ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ, ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Π.Μ.Σ. «ΠΡΟΗΓΜΕΝΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΔΙΚΤΥΑ»

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΕΞΥΠΝΕΣ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ (SMARTPHONES) ΛΕΙΤΟΥΡΓΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ANDROID

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Άγγελος Β. Παπακώστας

Επιβλέπων : Νικόλαος, Τσελίκας

Επίκουρος Καθηγητής

Τρίπολη, Οκτώβριος 2014

ΕΥΧΑΡΙΣΤΙΕΣ

Τον επιβλέποντα καθηγητή μου για την άμεση και αποτελεσματική επικοινωνία σε όποιο πρόβλημα και αν παρουσιάστηκε κατά την πορεία της εργασίας, ακόμα και τις πιο βάρβαρες ώρες. Τον μέντορα σε θέματα προγραμματισμού και ξάδελφο μου Μάνο Ψυχογυιόπουλο, ο οποίος φιλοξένησε την βάση δεδομένων μου στον web server του. Τέλος, την οικογένειά μου Βασίλη, Ντόρα και Δήμητρα που σε αυτούς τους δύσκολους καιρούς με στήριξαν οικονομικά, αλλά πολύ περισσότερο πνευματικά.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Η εποχή που η χρήση των κινητών συσκευών περιοριζόταν στις τηλεφωνικές κλήσεις και την ανταλλαγή μηνυμάτων έχει περάσει προ πολλού. Η ταχύτατη εξέλιξη των τηλεπικοινωνιών και του υλικού (hardware) των κινητών συσκευών, επέτρεψε την ανάπτυξη ενός λειτουργικού συστήματος, το οποίο είναι ικανό να μετατρέψει ένα κινητό τηλέφωνο σ' ένα εκπληκτικό και παράλληλα πολύ χρήσιμο εργαλείο. Αυτή η χρησιμότητα επιβεβαιώνεται από την ύπαρξη πληθώρας εφαρμογών για το συγκεκριμένο λειτουργικό σύστημα. Με το πέρασμα των χρόνων η εμπειρία χρήσης βελτιώνεται και οι δυνατότητες μιας εφαρμογής ολοένα και πολλαπλασιάζονται. Όλα αυτά φυσικά αφορούν τους χρήστες των κινητών τηλεφώνων.

Ο τομέας της ανάπτυξης λογισμικού για κινητά τηλέφωνα έχει εξελιχθεί σημαντικά τα τελευταία χρόνια. Το Android έχει κάνει την εμφάνισή του ως μία νέα πλατφόρμα ανάπτυξης εφαρμογών κινητών τηλεφώνων και έξυπνων συσκευών γενικότερα, βασιζόμενη σε επιτυχίες του παρελθόντος και αποφεύγοντας τα παλιά λάθη άλλων πλατφορμών. Το Android σχεδιάστηκε έτσι, ώστε να δώσει στον προγραμματιστή όλα , όσα χρειάζεται ώστε να δημιουργήσει καινοτόμες εφαρμογές.

Το αντικείμενο της παρούσας πτυχιακής είναι η σχεδίαση και η μελέτη της διαδικασίας ανάπτυξης εφαρμογών για το λειτουργικό σύστημα Android. Το λειτουργικό σύστημα Android αποτελεί μια τεχνολογία η οποία μέρα με τη μέρα κερδίζει μεγαλύτερο μερίδιο στην αγορά των έξυπνων τηλεφώνων. Σήμερα, κατέχει την πρώτη θέση σε πωλήσεις, με πεντακόσια εκατομμύρια ενεργές συσκευές.

ΣΚΟΠΟΣ

Η παρούσα πτυχιακή εργασία περιστρέφεται γύρω από δύο άξονες. Πρώτος άξονας είναι η μελέτη και η καταγραφή των τεχνολογιών και των εργαλείων που απαιτούνται για την ανάπτυξη μιας τέτοιας εφαρμογής, καθώς και η παρουσίαση της ανάλυσης και σχεδίασης της εφαρμογής αυτής. Ο δεύτερος άξονας, στον οποίο δίνεται και μεγαλύτερη έμφαση, αφορά στην υλοποίηση της εφαρμογής αυτής αποκλειστικά για το λειτουργικό σύστημα Android. Η διαδικασία ανάπτυξης της εφαρμογής περιγράφεται βήμα προς βήμα ενώ, ταυτόχρονα, παρατίθενται λεπτομερείς επεξηγήσεις για τα σημαντικότερα επί μέρους κομμάτια της.

ABSTRACT

The purpose of this thesis is based on two axes. The first one is to study and record the technologies and tools required for the development of such an application. Specifically, the design and analysis of our application is thoroughly explained. The second axis, in which greater emphasis is given, is the implementation of an Android native application. The application development phase is presented in a step-by-step analysis, while further details are presented for the most significant parts of it.

ΚΕΦΑΛΑΙΟ 1: ΤΕΧΝΟΛΟΓΙΕΣ

1.1 ANDROID

Η Google έχοντας εντοπίσει την αυξημένη χρήση του Internet και αναζητήσεων στον Παγκόσμιο Ιστό μέσω κινητών συσκευών εξαγοράζει το 2005 την Android Inc με σκοπό την ανάπτυξη μιας πλατφόρμας για τέτοιου είδους συσκευές. Το 2007 λοιπόν, δημιουργείται ένας οργανισμός που αποτελείται από μεγάλο αριθμό εταιρειών τηλεπικοινωνιακού εξοπλισμού, καθώς και εταιρείες πληροφορικής όπως η Google, η T-Mobile, η Motorola, η Samsung, η Sony Ericsson, η Intel, η Vodafone και άλλες, με το όνομα Open Handset Alliance (http://openhandsetalliance.com). Σκοπός του είναι η έρευνα και η ανάπτυξη τεχνολογιών για την παραγωγή συσκευών, που θα διευκολύνουν τόσο τους παρόχους κινητής τηλεφωνίας όσο και τους κατασκευαστές των συσκευών αλλά και των προγραμματιστικών εφαρμογών. Τα μέλη της συμμαχίας δεσμεύτηκαν να παρέχουν αυτές τις τεχνολογίες βάσει του μοντέλου ανοιχτού πηγαίου κώδικα Apache.

