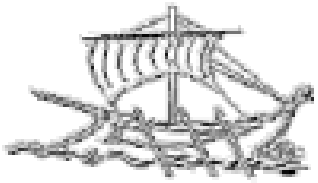


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Καταγραφή , υπενθύμιση και γνωστοποίηση γεωγραφικής θέσης στατικού / κινούμενου οχήματος (ή αντικειμένου), για Android συσκευές

Ιωάννης Δρυμούσης

Εισηγητής: Γεώργιος Πρεζεράκος, Καθηγητής

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2016

Καταγραφή , υπενθύμιση και γνωστοποίηση γεωγραφικής θέσης στατικού / κινούμενου οχήματος (ή αντικειμένου), για Android συσκευές

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Καταγραφή , υπενθύμιση και γνωστοποίηση γεωγραφικής θέσης στατικού / κινούμενου οχήματος (ή αντικειμένου), για Android συσκευές

Ιωάννης Δρυμούσης
Α.Μ. 42016

Εισηγητής:

Γεώργιος Πρεζεράκος, Καθηγητής

Εξεταστική Επιτροπή:

- **Ιωάννης Ν. Έλληνας, Καθηγητής**
- **Δημήτρης Νικολόπουλος, Αναπλ. Καθηγητής**

Ημερομηνία εξέτασης 23/09/2016

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/η ……, του ……, με αριθμό μητρώου …….. φοιτητής/τρια 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 είναι αφιερωμένη στον Θεό, για να μπορέσω να εκφράσω την ευγνωμοσύνη μου, που στην μέχρι τώρα πορεία της ζωής μου, μου έδωσε τα δύο σημαντικότερα πράγματα που χρειάζεται ένας άνθρωπος για να μεγαλουργήσει. Την υγεία και την αγάπη. Έχοντας την υγεία του, ένας άνθρωπος μπορεί να κάνει όνειρα, να έχει φιλοδοξίες και να δουλεύει αναπόσπαστα και ατάραχα για τον σκοπό του. Επίσης, ανεξάρτητα του ταλέντου και των ικανοτήτων που ένας άνθρωπος έχει, χρειάζεται και μια σταγόνα από το ισχυρότερο "αναβολικό" στον κόσμο προκειμένου να προοδεύσει. Την αγάπη. Τίποτα δεν είναι ωραίο χωρίς αυτή και τίποτα δεν δημιουργήθηκε χωρίς το κίνητρο της. Στη ζωή μου, ο Θεός με ευλόγησε δίνοντας μου αγάπη που πηγάζει από τα σημαντικότερα πρόσωπα της ζωής μου, που είναι φυσικά η οικογένεια μου και η σύντροφός μου που με στηρίζει με όλη την καρδία της και την ευχαριστώ. Πάνω απ 'όλα όμως Μαμά και Μπαμπά, θέλω να ξέρετε πως ευχαριστώ εσάς για κάθε σταγόνα ιδρώτα που ρίξατε για να με μορφώσετε και να με σπουδάσετε. Ευχαριστώ που πάντα ήσασταν εκεί και που ακόμα είστε. Ευχαριστώ για τους κόπους, τα άγχη και τις θυσίες σας. Βγάλατε έναν σωστό άνθρωπο και το πτυχίο αυτό σας ανήκει. Εύχομαι ο Θεός να με αξιώνει να προσφέρω κι άλλα πράγματα σαν αντίδωρο στην προσπάθεια που κάνατε για εμένα. Σας αγαπώ και ευχαριστώ για όλα.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη μιας εφαρμογής για συσκευές Android, η οποία πραγματεύεται δυο διαφορετικούς σκοπούς που έχουν όμως κοινή βάση. Ο πρώτος σκοπός είναι η εύρεση και αποθήκευση μιας γεωγραφικής θέσης έτσι ώστε αυτή να χρησιμοποιηθεί μελλοντικά και ο δεύτερος σκοπός είναι η ανταλλαγή γεωγραφικών συντεταγμένων μεταξύ δύο διαφορετικών χρηστών της εφαρμογής, έτσι ώστε να γνωρίζουν αμφότεροι ο ένας τη θέση του άλλου σε πραγματικό χρόνο, προκειμένου ο ένας να ακολουθεί τον άλλο. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν ενδιαφέρουσες τεχνολογίες με την βασικότερη να είναι τα Web Sockets τα οποία τρέχουν σε μια JavaEE εφαρμογή η οποία φιλοξενείτε στον Glassfish application server που έστησα. Η παρούσα παρουσίαση αμέσως μετά τις εισαγωγικές πληροφορίες που θα παραθέσει για το Android, θα αναλύσει όλες τις τεχνολογίες που χρησιμοποιήθηκαν και θα παρέχει πληροφορίες για το πώς στήθηκε το περιβάλλον ανάπτυξης, όπως επίσης θα παρέχει αναλυτική περιγραφή του τι κάνει η εφαρμογή, αναφέροντας τον ρόλο που επιτελεί κάθε κλάση της. Επίσης θα αναφερθούν και μερικά προβλήματα που προέκυψαν κατά την διάρκεια της υλοποίησης της, σε συνδυασμό με τις ενέργειες που έκανα για να τα επιλύσω. Τέλος θα πρέπει να αναφερθεί πως η παρούσα πτυχιακή εργασία δεν αποτελεί ένα κατά κάποιο τρόπο tutorial για το πώς φτιάχνουμε μια Android εφαρμογή, αλλά θεωρώντας πως ο χρήστης έχει κάποια βασική γνώση προγραμματισμού σε Android, επικεντρώνεται στο σκοπό που αυτή εξυπηρετεί.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη Mobile Εφαρμογών
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Android, GPS, Glassfish, Google maps, Web Sockets

ABSTRACT

The present thesis concerns the development of an Android application, which is about two different purposes with the same scope. The first one is about capturing and saving a geographical position for future use and the second one is about exchanging positions between two different application's users, in order both of them to know each other's position in real time, so that the one could follow the other. For application's development, there were used some interesting technologies with the most important of them to be Web Sockets, which are running on a JavaEE application being deployed on a Glassfish application server that I have set. Just after the basic information that the current presentation will provide about Android, it will be analyzed every technology used, as well as the way that the development environment was set and will also provide a detailed description of the application each self, mentioning the role of every application's class. Additionally, will be also mentioned some problems arose and their solutions. Finally, the current thesis should not be considered as an Android development tutorial, because it assumes that reader has a basic knowledge about developing Android applications, and it focuses on analyzing its own scope.

SCIENTIFIC SCOPE: Mobile application development
KEY WORDS: Android, GPS, Glassfish, Google maps, Web Sockets

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

GUI Graphical Unit Interface

VM Virtual Machine

NDK Native Developer Kit

UI User Interface

IDE Integrated Development Environment

APK Android application package

VCS Version Control Systems

JDK Java Development Kit

SDK Software Development Kit

AVD Android Virtual Device

ADB Android Debug Bridge

API Application Programming Interface

GIS Geographical Information Systems

AOSP Android Open Source Project

WS WebSocket

ΚΕΦΑΛΑΙΟ 1

ΤΟ ANDROID

Σε αυτό το κεφάλαιο γίνεται μια συνοπτική περιγραφή για το τι ακριβώς είναι το λειτουργικό σύστημα android, η οποία έχει σαν στόχο να εξοικειώσει τον αναγνώστη με το περιβάλλον στο οποίο έγινε η ανάπτυξη της εφαρμογής που παρουσιάζεται στην παρούσα πτυχιακή εργασία.

1.1 Τι είναι

Το Android **[4]** είναι ένα Unix-like λειτουργικό σύστημα που σχεδιάστηκε από την Open Handset Alliance η οποία είναι μια κοινοπραξία εταιριών σχετικών με τον κλάδο των τεχνολογιών και της πληροφορικής και της οποίας ηγείται η Google.

Το λειτουργικό σύστημα που πρωτοεμφανίστηκε το 2007 (έγινε διαθέσιμο στην αγορά το 2008), είναι σχεδιασμένο να εκτελείτε σε μια ποικιλία πλατφόρμων για κινητές συσκευές και δίνει μέγιστη έμφαση στην διαχείριση ενεργειών που γίνονται μέσω touch screens και εικονικών πληκτρολόγιων. Σήμερα το λειτουργικό android συναντάται επιπλέον σε τηλεοράσεις, συσκευές που μπορούν να φορεθούν (ρολόγια κτλ.) ακόμα και σε αυτοκίνητα.

Η ανάπτυξή του βασίστηκε στον πυρήνα του Linux, πάνω στον οποίο έγινα βελτιστοποιήσεις και προσθήκες με σκοπό να αποτελέσει ένα κατάλληλο για χρήση από κινητές συσκευές λειτουργικό σύστημα. Ο πυρήνας του Linux χρησιμοποιήθηκε κυρίως για υποστήριξη διεργασιών, μνήμης και οδηγών συσκευών, καθώς έχει επεκταθεί για να περιλαμβάνει και διαχείριση ισχύος. Πάντως παρόλο πού έχει βασιστεί αρκετά πάνω στο Linux, σήμερα βρίσκεται εκτός της κανονικής διανομής των εκδόσεων του Linux.

Ο πηγαίος κώδικάς του android, ο οποίος είναι γραμμένος πέρα από C (στον πυρήνα), σε Java και C++ , είναι open source και διατίθεται ελεύθερα από την Google. Αυτός είναι και ένας από τους λόγους που το κατέστησαν δημοφιλές. Ωστόσο πολλές εταιρίες έχουν βασιστεί πάνω στον κώδικα του Android και έχουν κάνει τις δικές τους προσθήκες με αποτέλεσμα να βγάζουν την δικιά τους ξεχωριστή έκδοση, η οποία περιλαμβάνει επιπρόσθετα προγράμματα ή δυνατότητες που παρέχει η εκάστοτε εταιρία όπως επίσης και ξεχωριστό GUI.

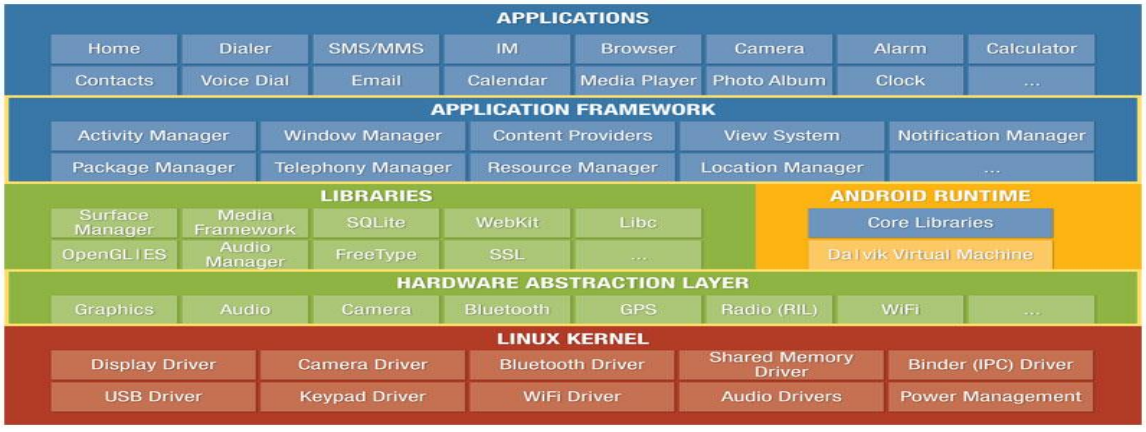
1.2 Πώς λειτουργεί

Σε αυτό το κεφάλαιο θα αναλυθεί περιληπτικά η αρχιτεκτονική του android και το πως οι διάφορες εφαρμογές τρέχουν πάνω σε αυτό.

Όπως προείπαμε το android βασίζεται στον πυρήνα του Linux (Linux kernel). Πάνω από αυτόν τον πυρήνα υπάρχει ένα middleware , βιβλιοθήκες και APIs που έχουν γραφτεί σε C, καθώς και ένα application framework το οποίο περιλαμβάνει βιβλιοθήκες συμβατές με Java. Όλα αυτά συνδυάζονται από το σύστημα για να παραχθεί το τελικό εκτελέσιμο αρχείο.

Πέρα όμως από τα παραπάνω middleware και βιβλιοθήκες, περιλαμβάνεται στην αρχιτεκτονική του Android και μια εικονική μηχανή (Virtual Machine ή απλά VM) η οποία είναι υπεύθυνη για την εκτέλεση των εφαρμογών, η οποία ονομάζεται Dalvic. Οι εφαρμογές λοιπόν που είναι γραμμένες σε Java, μεταγλωττίζονται αρχικά σε ένα δυαδικό κώδικα ο οποίος μπορεί να εκτελεστεί από την Java VM. Αυτός ο κώδικας στην συνέχεια μεταφράζεται σε δυαδικό κώδικα που αντιλαμβάνεται ο Dalvic και έτσι παράγεται ένα εκτελέσιμο αρχείο που ο Dalvic μπορεί να τρέξει. Ο λόγος που δεν μπορεί μια εφαρμογή να τρέξει άμεσα χωρίς την παρεμβολή του Dalvic είναι ότι ο Dalvic παράγει εκτελέσιμα αρχεία που έχουν βελτιστοποιηθεί για να εκτελούνται σε συσκευές με περιορισμένες δυνατότητες πόρων.

Είναι σημαντικό να αναφέρουμε πως στις νεότερες εκδόσεις του Android, ο Dalvic ο οποίος μέχρι πρότινος ήταν αναπόσπαστο κομμάτι της αρχιτεκτονικής του Android , έχει πλέον αντικατασταθεί από τον ART ο οποίος περιλαμβάνει ακόμα περισσότερες βελτιστοποιήσεις που αφορούν την αύξηση της απόδοσης του συστήματος.



Σχήμα 1.1 Η αρχιτεκτονική του Android

1.3 Πληροφορίες για Developers

Μια απορία που έχουν συνήθως οι νέοι developers που θέλουν να ασχοληθούν με την ανάπτυξη εφαρμογών για android , είναι το πώς μπορούμε να το πετύχουμε και το ποιά είναι η γλώσσα προγραμματισμού που χρησιμοποιούμε.

Η γλώσσα λοιπόν που χρησιμοποιείτε για την ανάπτυξη εφαρμογών σε android περιβάλλον είναι η Java για όλη την core λειτουργικότητα της εφαρμογής και η XML για τον σχεδιασμό του GUI. Για αποθήκευση σε βάση δεδομένων χρησιμοποιείτε η SQLite η οποία είναι ουσιαστικά SQL αλλά σε μια πιο «ελαφριά» μορφή και διαφέρει από τα συστήματα διαχείρισης βάσης δεδομένων καθώς δεν είναι μια ξεχωριστή διεργασία που προσπελάζεται από μια εφαρμογή, αλλά αποτελεί μέρος της εφαρμογής.

Επίσης δίνεται η δυνατότητα ανάλογα με τα εργαλεία ανάπτυξης που θα επιλέγουν να χρησιμοποιηθεί σαν γλώσσα προγραμματισμού η C ή C++. Αυτή την δυνατότητα την δίνει το Android NDK (Native Developer Kit) και χρησιμεύει στην υλοποίηση εφαρμογών που απαιτούν μεγάλη επεξεργαστική ισχύ όπως είναι για παράδειγμα τα παιχνίδια. Το NDK εκμεταλλεύεται το γεγονός ότι το Android είναι γραμμένο σε C και C++ και μας επιτρέπει να γράψουμε κώδικα σε αυτές τις native για το συγκεκριμένο λειτουργικό γλώσσες έτσι ώστε να αυξήσουμε την ταχύτητα εκτέλεσης του κώδικα καθώς δεν απαιτείτε η μεταγλώττισή του από τη JVM διότι μπορεί να εκτελεστεί κατευθείαν. Ωστόσο δεν προτείνεται η χρήση του NDK αν δεν συντρέχει κάποιος επιτακτικός για την ορθή λειτουργία της εφαρμογής μας λόγος, διότι κάνει το πρόγραμμά μας πιο περίπλοκο.

Σε αντίθεση λοιπόν με τον περιορισμό σχετικά με το ποιες γλώσσες προγραμματισμού πρέπει να χρησιμοποιηθούν, το android έχει αρκετές εναλλακτικές σχετικά με το προγραμματιστικό περιβάλλον που μπορούμε να χρησιμοποιήσουμε. Οι επιλογές αυτές μπορούν να είναι είτε τα διάσημα Eclipse και NetBeans εγκαθιστώντας πάνω τους και το σχετικό plug-in για το Android , είτε το Android Studio το οποίο είναι το επίσημο IDE που παρέχεται από την Google και το οποίο χρησιμοποίησα και εγώ για τις ανάγκες της παρούσας εργασίας. Πέρα από τα προαναφερθέντα IDEs κυκλοφορεί μια πληθώρα από άλλες επιλογές από τις οποίες ο χρήστης μπορεί να επιλέξει όποια θεωρεί ότι του ταιριάζει καλύτερα.

Επίσης τα τελευταία χρόνια είναι πολύ διαδεδομένη και η χρήση διάφορων frameworks όπως είναι για παράδειγμα το Ionic το οποίο μας επιτρέπει να φτιάξουμε το front-end μιας εφαρμογής χρησιμοποιώντας JavaScript. Τέτοιου είδους frameworks σε συνδυασμό με άλλα επίσης frameworks όπως είναι για παράδειγμα το Cordova που παρέχει υποστήριξη για βιβλιοθήκες που βρίσκονται στον πυρήνα της συσκευής, μας επιτρέπουν να φτιάξουμε μια ολοκληρωμένη εφαρμογή η οποία δεν θα τρέχει μόνο σε Android αλλά και σε άλλα λειτουργικά συστήματα κινητών, χωρίς να χρειαστεί εμείς να αναπτύξουμε κάτι παραπάνω.

Στην συνέχεια της εργασίας και συγκεκριμένα στο Κεφάλαιο 2, θα επανέλθουμε στα θέματα που αφορούν το περιβάλλον ανάπτυξης του android και θα δώσουμε πιο συγκεκριμένες πληροφορίες σχετικά με τις απαιτήσεις και τα βήματα που πρέπει να γίνουν για να το σετάρουμε.

