img": "http://androidun.eu.pn/android/showimage.php?id=86",
      "charger": "1",
      "powerbag": "1",
      "usb": "1",
      "usb_otg": "0",
      "battery": "15",
      "phone": "Samsung GT-I9505"
    }
  ],
  "success": 1
}
```


2.6 Ενημέρωση πληροφοριών χρήστη

Αυτό το php αρχείο είναι υπεύθυνο για την αλλαγή των προσωπικών στοιχείων του χρήστη. Αυτά αποστέλλονται μέσω της μεθόδου POST στο web service και άμα δεν υπάρξει κάποιο πρόβλημα επιστρέφεται ένα json response με success:1 αλλιώς το response είναι success:0. Έτσι ο χρήστης λαμβάνει την ανάλογη ανάδραση από την εφαρμογή.

2.7 Ενημέρωση εικόνας χρήστη

Αυτό το php αρχείο χρησιμοποιείται για την αλλαγή της φωτογραφίας του χρήστη. Και εδώ αποστέλλεται ένα json response που υποδηλώνει την επιτυχία ή την αποτυχία της ενέργειας του χρήστη, με στόχο την ανάλογη ανάδραση από την εφαρμογή.

2.8 Εμφάνιση εικόνας χρήστη

Το αρχείο αυτό χρησιμοποιείται για την ανάκτηση και εμφάνιση της εικόνας προφίλ του χρήστη. Αυτό το επιτυγχάνει λαμβάνοντας με τη μέθοδο POST το id του χρήστη. Άλλα αρχεία php, όπως το profile.php περιλαμβάνουν αυτό το αρχείο στο json response που αποστέλλουν στην εφαρμογή με το εκάστοτε id του χρήστη.

2.9 Βαθμολογία χρήστη

Το αρχείο αυτό, λαμβάνει το email του χρήστη και επιστρέφει την βαθμολογία του, καθώς και τον αριθμό τον ψήφων που έχει λάβει.

2.10 Αλλαγή στοιχείων εισόδου χρήστη

Το web service αυτό, όπως υποδεικνύει και το όνομά του, χρησιμοποιείται για να αλλάξουν το διαπιστευτήρια του χρήστη, σε περίπτωση που θέλει να προβεί σε τέτοια ενέργεια. Όταν αποθηκεύεται κάποιος κωδικός πρόσβασης στη βάση γίνεται η παρακάτω κρυπτογράφηση για επιπλέον ασφάλεια:

```
function cryptPass($input,$rounds=9){
  $salt="";

  //With range we create three random arrays and then we merge them into one
  $saltChars = array_merge(range('A','Z'),range('a','z'),range('0','9'));
  for($i=0;$i<22;$i++){

    //We randomise the saltChars array and store the result in parameter salt
    $salt .=$saltChars[array_rand($saltChars)];
  }
  //Crypt is used for string hashing
  //$2y$ is going to "tell" crypt that we are using the blowfish algorithm
  return crypt($input,sprintf('$2y$%02d$',$rounds) . $salt);
}
```

2.11 Αποσύνδεση

Με το αρχείο αυτό ο χρήστης αποσυνδέεται από την εφαρμογή. Πιο συγκεκριμένα, στη βάση, υπάρχει ένα πεδίο στον πίνακα «user» που υποδεικνύει αν ο χρήστης είναι online ή offline. Το πεδίο αυτό ονομάζεται active και μπορεί να έχει τιμή 1 ή 0 ανάλογα. Όταν ο χρήστης αποσυνδέεται το αρχείο logout.php αλλάζει το πεδίο active του χρήστη σε 0.

2.12 Επαφές χρήστη

Αυτό το web service χρησιμοποιείται για να επιστρέψει όλους τους χρήστες που ανήκουν σε μία απί τις παρακάτω κατηγορίες:

- Επαφές
- Αιτήματα φιλίας που έχουν αποσταλεί στον χρήστη
- Αιτήματα φιλίας που έχουν αποσταλεί από τον χρήστη

Στη συνέχεια παρουσιάζεται ένα json response που λαμβάνεται από αυτό το web service.