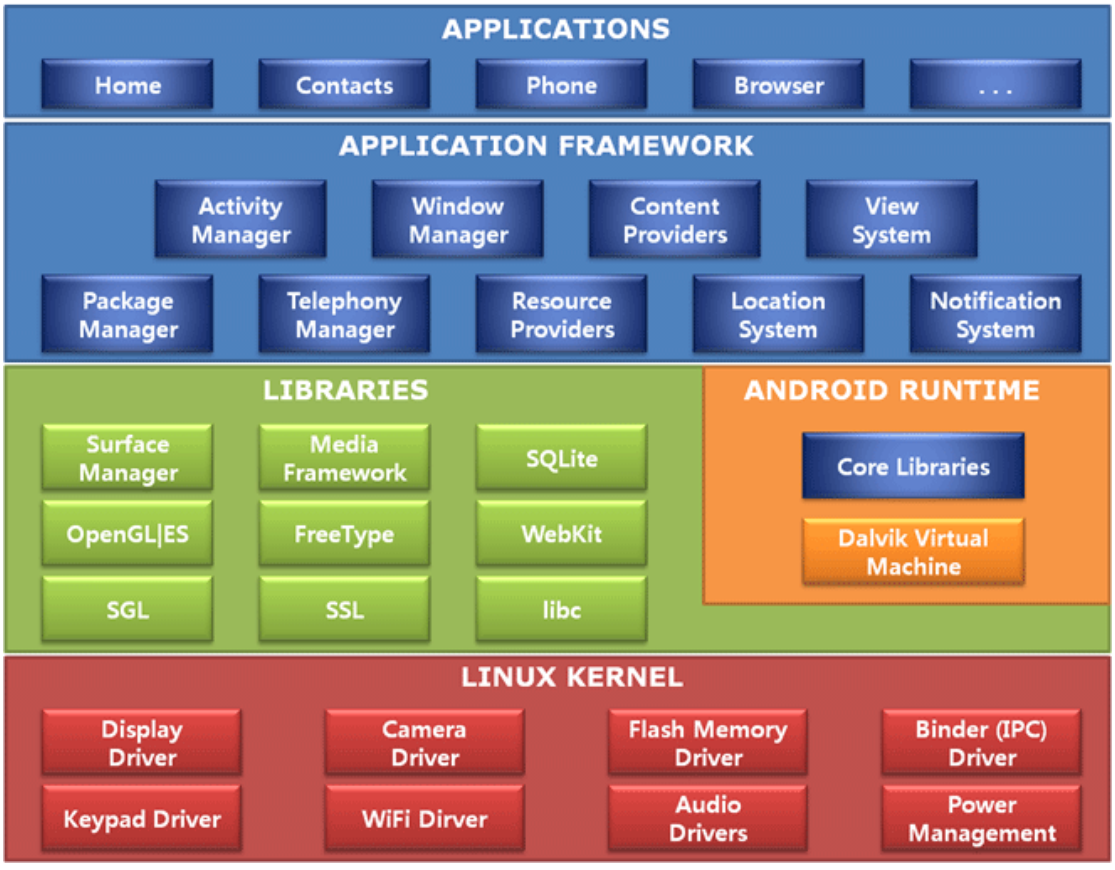
Η πρώτη έκδοση του Android SDK δημοσιεύτηκε τον Νοέμβριο του 2007, ενώ το πρώτο smartphone που έκανε χρήση λειτουργικού Android ήταν το G1 της T-Mobile. Οι συσκευές Android άρχισαν να διαδίδονται με γρήγορο ρυθμό κυρίως λόγω της δυνατότητας της πλατφόρμας να εκμεταλλεύεται το μοντέλο cloud computing αλλά και της έμφυτης υποστήριξης για συνεργασία με μια σχεσιακή βάση δεδομένων (SQLite). Ακολούθησαν αρκετές αναβαθμισμένες εκδόσεις του Android, κάθε μια προσθέτοντας νέα χαρακτηριστικά και λειτουργίες. Για κάποιον άγνωστο λόγο κάθε έκδοση του Android φέρει και μία κωδική ονομασία ενός γλυκού εδέσματος (1.5 – Cupcake, 1.6 – Donut, 4.4.2 – Kit Kat κ.ά.), ενώ από τις εκδόσεις αυτές, εκείνη που εισήγαγε την υποστήριξη πιο ανεπτυγμένων λειτουργιών ήταν σίγουρα η 2.0, στην οποία ενσωματώθηκε η υποστήριξη multitouch, HTML 5, text-to-speech και η δυνατότητα πιο προχωρημένων αναζητήσεων.

Η κοινότητα της ανάπτυξης εφαρμογών για κινητά τηλέφωνα βρίσκεται σε σημείο ανατροπής στην εποχή μας. Οι χρήστες κινητής τηλεφωνίας απαιτούν πλέον περισσότερες επιλογές, περισσότερες δυνατότητες προσαρμογής των τηλεφώνων τους και περισσότερες λειτουργίες. Οι πάροχοι υπηρεσιών κινητής τηλεφωνίας θέλουν να προσφέρουν στους συνδρομητές τους υλικό προστιθέμενης αξίας με εύχρηστο και επικερδή τρόπο. Αντίστοιχα, οι προγραμματιστές επιζητούν την ευχέρεια να αναπτύσσουν τις ισχυρές εφαρμογές, που απαιτούν οι χρήστες για τις κινητές συσκευές τους με τα λιγότερα δυνατά εμπόδια στον δρόμο για την επιτυχία. Τέλος, οι κατασκευάστριες εταιρείες κινητών τηλεφώνων αποζητούν μια σταθερή, οικονομική και κυρίως ασφαλή πλατφόρμα που θα τροφοδοτεί τις συσκευές τους. Μέχρι τώρα, μόνο μία πλατφόρμα έχει ικανοποιήσει επαρκώς τις ανάγκες όλων αυτών των πλευρών.

Αυτή η πλατφόρμα είναι το Android, μία καινοτόμος και ανοιχτή πλατφόρμα που επιχειρεί να καλύψει τις ανάγκες της αγοράς. Ένα από τα πιο δυνατά σημεία του Android είναι το ότι η ανάπτυξη των εφαρμογών γίνεται σχεδόν αποκλειστικά στη Java, ίσως την πιο δημοφιλή, πιο ολοκληρωμένη και καλύτερα δομημένη γλώσσα που υπάρχει σήμερα. Επιπλέον, η υλοποίηση

εφαρμογών Android μπορεί να γίνει με μικρό κόστος από την πλευρά του προγραμματιστή. Το Android βασίζεται στο Linux και έχει έναν αμιγώς open source χαρακτήρα. Σε αυτό το σημείο πρέπει να υπενθυμίσουμε κάτι σημαντικό αναφορικά με το Android. Υπάρχει πληθώρα συσκευών με διαφορετικά χαρακτηριστικά, με οθόνες διαφορετικών διαστάσεων και αναλύσεων, διαφορετικά σετ αισθητήρων, διαφορετικά περιβάλλοντα διάδρασης κ.α., τα οποία οι developers θα πρέπει να λαμβάνουν υπ' όψιν κατά την ανάπτυξη των εφαρμογών τους, ώστε αυτές να τρέχουν απροβλημάτιστα και με ομοιόμορφο τρόπο στις συσκευές. Αυτό φυσικά προϋποθέτει επιπλέον δουλειά από την πλευρά του προγραμματιστή.

Στην εικόνα που ακολουθεί παρατίθεται η αρχιτεκτονική του Android. Όπως παρατηρούμε, στον πυρήνα μιας πλατφόρμας Android βρίσκεται ένα Linux kernel το οποίο είναι υπεύθυνο για τη διαχείριση των device drivers, τον έλεγχο πρόσβασης στους πόρους του συστήματος, τη διαχείριση μνήμης και τις λοιπές υπηρεσίες που παρέχει ένα λειτουργικό σύστημα. Στους device drivers συγκαταλέγονται αυτοί της οθόνης, του Wi-fi, της κάμερας, του ήχου κ.ά.



Εικόνα 1: Αρχιτεκτονική του Android

Ένα επίπεδο πάνω βρίσκονται οι native βιβλιοθήκες του συστήματος που είναι γραμμένες σε C++ και περιλαμβάνουν το OpenGL, την SQLite, την Media Library κ.ά. Οι εφαρμογές που τρέχουν στο

κινητό μπορούν να έχουν πρόσβαση στις βιβλιοθήκες αυτές μέσω μιας Dalvik VM. Όπως έχει ήδη αναφερθεί, οι εφαρμογές Android είναι γραμμένες σε Java οπότε για να τρέξουν χρειάζονται το αντίστοιχο περιβάλλον. Όπως λοιπόν για να εκτελέσουμε μία εφαρμογή σε ένα PC είναι απαραίτητο να είναι εγκατεστημένο το κατάλληλο JRE (Java Runtime Environment), για τις εφαρμογές Android τον ρόλο του JRE παίζει η Dalvik VM. Δεδομένης της σαφώς πιο περιορισμένης επεξεργαστικής ισχύος καθώς και της ποσότητας διαθέσιμης μνήμης που έχουν οι κινητές συσκευές σε σχέση με τους υπολογιστές, η συγκεκριμένη VM είναι βελτιστοποιημένη να χρησιμοποιεί μικρότερα σε μέγεθος αρχεία ενδιάμεσου κώδικα, τα οποία σε αντίθεση με αυτά του Java SE έχουν κατάληξη .dex αντί για .class. Αν και υπάρχει έκδοση της Java για κινητά τηλέφωνα, η Java ME, η Google έκρινε σκόπιμο να χρησιμοποιήσει μια δική της υλοποίηση και έτσι γεννήθηκε η Dalvik. Κάθε εφαρμογή λοιπόν που, γράφουμε και εκτελούμε στο Android κινητό μας, χρησιμοποιεί την Dalvik και τρέχει σε ξεχωριστό instance του VM.