ΚΕΦΑΛΑΙΟ 2

Η ΕΦΑΡΜΟΓΗ ΜΟΥ

Σε αυτό το κεφάλαιο θα αναλυθεί το αντικείμενο της παρούσας πτυχιακής εργασίας. Θα γίνει αναφορά στο τι ακριβώς κάνει η εφαρμογή μου, ποιά βήματα έγιναν για να αναπτυχθεί, πως στήθηκε το περιβάλλον ανάπτυξης και το ποια προγραμματίστηκα εργαλεία χρησιμοποιήθηκαν.

2.1 Τι κάνει

Η εφαρμογή που μελετάται στη παρούσα πτυχιακή εργασία προέκυψε από μια έμπνευση της στιγμής όταν βρέθηκα στην δύσκολη θέση του να μην θυμάμαι που ακριβώς είχα παρκάρει το αυτοκίνητο μου. Η μια σκέψη διαδέχτηκε την άλλη και τελικά η αρχική μου ιδέα πήγε και ένα βήμα παραπέρα. Τι εξυπηρετεί λοιπόν η εφαρμογή μου; Εξυπηρετεί την καταγραφή και γνωστοποίηση μιας γεωγραφικής θέσης την οποία θέλω να θυμάμαι καθώς και την ανταλλαγή γεωγραφικών θέσεων μεταξύ δυο συσκευών με απώτερο σκοπό την διατήρηση επαφής μεταξύ τους καθώς αυτές αλλάζουν θέση στον χάρτη. Πρόκειται συνεπώς για δυο ξεχωριστές λειτουργίες που κοινό γνώμονα έχουν να βρίσκουν την θέση μας στον χάρτη και να την καθιστούν γνωστή.

Η πρώτη λειτουργία μπορεί να μας λύσει τα χέρια αν θέλουμε να βρούμε για παράδειγμα το που σταθμεύσαμε το αυτοκίνητο μας και το πως μπορούμε να πάμε σε αυτό με βάση την τωρινή μας θέση. Πρόκειται για μια απλή σχετικά λειτουργία που καταγράφει μέσω GPS ή άλλου location provider (π.χ. κεραίες τηλεφωνίας), την γεωγραφική μας θέση και βρίσκει μέσω geotaging (αντιστοίχηση θέσης σε οδό) την οδό και τον αριθμό στον οποίο αυτή αντιστοιχεί. Στη συνέχεια την αποθηκεύει με κάποιον τρόπο που θα αναλύσουμε στην συνέχεια και την επόμενη φορά που θα ανοίξουμε την εφαρμογή μας, θα μας την υπενθυμίσει.

Η δεύτερη λειτουργία είναι σαφώς πιο πολύπλοκη αλλά και συνάμα πιο ενδιαφέρουσα. Μας δίνει λοιπόν την δυνατότητα να συνδεθούμε με κάποια άλλη συσκευή (που πιθανότατα θα την χρησιμοποιεί κάποιος οικείος μας) για να βλέπουμε κάθε χρονική στιγμή το στίγμα της στον χάρτη και θα μας παρέχονται επίσης οδηγίες για το πώς μπορούμε να φτάσουμε σε αυτή.

Με απλά λόγια θα μπορούσαμε να πούμε πως αυτή η λειτουργία μας βοηθάει να ακολουθήσουμε κάποιον ακόμα και αν δεν έχουμε οπτική επαφή μαζί του. Η όλη ιδέα στηρίχτηκε στην σκέψη του πώς θα μπορούσαν να διευκολυνθούν δυο συνταξιδιώτες που επιβαίνουν σε διαφορετικά οχήματα στο να μην χάνουν οπτική επαφή μεταξύ τους. Όλοι ξέρουμε πως οι συνθήκες στον δρόμο δεν είναι πάντα ιδανικές και είναι πολύ πιθανόν δυο οχήματα που το ένα ακολουθεί το άλλο να χαθούν μεταξύ τους κάποια στιγμή. Αυτό είναι μια δυσάρεστη και στρεσογόνα κατάσταση που μπορεί να προκαλέσει ακόμα και ατυχήματα. Έτσι λοιπόν αυτό το άγχος έρχεται να καλύψει η παρούσα λειτουργία για να μην χρειάζεται τα δυο οχήματα να βρίσκονται συνεχώς σε οπτική επαφή γιατί πλέον και ο προπορευόμενος αλλά και αυτός που ακολουθεί, ανά πάσα στιγμή γνωρίζουν το που βρίσκεται ο άλλος και το πως θα οδηγηθούν σε αυτόν.

Για να χρησιμοποιήσει κάποιος αυτή την λειτουργία θα πρέπει να δημιουργήσει έναν λογαριασμό χρήστη όπου θα έχει ένα μοναδικό username και έναν κωδικό. Στη συνέχεια θα μπορεί να κάνει log in και να επιλέξει να παρακολουθήσει την γεωγραφική θέση ενός οικείου. Εδώ είναι σημαντικό να αναφέρω ότι η παρακολούθηση της θέσης είναι συναινετική και πρέπει να εγκριθεί και από τις δύο πλευρές. Αφού λοιπόν επιλέξει να συνδεθεί με κάποιον, θα πρέπει να ορίσει τον ρόλο που θα έχει κατά την διάρκεια της επαφής. Οι ρόλοι είναι δύο και έχουν σημασία στο πώς η εφαρμογή θα δείχνει την θέση του καθενός. Έτσι κάποιος μπορεί να επιλέξει είτε να είναι ο προπορευόμενος και άρα θα επιλέξει τον ρόλο lead ή θα είναι αυτός που ακολουθεί και θα επιλέξει τον ρόλο follow. Σημαντικό είναι να αναφέρουμε πως δεν μπορούν να πάρουν τον ίδιο ρόλο και οι δυο συσκευές.

Ασχέτως πάντως με τον ρόλο που θα επιλέξει ο καθένας είναι υποχρεωτικό να δώσει το username που έχει δηλώσει ο φίλος του στο πεδίο που ρωτάει με ποιον θέλουμε να συνδεθούμε. Αν για παράδειγμα εμείς είμαστε ο χρήστης John και θέλουμε να συνδεθούμε με τον χρήστη Nick, τότε θα δώσουμε το Nick σαν όνομα χρήστη με τον οποίο θέλουμε να συνδεθούμε και ο Nick θα δηλώσει το John σαν username, το οποίο είναι το δικό μας. Έτσι με αυτόν τον τρόπο γίνεται η συναίνεση και αρχίζει η ανταλλαγή θέσεων. Αν η διαδικασία που περιγράφτηκε παραπάνω δεν γίνει τότε δεν θα μπορεί κανένας από τους δύο να δει την θέση του άλλου. Έτσι αν για παράδειγμα εμείς δηλώσουμε σαν username με το οποίο

επιθυμούμε να συνδεθούμε το Nick και ο Nick δεν έχει την παραμικρή ιδέα για τις προθέσεις μας, τότε επ' ουδενί θα υπάρξει σύνδεση μιας και θα πρέπει υποχρεωτικά και ο Nick να δηλώσει πως θέλει να επικοινωνήσει με τον John.

Αφού λοιπόν οι δυο χρήστες συνδεθούν, ανά τακτά χρονικά διαστήματα διαβάζεται η γεωγραφική τους θέση και έπειτα μέσω web sockets με την παρεμβολή ενός application server γίνεται η ανταλλαγή των θέσεων. Αυτός που έχει τον ρόλο lead εμφανίζεται πρώτος στον χάρτη και βλέπει την διαδρομή που πρέπει ο χρήστης με τον ρόλο follow να διανύσει για να τον φτάσει ή με άλλα λόγια την διαδρομή που έκανε και πρέπει να καλύψει ο άλλος χρήστης. Ο χρήστης δε με τον ρόλο follow βλέπει την διαδρομή που πρέπει να καλύψει για να φτάσει τον leader.

Αυτή ήταν σε γενικές γραμμές η παρουσίαση του τί κάνει η εφαρμογή που παρουσιάζεται. Στην συνέχεια θα αναλύσουμε το πώς επετεύχθησαν όλα αυτά και με ποια βήματα.

2.2 Πορεία σχεδιασμού

Μετά την σύλληψη της ιδέας για την παρούσα εφαρμογή, ακολούθησε μια διαδικασία σχεδιασμού και ανάλυσης η οποία ουσιαστικά έδωσε στην ιδέα αυτή περιεχόμενο και ουσία.

Στα στενά πλαίσια μιας πτυχιακής εργασίας η διαδικασία που ακολουθήθηκε και θα περιγραφεί παρακάτω, μπορεί να φαίνεται κάπως επιπόλαιη και επιφανειακή. Στην πραγματικότητα όμως όλες αυτές οι διαδικασίες που αναφέρονται στην ανάλυση, σχεδίαση, υλοποίηση και συντήρηση ενός λογισμικού και αποτελούν την ουσία της σημαντικότατης επιστήμης που ονομάζεται Μηχανική Λογισμικού, είναι καθοριστικές για την πορεία και την ολοκλήρωση ενός project. Είναι σημαντικό να αναφερθεί πως τέτοιες διαδικασίες ακολουθούνται πιστά και αδιαπραγμάτευτα από κάθε εταιρία ή οργανισμό πληροφορικής που σέβεται το αντικείμενό της και παράγει ποιοτικό λογισμικό. Πρόκειται για βήματα που καθορίζουν το πως θα αναπτυχθεί ένα project και όπως είναι εύκολα αντιληπτό πρέπει να είναι πολύ προσεκτικά έτσι ώστε να μειωθούν οι πιθανότητες κάτι να μην πάει καλά στην πορεία και να αναγκαστεί η ομάδα υλοποίησης να κάνει βήματα πίσω ή ακόμα και σε ακραίες περιπτώσεις να συνειδητοποιήσει πως το project δεν είναι εφικτό και να σταματήσει την υλοποίησή του.

Για τις ανάγκες της παρούσας εργασίας, έγινε μια ανάλυση από εμένα για το τι θα χρειαστεί η εφαρμογή μου και το πως θα υλοποιήσω αυτά που απαιτούνται, δηλαδή το ποιες τεχνολογίες θα χρησιμοποιήσω. Είναι σημαντικό να κάνουμε μια καλή έρευνα για τις τεχνολογίες που έχουμε διαθέσιμες για το κάθε μας πρόβλημα, διότι πολλές φορές οι τεχνολογίες που επιλέγονται δεν κάνουν τελικά αυτό που νομίζαμε ή μπορεί να παρουσιάζουν ασυμβατότητες με τα υπόλοιπα μέρη της εφαρμογής μας. Βέβαια αυτού του είδους οι επιλογές απαιτούν εμπειρία και τριβή με το αντικείμενο καθώς οι παράγοντες που τις επηρεάζουν είναι πολλοί και απαιτούν διορατικότητα ως προς τα μελλοντικά μας σχέδια για ένα project και ως προς τα προβλήματα που μπορεί να προκύψουν.

Έτσι αρχικά ξεκίνησα σπάζοντας το αρχικό πρόβλημα σε μικρότερα και αναλύοντας το καθένα ξεχωριστά. Προσπάθησα να αντιληφθώ δηλαδή το κύριο project ως ένα σύνολο από επιμέρους λειτουργίες. Έκανα ένα πλάνο με τις κλάσεις τις οποίες θα χρειαστώ και προσπάθησα να ομαδοποιήσω τον κώδικα που ήταν κοινός σε κάθε κλάση. Για παράδειγμα όπως θα δούμε και στη συνέχεια, στην υλοποίησή μου υπάρχει μία κλάση που διαχειρίζεται τον εντοπισμό και την καταγραφή της γεωγραφικής θέσης (πρώτη λειτουργία) και μια διαφορετική η οποία διαχειρίζεται την ανταλλαγή γεωγραφικών θέσεων μεταξύ των χρηστών (δεύτερη λειτουργία). Σε αυτή την περίπτωση τόσο στη μια όσο και στην άλλη κλάση υπάρχει η ανάγκη να βρεθεί η γεωγραφική θέση μέσω ενός location provider. Αυτός ο location provider που χρησιμοποιείτε σε πολλές περιπτώσεις, θα πρέπει να αποτελεί μια αυτόνομη και επαναχρησιμοποιήσιμη οντότητα και δεν θα πρέπει να εμπεριέχεται ξανά και ξανά στις κλάσεις που τον χρειάζονται.

Σύμφωνα με τα παραπάνω λοιπόν, προσπάθησα να κάνω μια εκτίμηση των διαφορετικών οντοτήτων που θα απαρτίσουν το project και τελικά κατέληξα στην παρακάτω λίστα που περιέχει τα απαιτούμενα προς υλοποίηση tasks, τα οποία είναι :

- Ένας controller (μια κλάση που διαχειρίζεται κάτι) για την διαχείριση της γεωγραφικής θέσης της συσκευής που θα αντλείτε είτε μέσω GPS, είτε μέσω κεραιών τηλεφωνίας ή από Wi-Fi .
- Ένας controller για την σύνδεση στο internet μιας και η εφαρμογή θα πρέπει να συνδέεται στο διαδίκτυο για να αντλεί και να στέλνει πληροφορίες.

- Ένας τρόπος αποθήκευσης της γεωγραφικής θέσης για να είναι διαθέσιμη την επόμενη φορά που ο χρήστης θα ανοίξει την εφαρμογή. Οι προτάσεις ήταν είτε να χρησιμοποιήσω μια βάση δεδομένων είτε κάποιο εξωτερικό αρχείο. Τελικά κατέληξα στο δεύτερο και συγκεκριμένα σε έναν τρόπο που λέγεται Shared Preferences και θα αναλυθεί στην συνέχεια.
- Χρήση χάρτη για απεικόνιση της γεωγραφικής θέσης. Χρησιμοποιήθηκε τελικά το Google Maps API μιας και παρέχει έτοιμες, σταθερές και ολοκληρωμένες υπηρεσίες , όπως είναι για παράδειγμα το navigation (πλοήγηση). Επίσης είναι τεστραρισμένο και περιέχεται εξ' αρχής σε εκατομμύρια συσκευές.
- Εξωτερική βάση δεδομένων (σε κάποιον server) για αποθήκευση των χρηστών που γίνονται μέλη στην εφαρμογή, μιας και όπως έχει αναφερθεί για ανταλλαγή τοποθεσιών μεταξύ δυο χρηστών απαιτείτε να είναι και οι δύο μέλη.
- Στήσιμο Server για την διαχείριση μελών και ανταλλαγής γεωγραφικών θέσεων μεταξύ δυο συσκευών. Το μηχάνημα που χρησιμοποιήθηκε είναι ένα VM που έχει στηθεί για ακαδημαϊκούς σκοπούς στον Ωκεανό (https://okeanos.grnet.gr/home/). Χρησιμοποιεί Linux Centos για λειτουργικό σύστημα και έχει εγκατεστημένο apache server, MySQL για βάση δεδομένων, και τον Glassfish (θα αναφερθούμε προσεχώς) για application server.
- Σχεδιασμός του UI έτσι ώστε να έχει απλό και εύχρηστο περιβάλλον η εφαρμογή. Το interface της εφαρμογής άλλαξε αρκετά κατά την πορεία υλοποίησης για να εξυπηρετεί καλύτερα τις ανάγκες του χρήστη.

Όλα όσα προαναφέρθηκαν χρησιμοποιήθηκαν ως ραχοκοκαλιά πάνω στην οποία στηρίχτηκα για την ανάπτυξη της εφαρμογής. Στην πορεία όμως, αρκετά απ όσα αρχικά είχα σκεφτεί αλλάξανε και έτσι οι απαιτήσεις αναπροσαρμόστηκαν.

Παραπάνω είδαμε μια λίστα με τα υποπροβλήματα που είχα να αντιμετωπίσω και τον τρόπο που επέλεξα να το κάνω. Για μερικά είχα γνώση από πριν και τα επέλεξα εξ αρχής, για άλλα έπρεπε να κάνω μια έρευνα και να μάθω τι επιλογές έχω και ποια είναι καλύτερη για εμένα. Όλη αυτή η διαδικασία τελικά παρήγαγε την παραπάνω λίστα σε ένα εύλογο βάθος χρόνου.

2.3 Περιβάλλον ανάπτυξης

Ως προς τις επιλογές του περιβάλλοντος πάνω στο οποίο θα γινόταν η υλοποίηση της εφαρμογής , είναι αλήθεια πως υπάρχουν αρκετές επιλογές. Στο internet κυκλοφορούν αρκετά ελεύθερα IDEs (Integrated Development Environment) που προσφέρουν αρκετές διαφορετικές επιλογές. Για παράδειγμα δυο πολύ καλές επιλογές ήταν να χρησιμοποιηθεί είτε το Eclipse είτε το Netbeans, τα οποία αν και δεν είναι καθαρά android IDEs μπορούν με τα κατάλληλα plug ins που είναι διαθέσιμα για το καθένα, να αποτελέσουν μια καλή και αξιόλογη λύση. Πέρα όμως από αυτές τις κλασικές επιλογές, μπορούμε να συναντήσουμε και αρκετά ακόμα IDEs τα οποία στοχεύουν σε cross platform development . Δίνουν δηλαδή την επιλογή στον προγραμματιστή να γράψει εφαρμογή για android χωρίς να χρησιμοποιήσει την Java που είναι η προεπιλεγμένη γλώσσα ανάπτυξης, αλλά χρησιμοποιώντας γλώσσες όπως η C++ , C#, Basic ή ακόμα και HTML. Τέτοια παραδείγματα IDE είναι το Cordova , το AIDE, το Basic4Android κ.α. .

Εγώ ωστόσο επέλεξα ένα IDE που είναι ειδικευμένο για ανάπτυξη εφαρμογών android και πλέον αποτελεί και το επίσημο IDE του Android. Πρόκειται για το Android Studio **[5]** που κυκλοφόρησε πρώτη φορά σαν stable έκδοση τον Δεκέμβρη του 2014 και εκ τότε αντικατέστησε τα Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη Android εφαρμογών.