```
{
  "users": [
    {
      "id": "55",
      "name": "Tasos",
      "surname": "Taptas",
      "username": "DeTapt",
      "email": "t.taptas1@gmail.com",
      "gender": "Male",
      "age": "26",
      "active": "1",
      "rate_user": "5",
      "img": "http://androidun.eu.pn/android/showimage.php?id=55",
      "charger": "1",
      "powerbag": "1",
      "usb": "1",
      "rate": 5
    },
    {
      "id": "87",
      "name": "Αριστεα",
      "surname": "Κοντογιάννη",
      "username": "aristea",
      "email": "a@hotmail.com",
      "gender": "Female",
      "age": "26",
      "active": "1",
      "rate_user": "5",
      "img": "http://androidun.eu.pn/android/showimage.php?id=87",
      "charger": "1",
      "powerbag": "1",
      "usb": "1",
      "rate": 5
    },
```

```
 ],
 "request": [
  {
    "id": "90",
    "name": "Konstantinos ",
    "surname": "Bachlavas ",
    "username": "Valdegart ",
    "email": "kbachla@yahoo.com",
  "gender": "Male",
    "age": "28",
    "active": "1",
    "img": "http://androidun.eu.pn/android/showimage.php?id=90",
    "charger": "1",
    "powerbag": "0",
    "usb": "0"
  }
 ],
 "myrequest": [
  {
    "id": "61",
    "name": "kon",
    "surname": "bach",
    "username": "kon",
    "email": "bach@hotmaol.com",
    "gender": "Male",
    "age": "28",
    "active": "1",
    "img": "http://androidun.eu.pn/android/showimage.php?id=61",
    "charger": "1",
    "powerbag": "1",
    "usb": "1"
  }
 ],
 "success": 1,
 "message": "Friends Found",
 "success_r": 1,
 "message_r": "Requests Found",
 "my_success_r": 1,
 "my_message_r": "Requests Found"
}
```

2.13 Διαγραφή επαφής

Το αρχείο αυτό χρησιμοποιείται για την διαγραφή ενός χρήστη από τις παρακάτω λίστες που παρουσιάζονται στη κλάση Contacts.java.

- Επαφές
- Αιτήματα φιλίας που έχουν αποσταλεί στον χρήστη
- Αιτήματα φιλίας που έχουν αποσταλεί από τον χρήστη

Με άλλα λόγια, διαγράφεται ο συσχετισμός που έχουν οι χρήστες μεταξύ τους.

2.14 Αποδοχή αιτήματος

Όταν ο χρήστης δέχεται ένα αίτημα φιλίας, τότε αποστέλλεται το email του μαζί με το email του χρήστη σε αυτό το web service έτσι ώστε να ενημερωθεί η βάση και να είναι πλέον «φίλοι». Η εφαρμογή ενημερώνεται για την επιτυχία της ενέργειας αυτής με ένα json response.

2.15 Αποστολή ειδοποίησης

Το web service αυτό είναι υπεύθυνο για την αποστολή ειδοποιήσεων σε έναν χρήστη από έναν άλλο.Ο κώδικας του αρχείου, που έχει δημιουργηθεί με βάση τα πρότυπα του Firebase παρατίθεται στη συνέχεια.

```
<?php

   $token=$_GET['token'];
   $name=$_GET['name'];
   $surname=$_GET['surname'];
   $img=$_GET['img'];
   $email=$_GET['email'];
//time
ini_set('date.timezone', 'Europe/Athens');
$time1 = date("h:i:sa");
$date1= date("Y/m/d");

//FCM api URL
$url = 'https://fcm.googleapis.com/fcm/send';
//api_key available in Firebase Console -> Project Settings -> CLOUD MESSAGING -> Server key
$server_key = 'AIzaSyBHlMSY7mSybInGmp_Bm7ym8OoHrCgQtLU';

$message="User "+$name+" "+$surname+"requests battery from you.";