Στο αμέσως επόμενο επίπεδο βρίσκεται το Android SDK που περιέχει όλες τις απαραίτητες βιβλιοθήκες για τη συγγραφή εφαρμογών. Ο κώδικας που γράφεται για την εκτέλεση κάποιας λειτουργίας, όπως για παράδειγμα για την πραγματοποίηση μιας κλήσης, τον εντοπισμό της τρέχουσας θέσης κλπ, στην ουσία θα καλεί κάποια από τις μεθόδους που παρέχονται από το συγκεκριμένο SDK. Έτσι λοιπόν, οι εφαρμογές βρίσκονται στην κορυφή του application stack και ονομάζεται Application Layer.

1.2 JAVA

Οι εφαρμογές Android που αναπτύχθηκαν παλαιότερα χρησιμοποιούσαν ως γλώσσα προγραμματισμού την Java. Ερχόμενοι στο σήμερα, αυτή είναι πραγματικά η μόνη μας επιλογή για τις εφαρμογές. Οπότε ας αφιερώσουμε λίγο χρόνο να δούμε τι είναι στην πραγματικότητα η Java και γιατί μας είναι απαραίτητη. Η Java είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού που αναπτύχθηκε από την Sun Microsystems (η οποία τώρα ανήκει στην Oracle). Σαν γλώσσα προγραμματισμού αναπτύχθηκε αρκετό καιρό μετά την C και C++ και ενσωματώνει πολλά από τα χαρακτηριστικά αυτών των ισχυρών γλωσσών, ενώ διορθώνει ορισμένα από τα μειονεκτήματά τους. Μερικά από τα πιο ισχυρά χαρακτηριστικά της είναι τα εξής:

- Είναι εύκολη τόσο στη μάθηση όσο και στην κατανόηση
- Έχει σχεδιαστεί για να είναι ανεξάρτητη πλατφόρμας και ασφαλής, κάνοντας χρήση των virtual machines και τέλος
- Είναι αντικειμενοστραφής γλώσσα μηχανής

Η Java είναι εύκολη στη μάθηση για πολλούς λόγους. Ας ξεκινήσουμε από το γεγονός ότι πλέον υπάρχει απεριόριστο υλικό στο Διαδίκτυο σε ιστοσελίδες με tutorials, βιβλία αλλά και online σεμινάρια. Αναμφισβήτητα, στο Διαδίκτυο μπορεί κανείς να λύσει οποιαδήποτε απορία του για

διάφορα θέματα. Η διαφορά έγκειται στο ότι αναφερόμαστε στην πιο ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού παγκοσμίως. Ο λόγος είναι το ότι χρησιμοποιείται σε πολλά, διαφορετικού τύπου, προγραμματιστικά projects ανεξαρτήτως της κλίμακας τους. Από διαδικτυακές εφαρμογές έως εφαρμογές για κινητά τηλέφωνα (που στην παρούσα είναι αυτό που μας ενδιαφέρει περισσότερο). Αν κάποιος έχει ένα προγραμματιστικό υπόβαθρο από C ή C++, τότε θα βρει το συντακτικό της Java παρεμφερές. Σε περίπτωση όμως που θα αρχίσει από την Java τότε πρέπει να είναι σίγουρος ότι έχει διαλέξει την πιο εύκολη γλώσσα προγραμματισμού για να ξεκινήσει. Αυτό γιατί η Java είναι η πιο εύκολα αναγνώσιμη γλώσσα μηχανής από τον άνθρωπο, δηλαδή αν ένα πρόσωπο δεν έχει την παραμικρή ιδέα για τον προγραμματισμό, μπορεί να δει έναν κώδικα Java και να καταλάβει σε γενικές γραμμές τι είναι φτιαγμένος να κάνει αυτός ο κώδικας.

Με τόσες γλώσσες προγραμματισμού θα πρέπει να χρησιμοποιήσουμε έναν μεταγλωττιστή ώστε να μετατρέψουμε τον κώδικά τους σε γλώσσα μηχανής και να μπορεί να τον αντιληφθεί η συσκευή μας. Εδώ όμως υπάρχει ένα θέμα. Διαφορετικές συσκευές χρησιμοποιούν διαφορετικές γλώσσες μηχανής. Αυτό σημαίνει ότι πρέπει να μεταγλωττίζουμε τις εφαρμογές μας ανάλογα με τη γλώσσα μηχανής που χρησιμοποιεί η εκάστοτε συσκευή. Η Java μας προσφέρει τη λύση σε αυτό το πρόβλημα. Οι μεταγλωττιστές Java μετατρέπουν τον κώδικα από το αναγνώσιμο από τον άνθρωπο αρχείο πηγαίου κώδικα Java σε κάτι που ονομάζεται «bytecode» στον κόσμο της Java. Αυτό ερμηνεύεται από μία virtual machine της Java η οποία κινεί τη διαδικασία με τον τρόπο που θα λειτουργούσε μια φυσική CPU στην ερμηνεία του κώδικα σε γλώσσα μηχανής ώστε να εκτελέσει τον μεταγλωττισμένο κώδικα. Αν και φαίνεται αναποτελεσματικό, εντούτοις έχουν καταβληθεί τεράστιες προσπάθειες ώστε αυτή η διαδικασία να γίνεται με μεγάλη ταχύτητα και κυρίως εύστοχα. Οι Android εφαρμογές όπως ειπώθηκε παραπάνω τρέχουν σε μία ειδική virtual machine που ονομάζεται Dalvik VM.

Επειδή οι εφαρμογές Java τρέχουν σε μία virtual machine, είναι απομονωμένες από το υλικό της συσκευής. Ως εκ τούτου η VM μπορεί να ενσωματώσει, να περιλαμβάνει και να διαχειρίζεται την εκτέλεση του κώδικα με πιο ασφαλή τρόπο σε σχέση με τις γλώσσες που λειτουργούν σε κώδικα μηχανής απευθείας. Συγκεκριμένα η πλατφόρμα Android προχωράει ένα βήμα παραπέρα. Κάθε Android εφαρμογή εκτελείται στο λειτουργικό σύστημα, με βάση το Linux, χρησιμοποιώντας ένα διαφορετικό λογαριασμό χρήστη με πρωτοβουλία του Dalvik VM. Επιπρόσθετα, οι εφαρμογές αυτές παρακολουθούνται στενά από το λειτουργικό σύστημα και αν κάνουν χρήση υπερβολικής επεξεργαστικής δύναμης, είτε χρησιμοποιούν τους πόρους μιας συσκευής αλόγιστα, τότε κλείνουν. Επομένως είναι πολύ σημαντικό να δημιουργούμε εφαρμογές οι οποίες θα είναι σταθερές.

Στην πληροφορική, *αντικειμενοστρεφή προγραμματισμό* (*object-oriented programming),* ή ΑΠ, ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο

χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά, γίνεται από κοινού, μέσω μιας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται αντικείμενο και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων ονόματι *κλάση*. Η *κλάση* προδιαγράφει τόσο τα δεδομένα όσο και τις διαδικασίες, οι οποίες επιδρούν επάνω τους. Αυτή υπήρξε και η πρωταρχική καινοτομία του ΑΠ. Μόνο τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς «καλούπια». Οι αιτίες που ώθησαν στην ανάπτυξη του ΑΠ ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού (ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών), και τελικώς η αντικειμενοστρέφεια επικράτησε καθώς μπορούσε να αντεπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας.