Οι λόγοι που επέλεξα το Android Studio έναντι των υπολοίπων IDEs είναι αρκετοί. Ο κυριότερος από αυτούς που με έπεισε να το χρησιμοποιήσω, είναι πως επειδή αποτελεί το επίσημο IDE της Google για ανάπτυξη Android εφαρμογών έχει καλύτερη υποστήριξη και δίνει όλο του το βάρος στην εξυπηρέτηση των αναγκών των Android developers. Εκτός όμως από τα πλεονεκτήματα που η ειδίκευση του παρέχει, το Android Studio μας πλαισιώνει με μια πληθώρα εργαλείων και καινοτομιών που βελτιώνουν τις επιδόσεις του IDE και συμβάλουν στην ευκολότερη ανάπτυξη των εφαρμογών. Μερικές από αυτές τις καινοτομίες ή ευκολίες είναι οι ακόλουθες :

- Χρησιμοποιεί ένα Gradle-based building system (το σύστημα που μαζεύει όλο το πηγαίο κώδικα, τον συνδυάζει και τον μετατρέπει σε εκτελέσιμο κώδικα). Το Gradle εμπεριέχει κάποια ευφυΐα ως προς τον τρόπο που χειρίζεται τα διαφορετικά στοιχεία κατά την διάρκεια του Build και βελτιώνει

τους χρόνους του. Είναι ιδανικό για multi-project builds και εισάγει ένα Groovy-based τρόπο για να δηλώνει το configuration του project αντικαθιστώντας την XML που χρησιμοποιείται από τον Apache Maven.

- Μπορεί να παράγει πολλαπλά APKs (Android application package) με διαφορετικά περιεχόμενα χρησιμοποιώντας το ίδιο project και τα ίδια modules.
- Πλούσιο documentation (οδηγίες και πρότυπα κώδικα).
- Χρησιμοποιεί το lint tool το οποίο είναι ένα εργαλείο στατικής ανάλυσης του κώδικα που μεταξύ άλλων προβλέπει πιθανά bugs και προτείνει βελτιστοποιήσεις σχετικά με την ορθότητα, την ασφάλεια και τις επιδώσεις του κώδικα.
- Χρησιμοποιεί επίσης τον ProGuard που είναι ένα εργαλείο για να βελτιστοποιεί τον κώδικα της Java. Έχει την δυνατότητα να εντοπίζει κλάσεις , μεθόδους και ιδιότητες που δεν χρησιμοποιούνται ποτέ και να τις διαγράφει . Πέρα από αυτό όμως συμπιέζει τον κώδικα για να μπορεί να φορτώνεται ή να μεταφέρεται μέσω του διαδικτύου γρηγορότερα και χειρίζεται τις βιβλιοθήκες του προγράμματος με τέτοιον τρόπο, ώστε να τις καθιστά λιγότερο ευάλωτες στο reverse-engineer (αλίευση πληροφοριών για τον τρόπο που δουλεύει το πρόγραμμα) και συνεπώς πιο ασφαλείς.
- Περιέχει έναν πλούσιο layout editor (φτιάχνει το γραφικό περιβάλλον της εφαρμογής) με υποστήριξη drag and drop.
- Έχει build-in υποστήριξη από VCS (Version Control System).
- Έχει build-in υποστήριξη για την Google Cloud Platform (παρέχει έτοιμο backend κώδικα για την διαχείριση εκατομμυρίων χρηστών χωρίς να χρειάζεται να συντηρούμε ή να στήσουμε δικό μας server) και κάνει ευκολότερη την αλληλοσυσχέτιση της με την εφαρμογή μας.
- Έχει ωραίο και παραμετροποιήσιμο περιβάλλον, παρέχοντας πολλές ευκολίες κατά την παραγωγή του κώδικα.
- Παρέχει την δυνατότητα να βλέπουμε live γραφήματα σχετικά με την χρήση της CPU ή της μνήμης την συσκευής, καθώς η εφαρμογή μας εκτελείται.

Αυτοί είναι λοιπόν οι κυριότεροι λόγοι που χρησιμοποίησα το Android Studio. Στη συνέχεια θα αναλύσουμε το πώς εγκαθιστάτε και το τι ακόμα χρειάζεται για να μπορέσουμε να αναπτύξουμε μια Android εφαρμογή.

2.4 Απαιτούμενα ανάπτυξης

Σε αυτό το κεφάλαιο θα δούμε το πως, μπορούμε να ξεκινήσουμε να γράφουμε μια εφαρμογή Android. Θα επικεντρωθούμε στο τι χρειάζεται για να στήσουμε το κατάλληλο περιβάλλον ανάπτυξης και θα δούμε βήμα - βήμα το πως δημιουργούμε ένα νέο project στο Android Studio.

Το πρώτο πράγμα που θα πρέπει να υπάρχει εγκατεστημένο στον υπολογιστή μας είναι το Java JDK (Java Development Kit). Το JDK είναι ένα προγραμματιστικό περιβάλλον που μας επιτρέπει να αναπτύξουμε εφαρμογές που χρησιμοποιούν την Java σαν γλώσσα προγραμματισμού. Συμπεριλαμβάνει προγραμματιστικά εργαλεία χρήσιμα για προγραμματισμό και testing Java εφαρμογών. Μπορούμε να το κατεβάσουμε και να το εγκαταστήσουμε από την ακόλουθη διεύθυνση http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html . Η κατώτερη απαιτούμενη έκδοση για να καλύψουμε και τις εκδόσεις του Android που είναι μεγαλύτερες της έκδοσης Android 5.0 (API 21) είναι το JDK 7.

Αφού εγκαταστήσουμε επιτυχώς το JDK μπορούμε να κατεβάσουμε και να εγκαταστήσουμε το Android Studio. Από την ιστοσελίδα http://developer.android.com/sdk/index.html επιλέγουμε download και στη συνέχεια ακολουθούμε την διαδικασία εγκατάστασης. Όταν θα ξεκινήσουμε για πρώτη φορά το Android Studio, ενδέχεται να μην βρει το εγκατεστημένο JDK και να μας βγάλει κάποιο σχετικό μήνυμα. Σε αυτή την περίπτωση ανάλογα με την έκδοση του Android Studio που κατεβάσαμε είτε θα μας ζητήσει να του γνωστοποιήσουμε το path στο οποίο είναι εγκατεστημένο το JDK είτε θα πρέπει εμείς χειροκίνητα να πάμε στις ιδιότητες του υπολογιστή και στις μεταβλητές περιβάλλοντος να προσθέσουμε αν δεν υπάρχει, μια ακόμα με όνομα JAVA_HOME και σαν τιμή να θέσουμε το path στο οποίο είναι τοπικά εγκατεστημένο το JDK.

Το Android Studio περιέχει εξαρχής το Android SDK (Software Development Kit), το οποίο μας επιτρέπει να αναπτύξουμε Android εφαρμογές μιας και περιέχει όλες τις απαραίτητες βιβλιοθήκες της πλατφόρμας, προγραμματιστικά εργαλεία, έναν emulator (για να τρέχουμε τις εφαρμογές μας σε εικονικό smart phone αν δεν έχουμε πραγματική συσκευή Android) και αρκετά πρότυπα projects με τον κώδικά τους για εκπαιδευτικούς σκοπούς. Την πρώτη φορά που θα τρέξει το Android Studio θα κάνει αυτόματα μια ενημέρωση του SKD. Επιπρόσθετα εμείς μπορούμε να διαχειριστούμε όλα αυτά τα SDK tools μέσω του SKD Manager που διαθέτει το IDE. Συνεπώς είναι καλό να εγκαταστήσουμε όλα εκείνα να εργαλεία και τα API (περιέχει τον κώδικα που απαιτείτε για να αλληλεπιδράσει η εφαρμογή μας με το σύστημα του Android) που θεωρούμε απαραίτητα για την ανάπτυξη της εφαρμογής μας.

Στο παράδειγμά μας εγκατέστησα μέσω του SDK Manager το SDK platform και Google API για όλες εκείνες τις εκδόσεις του Android που είναι μεγαλύτερες από το API 15 το οποίο αποτελεί την κατώτερη έκδοση της εφαρμογής μου. Επίσης εγκατέστησα και κάποιες extra βοηθητικές βιβλιοθήκες όπως το Intel x86 Emulator Accelerator (HAXM installer) το οποίο βελτιώνει την απόδοση του emulator.

Τέλος, ένα ακόμα θέμα που θα πρέπει να ορίσουμε είναι αυτό της εκτέλεσης του προγράμματός μας. Για να τρέξουμε μια εφαρμογή σε Android περιβάλλον κατά την διάρκεια του developing υπάρχουν δυο κυρίως τρόποι. Ο ένας είναι να σετάρουμε ένα AVD (Android Virtual Device), δηλαδή μια εικονική μηχανή μέσα από το Android Studio και ο δεύτερος είναι να το τρέξουμε από την πραγματική συσκευή που έχουμε. Κάθε τρόπος έχει τα πλεονεκτήματα του και τα μειονεκτήματα του. Ο τρόπος με το AVD μας παρέχει μια πληθώρα επιλογών ως προς την έκδοση του λειτουργικού συστήματος, την μνήμη, το μέγεθος της οθόνης και την ανάλυση της, ωστόσο υστερεί κατά πολύ σε θέματα απόκρισης και καμιά φορά επιβραδύνει πολύ την διαδικασία παραγωγής. Αυτός ο τρόπος είναι ιδανικός για να δοκιμάσουμε όχι τόσο την λειτουργικότητα της εφαρμογής αλλά θέματα που αφορούν το πως εμφανίζεται ή αποκρίνεται η εφαρμογή σε διαφορετικού τύπου συσκευές, με άλλη έκδοση λογισμικού και διαφορετική οθόνη. Ο δεύτερος τρόπος εκτέλεσης, μέσω της φυσικής μας συσκευής, υστερεί σε ότι έχει να κάνει με την ποικιλομορφία των διαφορετικών χαρακτηριστικών που θα μπορούσαμε να συναντήσουμε αν είχαμε πολλές ακόμα διαφορετικές συσκευές, αλλά υπερτερεί

κατά πολύ σε θέματα απόκρισης και είναι ιδανικός αν θέλουμε επιτόπου να τεστάρουμε κάτι χωρίς σημαντικές καθυστερήσεις. Στα πλαίσια της δικιάς μου εφαρμογής χρησιμοποίησα κυρίως τον δεύτερο τρόπο για τους λόγους που ανέφερα.

Για να στήσουμε ένα AVD, θα πρέπει να τρέξουμε τον Android Virtual Device Manager που βρίσκεται μέσα στο Android Studio και μπορούμε να το βρούμε από το menu επιλέγοντας Tools > Android > AVD Manager. Στη συνέχεια συνεχίζουμε ως εξής :

1. Επιλέγουμε το κουμπί Create Virtual Device.
2. Στο νέο παράθυρο που θα εμφανιστεί από το αριστερό πλαίσιο επιλογής, επιλέγουμε τον τύπο της συσκευής που θέλουμε π.χ. Phone, Tablet κτλ.
3. Ανάλογα με την επιλογή μας, θα εμφανιστεί στο διπλανό πλαίσιο μια λίστα με τα διαθέσιμα μοντέλα. Το Android Studio περιέχει εξαρχής σεταρισμένα με τα πραγματικά τους χαρακτηριστικά όλα τα μοντέλα της γνωστής σειράς που παράγει η Google, τα Nexus. Εκτός όμως από τα Nexus έχει έτοιμα σεταρισμένα και κάποια άλλα μοντέλα που διαφέρουν σε ανάλυση οθόνης και τεχνικά χαρακτηριστικά. Αν κανένα από αυτά τα έτοιμα μοντέλα δεν μας ικανοποιεί, μπορούμε να φτιάξουμε ένα καινούργιο επιλέγοντας το κουμπί New Hardware Profile και συμπληρώνοντας τα κατάλληλα πεδία. Τελικά επιλέγουμε ένα μοντέλο από τη λίστα και πατάμε Next.
4. Στη νέα οθόνη που θα εμφανιστεί επιλέγουμε την έκδοση του λειτουργικού που θα τρέχει το AVD μας, επιβεβαιώνουμε τα εικονικά τεχνικά χαρακτηριστικά που θα έχει και του δίνουμε ένα όνομα. Τέλος πατάμε το κουμπί Finish για να δημιουργηθεί το νέο AVD.

Για να τρέξουμε ένα AVD μπορούμε να το κάνουμε είτε μέσω του Android Virtual Device Manager επιλέγοντας αυτό που επιθυμούμε , είτε πατώντας το κουμπάκι Run από το μενού του Android Studio και επιλέγοντας Launch Emulator και το όνομα του AVD που θέλουμε.

Για να σετάρουμε την πραγματική μας συσκευή να κάνει debug της εφαρμογής μας θα πρέπει να ακολουθήσουμε την παρακάτω απλή μεθοδολογία :

1. Θα πρέπει να βεβαιωθούμε πως η εφαρμογή μας είναι debuggable μέσω του αρχείου manifest ή μέσω του αρχείου build.gradle.

2. Θα πρέπει να ενεργοποιήσουμε την δυνατότητα debugging στην συσκευή μας. Αν έχουμε έκδοση του Android από 4.0 και πάνω τότε η ενεργοποίηση γίνεται αν επιλέξουμε στο κινητό μας Settings > Applications > Development. Αν έχουμε μικτότερη έκδοση από την 4.0 τότε την ενεργοποιούμε επιλέγοντας Settings > Developer options. Υπάρχει η περίπτωση σε μερικές νεότερες έκδοσης η ενότητα Developer options να είναι κρυφή, οπότε για να την κάνουμε ορατή πάμε στα Settings > About phone και πατάμε επτά φορές την καρτέλα Build number. Πηγαίνοντας πάλι πίσω η ενότητα Developer options θα έχει εμφανιστεί.
3. Θα πρέπει να συνδέσουμε με USB την συσκευή στο PC μας και να εγκαταστήσουμε τους κατάλληλους drivers για ADB (Android Debug Bridge) . Το ADB είναι ένα command line tool που μας επιτρέπει να επικοινωνούμε με μια συνδεδεμένη συσκευή Android ή έναν emulator για σκοπούς debugging. Για να εγκαταστήσουμε αυτούς τους drivers θα πρέπει να ακολουθήσουμε τα παρακάτω βήματα :
    - Κάνουμε δεξί κλικ στο *Computer* από την επιφάνεια εργασίας και επιλέγουμε *Manage*.
    - Επιλέγουμε *Devices* από το πλαϊνό πάνελ.
    - Στο κεντρικό πάνελ ανοίγουμε την κατηγορία *Other Devices*.
    - Βρίσκουμε την συσκευή μας, κάνουμε δεξί κλικ πάνω της και επιλέγουμε *Update Driver Software.*
    - Στον hardware update wizard που θα εμφανιστεί επιλέγουμε *Browse my computer for driver software* και μετά πατάμε *Next*.
    - Στη συνέχεια πατάμε *Browse* και πάμε να εντοπίσουμε τον φάκελο που περιέχει τους USB drivers. Ο Google USB Driver βρίσκεται μέσα στον φάκελο `<sdk>\extras\google\usb_driver\` .
    - Τελικά πατάμε *Next* για να εγκαταστήσουμε τον driver.

Η παραπάνω διαδικασία είναι για εγκατάσταση USB driver για πρώτη φορά σε περιβάλλον Windows 7.

Με την ολοκλήρωση της εγκατάστασης των ADB drivers έχουμε σείσει με επιτυχία το περιβάλλον πάνω στο οποίο θα αναπτύξουμε την Android εφαρμογή μας. Πλέον είμαστε έτοιμοι να περάσουμε στην συγγραφή του κώδικα!

2.5 Συγγραφή κώδικα

Στα προηγούμενα κεφάλαια αναλύσαμε το τι κάνει η εφαρμογή μας και γνωρίσαμε το περιβάλλον ανάπτυξης που χρησιμοποιήσαμε και το πώς αυτό στήθηκε. Στο παρόν κεφάλαιο θα μπούμε πιο βαθιά στον τρόπο με τον οποίο λειτουργεί η εφαρμογή, παρουσιάζοντας τις κλάσεις που χρησιμοποιήθηκαν και αναλύοντας την κάθε μια ξεχωριστά για να καταλάβουμε τον ρόλο που κάθε μια επιτελεί. Παρουσίαση κώδικα θα γίνει μόνο όπου είναι απαραίτητο μιας και στόχος αυτής της ενότητας δεν είναι να παρουσιάσει τον κώδικα της εφαρμογής (που έτσι κι αλλιώς υπάρχει στα συνοδευτικά αρχεία της εργασίας), αλλά να αναλύσει τον τρόπο που αυτή λειτουργεί μέσα από τις κλάσεις που χρησιμοποιήθηκαν. Με τον τρόπο αυτό ευελπιστώ να γίνει κατανοητός ο τρόπος σκέψης μου για την παρούσα εφαρμογή.

Η παρούσα εργασία για να υλοποιηθεί με την ανάλυση που έκανα και την πορεία που επέλεξα να ακολουθήσω, χρειάστηκε χωρία να συνυπολογιστούν τα έτσι κι αλλιώς απαραίτητα για ένα android project αρχεία (όπως για παράδειγμα το AndroidManifest.xml), 16 Java classes και 8 xml layout files από την μεριά του android application και 4 JavaEE classes συν 1 .php αρχείο για την υλοποίηση του back end το οποίο τρέχει στον server. Ο ρόλος των παραπάνω κλάσεων και αρχείων όπως και η δομή τους είναι αναλυτικά διαθέσιμη στον παρακάτω πίνακα.

Client side classes :

MainActivity.java	
Είναι ο κορμός ολόκληρης της εφαρμογής καθώς όλες οι συναρτήσεις που εκκινούν τις διάφορες λειτουργίες κρέμονται από αυτή την κλάση. Είναι επίσης υπεύθυνη για το ξεκίνημα της εφαρμογής και το σετάρισμα των αρχικών στοιχείων της, όπως είναι τα shared preferences και το tab layout.	
Σημαντικές συναρτήσεις κλάσης	
OnCreate()	• Δημιουργεί και αντικαταστεί την by default action bar με την custom toolbar που χρησιμοποίησα για να έχει η εφαρμογή περισσότερο material look and fill.