$fields = array(
   'to' => $token,
   'notification' => array('title' => 'BattBook', 'body' => 'You have a new battery request from'),
   'data' => array('message' => $message, 'img'=>$img, 'name'=>$name, 'surname'=>$surname,
'email'=>$email, 'date'=>$date1, 'time'=>$time1)
```

```
);
$headers = array(
  'Content-Type:application/json',
  'Authorization:key='.$server_key
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($fields));

$result = curl_exec($ch);
if ($result === FALSE) {
  die('FCM Send Error: ' . curl_error($ch));
}else{
echo $result;
}
curl_close($ch);
?>
```

2.16 Αποστολή μηνύματος

Το αρχείο αυτό, έχει ακριβώς την ίδια λογική με το παραπάνω, αλλά χρησιμοποιείται για αποστολή ειδοποιήσεων που έχουν να κάνουν με μηνύματα στο chat και όχι για αιτήματα για μπαταρία.

2.17 Μπαταρία

Το αρχείο αυτό, χρησιμοποιείται για να επιστρέφει το ποσοστό της μπαταρίας που είναι αποθηκευμένο στη βάση για τον χρήστη με αποτέλεσμα να είναι ορατό σε άλλους χρήστες. Είναι απαραίτητο ώστε να γίνεται ανά κάποια ώρα έλεγχος για το αν η μπαταρία που έχει ο χρήστης δε διαφέρει πάνω από 5% από αυτή στη βάση.

2.18 Ποσοστό μπαταρίας

Σε περίπτωση που η μπαταρία του χρήστη διαφέρει περισσότερο από 5% από αυτή στη βάση, το αρχείο αυτό καλείται και ενημερώνει τη βάση με το τρέχον ποσοστό μπαταρίας. Έτσι δεν υπάρχει μεγάλη απόκλιση ανάμεσα στα δύο ποσοστά.

2.19 Αποστολή αιτήματος

Το αρχείο αυτό χρησιμοποιείται για να σταλεί αίτημα φιλίας σε έναν χρήστη.

2.20 Φόρτιση

Σε περίπτωση που ο χρήστης φορτίζει την συσκευή του, είναι απαραίτητο να ενημερωθεί η βάση για να είναι ορατό και στους υπόλοιπους χρήστες. Για να επιτευχθεί αυτό, το email του χρήστη αποστέλλεται σε αυτό το web service, το οποίο με τη σειρά του ενημερώνει τη βάση.

2.21 Μη φόρτιση

Το αρχείο αυτό εκτελεί την αντίθετη ακριβώς ενέργεια από το παραπάνω αρχείο. Όταν ο χρήστης δε φορτίζει, χρησιμοποιείται αυτό το web service για την ενημέρωση της βάσης.

2.22 Τοποθεσία χρηστών

Το web service αυτό, ενημερώνει την εφαρμογή για τις γεωγραφικές συντεταγμένες του χρήστη που είναι αποθηκευμένες στη βάση, με αποτέλεσμα να είναι ορατές από τους άλλους χρήστες.

2.23 Ενημέρωση τοποθεσίας χρήστη

Το αρχείο αυτό ενημερώνει τη βάση σε περίπτωση που έχουν αλλάξει οι γεωγραφικές συντεταγμένες του χρήστη. Δεν υπάρχει κάποια ανάδραση στον χρήστη για αυτή την ενέργεια καθώς κάτι τέτοιο δε θα ήταν θεμιτό.