1.3 ECLIPSE

Το Eclipse IDE (Integrated Development Environment) είναι ίσως το πιο δημοφιλές πρόγραμμα IDE για τους προγραμματιστές Java, αν και χρησιμοποιείται ως ένα IDE για C++, PHP, Rails (μέσω aptana plugins), Javascript και Android (SDK), στην πραγματικότητα είναι το πιο δημοφιλές, ανοικτού κώδικα IDE, και είναι πολύ επεκτάσιμο (μέσω plugins), ώστε όποια γλώσσα και αν χρησιμοποιούμε το Eclipse να συμβάλλει δραστικά.

Το IDE βοηθάει πολύ στη δημιουργία ενός σύνθετου περιβάλλοντος ανάπτυξης (σε Java π.χ.), έτσι ώστε να μπορεί να βοηθήσει στην αύξηση της παραγωγικότητας του. Το Eclipse IDE (Integrated Development Environment) είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών. Δεν είναι απαραίτητο να έχει κάποιος ένα περιβάλλον ανάπτυξης προκειμένου να αναπτύξει εφαρμογές, ένα απλό σημειωματάριο αρκεί. Οι τεράστιες όμως διευκολύνσεις που παρέχει ένα τέτοιο περιβάλλον, το καθιστούν απαραίτητο για μεσαίου και μεγάλου μεγέθους εφαρμογές. Το Eclipse παρέχει ένα περιβάλλον για την οργάνωση του κώδικα ενός ολόκληρου project. Λειτουργίες όπως διατήρηση εκδόσεων (versioning), διαμοιρασμός αλλαγών στον κώδικα μεταξύ των διαφόρων μελών της ομάδας ανάπτυξης, εικονικοί εξυπηρετητές για έλεγχο ορθότητας λειτουργίας web εφαρμογών, παρέχονται εντός του ίδιου του περιβάλλοντος. Επιπλέον, με την υποστήριξη των plug-in το Eclipse μπορεί να επεκτείνει τις λειτουργίες που προσφέρει ακόμα παραπέρα όπως ανέφερα και παραπάνω. Χαρακτηριστικό παράδειγμα είναι το ADT Plug-in για την ανάπτυξη εφαρμογών Android, που θα δούμε στη συνέχεια.

Χάρη στα Plug-in, το Eclipse μπορεί να επεκτείνει τις λειτουργικότητες που προσφέρει. Εκμεταλλευόμενη το γεγονός αυτό η Google ανέπτυξε το ADT (Android Development Tools) Plugin για το Eclipse. Με το plug-in αυτό, το Eclipse προσφέρει τη δυνατότητα σχεδίασης διεπαφών χρήστη για τις εφαρμογές Android με γραφικό τρόπο (παρόμοια με το Visual Studio της Microsoft). Επεκτείνει έτσι τις επιλογές των μενού εισαγωγής του Eclipse έτσι ώστε, για παράδειγμα, να μπορεί κανείς να δημιουργήσει ένα καινούριο project Android με τη σωστή δομή και αρχεία

αυτοματοποιημένα εντός του περιβάλλοντος. Επιπλέον, παρέχεται ειδικός editor για τα διάφορα XML αρχεία που περιέχονται σε ένα project Android, καθώς και εργαλεία για την παραγωγή των τελικών πακέτων, μιας εφαρμογής ώστε να διατεθεί προς πώληση. Η εγκατάσταση του plug-in μπορεί να γίνει εντός του ίδιου του Eclipse (ένα ακόμη σημαντικό χαρακτηριστικό του περιβάλλοντος αυτού).

Η ανάπτυξη της εφαρμογής θα γίνει χρησιμοποιώντας τα παραπάνω εργαλεία και ο έλεγχος της θα γίνει χρησιμοποιώντας τον εξομοιωτή που έχει εγκατασταθεί μαζί με το Android SDK. Για να γίνει χρήση του εξομοιωτή πρέπει φυσικά πρώτα να τον ρυθμίσουμε. Γενικά το Eclipse μας προσφέρει τη δυνατότητα να επιλέξουμε πολλά χαρακτηριστικά μιας εικονικής συσκευής όπως το μέγεθος της οθόνης της, το μέγεθος της μνήμης της, την έκδοση του Android που θα τρέχει και άλλα. Όλα αυτά θα τα δούμε πιο αναλυτικά όταν θα έρθει η ώρα να δοκιμάσω την εφαρμογή.

Η Google υποστηρίζει επίσημα το Eclipse και έχει αναπτύξει ειδικά για αυτό το ADT plugin, το οποίο παρέχει σύνδεση με το Android SDK και με όλες τις δυνατότητες που περιλαμβάνει αυτό. Επίσης το plugin παρέχει σύνδεση με τον AVD Manager για διαχείριση και εκκίνηση από το GUI του, εικονικών συσκευών Android για δοκιμές και αποσφαλμάτωση των εφαρμογών. Φυσικά, όπως αναφέρθηκε παραπάνω, ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει τον Text Editor ή ένα IDE της επιλογής του για τη δημιουργία του κώδικα και μετέπειτα να χρησιμοποιήσει τα εργαλεία JDK και Apache Ant μέσω γραμμής εντολών για να μεταγλωττίσει την εφαρμογή του ώστε να την τεστάρει με όλες τις δυνατότητες που του παρέχει το Android SDK.

Η επιλογή του IDE, που κάνει όλη την πολύπλοκη δουλειά είναι λοιπόν προφανής, καθώς τα περισσότερα παραδείγματα και άρθρα για το Android στηρίζονται στο γεγονός ότι η πλειονότητα των developers χρησιμοποιεί το Eclipse μαζί με το ADT plugin. Οπότε ξεκινάμε με αυτό σαν δεδομένο.

Για να ξεκινήσουμε τη σχεδίαση του user interface της εφαρμογής μας είναι σκόπιμο να αναφερθούμε στον τρόπο με τον οποίο μπορούμε να επεξεργαζόμαστε τα αρχεία XML μέσα από το περιβάλλον του Eclipse, μιας και τα αρχεία αυτά χρησιμοποιούνται εκτενώς στα Android projects. Π.χ. σε κάθε τέτοιο project υπάρχει ένα αρχείο (συνήθως το strings.xml) στο οποίο αποθηκεύουμε τα resources που χρησιμοποιούμε στην εφαρμογή μας. Με τον όρο resources αναφερόμαστε σε όλα τα συστατικά της εφαρμογής μας. Το περιβάλλον του eclipse λοιπόν μας παρέχει τον graphical XML editor για την επεξεργασία αυτών των αρχείων. Φυσικά, εκτός από αυτόν και για όσους προτιμούν τη χειροκίνητη σύνταξη XML αρχείων, υπάρχει και ο αντίστοιχος XML text editor. Το ποιόν από τους δύο editors θα χρησιμοποιήσουμε για την επεξεργασία αυτού του τύπου αρχείων, εξαρτάται αποκλειστικά και μόνο από εμάς.

Όταν έρθει λοιπόν η στιγμή να ξεκινήσουμε τη σχεδίαση του γραφικού περιβάλλοντος της εφαρμογής μας, συνήθως πρώτα ζωγραφίζουμε ένα προσχέδιο του interface σε ένα κομμάτι χαρτί το οποίο στη συνέχεια θα αναπαράγουμε σε κώδικα. Έχοντας καταλήξει λοιπόν στο GUI (Graphical User Interface) της εφαρμογής μας μπορούμε να ξεκινήσουμε την υλοποίησή του στο Eclipse. Όπως και στην περίπτωση του XML editor έτσι και στην περίπτωση του GUI editor μας επιτρέπει να

επεξεργαστούμε το XML αρχείο είτε χρησιμοποιώντας το γραφικό περιβάλλον (Graphical Layout), είτε χειροκίνητα (που οδηγεί στον αντίστοιχο XML editor). Εκεί τα Layouts του Android έχουν αντίστοιχη λειτουργία με τους Layout Managers που χρησιμοποιούμε σε παραθυρικές εφαρμογές για PC σε Java, με την διαφορά ότι υπάρχει σαφώς μικρότερη ποικιλία. Κάθε Android GUI θα περιλαμβάνει τουλάχιστον ένα τέτοιο Layout, ενώ τα πιο σύνθετα GUIs θα αποτελούνται από περισσότερα και σε διάφορους συνδυασμούς.