	• Φτιάχνει ένα sliding tab layout (tabs που τα αλλάζουμε σέρνοντας τα δάχτυλά μας στην οθόνη), με την βοήθεια των κλάσεων TabPagerAdapter, ViewPager και SlidingTabLayout. • Κάνει get τα shared preferences και τα κρατάει σε μια global μεταβλητή για μελλοντική χρήση.
getSharedPreferencesOnStart()	Κάνει initiate της κλάση LocationListener και κοιτάει να δει αν υπάρχουν αποθηκευμένα latitude και longitude στα sharedPreferenses. Αν βρει, τότε δημιουργεί ένα object τύπου GoogleMapsGPSParameters για να κρατήσει τις συντεταγμένες που βρήκε με σκοπό να τις δείξει στον χάρτη. Τέλος καλεί την συνάρτηση ReplaceViews. Αν δεν βρεί δεν κάνει κάτι επιπλέον.
replaceViews()	Η συνάρτηση αυτή καλείται όταν υπάρχει αποθηκευμένη γεωγραφική θέση και δουλειά της είναι να αλλάξει το GUI που φαίνεται στον χρήστη. Έτσι στο spot_position.xml κάνει invisible το βασικό στοιχείο που δείχνει το κουμπί για εύρεση θέσης και το αντικαταστεί με την οθόνη που δείχνει ποιά είναι η θέση αυτή και διάφορα actions επί της θέσης, έτσι όπως ορίζονται στο data_container.xml. Έτσι αρχικοποιεί τα elements της οθόνης και αλλάζει την γραμματοσειρά μέσω της κλάσης RobotoFont.
captureButtonHandler()	Βλέπει αν υπάρχει ενεργοποιημένος κάποιος provider της κλάσης LocationListener και αν βρεί τραβάει την γεωγραφική θέση που

	επιστρέφεται και την αποθηκεύει στα sharedPreferenses, καθώς επίσης καλεί την replaceViews. Αν δεν βρεί ενεργό provider ο location listener θα προτείνει στον χρήστη να ενεργοποιήσει έναν.
navigateToPoint()	Ξεκινάει την διεργασία που είναι υπεύθυνη να τρέξει τους GoogleMaps. Σετάρει την διεργασία παρέχοντας της ένα Uri στο οποίο βάζουμε τον προορισμό στον οποίο επιθυμούμε να οδηγηθούμε από το Google Maps navigation.
showMapButtonHandler()	Η συνάρτηση αυτή καλείτε αν ο χρήστης πατήσει το κουμπί "Show on map" από την οθόνη που δείχνει την αποθηκευμένη θέση του χρήστη. Δουλειά της είναι να καλέσει την κλάση SpotPositionGoogleMapActivity περνώντας σαν παραμέτρους στην διεργασία που ξεκινάει το γεωγραφικό μήκος και πλάτος της θέσης του χρήστη.
clearData()	Η συνάρτηση αυτή καθαρίζει τα sharedPreferenses και επαναφέρει το layout της πρώτης λειτουργιάς στην οθόνη με το κουμπάκι που προτρέπει τον χρήστη να ανιχνεύσει την θέση του.
logIn()	Η συνάρτηση αυτή καλείτε από το αρχικό layout της δεύτερης λειτουργίας (track_vehicle.xml) που αφορά την ανταλλαγή θέσεων μεταξύ δυο συνδεδεμένων χρηστών. Σκοπός της είναι να πάρει το username και password που θα δώσει ο χρήστης και μέσω της κλάσης SignInAsynkTask να κάνει ένα call στον server προκειμένου να αποκτήσει πρόσβαση στην λειτουργία.

replaceTrackVehicleView()	Αλλάζει το αρχικό layout της δεύτερης λειτουργίας της εφαρμογής κρύβοντας τα elements του track_vehicle.xml και κάνοντας μέσα του inflate το layout user_page.xml που περιλαμβάνει τις ενέργειες που μπορεί να κάνει ο συνδεδεμένος χρήστης.
signOut()	Από το layout user_page.xml μπορούμε να πατήσουμε το κουμπί "Sign out" το οποίο καλεί την παρούσα συνάρτηση που έχει σαν σκοπό να επαναφέρει το αρχικό track_vehicle.xml layout και να κρύψει το παρόν.
leadOrFollowFriend()	Αν από το layout user_page.xml και ενώ είμαστε λογκαρισμένοι στην εφαρμογή μπορούμε να πατήσουμε το κουμπί "WATCH FRIEND'S LOCATION" και έτσι θα αναλάβει η παρούσα συνάρτηση να σετάρει την κλάση TrackVehicleConnectionDialog η οποία θα δρομολογήσει την σύνδεση των χρηστών μεταξύ τους.

TabPagerAdapter.java

Η κλάση αυτή είναι υπεύθυνη για την ενσωμάτωση των δυο fragments της εφαρμογής σε ένα tab layout. Αυτό το κάνει υλοποιώντας την κλάση του android FragmentPagerAdapter. Τα δύο fragments που έχει η εφαρμογή μας είναι το SpotPosition και το TrackVehicle τα οποία αναφέρονται στις δυο ξεχωριστές λειτουργίες που περιέχει η εφαρμογή.	
Σημαντικές συναρτήσεις κλάσης	
getItem()	Ανάλογα με το fragment που επιλέγει ο χρήστης (ή πιο απλά την σελίδα που επιλέγει) από το tab layout, η συνάρτηση αυτή του επιστρέφει την αντίστοιχη κλάση που χειρίζεται το κάθε fragment.
getCount()	Επιστρέφει τον αριθμό των fragments που έχουν

	ενσωματωθεί στον tabPagerAdapter.
getPageTitle()	Επιστρέφει το όνομα του επιλεγμένου fragment.

SpotPosition.java

Η κλάση αυτή υλοποιεί το fragment της λειτουργίας που αφορά την καταγραφή και υπενθύμιση μιας γεωγραφικής θέσης. Κάθε φορά που επιλέγεται το συγκεκριμένο fragment, η κλάση αναλαμβάνει να το φορτώσει στο ViewGroup της εφαρμογής μέσω του inflater.	
Σημαντικές συναρτήσεις κλάσης	
onResume()	Κάθε φορά που εκκινεί η εφαρμογή καλεί την συνάρτηση getSharedPreferencesOnStart της MainActivity.

TrackVehicle.java

Η κλάση αυτή υλοποιεί το fragment της λειτουργίας που αφορά την ανταλλαγή γεωγραφικών συντεταγμένων μεταξύ των συνδεδεμένων χρηστών. Κάθε φορά που επιλέγεται το συγκεκριμένο fragment, η κλάση αναλαμβάνει να το φορτώσει στο ViewGroup της εφαρμογής μέσω του inflater.

LocationListener.java

Είναι ουσιαστικά ένας custom location listener. Δουλεία του είναι να συνδέεται και να λαμβάνει τις γεωγραφικές συντεταγμένες στις οποίες βρίσκεται η εφαρμογή, από το GPS ή το Internet ή και συνδυασμό αυτών. Ο συγκεκριμένος location listener χρησιμοποιείτε παντού στην εφαρμογή όπου χρειάζεται να αντλήσουμε τη θέση μας.	
Σημαντικές συναρτήσεις κλάσης	
initialize()	Η συνάρτηση αυτή βλέπει αν υπάρχει ενεργός location provider όπως το GPS ή το Internet (Wi-Fi ή κεραίες τηλεφωνίας). Αν υπάρχει λαμβάνει την τελευταία γνωστή θέση και την ορίζει ως την καλύτερη ως τώρα γνωστή θέση. Επίσης κάνει request για να λαμβάνει νεότερες γεωγραφικές συντεταγμένες αν αυτές προκύψουν. Αν δεν υπάρχει ενεργός provider τότε προτείνει στον χρήστη να ενεργοποιήσει έναν.

onLocationChanged()	Κάθε φορά που έρχεται μια νέα θέση κοιτάει να δει αν είναι πιο ακριβής από την προηγούμενη, καλώντας τη συνάρτηση isBetterLocation. Αν είναι καλύτερη την κρατάει, αλλιώς κρατάει την παλιά.
isBetterLocation()	Χρησιμοποιεί έναν αλγόριθμο, ο οποίος με κάποια συγκεκριμένα κριτήρια όπως το πόσο παλιά ή πόσο ακριβής είναι μια θέση ή αν προέρχεται από τον ίδιο provider, αποφασίζει αν η θέση αυτή είναι καλύτερη από την παλιά.
onProviderEnabled()	Αν ενεργοποιηθεί κάποιος provider κατά την χρήση της εφαρμογής κάνει request για να λαμβάνει θέσεις και από τον νέο provider.

GeocoderUtility.java

Η κλάση αυτή περιέχει το function που μας μετατρέπει τις συντεταγμένες σε διεύθυνση.	
Σημαντικές συναρτήσεις κλάσης	
getAddressNameByCoordinates()	Κάνει reverse geocoding και επιστρέφει το όνομα της οδού που βρίσκεται ο χρήστης αντί για την γεωγραφική του θέση.

GoogleMapsGPSParameters.java

Η κλάση αυτή δημιουργεί ένα object για να αποθηκεύει τα latitudes και longitudes που λαμβάνονται από τα sharedPreferenses , και περνάει σαν παράμετρος στους GoogleMaps.

RobotoFont.java

Σετάρει το roboto font style που κατέβασα από τα Google fonts.

SpotPositionGoogleMapActivity.java

Αυτή η κλάση είναι υπεύθυνη για να μας δείξει στον χάρτη τη γεωγραφική θέση που αποθήκευσε ο χρήστης στα sharedPreferenses.

Σημαντικές συναρτήσεις κλάσης	
onCreate()	• Φορτώνει το layout των GoogleMaps από το google_maps.xml και το σετάρει με συγκεκριμένες προτιμήσεις. • Διαβάζει το γεωγραφικό πλάτος και μήκος που περάστηκαν σαν παράμετροι στην διεργασία που εκτέλεσε την κλάση και φτιάχνει ένα αντικείμενο τύπου LatLng το οποίο προσθέτει σαν Marker στον χάρτη. • Με την συνάρτηση animateCamera των GoogleMaps κεντράρει το σημείο που θέλουμε να δείξουμε στον χάρτη.

SignInAsynkTask.java

Η κλάση αυτή επεκτείνει την κλάση του android AsyncTask της οποίας ο ρόλος είναι να τρέχει παρασκηνιακά σε ένα νέο thread και να επιστρέφει κάποια αποτελέσματα στο κύριο νήμα της εφαρμογής, χωρίς αυτό να διακόπτεται και να περιμένει την ολοκλήρωσή του. Σκοπός της δικιάς μας κλάσης είναι να κάνει ασύγχρονα ένα HTTP call στον server για να διαπιστεύσει τον χρήστη που αιτείτε σύνδεση στην εφαρμογή.	
Σημαντικές συναρτήσεις κλάσης	
doInBackground()	Η συνάρτηση αυτή που ανήκει στη κλάση AsyncTask ορίζει τις ενέργειες που θα γίνου παρασκηνιακά. Στη περίπτωσή μας καλεί τη συνάρτηση signIn().
signIn()	Κάνει ένα HTTP call τύπου POST στο sign_in.php αρχείο που τρέχει στον server και κάνει την ταυτοποίηση του χρήστη. Η συνάρτησή αυτή περιμένει ένα response από τον server το οποίο αν είναι θετικό (200) τότε παρσάρει το response του server, γνωστοποιεί αν η ταυτοποίηση χρήστη πέτυχε και τερματίζει την σύνδεση.
onPostExecute()	Η συνάρτηση αυτή που ανήκει στη κλάση AsyncTask ορίζει τις ενέργειες που θα γίνου αφού το thread ολοκληρώσει τις ενέργειές του. Στη περίπτωσή μας

	βλέπει αν ο χρήστης βρέθηκε και αν βρέθηκε καλεί την συνάρτηση της MainActivity replaceTrackVehicleView() με παράμετρο το username του χρήστη.

TrackVehicleConnectionDialog.java

Η κλάση αυτή που επεκτείνει την κλάση του android DialogFragment είναι υπεύθυνη για την δημιουργία και την διαχείριση ενός dialog box που έχει σαν σκοπό να παρακινήσει τον χρήστη να εισάγει το username του χρήστη με τον οποίο επιθυμεί να συνδεθεί και να ορίσει ποιός θα είναι ο ρόλος του κατά την διάρκεια της σύνδεσης.	
Σημαντικές συναρτήσεις κλάσης	
onCreateDialog()	Αυτή η κλάση πάει και ενσωματώνει το dialog_connect_with.xml που είναι το εσωτερικό interface του dialog box, στο parent view της διεργασίας που κάλεσε την κλάση. Επίσης σετάρει τα actions που θα εφαρμοστούν σε κάθε επιλογή του χρήστη και τελικά αν ο χρήστης δώσει τα απαιτούμενα στοιχεία ξεκινάει την διεργασία που θα τρέξει την κλάση TrackVehicleGoogleMapActivity, στην οποία περνάει σαν όρισμα το username του χρήστη, το username του απομακρυσμένου χρήστη και το action (ρόλο) που επέλεξε ο παρών χρήστης να έχει κατά την σύνδεση.

TrackVehicleGoogleMapActivity.java

Η κλάση αυτή αποτελεί τον κορμό της αλληλεπίδρασης των δύο χρηστών καθώς συντονίζει μια πληθώρα από διαδικασίες που έχουν σαν σκοπό την ανταλλαγή θέσεων και την συγχρονισμένη αναπαράστασή τους στον χάρτη.	
Σημαντικές συναρτήσεις κλάσης	
onCreate()	Αρχικά διαβάζει όλες τις παραμέτρους που έχουν περάσει μέσω της διεργασίας (με χρήση Bundle) στην κλάση, με σκοπό να σετάρει τις απαιτούμενες μεταβλητές. Στη συνέχεια φορτώνει το google_maps.xml layout για να φορτωθεί ο

	χάρτης στον οποίο μαρκάρει την θέση του χρήστη που την ξέρει από τον location listener. Ακολούθως ανοίγει το web socket που χρειάζεται για την επικοινωνία των δύο χρηστών μέσω της κλάσης MyWebSocketClient, ενώ παράλληλα δημιουργεί ένα google maps location listener για να λαμβάνει μέσω αυτού τις νέες θέσης του χρήστη. Τέλος δημιουργεί ένα scheduled thread το οποίο εκτελείτε κάθε 5sec με σκοπό να ανανεώνει τις θέσεις των χρηστών στον χάρτη μέσω της συνάρτησης periodicallyReadLocations()
periodicallyReadLocations()	Ανάλογα με τον ρόλο που έχει επιλέξει ο χρήστης να έχει κατά την διάρκεια της ανταλλαγής θέσεων με τον απομακρυσμένο χρήστη (lead ή follow), η συνάρτηση αυτή εκτελεί την συνάρτηση drawRoute() με συγκεκριμένες παραμέτρους.
drawRoute()	Η συνάρτηση αυτή είναι ο κορμός της απεικόνισης των θέσεων των χρηστών και της μεταξύ τους διαδρομής στο χάρτη. Έτσι αρχικά, εκτελεί σε ένα νέο νήμα τον έλεγχο για το ποιός είναι ο leader και ποιός ο follower και αναλόγως σετάρει της μεταβλητές origin και dest που θα χρειαστούν για τον υπολογισμό της διαδρομής από τον ένα χρήστη στον άλλο. Δηλώνουν δηλαδή οι μεταβλητές αυτές από πού ξεκινάει η διαδρομή και που καταλήγει. Επίσης μέσω της συνάρτησης getDirectionsUrl() και της εμφωλευμένης κλάσης DownloadTask σετάρει και εκτελεί το service που είναι υπεύθυνο για την λήψη οδηγιών πλοήγησης από τον ένα χρήστη στον άλλο.
getDirectionsUrl()	Φτιάχνει το URL που θα χτυπήσει το κατάλληλο

	service για λήψη οδηγιών πλοήγησης. Το συγκεκριμένο URL χτυπάει τους servers της Google με παραμέτρους το σημείο εκκίνησης και το σημείο προορισμού, για να πάρει σαν response ένα JSON με τις οδηγίες.
downloadURL()	Κάνει ένα HTTP call στο URL που σετάραμε προηγουμένως και παίρνει το response.
showAllMembersOnScreen()	Παίρνει όλα τα markerPoints μαζί με τα southwest και northeast σημεία που επέστρεψε το response παραπάνω και τα συνδυάζει έτσι ώστε να κεντραριστούν στο χάρτη οι θέσεις των χρηστών και η μεταξύ τους διαδρομή. Αυτό γίνεται για λόγους παρουσίασης.
Εμφωλευμένες Κλάσεις	
DownloadTask	Μια ασύγχρονη διαδικασία που καλεί την συνάρτηση downloadURL() για να γίνει το download των οδηγιών. Όταν αυτή ολοκληρωθεί μέσω της εμφωλευμένης κλάσης ParserTask χειρίζεται τις οδηγίες που έλαβε.
ParserTask	Με ασύγχρονο τρόπο, μέσω της κλάσης DirectionsJSONParser παρσάρει τις οδηγίες που ήρθαν σαν response, σε μορφή που μπορεί να χρησιμοποιηθεί εύκολα από την Java. Όταν τελειώσει η μετατροπή σχεδιάζει την διαδρομή μεταξύ των χρηστών στο χάρτη και καλεί την συνάρτηση showAllMembersOnScreen() για να κεντράρει τα αποτελέσματα.

MyWebSocketClient.java
Η κλάση αυτή είναι υπεύθυνη για να ανοίξει ένα web socket μεταξύ του χρήστη και του server, με απώτερο σκοπό την επικοινωνία δυο χρηστών. Διαχειρίζεται την σύνδεση των δυο χρηστών, την αποστολή και λήψη γεωγραφικών συντεταγμένων.

Σημαντικές συναρτήσεις κλάσης	
connectWebSocket()	Η συνάρτηση αυτή εκκινεί την σύνδεση μεταξύ του χρήστη και του server. Επειδή το android δεν έχει κάποια δικιά του βιβλιοθήκη για την υλοποίηση των WS, χρησιμοποίησα μια third party βιβλιοθήκη που τα υλοποιεί. Μέσα στη συνάρτηση αυτή λοιπόν, υπάρχουν εμφωλευμένες συναρτήσεις που υλοποιούν την βιβλιοθήκη που επέλεξα (Java WebSockets). Οι συναρτήσεις αυτές είναι οι onOpen(), onMessage(), onClose() και onError(). Από αυτές σημαντικές είναι οι δύο πρώτες.
onOpen()	Διαχειρίζεται τις ενέργειες που γίνονται την στιγμή που ανοίγει το WS. Στη περίπτωσή μας στέλνει στον server ένα JSON με τα στοιχεία του χρήστη και την θέση του καθώς και το username του χρήστη με τον οποίο επιθυμεί να συνδεθεί, έτσι ώστε ο server να μαρκάρει τον χρήστη που πραγματοποίησε το call σαν συνδεδεμένο.
onMessage()	Αυτή η συνάρτηση διαχειρίζεται τις ενέργειες που πρέπει να γίνουν κάθε φορά που έρχεται ένα message από το WS. Ρόλος της είναι να λαμβάνει το μήνυμα του απομακρυσμένου χρήστη και να προωθεί την θέση του στη συνάρτηση friendPositionListener() της κλάσης TrackVehicleGoogleMapActivity.
sendMessage()	Πραγματοποιεί την αποστολή μηνύματος μέσω του WS.
isWSConnected()	Κάνει έλεγχο για να δει αν υπάρχει συνδεδεμένο WS.