Κεφάλαιο 3 Η βάση δεδομένων

Η βάση δεδομένων της παρούσας διπλωματικής παρουσιάζεται στην συνέχεια.

- Όνομα βάσης: 2157640_android
- Τύπος: MySQL
- Host: fdb3.freehostingeu.com
- Management: phpMyAdmin 4

Οι πίνακες που περιέχονται στην βάση είναι οι εξής:

- user
- phone
- rating
- charging_options
- token
- contacts

Η μηχανή αποθήκευσης που χρησιμοποιήθηκε είναι η innoDB και η σύνθεση σύνδεσης η utf8_unicode_ci.Παρακάτω παρουσιάζονται αναλυτικά τα πεδία όλων των πινάκων ακολουθούμενα από ένα παράδειγμα αποθήκευσης δεδομένων.

3.1 Πίνακας user

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των βασικών στοιχείων του χρήστη με την εγγραφή του στην εφαρμογή. Επίσης, χρησιμοποιείται για την αποθήκευση των γεωγραφικών συντεταγμένων του, τη βαθμολογία του και για να ορίζεται αν είναι συνδεδεμένος στην εφαρμογή και αν φορτίζει.

#	Column	Type	Collation	Attributes	Null	Comments	Default	Extra
1	id	int(100)	utf8_unicode_ci		No		None	AUTO_INCREMENT
2	name	varchar(50)	utf8_unicode_ci		No		None	
3	surname	varchar(50)	utf8_unicode_ci		No		None	
4	email	varchar(50)	utf8_unicode_ci		No		None	
5	username	varchar(50)	utf8_unicode_ci		No		None	
6	password	varchar(50)	utf8_unicode_ci		No		None	
7	gender	set('Male', 'Female')	utf8_unicode_ci		No		None	
8	age	varchar(50)	utf8_unicode_ci		No		None	
9	lastlogin	date	utf8_unicode_ci		No		None	
10	longitude	double			No		None	
11	latitude	double			No		None	
12	active	enum('0', '1')	utf8_unicode_ci		No		None	
13	img	blob			No		None	
14	rate	float			No		None	

15	charging	set(‘0’, ‘1’)	utf8_unicode_ci		No		None	

Πίνακας 1:Πίνακας User

Παράδειγμα:

id	name	surname	email	username	password	gender	age	lastlogin
50	aristea	kontogianni	aristeabd@hotmail.com	karistea	$2TvlZAT.hcA2	Female	25	2016-12-5

longitude	latitude	active	img	rate	charging
23.740796	37.977158	1	[BLOB - 21.3 KiB]	0	0

Πίνακας 2: Παράδειγμα εγγραφής στον πίνακα user

3.2 Πίνακας phone

Στον πίνακα αυτόν, αποθηκεύονται τα δεδομένα που αφορούν το μοντέλο κινητού του χρήστη και το ποσοστό μπαταρίας που έχει. Πιο συγκεκριμένα, χρησιμοποιείται το ξένο κλειδί user_id που υποδεικνύει τον χρήστη από το πίνακα user, ώστε να προσδιοριστεί το μοντέλο (phone_model) και το ποσοστό μπαταρίας που έχει (battery).