Ανέφερα προηγουμένως πως κατά την υλοποίηση ενός UI χρησιμοποιώ τόσο τον GUI editor όσο και τον XML editor. Κάθε ένας από αυτούς προσφέρει τα δικά του πλεονεκτήματα, κάτι που θα διαπιστώσουμε στη χρήση τους. Ο GUI editor είναι πολύ βολικός γιατί μέσω της λειτουργίας drag ή drop διευκολύνει και επιταχύνει κατά πολύ την όλη διαδικασία της σχεδίασης, προβάλλοντας παράλληλα μία επισκόπηση του πώς θα φαίνεται το interface στην οθόνη μιας συσκευής. Αυτό που πρέπει να γνωρίζουμε είναι ότι ο GUI editor παράγει αυτόματα το κείμενο XML που απαιτείται για την επίτευξη του συγκεκριμένου οπτικού αποτελέσματος.

1.4 SQLITE

Για την αποθήκευση πιο πολύπλοκων δεδομένων, μία σχεσιακή βάση δεδομένων προσφέρει ταχύτερη και πιο ευέλικτη πρόσβαση από ότι τα απλά αρχεία ή ο μηχανισμός των Shared Preferences. Η πλατφόρμα Android παρέχει έμφυτη υποστήριξη της SQLite, μιας ελαφριάς έκδοσης αλλά με ολοκληρωμένη λειτουργικότητα σχεσιακής βάσης δεδομένων την οποία μπορούμε να χρησιμοποιούμε για την αποθήκευση των δεδομένων των εφαρμογών μας. Κάθε εφαρμογή μπορεί να αποκτήσει το δικό της στιγμιότυπο της βάσης. Τα δεδομένα που έχουν αποθηκευτεί στη συνέχεια μπορούν να γίνουν διαθέσιμα σε άλλες εφαρμογές μέσω του Content Provider.

Η SQLite είναι στην ουσία ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων, το οποίο σε αντίθεση με τα άλλα συστήματα δεν είναι μια ξεχωριστή διαδικασία που είναι προσβάσιμη από την εφαρμογή του πελάτη, αλλά αναπόσπαστο κομμάτι της. Είναι αναμφισβήτητα η πιο ευρέως διαδεδομένη μηχανή διαχείρισης βάσεων δεδομένων, καθώς χρησιμοποιείται από πολλούς διαδικτυακούς περιηγητές όπως ο Mozilla Firefox και ο Google Chrome. Παρακάτω θα εξετάσουμε τις αιτίες, που την κάνουν τόσο δημοφιλή.

Ξεκινώντας, πρέπει να επισημάνουμε ότι η SQLite είναι μια βιβλιοθήκη, που εφαρμόζει έναν αυτοδύναμο, χωρίς server δηλαδή, και χωρίς καμία αρχική διαμόρφωση μηχανισμό διαχείρισης βάσεων δεδομένων SQL. Ο κώδικας για την SQLite παρέχεται για δημόσια χρήση και συνεπώς είναι ελεύθερος για οποιονδήποτε σκοπό, εμπορικό ή ιδιωτικό. Η SQLite αυτή τη στιγμή χρησιμοποιείται σε περισσότερες εφαρμογές από ό,τι μπορούμε να φανταστούμε, συμπεριλαμβανομένων αρκετών έργων υψηλού προφίλ.

Η SQLite είναι μια ενσωματωμένη μηχανή βάσης δεδομένων SQL. Σε αντίθεση με τις περισσότερες βάσεις δεδομένων SQL, η SQLite δεν έχει μια ξεχωριστή διεργασία διακομιστή, καθώς διαβάζει και

γράφει απευθείας σε συνηθισμένα αρχεία στο δίσκο. Ουσιαστικά είναι μια πλήρης βάση δεδομένων τύπου SQL με πολλούς πίνακες, δείκτες και απόψεις η οποία περιέχεται σε ένα ενιαίο αρχείο στο δίσκο. Η μορφή του αρχείου της βάσης δεδομένων είναι cross-platform, δηλαδή μπορείτε να αντιγράψετε ελεύθερα μια βάση δεδομένων μεταξύ των συστημάτων 32-bit και 64-bit ή μεταξύ διαφόρων μορφών αρχιτεκτονικής. Αυτά τα χαρακτηριστικά καθιστούν την SQLite μια πολύ δημοφιλή επιλογή ως μορφή αρχείου εφαρμογής. Σκεφτείτε την SQLite όχι ως υποκατάστατο της Oracle, αλλά ως υποκατάστατο της συνάρτησης fopen.

Πέρα από όλα τα προηγούμενα που αναφέρθηκαν, η SQLite χαρακτηρίζεται ως μια συμπαγής βιβλιοθήκη. Με ενεργοποιημένα όλα τα χαρακτηριστικά, το μέγεθος της βιβλιοθήκης μπορεί να είναι μικρότερο και από 500KB, ανάλογα με τον στόχο της πλατφόρμας και τις βέλτιστες ρυθμίσεις του compiler (π.χ. ο κώδικας 64-bit είναι μεγαλύτερος σε μέγεθος). Σε περίπτωση που οι προαιρετικές επιλογές παραλείπονται, το μέγεθος της βιβλιοθήκης SQLite μπορεί να μειωθεί κάτω από τα 300KB. Η SQLite μπορεί έτσι να τρέξει σε ελάχιστο χώρο στοίβας (4KB) και πολύ λίγο σωρού (100KB), κάνοντας την SQLite την πιο δημοφιλή επιλογή βάσης δεδομένων για συσκευές με περιορισμένη μνήμη, όπως τα κινητά τηλέφωνα, τα PDAs, και τα MP3 players. Εδώ υπάρχει ένα δίλημμα μεταξύ της χρήσης μνήμης και της ταχύτητας. Γενικά η SQLite τρέχει πιο γρήγορα, όσο περισσότερη μνήμη της δώσουμε. Παρ' όλα αυτά, οι επιδόσεις είναι συνήθως αρκετά καλές, ακόμη και σε περιβάλλοντα με χαμηλή μνήμη.

Η SQLite δοκιμάζεται πολύ προσεκτικά πριν από κάθε έκδοσή της και έχει τη φήμη ότι είναι αρκετά αξιόπιστη. Το μεγαλύτερο κομμάτι του πηγαίου κώδικα της SQLite είναι αφιερωμένο αποκλειστικά σε δοκιμές και ελέγχους. Μια αυτοματοποιημένη δοκιμαστική ακολουθία τρέχει εκατομμύρια σενάρια δοκιμών τα οποία περιέχουν εκατοντάδες εκατομμύρια ατομικές δηλώσεις SQL και επιτυγχάνει το 100% της δοκιμαστικής κάλυψης ανά παρακλάδι κώδικα. Η SQLite ανταποκρίνεται αρκετά καλά σε αποτυχίες εκχώρησης μνήμης και σε λάθη στον δίσκο I / O. Οι συναλλαγές είναι ACID (ένα σετ ιδιοτήτων το οποίο όταν το αναφέρουμε εννοούμε ότι οι συναλλαγές των βάσεων δεδομένων είναι αξιόπιστες), ακόμη και αν κολλήσει το σύστημα ή υπάρξουν διακοπές ρεύματος. Όλο αυτό επαληθεύεται από τις αυτοματοποιημένες δοκιμές με τη χρήση ειδικών σεναρίων δοκιμών που προσομοιώνουν τις αποτυχίες του συστήματος. Φυσικά, ακόμη και μετά από όλη αυτή την δοκιμαστική διαδικασία, εξακολουθούν να υπάρχουν σφάλματα. Αλλά σε αντίθεση με κάποια παρόμοια projects (κυρίως από τους εμπορικούς ανταγωνιστές) η SQLite είναι ανοιχτή και ειλικρινής για όλα τα σφάλματα. Έτσι παρέχει καταλόγους σφαλμάτων, και καταλόγους των κρίσιμων σφαλμάτων, καθώς και λεπτό-προς-λεπτό παρακολούθηση και καταγραφή των αναφορών σφαλμάτων και οποιωνδήποτε άλλων αλλαγών γίνουν στον κώδικα.