DirectionsJSONParser.java
Η κλάση αυτή παίρνει σαν όρισμα ένα JSON με οδηγίες πλοήγησης και το

μετατρέπει σε μια λίστα από λίστες τύπου LatLng. Επίσης μετατρέπει σε LatLng και τα βάζει σε λίστα και τα polylines που πείρε από το response.

Server side classes :

sign_in.php

Αυτό το αρχείο php που τρέχει στον server που έχω στήσει για την εφαρμογή, συνδέεται στη mySQL βάση που τρέχει επίσης στον server και περιέχει τους χρήστες που έχουν δικαίωμα εισόδου στην εφαρμογή και κάνει query για να δει αν ο χρήστης που έκανε το αίτημα εισόδου υπάρχει. Αν υπάρχει σετάρει σε ένα JSON τα στοιχεία του και τα στέλνει στην εφαρμογή σαν response. Αν δεν βρεθεί ο χρήστης στέλνει response "Not Signed in".

LocationMessage.java

Είναι το DTO που περιγράφει το JSON που ανταλλάσσεται μεταξύ των χρηστών κατά την επικοινωνία τους. Με άλλα λόγια το object στο οποίο θα κρατούνται οι τιμές που θα έρχονται μέσω των μηνυμάτων του WS.

LocationMessageEncoder.java

Η εφαρμογή μας χρησιμοποιεί το Java API για JSON Processing που είναι κομμάτι της Java EE 7, το οποίο επιτρέπει η επικοινωνία μεταξύ client και server να γίνεται με JSON Objects. Αυτό σημαίνει πως για να στείλουμε κάτι στο WS πρέπει από text να το μετατρέψουμε σε JSON Object. Την δουλειά αυτή (encoding) την αναλαμβάνει η παρούσα κλάση, η οποία δηλώνεται σαν κυρίως κλάση (LocationWebSoket) ως ο Server endpoint encoder.

LocationMessageDecoder.java

Με βάση τα όσα αναφέραμε στην κλάση LocationMessageEncoder, η παρούσα κλάση έχει ως αντικείμενο την αντίθετη δουλειά. Σκοπός της είναι να πάρει το JSON Object που έρχεται και να το μετατρέψει πάλι σε text. Η κλάση αυτή λοιπόν όπως και η προηγούμενη δηλώνεται ως Server endpoint decoder, στη κύρια κλάση.

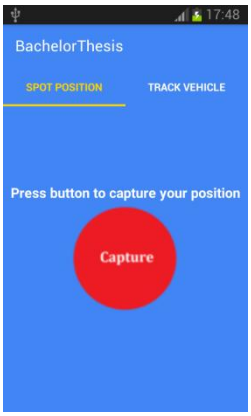
LocationWebSoket.java	
Η παρούσα κλάση είναι η κύρια κλάση της υλοποίησης του WS στον server μας. Εδώ δηλώνονται και διαχειρίζονται όλες οι ενέργειες του WS. Έτσι ανάλογα με το τι συμβαίνει στο WS μας, η κλάση αυτή αναλαμβάνει να κάνει τις ενέργειες που απαιτούνται.	
Σημαντικές συναρτήσεις κλάσης	
onOpen()	Κάθε φορά που ανοίγει επιτυχώς μια νέα σύνδεση στο WS στέλνει μήνυμα "200", για να υποδηλώσει πως η σύνδεση έγινε με επιτυχία. Επίσης αποθηκεύει το session που δημιουργήθηκε σε μια λίστα που κρατάει όλα τα ενεργά sessions.
onMessage()	Η συνάρτηση αυτή έχει μια κάπως σύνθετη λειτουργία. Σκοπός της είναι αρχικά να δει αν το μήνυμα που ήρθε, εστάλει από κάποιο μέλος που έχει ήδη δηλωθεί στη λίστα των χρηστών. Αν βρεθεί στη λίστα χρηστών (αν έχει ξαναστείλει δηλαδή μήνυμα), τότε η συνάρτηση προωθεί το μήνυμα αυτό στον δηλωμένο παραλήπτη. Να θυμίσω πως η πληροφορία σχετικά με το ποιος είναι ο χρήστης και σε ποιον στέλνει αποτυπώνεται στο JSON Object που ανταλλάσατε κατά την επικοινωνία. Αν ο χρήστης δεν βρεθεί, τότε τον προσθέτει στη λίστα χρηστών και στη συνέχεια προωθεί κανονικά το μήνυμα του. Η προώθηση του μηνύματος γίνεται μέσω της συνάρτησης sendMessage().
sendMessage()	Η συνάρτηση αυτή έχει το ρόλο του να στέλνει μηνύματα μέσω του WS στους συνδεδεμένους χρήστες. Έτσι εισάγοντας κατά κάποιον τρόπο μια δικλείδα ασφαλείας, ώστε να μην πηγαινοέρχονται ανεξέλεγκτα και προς κάθε κατεύθυνση τα διάφορα μηνύματα, κάνει αρχικά έναν έλεγχο για να δει αν ο παραλήπτης έχει εξουσιοδοτήσει τον αποστολέα

	να του στέλνει μηνύματα και αναλόγως τα προωθεί ή τα ακυρώνει. Θυμίζω πως όταν ο αποστολέας στέλνει ένα μήνυμα, το μήνυμα περιέχει το username αυτού με τον οποίο θέλει να επικοινωνήσει. Αν και ο παραλήπτης αντίστοιχα κατά την σύνδεση έχει δηλώσει το όνομα του αποστολέα ως επιθυμητό για σύνδεση, τότε η συνάρτηση του προωθεί το μήνυμα. Με αυτό τον τρόπο θα πρέπει και οι δύο πλευρές να έχουν συναινέσει για την πραγματοποίηση της επικοινωνίας, διαφορετικά έστω ένας να διαφωνεί, ποτέ δεν θα πάει μήνυμα από τον ένα στον άλλο.
onClose()	Κάθε φορά που κλείνει μια σύνδεση στο WS, η συνάρτηση αυτή φροντίζει έτσι ώστε να αφαιρέσει τον χρήστη που διατηρούσε την σύνδεση από τη δηλωμένη λίστα χρηστών.

2.6 Παράδειγμα χρήσης

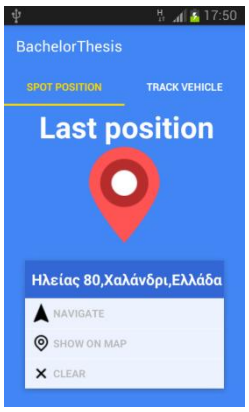
Στη παρακάτω ενότητα που είναι η τελευταία που αφορά την ανάλυση της εφαρμογής, θα δούμε με εικόνες από ένα παράδειγμα χρήσης για κάθε λειτουργία της.

Όταν αρχικά ο χρήστης εκκινεί την εφαρμογή οδηγείτε στη παρακάτω οθόνη, η οποία περιέχει ένα κουμπί το οποίο πατώντας ο χρήστης, μπορεί να βρει την γεωγραφική θέση στην οποία βρίσκεται και να την καταγράψει.



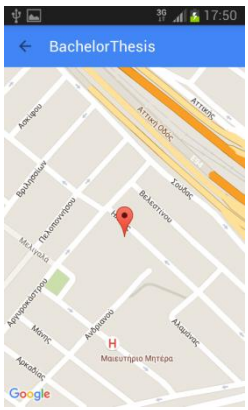
Εικόνα 1 - Αρχική οθόνη

Όταν η θέση καταγραφεί, ο χρήστης θα βλέπει πλέον την παρακάτω εικόνα, η οποία περιέχει το όνομα, τον αριθμό και την περιοχή της διεύθυνσης στην οποία βρίσκεται.

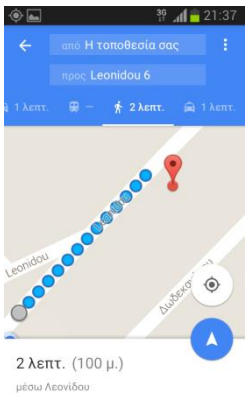


Εικόνα 2 - Λεπτομέρειες καταγεγραμμένης διεύθυνσης

Όπως βλέπουμε ο χρήστης έχει πλέον τρεις επιλογές σχετικές με την θέση την οποία κατέγραψε. Μπορεί είτε να την διαγράψει πατώντας "CLEAR" και να επιστρέψει στην αρχική οθόνη (Εικόνα 1), είτε να δει στον χάρτη που βρίσκεται αυτή η θέση πατώντας "SHOW ON MAP" (Εικόνα 3) ή να πάρει οδηγίες πλοήγησης προς αυτή την θέση πατώντας "NAVIGATE" (Εικόνα 4). Οι οθόνες που θα δει ο χρήστης ανάλογα με την επιλογή που θα κάνει είναι οι παρακάτω.



Εικόνα 3 - Show on map

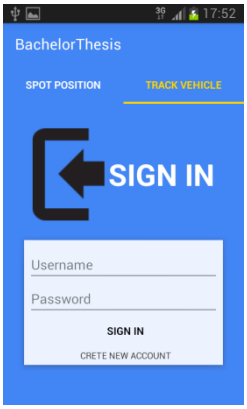


Εικόνα 4 - Navigate

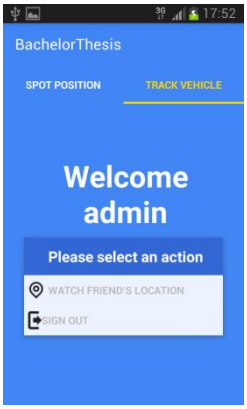
Κλείνοντας με τις παραπάνω εικόνες έχω δώσει ένα πλήρες παράδειγμα του τι μπορεί ο χρήστης να κάνει με την πρώτη λειτουργικότητα της εφαρμογής.

Όσον αφορά την δεύτερη λειτουργικότητα, η οποία αφορά την σύνδεση και ανταλλαγή γεωγραφικών θέσεων μεταξύ δυο χρηστών με σκοπό το συσχετισμό τους στο χάρτη, ο χρήστης πρέπει από το tab layout της εφαρμογής να επιλέξει το tab "TRACK VEHICLE", για να μεταφερθεί στην αρχική οθόνη της δεύτερης λειτουργικότητας. Σε αυτή την οθόνη θα κληθεί να εισάγει τα credentials του (

username και password, αν έχει προηγούμενος δημιουργήσει κάποιον λογαριασμό) προκειμένου να του επιτραπεί η χρήση της λειτουργικότητας. Ο λόγος για τον οποίο απαιτούνται credentials είναι διότι κάθε χρήστης πρέπει να έχει ένα username, το οποίο θα μπορεί να χρησιμοποιήσει για να αναζητήσει και να συνδεθεί με άλλον χρήστη. Αφού η σύνδεση πραγματοποιηθεί επιτυχώς ο χρήστρης θα οδηγηθεί σε ένα menu απ' όπου θα μπορεί είτε να κάνει log out είτε να επιλέξει "WATCH FRIEND'S LOCATION" προκειμένου να συνδεθεί με έναν άλλο χρήστη. Οι δύο οθόνες φαίνονται παρακάτω.

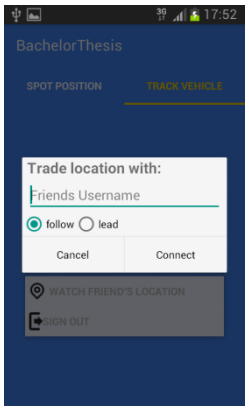


Εικόνα 5 - Sign in track vehicle operation



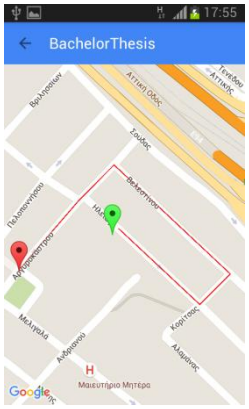
Εικόνα 6 - Track vehicle menu

Πατώντας το κουμπί για επιτήρηση της θέσης ενός άλλου χρήστη, ο παρών χρήστης θα χρειαστεί να εισάγει το username αυτού με τον οποίο θέλει να συνδεθεί. Επίσης θα πρέπει να επιλέξει έναν από τους δύο ρόλους, τον οποίο θα έχει κατά την διάρκεια της σύνδεσης follow για να ακολουθήσει τον άλλο χρήστη ή lead για να τον ακολουθήσει ο άλλος χρήστης. Εδώ να σημειώσουμε πως αυτό θα γίνει με συνεννόηση των δύο χρηστών, δεν γίνεται και οι δύο να έχουν τον ίδιο ρόλο. Η παρακάτω εικόνα δείχνει το dialog box το οποίο πρέπει να συμπληρώσει ο χρήστης.



Εικόνα 7 - Connect with friend

Αμέσως μετά την επιτυχή σύνδεση, ο χρήστης θα οδηγηθεί σε έναν χάρτη όπου με πράσινο marker θα βλέπει το στίγμα του και με κόκκινο το στίγμα του φίλου του. Εδώ πρέπει να σημειώσω ότι αν παράλληλα ο άλλος χρήστης δεν έχει συνδεθεί, τότε ο παρών χρήστης θα βλέπει μόνο το δικό του στίγμα. Το στίγμα του άλλου χρήστη θα το δει όταν αυτός συνδεθεί. Όταν έχουν πλέον συνδεθεί και οι δύο μία κόκκινη γραμμή με κατεύθυνση από τον follower προς τον leader θα εμφανιστεί για να υποδείξει στον follower το ποιό δρόμο πρέπει να ακολουθήσει για να φτάσει στον leader. Η κόκκινη γραμμή που εμφανίστηκε μεταξύ τον χρηστών δεν είναι αυθαίρετη, αλλά όπως έχουμε ήδη αναφέρει είναι αποτέλεσμα του http call που γίνεται στη Google για να επιστρέψει τις οδηγίες πλοήγησης από τον ένα χρήστη στον άλλο, σεβόμενη πάντα τις κυκλοφοριακές ρυθμίσεις της εκάστοτε περιοχής (δεν θα δείξει στον χρήστη να πάρει έναν δρόμο στον οποίο απαγορεύεται να μπει). Η διαδρομή και οι θέσεις των χρηστών ανανεώνονται αυτόματα ανά τακτά χρονικά διαστήματα. Ακολουθεί η εικόνα που δείχνει τα παραπάνω.



Εικόνα 8 - Συνδεδεμένοι χρήστες

ΚΕΦΑΛΑΙΟ 3

ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΣΑ

Σε αυτό το κεφάλαιο γίνεται μια αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής μου. Για κάθε μια από αυτές θα γίνει μια όσο το δυνατόν ουσιαστικότερη περιγραφή χωρίς να αναλύονται εις βάθος. Ο λόγος που θα συμβεί αυτό, είναι διότι ορισμένες όπως είναι ο application server Glassfish και τα web sockets, θα μπορούσαν να αποτελέσουν από μόνες τους μια ξεχωριστή εργασία. Επίσης για κάθε μια από αυτές τις τεχνολογίες θα αναφέρουμε αν υπάρχει και μια εναλλακτική της, έτσι ώστε να υπάρχει μια όσο το δυνατόν σφαιρικότερη άποψη για τις τεχνολογίες που χρησιμοποιήθηκαν.

3.1 Google Maps

Η εφαρμογή που περιγράφεται στην παρούσα πτυχιακή εργασία, είναι μια εφαρμογή που κατεξοχήν ασχολείται με την καταγραφή και προβολή γεωγραφικών συντεταγμένων στον χάρτη για να ικανοποιεί το business κομμάτι που εξυπηρετεί. Ως προς αύτη την κατεύθυνση λοιπόν, δεν θα μπορούσε παρά να μην υπάρχει ένας προβληματισμός ως προς το πώς θα βάλουμε έναν χάρτη στην εφαρμογή μας και πως θα τον χειριστούμε για να πετύχουμε τον σκοπό μας. Την λύση στον προβληματισμό αυτό έδωσαν οι Google Maps **[11-13]**.

Οι Google Maps ξεκίνησαν ως μια desktop web mapping εφαρμογή (χρησιμοποιεί δηλαδή χάρτες που παράγονται από τα λεγόμενα GIS [Geographical Information Systems] , τα οποία καταγράφουν, αποθηκεύουν, διαχειρίζονται και παρουσιάζουν κάθε γεωγραφική πληροφορία.) που στη συνέχεια μετατράπηκε σε ένα web application, ενώ κάποια στιγμή αργότερα κυκλοφόρησε και ως mobile application. Αυτό που προσφέρει είναι δορυφορικές εικόνες τοποθεσιών, προβολή οδικών δικτύων, προγραμματισμό διαδρομής και οδηγίες πλοήγησης για μεταφορά από μια τοποθεσία σε μια άλλη, μέσω αυτοκινήτου, ποδηλάτου, μέσων μαζικής μεταφοράς ή και με τα πόδια.

Ωστόσο δεν είναι διαθέσιμοι μόνο ως αυτόνομη εφαρμογή. Οι Google Maps διαθέτουν ένα API (Application Programming Interface) μέσω του οποίου μπορούμε να εισάγουμε τους χάρτες και τις υπηρεσίες που τους συνοδεύουν και σε άλλες εφαρμογές. Αυτός είναι και ο τρόπος με τον οποίο χρησιμοποιήθηκαν οι

Google Maps στην εφαρμογή μας. Εκμεταλλεύτηκα όλα αυτά τα έτοιμα χαρακτηριστικά που μας χαρίζουν και μέσω του API που παρέχουν τους προσάρμοσα στις ανάγκες της παρούσας εφαρμογής.