#	Column	Type	Collation	Attributes	Null	Comments	Default	Extra
1	user_id	int(100)	utf8_unicode_ci		No		None	
2	phone_model	varchar(100)	utf8_unicode_ci		No		None	
3	battery	int(100)	utf8_unicode_ci		No		None	

Πίνακας 3:Πίνακας phone

Παράδειγμα:

user_id	phone_model	battery
50	Samsung GT-I9505	84

Πίνακας 4:Παράδειγμα εγγραφής στον πίνακα phone

3.3 Πίνακας rating

Στον πίνακα αυτόν, αποθηκεύονται τα δεδομένα που αφορούν τη βαθμολογία του χρήστη.Πιο συγκεκριμένα, ορίζεται ο χρήστης ο οποίος βαθμολογείται (rated_user), από ποιόν χρήστη (rates) και με τι βαθμολογία (rate).

#	Column	Type	Collation	Attributes	Null	Comments	Default	Extra
1	id	int(255)			No		None	AUTO_INCR

								EMENT
2	rated_user	int(255)			No		None	
3	rate	float			No		None	
4	rates	int(255)			No			

Πίνακας 5:Πίνακας rating

Παράδειγμα:

id	rated_user	rate	rates
87	87	5	86

Πίνακας 6:Παράδειγμα εγγραφής στον πίνακα rating

3.4 Πίνακας charging_options

Στον πίνακα αυτόν, αποθηκεύονται οι επιλογές φόρτισης που προσφέρει ο κάθε χρήστης. Ένα πεδίο user_id, αποθηκεύει το id του χρήστη το οποίο είναι μοναδικό στον πίνακα user. Επιπλέον, υπάρχουν τα πεδία που αντιπροσωπεύουν το κάθε είδος φόρτισης που ο εκάστοτε χρήστη προσφέρει: charger, powerbag, usb, usb_otg.

#	Column	Type	Collation	Attributes	Null	Comments	Default	Extra
1	user_id	int(100)	utf8_unicode_ci		No		None	
2	charger	enum(‘0’, ‘1’)	utf8_unicode_ci		No		None	
3	powerbag	enum(‘0’, ‘1’)	utf8_unicode_ci		No		None	
4	usb	enum(‘0’, ‘1’)	utf8_unicode_ci		No			
5	usb_otg	enum(‘0’, ‘1’)	utf8_unicode_ci					

Πίνακας 7:Πίνακας charging_options

Παράδειγμα:

user_id	charger	powerbag	usb	usb_otg
86	1	1	0	1

Πίνακας 8:Παράδειγμα εγγραφής στον πίνακα charging_options

3.5 Πίνακας token

Στον πίνακα αυτόν, αποθηκεύονται τα δεδομένα που αφορούν τα token του χρήστη. ειδοποιήσεων Πιο συγκεκριμένα, για τις ειδοποιήσεις είναι απαραίτητο να υπάρχει ένα token το οποίο παράγεται στη κάθε συσκευή και είναι μοναδικό.

#	Column	Type	Collation	Attributes	Null	Comments	Default	Extra

1	id	int(255)			No		None	AUTO_INCREMENT
2	user_id	int(255)			No		None	
3	token	varchar(1000)	utf8_unicode_ci		No		None	

Πίνακας 9:Πίνακας token

Παράδειγμα:

id	user_id	token
1	86	e0WeMl3ra58:APA91bGJ0rOXMPTjze7gfCLrG25pRSJgQFf7l5kyX_CFv09-lcnm0G2MnFMwW5h1BAPSZeVPSjxo_bR4pXunnx-klC00OWQNVAIem1vhpUwJIhYIdqlQsFcUq8JqxPLB7WTogW3VgtQ2