Η βάση του κώδικα της SQLite υποστηρίζεται και αναβαθμίζεται από μια διεθνή ομάδα

προγραμματιστών που εργάζονται για την SQLite σε καθεστώς πλήρους απασχόλησης. Οι προγραμματιστές λοιπόν, συνεχίζουν να επεκτείνουν τις δυνατότητες της SQLite και την ενίσχυση της αξιοπιστίας και της απόδοσής της. Προσπαθούν φυσικά να διατηρήσουν παράλληλα τη συμβατότητα με τις παλαιότερες εκδόσεις, με το δημοσιευμένο spec interface, τη σύνταξη SQL και τη μορφή του αρχείου βάσης δεδομένων. Ο πηγαίος κώδικας είναι απολύτως ελεύθερος και προσβάσιμος σε όποιον ενδιαφέρεται, αλλά και η επαγγελματική υποστήριξη είναι επίσης διαθέσιμη.

Τα βήματα που απαιτούνται γενικά για τη χρήση μιας βάσης SQLite είναι τα εξής:

1. Δημιουργία και άνοιγμα της βάσης
2. Δημιουργία πίνακα
3. Δημιουργία διαδικασίας εισαγωγής εγγραφών
4. Δημιουργία διαδικασίας ανάκτησης δεδομένων μέσω ερωτήματος
5. Κλείσιμο της βάσης

Για να δούμε στην πράξη την όλη διαδικασία θα πρέπει φυσικά να αναπτύξουμε μια εφαρμογή που θα κάνει χρήση μιας βάσης δεδομένων.

1.5 LOCATION BASED SERVICES

Είτε για λόγους ασφαλείας, είτε για λόγους ευκολίας, τα χαρακτηριστικά θέσης σε κινητά τηλέφωνα είναι τα πλέον βασικά στις μέρες μας. Ως τέτοια, η ενσωμάτωση πληροφοριών θέσης, πλοήγησης και χαρακτηριστικών χαρτογράφησης στο έργο μας μπορεί να καταστήσει την εφαρμογή μας πιο ανθεκτική.

Το Android SDK παρέχει μέσα για την πρόσβαση στοιχείων θέσης μέσω ενσωματωμένου υλικού GPS, όταν υπάρχει. Γενικά, όλα σχεδόν τα τηλέφωνα Android έχουν κάποιες δυνατότητες LBS. Για παράδειγμα στις Ηνωμένες Πολιτείες οι πληροφορίες θέσης σε κινητά τηλέφωνα χρησιμοποιούνται από εφαρμογές επείγουσας ανάγκης. Ωστόσο, ούτε όλες οι συσκευές Android είναι τηλέφωνα, ούτε όλα τα τηλέφωνα παρέχουν υπηρεσίες LBS, που μπορούν να χρησιμοποιηθούν από χρήστες. Εάν τα χαρακτηριστικά GPS έχουν απενεργοποιηθεί ή μια συσκευή Android δεν έχει υλικό LBS, το Android SDK παρέχει πρόσθετα APIs για τον προσδιορισμό εναλλακτικών παρόχων θέσης. Αυτοί οι πάροχοι μπορεί να έχουν πλεονεκτήματα και μειονεκτήματα όσον αφορά στη χρήση ισχύος, στην ταχύτητα και στην ακρίβεια της αναφοράς.

Οι υπηρεσίες LBS και το υλικό, όπως ένα ενσωματωμένο σύστημα GPS υψηλής ακρίβειας, είναι προαιρετικά χαρακτηριστικά για συσκευές Android. Εκτός από την αίτηση των κατάλληλων δικαιωμάτων, μπορούμε να καθορίσουμε ποια προαιρετικά χαρακτηριστικά θα ζητά η εφαρμογή μας.

Οι υπηρεσίες βάσει τοποθεσίας (Location-Based Services) αποτελούν μία από τις πιο ενδιαφέρουσες δυνατότητες της πλατφόρμας, δεν είναι τυχαίο άλλωστε πως γύρω από αυτήν έχουν χτιστεί οι πιο δημοφιλείς εφαρμογές για Android. Η δυνατότητα αυτή σε συνδυασμό με την έμφυτη υποστήριξη της κορυφαίας εφαρμογής χαρτών της Google (Google Maps) από την πλατφόρμα αποτελεί ένα από τα δυνατά σημεία των συσκευών Android έναντι των ανταγωνιστών τους. Οι υπηρεσίες βάσει τοποθεσίας μπορούν να ενσωματωθούν σε πολλές λειτουργίες όπως για παράδειγμα σε μία αναζήτηση web, στη λήψη φωτογραφιών , την κοινωνική δικτύωση που είναι και το πιο σύνηθες και πολυχρησιμοποιημένο κ. ά.

Η ένωση του Διαδικτύου και του κινητού τηλεφώνου έγιναν η αιτία για τη μεγάλη αύξηση στην αγορά των φορητών συσκευών. Έτσι οι δύο αυτές τεχνολογίες υπόσχονται στο κοινό ακόμα περισσότερες υπηρεσίες. Για παράδειγμα, η γνώση της θέσης του χρήστη αποκτά όλο και μεγαλύτερο ενδιαφέρον για έναν πάροχο κινητής τηλεφωνίας. Φυσικά αυτή η ιδέα υπήρχε από παλιά με τις υπηρεσίες εντοπισμού θέσης του οχήματος. Η τελειοποίηση των υπηρεσιών με βάση τη γεωγραφική θέση έδωσε το έναυσμα στην αγορά για πολλούς προγραμματιστές. Τα τελευταία χρόνια έχουμε την πλήρη εμπορευματοποίηση των LBS με μια πληθώρα εμπορικών υπηρεσιών και εφαρμογών με βάση την τοποθεσία και πληροφορίες που αφορούν την περιοχή στην οποία βρίσκεται ο χρήστης.

Οι εξελίξεις στις τηλεπικοινωνίες και στις τεχνολογίες πληροφόρησης, καθιστούν αναγκαία την ανάπτυξη υψηλής ποιότητας αλλά και ποσότητας υπηρεσιών που βασίζονται στη γεωγραφική θέση των χρηστών, που τις χρησιμοποιούν. Όταν ένας χρήστης ζητά την παροχή κάποιας LBS υπηρεσίας, στέλνει μέσω μιας συσκευής του την τρέχουσα θέση του σε έναν απομακρυσμένο πάροχο μιας υπηρεσίας, που βρίσκεται εγκατεστημένος στον τηλεπικοινωνιακό φορέα στον οποίο είναι εγγεγραμμένος ο χρήστης. Οι θέσεις από τις οποίες διέρχεται ο κάθε χρήστης συλλέγονται από ειδικά συστήματα (GPS) που είναι ενσωματωμένα σε διάφορες συσκευές (smartphones, tablets, GPS devices κ.ά.). Για να υπάρξει μια σωστή τεχνολογία LBS πρέπει να πληρούνται κάποιες προϋποθέσεις για τον εντοπισμό θέσης οι οποίες είναι οι εξής:

- Συντεταγμένες ακριβείας που καθορίζονται από την αρμόδια υπηρεσία
- Όσο το δυνατό χαμηλότερο κόστος
- Ελάχιστες επιπτώσεις στον εξοπλισμό και στο δίκτυο.