Για να έχουμε πρόσβαση και να μπορούμε να χρησιμοποιήσουμε τους Google Maps πρέπει αρχικά να εγκαταστήσουμε μέσω του Android SDK τα Google Play services. Είναι ουσιαστικά οι βιβλιοθήκες μέσω των οποίων έχουμε πρόσβαση σε διάφορες υπηρεσίες της Google μεταξύ των οποίων είναι και οι Google Maps. Στη συνέχεια θα πρέπει να αποκτήσουμε ένα Google Maps API key. Πρόκειται για ένα αναγνωριστικό που θα μας επιτρέψει να έχουμε πρόσβαση στους Google Maps servers και στο σχετικό API. Το key αυτό παρέχεται δωρεάν, υποστηρίζει απεριόριστο αριθμό χρηστών και πολύ συνοπτικά ο τρόπος για να το αποκτήσουμε είναι ο ακόλουθος :

- Αποκτούμε ένα SHA-1 fingerprint το οποίο είναι το ψηφιακό πιστοποιητικό της εφαρμογής μας. Ο τρόπος για να γίνει αυτό στα Windows είναι να πάμε στο path `C:\Users\your_user_name\.android\` , να ανοίξουμε ένα cmd και να τρέξουμε την παρακάτω εντολή `keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android`. Από τη πληροφορία που θα εμφανιστεί αντιγράφουμε τον κωδικό που έχει ετικέτα SHA1.
- Στη συνέχεια κάνουμε log in στη Google Developers Console με τον λογαριασμό που πρέπει να έχουμε δημιουργήσει.
- Δημιουργούμε ή επιλέγουμε ένα υπάρχον project και πατάμε *Continue* για να ενεργοποιηθεί το API.
- Από το menu πάμε στα Credentials για να αποκτήσουμε ένα Android key.
- Στο παράθυρο που θα εμφανιστεί κάνουμε επικόλληση το SHA1 fingerprint που αποκτήσαμε παραπάνω και το όνομα του πακέτου της εφαρμογής μας.
- Τέλος αντιγράφουμε το Android key που θα εμφανιστεί και το βάζουμε στο AndroidManifest.xml αρχείο της εφαρμογής μας ως εξής :

```
<meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="YOUR_API_KEY"/>
```

Με αυτό τον τρόπο έχουμε πλέον πρόσβαση στο Google Maps API. Για να το χρησιμοποιήσουμε θα πρέπει να γράψουμε τον κατάλληλο κώδικα, ο οποίος

αποτελείτε από δύο αρχεία. Το πρώτο είναι το XML layout αρχείο με το οποίο θα εμφανίζουμε γραφικά τον χάρτη στην οθόνη και το δεύτερο είναι μια Java κλάση μέσα στην οποία θα χειριζόμαστε και θα σετάρουμε τους Google Maps. Έτσι ο ελάχιστος κώδικας που απαιτείτε είναι ο ακόλουθος:

XML:

```
<fragment
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/map"
  tools:context=".MapsActivity"
  android:name="com.google.android.gms.maps.SupportMapFragment" />
```

Java :

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
...

public class MapsActivity extends FragmentActivity{

    private GoogleMap mMap;

    @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);
    }
...
}
```

Έτσι όπου χρειάστηκε να χρησιμοποιήσω τους Google Maps αρκούσε να συμπεριλάβω τον παραπάνω κώδικα στα κατάλληλα σημεία, προσαρμοσμένο στις δικές μου ανάγκες. Ο παραπάνω κώδικας είναι απλά μια βάση για το τι μπορούμε να κάνουμε, συνεπώς υπάρχουν και πολλά ακόμα πράγματα που μας παρέχει το Google Maps API και μερικά από αυτά μπορούμε να τα δούμε αναλυτικότερα στον κώδικα της εφαρμογής, όπως το πώς προσθέτουμε markers ή animation στον χάρτη.

Μία άλλη εναλλακτική για προβολή και χειρισμό χαρτών είναι το open source toolset που παρέχει η Mapbox. Ωστόσο η άνεση, η ευκολία, η σταθερότητα και η εξοικείωση που είχα με τους Google Maps με έκαναν να τους χρησιμοποιήσω χωρίς ιδιαίτερη σκέψη. Πέρα από το Mapbox υπάρχουν και μερικά ακόμα mapping APIs που θα μπορούσαν να χρησιμοποιηθούν όπως είναι το δημοφιλές και με καλές κρητικές Bing Maps της Microsoft.

3.2 Geocoder

Μία ακόμα τεχνολογία που χρησιμοποιήθηκε στην εφαρμογή μας και αφορά την αντιστοίχηση ενός γεωγραφικού πλάτους και μήκους με το όνομα μιας οδού είναι η κλάση Geocoder **[8]**. Γενικά το geocoding είναι η διαδικασία με την οποία αντιστοιχούμε μια διεύθυνση στο αντίστοιχο γεωγραφικό πλάτος και μήκος. Στην παρούσα εφαρμογή η συγκεκριμένη μετατροπή δεν ήταν χρήσιμη μιας και ρόλος της εφαρμογής ήταν να διαχειρίζεται γεωγραφικά πλάτη και μήκη και όχι διευθύνσεις. Άρα το ερώτημα είναι, πώς κατάφερα και εμφάνισα στον χρήστη το όνομα της διεύθυνσης (οδός και αριθμός) στην οποία βρίσκεται ξερώντας μόνο το γεωγραφικό πλάτος και μήκος, αφού το geocoding είναι χρήσιμο όταν ξέρω μόνο το όνομα της διεύθυνσης και ψάχνω τις αντίστοιχες γεωγραφικές συντεταγμένες.

Ο τρόπος που το κατάφερα αυτό λέγεται reverse geocoding. Reverse geocoding είναι η διαδικασία κατά την οποία ξέροντας ένα γεωγραφικό πλάτος και μήκος μπορούμε να πάρουμε την διεύθυνση στην οποία αυτά αντιστοιχούν στον χάρτη. Όπως γίνεται αντιληπτό αυτή είναι η διαδικασία με την οποία η εφαρμογή εμφανίζει στον χρήστη την διεύθυνση στην οποία βρίσκεται και όχι το γεωγραφικό πλάτος και μήκος. Ο λόγος που χρειάζομαι το reverse geocoding είναι διότι αυτό που λαμβάνει η εφαρμογή μέσω του location listener είναι το γεωγραφικό πλάτος και μήκος το οποίο στέλνει το GPS ή οι κεραίες κινητής τηλεφωνίας. Αυτό σημαίνει πως για τον χρήστη θα ήταν άχρηστο να ξέρει πως βρίσκεται για παράδειγμα στην τοποθεσία με γεωγραφικό πλάτος και μήκος 37.975150°, 23.735691°. Αυτό που θα είχε νόημα είναι να ξέρει το όνομα της διεύθυνσης στην οποία αντιστοιχούν αυτές οι συντεταγμένες, το οποίο είναι η «Λεωφόρος Βασιλίσσης Αμαλίας».

Geocoding δεν υπάρχει μόνο στο Android αλλά και σε άλλες διαδικτυακές ή desktop εφαρμογές, άρα δεν υπάρχει και ένας στάνταρ τρόπος με τον οποίο αυτό υλοποιείτε. Για το Android συγκεκριμένα, υπάρχει η κλάση Geocoder η οποία είναι υπεύθυνη για να κάνει τις μετατροπές που αναφέρθηκαν παραπάνω. Το πρόβλημα με αυτή την κλάση είναι πως δεν αποτελεί κομμάτι του AOSP (Android Open Source Project) αλλά συμπεριλαμβάνεται στο Google API add-on. Αυτό πρακτικά σημαίνει πως αν δεν έχουμε μια συσκευή με εγκατεστημένες εφαρμογές της Google έτσι ώστε να είναι εγκατεστημένο και το Google API, δεν θα έχουμε πρόσβαση στην κλάση Geocoder μιας και δεν θα βρίσκεται πουθενά στη συσκευή μας. Γι αυτό τον λόγο όπως θα δούμε και στον demo κώδικα που θα επακολουθήσει για το πώς χρησιμοποιούμε τον Geocoder, πρώτα χρησιμοποιούμε την συνάρτηση Geocoder.isPresent() για να δούμε αν η κλάση υπάρχει στην συσκευή μας. Αν υπάρχει μπορούμε να τη χρησιμοποιήσουμε κάνοντας import την κατάλληλη βιβλιοθήκη. Αν δεν υπάρχει, θα πρέπει να φτιάξουμε τον δικό μας Geocoder χρησιμοποιώντας απευθείας το Goggle Maps Geocoding API.

Η κλάση Geocoder ουσιαστικά αυτοματοποιεί μια διαδικασία που θα χρειαζόταν να κάνουμε μόνοι μας με περισσότερες γραμμές κώδικα και περισσότερο κόπο. Έτσι ανάλογα με την συνάρτηση της κλάσης που χρησιμοποιούμε, μπορούμε εύκολα όπως θα δούμε παρακάτω να πάρουμε μια λεπτομερή περιγραφή της διεύθυνσης στην οποία βρισκόμαστε στέλνοντας μόνο σαν παράμετρο τις συντεταγμένες που πήραμε από τον location listener.

Εναλλακτικά αυτό που έπρεπε να κάνουμε είναι να στείλουμε ένα http request στο Google Maps API με query parameters το γεωγραφικό πλάτος και μήκος με το ακόλουθο URL :

http://maps.googleapis.com/maps/api/geocode/json?latlng=37.975150,23.735691.

Το response στο παρακάτω call θα ήταν ένα JSON που θα είχε πάνω του όλη την απαραίτητη πληροφορία για την διεύθυνση που αναζητούμε. Έτσι θα έπρεπε να γράψουμε και κώδικα που θα χειριζόταν την ανάγνωση του JSON. Όλο αυτό εισάγει πολυπλοκότητα που καλό θα ήταν να αποφύγουμε χρησιμοποιώντας την Geocoder class.

Έτσι λοιπόν καταλήγουμε στη χρήση του Geocoder και ο κώδικας που τελικά απαιτείτε για την χρήση του είναι ο ακόλουθος, έτσι όπως έχει χρησιμοποιηθεί και στην εφαρμογή μου :

Java

```
public String getAddressName() {
    if (!Geocoder.isPresent()) {
        return "";
    }
    String addressText = null;
    Geocoder geo = new Geocoder(this, Locale.getDefault());
    try {
        List<Address> addresses = geo.getFromLocation(this.latitude,
this.longitude, 1);
        if (addresses != null && addresses.size() > 0) {
            Address address = addresses.get(0);
            addressText = String.format("%s,%s,%s",
address.getMaxAddressLineIndex() > 0 ? address.getAddressLine(0) :
"",address.getLocality(),address.getCountryName());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return addressText;
}
```

Αυτός είναι λοιπόν ο τρόπος με τον οποίο σε λίγες γραμμές κώδικα καταφέρνουμε να πάρουμε μια ανθρωπίνως κατανοητή διεύθυνση ξέροντας τις γεωγραφικές τις συντεταγμένες.

3.3 Android Shared Preferences

Μία ακόμα απόφαση που έπρεπε να παρθεί και αφορούσε το στάδιο ανάπτυξης της πρώτης λειτουργικότητας στην οποία ο χρήστης λαμβάνει την τοποθεσία του και την αποθηκεύει, ήταν το πώς θα αποθηκεύσουμε τις συντεταγμένες (άρα κατ επέκταση την διεύθυνση) τις οποίες έλαβε ο χρήστης.

Γενικά στο Android υπάρχουν τέσσερεις κυρίως τρόποι να αποθηκεύσουμε δεδομένα και αυτοί είναι οι παρακάτω **[18]** :

- Αποθήκευση δεδομένων σε αρχεία στον εσωτερικό ή εξωτερικό χώρο αποθήκευσης της συσκευής.
- Αποθήκευση σε SQLite βάση δεδομένων. Πρόκειται για την βάση δεδομένων που χρησιμοποιεί το Android. Μοιάζει πολύ με MySQL ως προς την σύνταξη αλλά είναι περιορισμένων δυνατοτήτων και πολύ πιο ελαφριά μιας και έχει σχεδιαστεί να παίζει σε κινητά.

- Αποθήκευση σε Cloud μέσω σύνδεσης στο internet.
- Αποθήκευση δεδομένων σε συλλογή ζευγών ονόματος-τιμής (key-values), μέσω του SharedPreferences API.

Οι παραπάνω τρόποι αποθήκευσης έχουν τα θετικά και αρνητικά τους μέρη τα οποία δεν θα αναλυθούν στην παρούσα εργασία και χρησιμοποιούνται το καθένα για διαφορετικό σκοπό. Ωστόσο από τα προαναφερθέντα θα αναλύσουμε περισσότερο τον τρόπο που εγώ επέλεξα να σώσω τις συντεταγμένες του χρήστη και να τις καταστήσω γνωστές σε μελλοντική χρήση, ο οποίος είναι οι SharedPreferences.

Τα SharedPreferences είναι ιδανικά για αποθήκευση μικρών συλλογών πρωτόγονων δεδομένων (primitive data types : boolean, float, int, long, string) που είναι οργανωμένα σε ζευγάρια τα οποία έχουν ένα χαρακτηριστικό όνομα (key) και μια τιμή (value). Αυτά τα δεδομένα αποθηκεύονται σε ιδικά αρχεία τα οποία μπορεί να είναι κοινόχρηστα ή ιδιωτικά και για τα οποία μας παρέχονται απλές μέθοδοι με τις οποίες μπορούμε να γράψουμε και να διαβάσουμε τα ζεύγη τιμών. Το κυριότερο είναι πως τα δεδομένα παραμένουν διαθέσιμα ακόμα και όταν η εφαρμογή κλείσει και σε συνδυασμό με την απλότητα που παρουσιάζουν επέλεξα να τα χρησιμοποιήσω.

Στα πλαίσια της εφαρμογής που εξετάζουμε το ζητούμενο ήταν να αποθηκεύσουμε και να διαβάσουμε σε μεταγενέστερο χρόνο τα δύο ζεύγη τιμών που καθορίζουν μια γεωγραφική θέση. Το γεωγραφικό πλάτος (latitude) και το γεωγραφικό μήκος (longitude). Έτσι για να υλοποιήσουμε το γράψιμο και το διάβασμα αυτών των δεδομένων χρησιμοποιήσαμε τον παρακάτω κώδικα :

Δημιουργία Shared Preferences File:

```
SharedPreferences sharedPref =
context.getSharedPreferences("net.ddns.drimou.bachelorthesis", Context.MODE_PRIVATE);
```

Όταν δημιουργούμε shared preferences files θα πρέπει να χρησιμοποιούμε ονόματα που χαρακτηρίζουν μοναδικά την εφαρμογή μας όπως είναι για παράδειγμα το όνομα του package της εφαρμογής.

Γράψιμο στο Shared Preferences File:

```
SharedPreferences.Editor editor = sharedPref.edit();
editor.clear();
editor.commit();
editor.putLong("net.ddns.drimou.bachelorthesis.latitude", (long) gps.getLatitude());
editor.putLong("net.ddns.drimou.bachelorthesis.longitude", (long) gps.getLongitude());
editor.commit();
```

Το ίδιο μοναδικά πρέπει να ονοματίζουμε και τα ζευγάρια τιμών που αποθηκεύουμε . Άρα καλό θα ήταν να βάζουμε το όνομα του package της εφαρμογής συν το όνομα της μεταβλητής που αποθηκεύσουμε. Επίσης για να βάλουμε τιμές σε ένα shared preferences file πρέπει να δηλώσουμε και να χρησιμοποιήσουμε τον shared preferences editor.

Διάβασμα από το Shared Preferences File:

```
if (sharedPref.contains("net.ddns.drimou.bachelorthesis.latitude") ||
sharedPref.contains("net.ddns.drimou.bachelorthesis.longitude")) {
        gps.setLatitude((double) sharedPref.getLong("net.ddns.drimou.bachelorthesis.latitude", 0));
        gps.setLongitude((double) sharedPref.getLong("net.ddns.drimou.bachelorthesis.longitude", 0));
 }
```

Αυτός είναι ο κώδικας με τον οποίο μπορώ να αποθηκεύσω και να προσπελάσω τα SharedPreferences files. Όπως γίνεται εύκολα αντιληπτό είναι εύκολος και γρήγορος τρόπος και ιδανικός για περιπτώσεις όπου πρέπει να αποθηκευτούν δεδομένα σε μορφή key-value. Η δημιουργία και ανάγνωσή τους γίνεται στο OnCreate() method της κύριας κλάσης της εφαρμογής και το γράψιμο στον handler του κουμπιού «Capture Position».

3.4 Glassfish

Όπως είναι γνωστό η εφαρμογή που εξετάζουμε στη παρούσα εργασία υλοποιεί μια λειτουργικότητα που απαιτεί την ανταλλαγή συντεταγμένων μεταξύ δυο χρηστών. Συνεπώς γεννάται το ερώτημα πώς γίνεται αυτή η ανταλλαγή δεδομένων μεταξύ των χρηστών και ποιος την διαχειρίζεται;

Η απάντηση στο ερώτημα αυτό είναι ο open-source application server Glassfish. Με τον όρο application server εννοούμε κάθε server μέσω του οποίου έχουμε πρόσβαση σε λογισμικό που τρέχει και εξυπηρετεί τις ανάγκες μιας εφαρμογής. Είναι οι servers που εξυπηρετούν το business μιας εφαρμογής και μέσω του API που δημιουργούν δίνουν πρόσβαση στους προγραμματιστές λογισμικού να το χρησιμοποιήσουν και να το εντάξουν στις εφαρμογές που αναπτύσσουν.

Ο Glassfish συγκεκριμένα αποτελεί ένα μοντέλο αναφοράς (model implementation) για την JavaEE. Με απλά λόγια θα μπορούσαμε να πούμε πως με τον Glassfish μπορούμε να δημιουργήσουμε enterprise εφαρμογές που τρέχουν σε Java και ως τέτοιες υποστηρίζουν κάθε τεχνολογία που τρέχει πάνω στη Java platform όπως είναι οι Enterprise JavaBeans, JPA, JavaServer Faces, Java servlets κτλ. . Για όλες αυτές τις τεχνολογίες αλλά και ακόμα περισσότερο για τον Glassfish θα μπορούσαμε να αφιερώσουμε πολλά κεφάλαια ωστόσο κάτι τέτοιο ξεφεύγει από τα στενά όρια της παρούσας πτυχιακής εργασίας και γι αυτό τον λόγο θα εστιάσουμε κυρίως στον τρόπο με τον οποίο τον χρησιμοποιούμε ως ένα βασικό επίπεδο.

Από τα λίγα που προαναφέρθηκαν σχετικά με τι ακριβώς είναι ο Glassfish συμπεραίνουμε σιγά σιγά και τον λόγο για τον οποίο μας ήταν χρήσιμος και τον χρησιμοποιήσαμε. Όταν λοιπόν στο παράδειγμα της εφαρμογής μας, ένας από τους δύο χρήστες πρέπει να στείλει στον άλλο χρήστη της γεωγραφική του θέση το πρώτο πράγμα που κάνει είναι να λάβει τις συντεταγμένες του από τον location listener. Στη συνέχεια αφού κάνει τους απαραιτήτους μετασχηματισμούς και φτιάξει ένα JSON που θα μεταφέρει την σχετική πληροφορία, ανοίγει ένα κανάλι επικοινωνίας με τον άλλο χρήστη χρησιμοποιώντας Web Sockets τα οποία θα δούμε στο επόμενο κεφάλαιο. Όλα αυτά μέχρι εδώ είναι ενέργειες που γίνονται τοπικά στη συσκευή του χρήστη (client) . Όμως πως από εκείνο το σημείο και έπειτα καταλήγει η πληροφορία στον παραλήπτη. Η αναμενόμενη πλέον απάντηση είναι μέσω ενός application server, συγκεκριμένα του Glassfish ο οποίος διαχειρίζεται όλο αυτό το "αλισβερίσι" πληροφοριών.

Έτσι στο ενδιάμεσο της πορείας που έχει να διανύσει η πληροφορία από την μια συσκευή στην άλλη, υπάρχει ένα μηχάνημα (server) που τρέχει τον Glassfish στον οποίο είναι deployed η εφαρμογή που διαχειρίζεται την επικοινωνία των χρηστών. Η εφαρμογή αυτή είναι γραμμένη σε Java και υλοποιεί την τεχνολογία Web Sockets μέσω των οποίων επέλεξα να μεταφέρω τα δεδομένα που ανταλλάσουν οι χρήστες. Ο ρόλος που έχει η συγκεκριμένη εφαρμογή είναι να δέχεται αιτήματα από μια συσκευή (client) τα οποία τελικώς τα δρομολογεί σε μια άλλη συσκευή που έχει ενεργή σύνδεση με το Web Socket. Αν δεν μεσολαβούσε αυτό το application τότε κάθε αίτημα που έφευγε από μια συσκευή δεν θα έβρισκε ποτέ τον παραλήπτη του, διότι πολύ απλά δεν θα ήξερε ποιος είναι αυτός ή αν

είναι συνδεδεμένος. Επίσης αν το επιθυμούσαμε θα μπορούσαμε να εντάξουμε extra logic στην εφαρμογή μας για να προσθέσουμε περισσότερη ασφάλεια ή να επέμβουμε στα αιτήματα που μας έρχονται για οποιοδήποτε σκοπό θα θέλαμε να εξυπηρετήσουμε.

Αφού λοιπόν έχουμε πλέον μια σφαιρική άποψη για το τι είναι ο Glassfish και ποίος ο ρόλος του στην παρούσα εφαρμογή, θα δούμε το πώς τον εγκατέστησα και πως τον σέταρα στον server μου. Να σημειώσω πως ο Glassfish που χρησιμοποίησα τρέχει σε ένα virtual machine το οποίο διατηρώ και στο οποίο χρησιμοποιώ για λειτουργικό σύστημα το Linux Centos 6.7 (Final).

Το πρώτο πράγμα που πρέπει να ελέγξουμε στο σύστημά μας είναι αν υπάρχει εγκατεστημένο το JDK. Αν δεν το έχουμε θα πρέπει να το εγκαταστήσουμε και να βεβαιωθούμε πως το path της μεταβλητής συστήματος JAVA_HOME δείχνει στο JDK που εγκαταστήσαμε.

Στη συνέχεια μπορούμε πλέον να εγκαταστήσουμε τον Glassfish Server **[14]** στο μηχάνημά μας. Τα βήματα της εγκατάστασης είναι τα παρακάτω :

- Πηγαίνουμε στο directory που θα τον εγκαταστήσουμε.
- Κατεβάζουμε την έκδοση του Glassfish που μας ενδιαφέρει σε zip μορφή ως εξής :
  # wget http://download.oracle.com/glassfish/4.1/release/glassfish-4.1.zip
- Κάνουμε unzip το αρχείο που κατεβάσαμε ως εξής : # unzip glassfish-4.1.zip Συνήθως το όνομα του φακέλου στον οποίο εξάγεται το zip είναι 'glassfish4' .
- Μπαίνουμε στο directory glassfish4/bin και αλλάζουμε το username και το password για το login στον Glassfish με τη χρήση του εργαλείου asadmin που βρίσκεται μέσα στον φάκελο bin εκεί όπου είναι εγκατεστημένος ο Glassfish. Για να αλλάξουμε λοιπόν τα credentials πληκτρολογούμε την παρακάτω εντολή: # ./asadmin change-admin-password και ακολουθούμε τις οδηγίες που θα εμφανιστούν στη κονσόλα.
- Τέλος για να σηκωθεί ο Glassfish πληκτρολογούμε το παρακάτω: # asadmin start-domain domain1 .

Αυτά ήταν τα βασικά βήματα για να εγκαταστήσουμε και να τρέξουμε τον Glassfish server. Για περισσότερη ασφάλεια θα μπορούσαμε να δημιουργήσουμε χρήστες και να παραχωρήσουμε συγκεκριμένα δικαιώματα έτσι ώστε να μην μπορεί ο οποιοσδήποτε να έχει πρόσβαση σε κάθε λεπτομέρεια του Glassfish.

Ωστόσο κάτι τέτοιο δεν θα δούμε πως γίνεται στη παρούσα εργασία καθώς είναι εκτός εμβέλειας του θέματος, μιας και στη περίπτωσή μας μόνος χρήστης του Glassfish είμαι εγώ και εγώ χρησιμοποιώ την by default root access.

Άλλο ένα βασικό εργαλείο του είναι το Glassfish admin console μέσα από την οποία μπορούμε να έχουμε πλήρη πρόσβαση σε κάθε configuration του server όπως settings για θέματα ασφαλείας, ορισμός dependencies σε διάφορα projects, δήλωση βάσεων δεδομένων με τις οποίες συνεργάζεται κάποιο project και πολλά ακόμα. Το σημαντικότερο είναι όμως πως μπορούμε να κάνουμε εύκολα deploy (με απλά λόγια να ανεβάσουμε στον server και να τρέξει) το project που φτιάξαμε τοπικά στον υπολογιστή μας και θέλουμε να κοινοποιήσουμε. Την διαδικασία του deploy χρειάστηκε αν την κάνω και εγώ για να ανεβάσω στον server το back end που υλοποιεί τα Web Sockets. Η Glassfish admin console είναι διαθέσιμη μέσω browser αν πάμε στη διεύθυνση http://your_domain:4848. Αν λοιπόν χτυπήσουμε την παραπάνω διεύθυνση θα εμφανιστεί μπροστά μας η login σελίδα του Glassfish για να μπούμε στην διαχειριστική κονσόλα. Σαν credentials βάζουμε αυτά που δηλώσαμε στα παραπάνω βήματα.

Σε γενικές γραμμές αυτή είναι μια υπεραπλουστευμένη παρουσίαση του τι είναι τελικά ο Glassfish και πως τον χρησιμοποιούμε με έναν πολύ βασικό τρόπο. Γενικά θα μπορούσαν να γραφτούν και να ειπωθούν πολλά ακόμα για αυτόν, αλλά αυτό είναι κάτι που ξεφεύγει από τα όρια αυτής της εργασίας.

Τέλος για να έχουμε μια ολοκληρωμένη άποψη για το πώς αλλιώς θα μπορούσαμε να διαχειριστούμε και να υλοποιήσουμε όλη αυτή την υλοποίηση με τα Web Sockets, ο glassfish δεν είναι ο μόνος application server που υπάρχει. Η αλήθεια είναι πώς υπάρχουν πολλοί ακόμα και ειδικά σε αυτούς που υποστηρίζουν Java υπάρχει ένας μεγάλος κατάλογος στον οποίο δεσπόζουν και άλλα ονόματα application servers όπως είναι αυτό του δημοφιλή Tomcat.

Μία άλλη προσέγγιση θα ήταν να χρησιμοποιήσω τον όλο και δημοφιλέστερο node.js, κάτι που θα σήμαινε πως θα υλοποιούσα την back end υποστήριξη των Web Sockets με JavaScript αντί για Java, κάτι που θα ήταν εξίσου ενδιαφέρον. Ωστόσο για λόγους προσωπικής εξοικείωσης επέλεξα να χρησιμοποιήσω Glassfish και να γράψω το back end σε Java.

3.5 Web Sockets

Το WebSocket **[20,15]** (WS) είναι ένα πρωτόκολλο που παρέχει full-duplex communication channels (και προς τις δυο κατευθύνσεις κανάλια επικοινωνίας) πάνω σε μία μόνο σύνδεση TCP. Η μόνη σχέση που έχει με το HTTP είναι πως η handshake διαδικασία του WebSocket μεταφράζεται από τους HTTP servers σαν ένα upgrade request. Το WS μας παρέχει την δυνατότητα να φτιάξουμε ποιο διαδραστικές διαδικτυακές (και όχι μόνο) εφαρμογές μιας και έρχεται να καλύψει κάποια μειονεκτήματα που έχει το HTTP το οποίο ανακαλύφθηκε για το World Wide Web και γι αυτό τον λόγο παρουσιάζει κάποιους περιορισμούς.

Ο τρόπος που το HTTP λειτουργεί δεν είναι βέλτιστος για κάθε είδους εφαρμογή ή ανάγκη. Όπως είναι γνωστό κάθε φορά που κάνουμε ένα HTTP request (πχ. για να κατεβάσουμε ένα τραγούδι) ανοίγει ένα port/socket και μεταφέρεται η πληροφορία. Μόλις η πληροφορία μεταφερθεί αυτό το socket κλείνει αυτόματα. Το να ανοίγουμε και να κλείνουμε sockets δημιουργεί καθυστερήσεις ή αλλιώς το λεγόμενο overhead, πράγμα που είναι πολύ αρνητικό για εφαρμογές που απαιτούν άμεση ενημέρωση από κάποιον server. Ένα ακόμα μειονέκτημα που έχει το HTTP είναι πως δεν μπορεί ένας server να σπρώξει πληροφορίες σε έναν client αν αυτός πρώτα δεν του το ζητήσει. Έτσι για εφαρμογές που χρειάζονται real-time ενημέρωση θα πρέπει ο client να κάνει κάθε λίγο ένα request για να δει αν υπάρχουν διαθέσιμες ενημερώσεις. Αυτό εκτός του ότι είναι αργό αυξάνει και τον διαδικτυακό φόρτο καθώς γεμίζει το διαδίκτυο με συνεχόμενα requests που τελικά πολλές φορές δεν έχουν κανένα αποτέλεσμα.

Συνεπώς για διαδραστικές εφαρμογές ή εφαρμογές που απαιτούν real-time ενημέρωση από τον server απαιτείτε διαφορετικό πρωτόκολλο επικοινωνίας και ένα από αυτά είναι το WS. Όπως προαναφέραμε ο τρόπος που το WS λειτουργεί είναι να κάνει ένα αρχικό call (handshake) για να ανοίξει ένα κανάλι επικοινωνίας και έπειτα το κρατάει συνέχεια ανοικτό, χωρίς να χρειάζεται να ξανασυνδεθεί. Επίσης το κανάλι επικοινωνίας που ανοίχτηκε είναι bidirectional (και προς τις δυο κατευθύνσεις) και συνεπώς αυτό σημαίνει πως η πληροφορία μπορεί να μεταδοθεί είτε real-time από κάποιον server είτε μετά από απαίτηση.

Ένα παράδειγμα για το πώς λειτουργούν αυτά τα δυο πρωτόκολλα είναι το να υποθέσουμε πως κάπου υπάρχει ένας στρατός. Ο στρατός αυτός έχει έναν στρατηγό (server) και πολλούς στρατιώτες (clients). Αν χρησιμοποιούσαμε

HTTP θα έπρεπε κάθε στρατιώτης ξεχωριστά να ρωτήσει τον στρατηγό αν υπάρχουν νέες εντολές. Καταλαβαίνουμε το τι φόρτο θα είχε ο στρατηγός να απαντάει συνεχόμενα και για πολλή ώρα στο κάθε στρατιώτη ξεχωριστά σχετικά με το αν υπάρχουν νέες εντολές. Επίσης πέρα του φόρτου που θα είχε ο στρατηγός θα υπήρχε και μια καθυστέρηση μέχρι όλοι οι στρατιώτες να μάθουν την απάντησή του, καθώς από αυτόν που ρώτησε πρώτος μέχρι αυτόν που ρώτησε τελευταίος υπάρχει διαφορά χρόνου. Με το WS τα παραπάνω προβλήματα επιλύονται πολύ εύκολα, διότι πλέον ο στρατηγός είναι σαν να έχει εφοδιάσει κάθε στρατιώτη με έναν ασύρματο και να τους μεταβιβάζει από εκεί τις εντολές του, που φυσικά θα φτάνουν σε όλους ταυτόχρονα. Επίσης αν θέλει μπορεί να αλλάξει μπάντα συχνοτήτων (κανάλι επικοινωνίας) και να μιλήσει με συγκεκριμένα άτομα χωρίς να τον ακούνε όλοι. Άρα βάση του παραπάνω παραδείγματος καταλαβαίνουμε το πόσο γρηγορότερα και αποδοτικότερα γίνονται όλα στο WS.

Έχοντας λοιπόν αναλύσει τα παραπάνω η ερώτηση του γιατί χρησιμοποιήσαμε WS στην εφαρμογή μας είναι μάλλον περιττή. Έχουμε μια εφαρμογή στην οποία απαιτείται σε πραγματικό χρόνο να ανταλλάσσονται συντεταγμένες μεταξύ δυο χρηστών (με τα WS μπορούμε να έχουμε και παραπάνω χρήστες μελλοντικά). Καταλαβαίνουμε λοιπόν πως θέλουμε κάτι γρήγορο και αποδοτικό. Στην εφαρμογή μας έχουμε ένα application server (τον glassfish όπως είδαμε) που τρέχει την εφαρμογή που διαχειρίζεται τα request και responses που έρχονται μέσω του WS στον server. Επίσης από την άλλη έχουμε σε κάθε client μια αντίστοιχη κλάση που είναι υπεύθυνη για να ανοίγει κανάλι επικοινωνίας με τον server και να διαχειρίζεται την αποστολή και λήψη μηνυμάτων.

Η JavaEE7 εφαρμογή που τρέχει στον server απαιτεί τέσσερεις κλάσεις για να λειτουργήσει (ολόκληρες οι κλάσεις είναι διαθέσιμες στον κώδικα που έχω παραδώσει μαζί με την παρούσα παρουσίαση) . Η πρώτη κλάση είναι αυτή που καθορίζει το object που θα ανταλλάσσεται κατά την επικοινωνία μεταξύ των χρηστών. Αυτή την κλάση την ονόμασα LocationMessage.java και περιέχει τον constructor, τους getters και τους setters για το object που θα χρησιμοποιήσουμε στην επικοινωνία. Η δεύτερη κλάση που ονόμασα LocationMessageEncoder.java είναι ουσιαστικά ένας encoder που έχει τον ρόλο του να δημιουργεί ένα JSON αφού τραβήξει τιμές από το object στο οποίο αναφέρθηκα προηγούμενος. Στη

συνέχεια αυτό το JSON που παρήγαγε ο encoder θα μεταδοθεί μέσω του WS στους χρήστες. Η επόμενη κλάση όπως είναι αναμενόμενο κάνει την ανάποδη λειτουργία (decoding) από αυτή που αναφέρθηκε προηγούμενος και την ονόμασα LocationMessageEncoder.java . Ο ρόλος της είναι να παρσάρει το JSON που έρχεται μέσω της επικοινωνίας σε object. Τέλος υπάρχει και η LocationWebSocket.java κλάση η οποία είναι υπεύθυνη να διαχειρίζεται την κίνηση από και προς τον server. Εδώ δηλώνουμε το endpoint που θα πρέπει να "χτυπάνε" τα WS requests (στη περίπτωση μας ws:domain_name*/ws/location*), όπως επίσης ορίζουμε την συμπεριφορά του server κάθε φορά που θα λαμβάνει ή θα στέλνει κάποιο μήνυμα ή όταν ένα νέο κανάλι επικοινωνίας ανοίγει ή ένα είδη ανοικτό κλείνει. Είναι δηλαδή το κέντρο ελέγχου της εφαρμογής που διαχειρίζεται την μεταφορά πληροφοριών μεταξύ των χρηστών. Στην τελευταία κλάση έχει προστεθεί και επιπλέον logic έτσι ώστε να διασφαλίσουμε αυτό που αρχικά είχαμε αναφέρει, την αποκλειστικότητα δηλαδή την πληροφορίας που μεταφέρεται έτσι ώστε μόνο οι χρήστες που πρέπει να λαμβάνουν τα μηνύματα που προορίζονται για αυτούς. Αυτό το πέτυχα καταγράφοντας σε ένα Java Set collection κάθε νέο χρήστη που άνοιγε ένα κανάλι επικοινωνίας με τον server. Σε αυτό το Java Set έσωζα για κάθε χρήστη το ποιος είναι και με ποιον θέλει να επικοινωνήσει. Έτσι τραβώντας την κατάλληλη πληροφορία έστελνα τα μηνύματα μόνο στους χρήστες των οποίον τα στοιχεία ήταν σε αντιστοιχία.

Αντίστοιχα και στον client υπάρχει κώδικας για την αλληλεπίδραση με τον WS server. Επειδή το Android δεν περιέχει από μόνο του κάποια κλάση για τον χειρισμό WS, έπρεπε να συμπεριλάβω κάποιον ήδη έτοιμο WS client για Android. Η βιβλιοθήκη που χρησιμοποίησα για να φτιάξω τον δικό μου WS client είναι η «Java WebSockets». Για να την ενσωματώσω στο project μου, επειδή το development γίνεται στο Android Studio το οποίο χρησιμοποιεί το Gradle, αρκούσε να δηλώσω στο build.gradle αρχείο του project μου στα dependencies τη βιβλιοθήκη που θέλω να χρησιμοποιήσω. Το Gradle αυτόματα θα την κατεβάσει και θα την ενσωματώσει στο project. Στη συνέχεια είμαι ελεύθερος να χρησιμοποιήσω τις μεθόδους που μου παρέχει και να προσαρμόσω τον WS client στα δικά μου δεδομένα. Για το πως χρησιμοποιείτε η βιβλιοθήκη αυτή ανέτρεξα στα manuals στην σχετική σελίδα στο GitHub από όπου οι δημιουργοί της την έκαναν διαθέσιμη στο κοινό.

Αυτός είναι ο τρόπος λοιπόν που χρησιμοποίησα το WS πρωτόκολλο στην εφαρμογή μου. Αναλυτικά ο κώδικας για κάθε υλοποίηση υπάρχει στον συνοδευτικό κώδικα της εφαρμογής.

3.6 JSON

Το JSON **[16]** αποτελεί ακρωνύμιο των λέξεων JavaScript Object Notation και αποτελεί έναν οργανωμένο και εύκολα προσβάσιμο τρόπο για να αποθηκεύουμε πληροφορίες. Είναι self describing (διαβάζεται από ανθρώπους), ιεραρχικό και μπορεί να διαβαστεί από πολλές γλώσσες προγραμματισμού. Επίσης ένα ακόμα χαρακτηριστικό του που μας φάνηκε πολύ χρήσιμο είναι ότι μπορεί να ενσωματωθεί στο body ενός http response ή request και άρα μπορεί να μεταφέρει πληροφορία μέσω διαφόρων services.

Η σύνταξή του JSON προέρχεται από τον τρόπο δήλωσης ενός object στη JavaScript και ακολουθεί μερικούς απλούς κανόνες:

- Τα δεδομένα είναι οργανωμένα σε ζευγάρια που από την μία έχουν ένα όνομα και από την άλλη μια τιμή (name/value pairs).
- Τα δεδομένα διαχωρίζονται με κόμμα.
- Με άγκιστρα δηλώνουμε ένα object.
- Με αγκύλες δηλώνουμε έναν πίνακα.
- Μπορούν να περιέχουν εμφωλευμένα objects.

Ο λόγος για τον οποίο χρησιμοποίησα JSON στην εφαρμογή μου ήταν για να μπορέσω να περάσω πληροφορίες μέσω http services από τον server μου στην εφαρμογή και αντίστροφα. Όπως είδαμε και σε προηγούμενο κεφάλαιο για να μπορέσουμε να υλοποιήσουμε την λειτουργικότητα με την ανταλλαγή τοποθεσιών μεταξύ δυο χρηστών έπρεπε να στήσουμε έναν application server που θα διαχειρίζεται αυτή την σύζευξη. Για να διαχειριστεί αυτή την λειτουργιά το πρόγραμμα που τρέχει στον server έπρεπε να έχει κάποιες απαραίτητες πληροφορίες από την εφαρμογή μας και θα έπρεπε αντίστοιχα να μεταφέρει και πληροφορίες στην εφαρμογή. Έτσι η ανταλλαγή πληροφοριών επέλεξα να γίνει με την τεχνολογία JSON.

Η πληροφορία που πρέπει να ανταλλάσσεται αφορά το ποιοί χρήστες θα συμμετέχουν, τι ρόλο θα έχει ο κάθε χρήστης κατά την διάρκεια της σύζευξης και

ποια είναι η γεωγραφική θέση του κάθε χρήστη. Έτσι κατέληξα στο μοντέλο που φαίνεται παρακάτω :