Πίνακας 10:Παράδειγμα εγγραφής στον πίνακα token

3.6 Πίνακας contacts

Στον πίνακα αυτόν, αποθηκεύονται τα δεδομένα που αφορούν τη σχέση ενός χρήστη με άλλους χρήστες. Από εδώ ορίζεται αν δύο χρήστες είναι φίλοι, αν έχει αποσταλεί αίτημα φιλίας από τον έναν στον άλλο και αν ναι από ποιόν, ή αν δεν υπάρχει καμία συσχέτιση σε αυτό τον πίνακα, οι χρήστες είναι άγνωστοι.

#	Column	Type	Collation	Attributes	Null	Comments	Default	Extra
1	id	int(100)			No		None	AUTO_INCREMENT
2	request_userid	int(100)			No		None	
3	accept_userid	int(100)			No		None	
4	request	int(100)			No		None	
5	accept	int(100)			No		None	
6	friends	int(100)			No		None	

Πίνακας 11:Πίνακας contacts

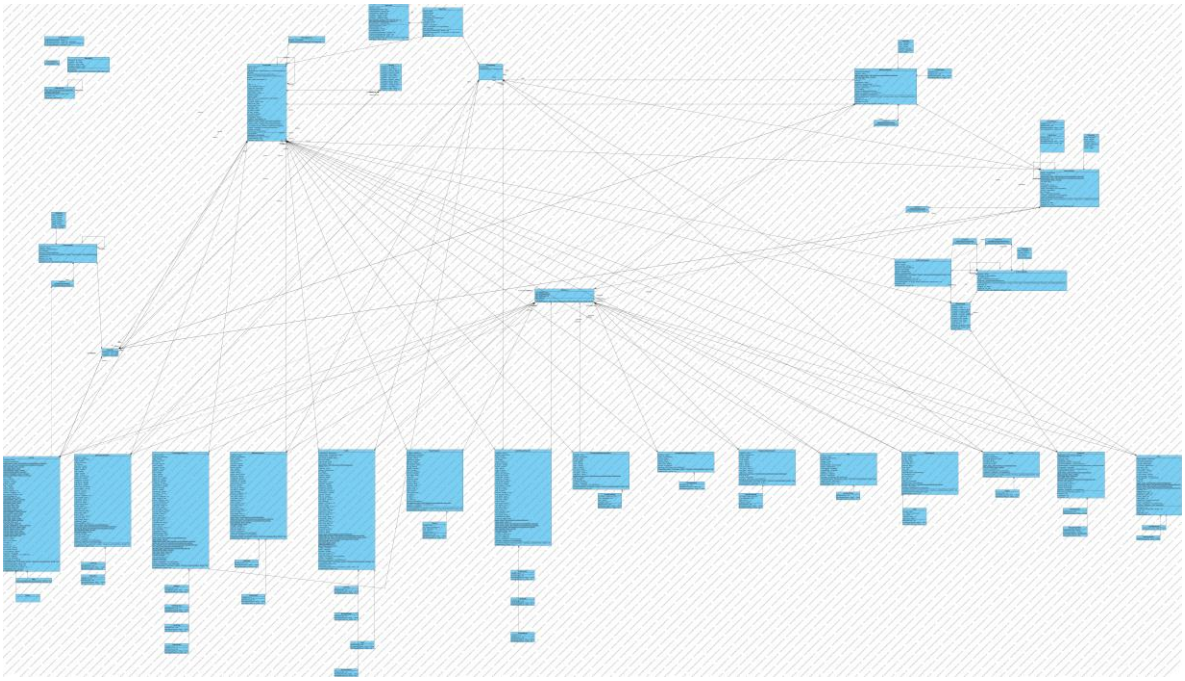
Παράδειγμα:

id	request_userid	accept_userid	request	accept	friends
46	86	97	1	1	1

Πίνακας 12:Παράδειγμα εγγραφής στον πίνακα contacts

Διάγραμμα UML

Παρακάτω παρατίθεται σε μορφή εικόνας το διάγραμμα UML που αφορά το project. Επίσης παρατίθεται σύνδεσμος για κατέβασμα του αρχείου καθότι το μέγεθος της εικόνας σε συνδυασμό με το μέγεθος της πληροφορίας δεν επιτρέπει να παρατεθεί σε υψηλότερη ανάλυση.



https://www.dropbox.com/s/9zx8r05hors9lka/com.androidunipi.jpg?dl=0

Συμπεράσματα

Μέσα από την διαδικασία υλοποίησης αυτής της εφαρμογής προσπαθήσαμε να δώσουμε μια λύση στο πρόβλημα της φόρτισης της μπαταρίας μέσω της δημιουργίας ενός δικτύου χρηστών που μοιράζονται την ανάγκη αλλά και την λύση αυτής. Παρόλα αυτά η εξέλιξη της τεχνολογίας οφείλει να δώσει μια οριστική λύση στο πρόβλημα αυτό των χρηστών έξυπνων συσκευών, επεκτείνοντας τη ζωή της μπαταρίας αλλά και της αυτονομίας των συσκευών. Κάτι τέτοιο ήδη έχει να επιτυγχάνεται μέσω της τεχνολογίας USB Type C για δημιουργία host device με σκοπό την ανταλλαγή μπαταρίας αλλά μια λύση που αφορά αποκλειστικά την μπαταρία κρίνεται αναγκαία. Όσον αφορά την εφαρμογή μπορεί να υλοποιηθεί και σε περιβάλλον iOS με σκοπό την εξυπηρέτηση και των iOS χρηστών.

Βιβλιογραφία

- https://developer.android.com/index.html
- http://www.androidhive.info/
- https://firebase.google.com/
- http://square.github.io/picasso/
- https://github.com/bumptech/glide
- https://github.com/google/gson
- http://www.vogella.com/tutorials/DesignPatternSingleton/article.html