Επίσης υπάρχουν κάποιες απαιτήσεις, που κάνουν μια εφαρμογή LBS εκμεταλλεύσιμη από το κοινό και αυτές είναι οι εξής:

- Ελάχιστη χρήση του δικτύου του παρόχου για ανάκτηση πληροφοριών
- Φιλικό γραφικό περιβάλλον για τον χρήστη
- Χρήση διαθέσιμων διαδικτυακών υπηρεσιών
- Υψηλή διαθεσιμότητα των υπηρεσιών ακόμα και σε περιόδους υψηλής ζήτησης
- Δυνατότητα Αναβάθμισης
- Χαμηλό κόστος

Φυσικά υπάρχουν και οι περιπτώσεις που οι εφαρμογές LBS αναφέρονται σε περιοχές εντός κτιρίων. Σε αυτές λοιπόν, πρέπει να πληρούνται τα παρακάτω:

- Ελάχιστη γραφική αλληλεπίδραση
- Εύκολη καθοδήγηση μέσω του φυλλομετρητή
- Υποστήριξη προσανατολισμού στο χώρο

Η εφαρμογή Maps της Google είναι αναμφισβήτητα μια από τις πιο ενδιαφέρουσες και πιο εντυπωσιακές (ειδικά μετά τις πρόσφατες αναβαθμίσεις της) μαζί με τις Earth, Sky κ.ά. Δεδομένου του ότι και η ίδια η πλατφόρμα είναι προϊόν της Google, δεν θα μπορούσε παρά να παρέχεται υποστήριξη για τις εφαρμογές στις συσκευές Android. Αν και το ίδιο το Android SDK δεν παρέχει έμφυτη υποστήριξη για τις εφαρμογές της Google, μπορούμε να κατεβάσουμε και να εγκαταστήσουμε το Google API ως Third Party πακέτο αν δεν το έχουμε κάνει ήδη, μέσω του εργαλείου AVD Manager, απ' όπου μπορούμε να εγκαταστήσουμε τις διαφορετικές εκδόσεις του Google API καθώς υπάρχει ξεχωριστή έκδοση του Google API για κάθε έκδοση του Android. Για να λειτουργήσει σωστά το Google Maps API απαιτείται η χρήση ενός έγκυρου κλειδιού, το οποίο μπορούμε να το προμηθευτούμε από την Google. Το πώς θα το δούμε παρακάτω.

ΚΕΦΑΛΑΙΟ 2: ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

2.1 ΚΑΤΗΓΟΡΙΕΣ ΣΥΣΤΑΤΙΚΩΝ ANDROID

Στις προηγούμενες ενότητες της εργασίας είδαμε όλες αυτές τις τεχνολογίες που χρησιμοποιούμε κατά κόρον στην ανάπτυξη εφαρμογών για έξυπνες συσκευές. Παρόλα αυτά δεν έχουμε κάνει κάποια ανάλυση για το πώς ξεκινάμε την κατασκευή της εφαρμογής μας, ενώ μέχρι στιγμής δεν έχουμε μιλήσει καθόλου για το ποια είναι τα συστατικά από τα οποία αποτελείται μια τυπική Android εφαρμογή.

Μια εφαρμογή για κινητό Android λοιπόν αποτελείται από διάφορες λειτουργίες. Ορισμένα παραδείγματα είναι η επεξεργασία μιας σημείωσης, η προβολή των στοιχείων μιας επαφής, η αναπαραγωγή ενός αρχείου μουσικής κ.ο.κ. Όλες οι προαναφερθείσες λειτουργίες μπορούν να ομαδοποιηθούν σε 4 διαφορετικές κατηγορίες συστατικών του Android, κάθε μία από τις οποίες αντιστοιχεί σε μια Java κλάση. Οι κατηγορίες αυτές είναι τα Intents, τα Activities, τα Services και οι Content Providers. Πάμε να δούμε την κάθε μία ξεχωριστά λοιπόν:

1) Activity: αποτελεί τον θεμέλιο λίθο ενός User Interface. Κατ' αναλογία, ένα Activity είναι για μια Android εφαρμογή ότι ένα παράθυρο ή ένας διάλογος για μια παραθυρική εφαρμογή στον προσωπικό μας υπολογιστή. Κάθε εφαρμογή λοιπόν αποτελείται από

τουλάχιστον ένα Activity ή και περισσότερα, αν η εφαρμογή χρησιμοποιεί περισσότερες της μίας οθόνες διάδρασης.

2) Intent: είναι ίσως το πιο σημαντικό συστατικό της πλατφόρμας και περιγράφει ένα μήνυμα συστήματος. Τα μηνύματα αυτά παράγονται και «κυκλοφορούν» στο σύστημα καθ' όλη τη χρονική περίοδο κατά την οποία η συσκευή είναι σε λειτουργία και ενημερώνουν τις εφαρμογές για το κάθε είδος συμβάντος , που λαμβάνει χώρα ανά πάσα χρονική στιγμή. Ένα τέτοιο συμβάν μπορεί να είναι μια αλλαγή στην κατάσταση μιας hardware συσκευής (π.χ. ο χρήστης ενεργοποίησε το GPS), ή μια εισροή δεδομένων (π.χ. ο χρήστης δέχθηκε ένα SMS), ή ένα συμβάν που προκλήθηκε από μία εφαρμογή μετά από χειρισμό του χρήστη (π.χ. ο χρήστης επέλεξε ένα εξωτερικό link της εφαρμογής και άνοιξε ο περιηγητής για να προβάλει την αντίστοιχη ιστοσελίδα). Ως προγραμματιστές όχι μόνο μπορούμε να εκτελούμε μια ενέργεια ανταποκρινόμενοι σε ένα τέτοιο συμβάν, αλλά έχουμε τη δυνατότητα να δημιουργούμε τα δικά μας συμβάντα ούτως ώστε να λανσάρουμε κάποιο Activity ή για να ενημερωθούμε για μια συνθήκη που επαληθεύεται.
3) Content Provider: αποτελούν το μέσο που δίνει πρόσβαση στην εφαρμογή σε δεδομένα που βρίσκονται αποθηκευμένα στο σύστημα και που είναι χρήσιμα για τη λειτουργία της. Για παράδειγμα, μία εφαρμογή που θέλει να έχει πρόσβαση στις επαφές του τηλεφώνου για να πραγματοποιήσει μία κλήση, θα πρέπει να κάνει χρήση του αντίστοιχου Content Provider. Εκτός από τους ήδη υπάρχοντες μπορούμε να δημιουργήσουμε και τους δικούς μας Content Providers ώστε να δίνουμε πρόσβαση με ελεγχόμενο τρόπο σε δεδομένα, που θέλουμε ως κοινόχρηστα σε άλλες εφαρμογές.
4) Services: όλα τα προηγούμενα που αναφέραμε έχουν μια τυπικά μικρή διάρκεια ζωής και μπορούν να τερματίσουν από το λειτουργικό σύστημα ανά πάσα χρονική στιγμή. Τα services είναι σχεδιασμένα να εκτελούνται στο παρασκήνιο ανεξάρτητα από κάποιο Activity. Θα μπορούσαμε δηλαδή να γράψουμε κάποιο Service το οποίο θα παίζει διαρκώς ένα video ακόμα και αν το Activity, που το ελέγχει, έχει τερματιστεί. Τα Android Services είναι κάτι αντίστοιχο των Windows Services του λειτουργικού των Windows.