```
{
  "action": "lead",
  "un": "user1",
  "authorizedUn": " user2",
  "latitude": "37.975150",
  "longitude": "23.735691"
}
```

Το παραπάνω JSON ανταλλάσουν οι δύο χρήστες που είναι συνδεδεμένοι μεταξύ τους καθ όλη την διάρκεια της επαφής τους. Αυτό που αλλάζει είναι ότι ο ένας έχει action = lead και ο άλλος έχει action = follow. Πρόκειται ουσιαστικά για τον ρόλο του κάθε χρήστη. Έτσι ο πρώτος δηλώνει ότι είναι ο οδηγός της πορείας και ο δεύτερος ότι είναι ο ακόλουθος. Αυτό έχει σημασία για να μπορούμε να εμφανίσουμε σωστά τις θέσεις και την πορεία των δύο χρηστών στο χάρτη διότι ξέρουμε για παράδειγμα πως η κίνηση είναι από τον user2 προς τον user1 και όχι το ανάποδο.

Ένα ακόμα πράγμα που αλλάζει είναι πως οι θέσεις των un και authorizedUn είναι ανάποδα σε κάθε χρήστη. Έτσι ο user1 θέτει μονίμως στο JSON un = user1 και authorizedUn = user2 και αντιστρόφως ο user2 θέτει μόνιμος στο δικό του JSON un = user2 και authorizedUn = user1. Αυτός είναι ο τρόπος που επέλεξα να αποτυπώσω το από ποιον και προς ποιον φεύγει η πληροφορία. Μέσα σε ένα web socket όπως είδαμε μπορεί να έχει πρόσβαση ο οποιοσδήποτε σε κάθε μήνυμα που διαδίδεται. Εμείς όμως δεν θέλουμε να μπορεί ο καθένας να "ακούει" την ανταλλαγή των πληροφοριών. Έτσι με αυτόν τον τρόπο γίνεται μια διαπίστευση και ορίζεται ότι για παράδειγμα ο user1 μπορεί να λαμβάνει και να στέλνει μηνύματα μόνο από και προς τον user2 γιατί αυτόν έχει δηλώσει ως authorizedUn κατά την διαδικασία της αρχικοποίησης της σύνδεσης. Το ίδιο με αντίστροφο τρόπο έχει κάνει και ο user2 και έτσι με τον κατάλληλο κώδικα που έχει γραφτεί για την υλοποίηση του web socket της εφαρμογής δεν υπάρχει περίπτωση κάποιος χρήστης να λάβει ή να στείλει μήνυμα σε κάποιον μη εξουσιοδοτημένο χρήστη.

Τέλος τα latitude και longitude αντιπροσωπεύουν τις γεωγραφικές συντεταγμένες στις οποίες αντιστοιχεί η θέση στην οποία βρίσκεται ο κάθε χρήστης και αποστέλλεται με σκοπό την σχετική ενημέρωση του συζευγμένου με αυτόν χρήστη.

Το επόμενο θέμα που αξίζει να αναφέρουμε είναι το πώς μπορούμε να δημιουργήσουμε ή να διαβάσουμε ένα JSON στο Android και πώς μπορούμε να χειριστούμε επίσης αυτή την πληροφορία στον server. Στο παρακάτω demo κώδικα φαίνετε το πώς μπορεί να συμβεί αυτό :

Διάβασμα JSON :

```
JSONObject locationResponce  = new JSONObject(s);
final double latitube = Double.parseDouble(locationResponce.getString("latitube"));
final double longitube = Double.parseDouble(locationResponce.getString("longitube"));
final String action = locationResponce.getString("action");
final String un = locationResponce.getString("un");
final String authorizedUn = locationResponce.getString("authorizedUn");
```

Δημιουργία JSON :

```
JSONObject sendObjectOnOpen = new JSONObject();
sendObjectOnOpen.put("action", "follow");
sendObjectOnOpen.put("un", "user1");
sendObjectOnOpen.put("authorizedUn", " user2");
sendObjectOnOpen.put("latitube", "37.975150");
sendObjectOnOpen.put("longitube", "23.735691");
```

Διάβασμα στον server :

```
try (JsonReader jsonReader = Json.createReader(reader)) {
        JsonObject jsonMessage = jsonReader.readObject();
        LocationMessage message = new LocationMessage();
        message.setAction(jsonMessage.getString("action"));
        message.setUn(jsonMessage.getString("un"));
        message.setAuthorizedUn(jsonMessage.getString("authorizedUn"));
        message.setLatitube(jsonMessage.getString("latitube"));
        message.setLongitube(jsonMessage.getString("longitube"));
        return message;
}catch(Exception e){
        return null;
}
```

Δημιουργία στον server :

```
JsonObject jsonMessage = Json.createObjectBuilder()
            .add("action", message.getAction())
            .add("un", message.getUn())
            .add("authorizedUn", message.getAuthorizedUn())
            .add("latitube", message.getLatitube())
            .add("longitube", message.getLongitube())
            .build();
      try (JsonWriter jsonWriter = Json.createWriter(writer)) {
         jsonWriter.write(jsonMessage);
      }
```

Αυτή ήταν μια όσο το δυνατόν περιεκτικότερη αναφορά στην τεχνολογία JSON. Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε XML που παρουσιάζει κάποιες ομοιότητες με το JSON αλλά είναι μεγαλύτερο από άποψη όγκου και δυσκολότερο στο να γραφτεί ή να διαβαστεί. Έτσι για τον λόγο αυτό όπως και για τον λόγο του ότι έχω μεγαλύτερη εξοικείωση με το JSON τελικά το επέλεξα.

ΚΕΦΑΛΑΙΟ 4

Προβλήματα που αντιμετώπισα

Σε αυτή την ενότητα θα αναφερθώ σε μερικά προβλήματα που αντιμετώπισα κατά την διάρκεια της εκπόνησης της πτυχιακής μου εργασίας και θα σημειώσω το πως αντιμετώπισα το καθένα. Βέβαια θα πρέπει να επισημάνω πως η διαδικασία εκπόνησης της πτυχιακής εργασίας είναι ένα απαιτητικό πρόβλημα από μόνης της, καθώς σε όλη της τη διάρκεια κλήθηκα να αντιμετωπίσω επιμέρους προβλήματα για να φτάσω στο τελικό αποτέλεσμα. Υπό αυτό το πρίσμα δεν είναι δυνατόν να αναφερθώ σε κάθε πρόβλημα που αντιμετώπισα, αλλά σε μερικά που ήταν περισσότερο χρονοβόρα.

Πρόβλημα	Τη πρώτη φορά που πήγα να τρέξω το project και ενώ ήταν σε εντελώς αρχικό στάδιο (αμέσως μετά την εγκατάσταση του περιβάλλοντος ανάπτυξης), πείρα ένα error από το Android studio κατά την διαδικασία του build που έλεγε "failed to find Build Tools version 23.0.0".
Επίλυση	Έκανα δεξί κλικ στο project->Open module settings και όρισα την τιμή του build tools version σε 22.0.0, όσο είναι και το target sdk που έχει οριστεί για το project.

Πρόβλημα	Επειδή στην εφαρμογή μου, σετάρω την Action bar ώστε να έχει material look and feel, όταν πήγα αρχικά να την τρέξω πήρε ένα nullPointerException από την συνάρτηση getActionBar().
Επίλυση	Το θέμα που είχε by default το project μου δεν υποστήριζε Action bar. Οπότε επέλεξα το θέμα Theme.Holo.Light το οποίο υποστηρίζει Action Bar.

Πρόβλημα	Έπαιρνα γεωγραφικό πλάτος και μήκος χωρίς να υπάρχει σύνδεση στο GPS ή στο Internet. Για να πάρω μια γεωγραφική θέση πρέπει πρώτα να βεβαιωθώ πως λειτουργεί ο location listener μου. Ότι δηλαδή είτε μέσω GPS είτε μέσω

	Internet λαμβάνω θέσεις. Αυτό τον έλεγχο τον κάνει η συνάρτηση του Android isProviderEnabled(LocationManager.PROVIDER_VALUE). Αυτή πάντα επέστρεφε true, είτε ήταν ενεργοποιημένοι οι providers είτε όχι και έτσι η εκτέλεση του προγράμματος προχωρούσε και καλούσε την .getLastKnownPosition() η οποία μου επέστρεφε την τελευταία θέση που είχε αποθηκεύσει η συσκευή. Αυτό προκαλούσε μπέρδεμα.
Επίλυση	Διαπίστωσα πως αυτό δεν είναι bug αλλά φυσιολογικό, διότι η isProviderEnabled() δεν βλέπει αν υπάρχει σύνδεση αυτή την στιγμή, αλλά αν ο χρήστης της συσκευής έχει επιτρέψει την χρήση GPS ή Network Location στα settings του τηλεφώνου. Εγώ τα είχα επιλεγμένα και γι αυτό πάντα έπαιρνα true. Τελικά χειρίστηκα διαφορετικά αυτή την λογική χωρίς πρόβλημα.

Πρόβλημα	Γενικά το μεγαλύτερο πρόβλημα που αντιμετώπισα είχε να κάνει με το πως θα εναλλάσσω τα layout των fragments, για να επιτύχω την δημιουργία ενός δυναμικού GUI. Για παράδειγμα όπως έχει ήδη αναφερθεί, τα sliding tabs που βλέπουμε στο GUI κρατάνε το καθένα πάνω του από ένα fragment. Αυτό που ήθελα είναι να μπορώ να επέμβω και προγραμματιστικά να κρύψω κάποια elements και στη θέση τους να κάνω inflate κάποιο άλλο layout. Στη προσπάθεια μου αυτή αρχικά προσπαθούσα κατά την εκτέλεση της συνάρτησης onStart() της MainActivity.class, να πάρω το view του fragment και να το τροποποιήσω. Τελικά έπαιρνα ένα nullPointerException.
Επίλυση	Το πρόβλημα εδώ ήταν πως προσπαθούσα να πάρω το view από ένα fragment που ακόμα δεν είχε δημιουργηθεί. Ο λόγος είναι το ότι το onStart() της MainActivity με το onStart() του fragment διαφέρουν χρονικά μιας και πρέπει πρώτα να εκτελεστεί το πρώτο για να ξεκινήσει η εκτέλεση του δεύτερου. Έτσι σε αυτό το σημείο συνειδητοποίησα το πόσο σημαντικό

	είναι το fragment και activity lifecycle, διότι σε κάθε φάση του γίνονται διαθέσιμα και διαφορετικά πράγματα. Έτσι τελικά αυτό που έκανα, ήταν στη συνάρτηση onResume() του fragment, να ξεκινάω τις διαδικασίες τροποποίησης του layout διότι από αυτό το step και έπειτα είναι διαθέσιμο το view του.

Πρόβλημα	Δεν μπορούσα να χρησιμοποιήσω το geolocation για να κάνω debug στον android emulator.
Επίλυση	Το πρόβλημα αυτό δεν επιλύθηκε, αλλά θα πρέπει να γνωρίσουμε ότι για να παίξει το geolocation θα πρέπει να έχουν υλοποιηθεί από το σύστημα οι συναρτήσεις getFromLocation και getFromLocationName. Αν δεν έχουν υλοποιηθεί τότε το geolocation δεν λειτουργεί. Στον emulator οι συναρτήσεις αυτές δεν υπάρχουν συνεπώς αποτυγχάνει η λειτουργία του. Συνήθως αν ένα σύστημα έχει εγκατεστημένα τα Google play services τότε ο geolocator θα βρει ότι χρειάζεται και θα λειτουργήσει.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] 'Adding Maps'. Προσβάσιμο: https://developer.android.com/training/maps/index.html (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[2] Akash Bangad, 'How to make material design sliding Tabs'. Προσβάσιμο: http://www.android4devs.com/2015/01/how-to-make-material-design-sliding-tabs.html (Ημερομηνία επίσκεψης: 2016, Οκτώβριος).

[3] Anders Carisson, 'Using websockets in native iOS and Android apps'. Προσβάσιμο : http://www.elabs.se/blog/66-using-websockets-in-native-ios-and-android-apps (Ημερομηνία επίσκεψης: 2016, Ιανουάριος).

[4] 'Android (operating system)'. Προσβάσιμο: https://en.wikipedia.org/wiki/Android_(operating_system) (Ημερομηνία επίσκεψης: 2016, Μάρτιος).

[5] 'Android Studio'. Προσβάσιμο: https://developer.android.com/studio/index.html (Ημερομηνία επίσκεψης: 2016, Οκτώβριος).

[6] Aron Sreder (15 Μαΐου 2014) - τελευταία ανανέωση, 'Android: GPS positioning and location strategies'. Προσβάσιμο: https://blog.codecentric.de/en/2014/05/android-gps-positioning-location-strategies/ (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[7] ddewaele (25 Δεκεμβρίου 2010) - τελευταία ανανέωση, 'Understanding the LocationListener in Android'. Προσβάσιμο: http://blog.doityourselfandroid.com/2010/12/25/understanding-locationlistener-android/ (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[8] 'Developer's Guide'. Προσβάσιμο: https://developers.google.com/maps/documentation/geocoding/intro (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[9] 'Developer's Guide'. Προσβάσιμο: https://developers.google.com/maps/documentation/directions/intro (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[10] George Mathew (12 Μαρτίου 2013) - τελευταία ανανέωση, 'Drawing driving route directions between two locations using Google Directions in Google Map Android API v2'. Προσβάσιμο: http://wptrafficanalyzer.in/blog/drawing-driving-route-directions-between-two-locations-using-google-directions-in-google-map-android-api-v2/ (Ημερομηνία επίσκεψης: 2016, Δεκέμβριος).

[11] 'Get API key'. Προσβάσιμο: https://developers.google.com/maps/documentation/android-api/signup (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[12] 'Getting Started'. Προσβάσιμο: https://developers.google.com/maps/documentation/android-api/start (Ημερομηνία επίσκεψης: 2016, Οκτώβριος).

[13] 'Google Maps Intents'. Προσβάσιμο: https://developers.google.com/maps/documentation/android-api/intents (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[14] 'Install GlashFish on a CentOS 6 VPS' , (26 Φεβρουαρίου 2014). Προσβάσιμο: https://www.rosehosting.com/blog/install-glassfish-on-a-centos-6-vps/ (Ημερομηνία επίσκεψης: 2016, Δεκέμβριος).

[15] Jasdeep Jaitla (5 Ιανουαρίου 2015) - τελευταία ανανέωση, 'WebSockets vs REST: Understanding the Difference'. Προσβάσιμο: https://www.pubnub.com/blog/2015-01-05-websockets-vs-rest-api-understanding-the-difference/ (Ημερομηνία επίσκεψης: 2016, Ιανουάριος).

[16] 'JSON Tutorial'. Προσβάσιμο: http://www.w3schools.com/json/ (Ημερομηνία επίσκεψης: 2016, Ιανουάριος).

[17] 'Location Strategies'. Προσβάσιμο: https://developer.android.com/guide/topics/location/strategies.html (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[18] 'Storage Options'. Προσβάσιμο: https://developer.android.com/guide/topics/data/data-storage.html#filesInternal (Ημερομηνία επίσκεψης: 2016, Νοέμβριος).

[19] 'Using WebSocket for Real-Time Communication in Java Platform, Enterprise Edition 7'. Προσβάσιμο: http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/WebSocket/WebSocket.html (Ημερομηνία επίσκεψης: 2016, Ιανουάριος).

[20] 'WebSocket'. Προσβάσιμο: https://en.wikipedia.org/wiki/WebSocket (Ημερομηνία επίσκεψης: 2016, Ιανουάριος).