Κάθε εφαρμογή λοιπόν απαρτίζεται από ένα ή περισσότερα από τα προαναφερθέντα συστατικά, τα οποία αρχικοποιούνται από το Android, όταν χρειαστεί. Οι υπόλοιπες εφαρμογές που εκτελούνται μπορούν να τα χρησιμοποιήσουν, ανάλογα με τα καθορισμένα δικαιώματα πρόσβασης. Κάθε συστατικό περνάει από διάφορες φάσεις του κύκλου ζωής του, παράλληλα με τις λειτουργίες που γίνονται κατά τη χρήση της συσκευής.

Όλα αυτά τα συστατικά, που περιγράψαμε πιο πάνω, υλοποιούνται σε αρχεία που περιέχουν κώδικα είτε Java είτε XML.

2.2: ΔΟΜΗ ΚΑΙ ΟΡΓΑΝΩΣΗ ΕΝΟΣ ANDROID PROJECT

Αφού είδαμε ποια είναι τα συστατικά που αποτελούν μια εφαρμογή, ας περάσουμε να δούμε και το πώς είναι δομημένα και οργανωμένα τα αρχεία σε ένα project Android εφαρμογής. Τα αρχεία λοιπόν και οι φάκελοι είναι δομημένα ως εξής:

-Main Activity.java: είναι το αρχείο που περιέχει την default οθόνη που εμφανίζεται στον χρήστη, όταν εκτελείται η εφαρμογή. Βρίσκεται στον φάκελο src που περιέχει όλον τον πηγαίο κώδικα της εφαρμογής και παράγεται από το περιβάλλον υλοποίησης με αυτόματο τρόπο. Το όνομά του το ορίζουμε εμείς στην αρχή της δημιουργίας του project μας.

-R.java: είναι ένα αρχείο που περιέχει όλες τις σταθερές ταυτοποίησης για τα συστατικά της εφαρμογής. Παράγεται και συντηρείται αυτόματα από το Eclipse και δεν θα πρέπει να γίνεται κάποια προσπάθεια επεξεργασίας του. Βρίσκεται μέσα στον φάκελο gen.

-Android 4.3: ο φάκελος που περιέχει τα αρχεία του Android SDK.

-Android Private Libraries: οι βιβλιοθήκες Android επιτρέπουν σε κάποιον να αποθηκεύσει τον πηγαίο κώδικα και τους πόρους που χρησιμοποιούνται από πολλά άλλα Android projects. Τα Android Development Tools (ADT) συγκεντρώνουν το περιεχόμενο της βιβλιοθήκης στο Android project, δημιουργώντας ένα αρχείο JAR. Χρησιμοποιώντας τις βιβλιοθήκες, παίρνουμε σημαντική βοήθεια στο να δομήσουμε σωστά τον κώδικα της εφαρμογής μας . Επίσης, όλο και περισσότερες σημαντικές βιβλιοθήκες ανοικτού κώδικα είναι διαθέσιμες για το Android. Είναι πολύ σημαντικό για έναν Android προγραμματιστή να κατανοήσει τα projects των βιβλιοθηκών.

-Android Dependencies: είναι ένας εικονικός φάκελος όπου το Eclipse υποδεικνύει από ποιά JAR αρχεία εξαρτάται το project. Δεν είναι ένα φυσικός φάκελος, δηλαδή δεν θα το βρούμε στο σκληρό μας δίσκο . Η διαγραφή αυτού του φακέλου θα καταστρέψει το project μας .

-assets: Φάκελος που περιέχει όλα τα multimedia και λοιπά αρχεία που μπορεί να χρησιμοποιήσει η εφαρμογή μας.

-bin: είναι ο φάκελος στον οποίο αποθηκεύονται όλα τα εκτελέσιμα (binary) αρχεία. Τέτοια είναι τα Java tools π.χ.

-libs: είναι ο φάκελος στον οποίον αποθηκεύονται τα αρχεία βιβλιοθήκης, κοινώς είναι κλάσεις της Java, που μας βοηθάνε στην ανάπτυξη της εφαρμογής μας.

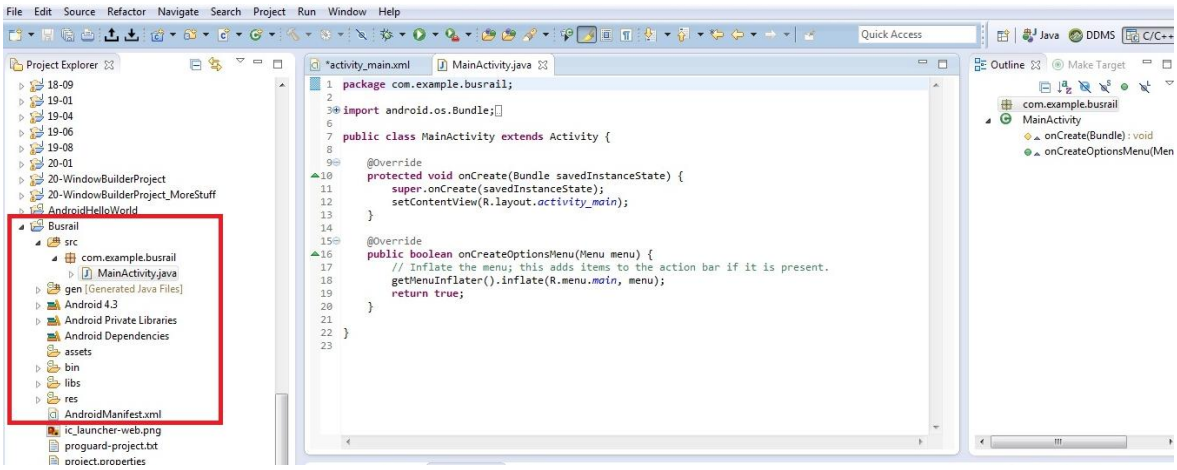
-res: είναι ο φάκελος, στον οποίο υπάρχουν όλα τα resources που χρησιμοποιεί η εφαρμογή μας. Με τον όρο resources μιας εφαρμογής εννοούμε τις διαστάσεις της, τα λεκτικά της, τα εικονίδια της κ.ά.

-res/drawable-xxxx: σε αυτούς τους φακέλους τοποθετούμε τα διάφορα εικονίδια και εικόνες που χρησιμοποιεί η εφαρμογή μας.

-res/layout: αυτός ο φάκελος περιέχει τα αρχεία XML που περιγράφουν τη μορφή μιας οθόνης GUI που χρησιμοποιεί η εφαρμογή.

-res/values: σε αυτούς τους φακέλους αποθηκεύονται πάλι αρχεία XML, στα οποία περιέχονται οι ορισμοί των λεκτικών, κωδικών χρωμάτων κτλ., που χρησιμοποιεί η εφαρμογή.

-AndroidManifest.xml: είναι το XML αρχείο που περιγράφει την εφαρμογή στο λειτουργικό σύστημα.



Εικόνα 2: Δομή ενός Android project

2.3 ΣΤΟΧΟΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Τώρα που είδαμε ποια είναι τα συστατικά μιας εφαρμογής Android, καθώς και τη δομή της, μπορούμε να δώσουμε μια αρχική περιγραφή για το τι θα κάνει αυτή η εφαρμογή. Ο σκοπός της εφαρμογής λοιπόν, είναι να λαμβάνει τη θέση στην οποία βρίσκεται ο χρήστης και να παρουσιάζει την πλησιέστερη στάση λεωφορείου. Είναι πολύ εύκολο να καταλάβουμε ότι ουσιαστικά σχεδόν ολόκληρη η εφαρμογή στηρίζεται στις υπηρεσίες βάσει τοποθεσίας, όπως αναφέραμε στην εισαγωγή. Σε αυτό το σημείο για να κατανοήσουμε καλύτερα το πώς θα λειτουργεί η εφαρμογή είναι απαραίτητο να εισάγουμε τον όρο Geocoding και Reverse geocoding.

2.4 GEOCODING ΚΑΙ REVERSE GEOCODING

Με τον όρο *geocoding* αναφερόμαστε στην διαδικασία αντιστοίχισης μιας διεύθυνσης στο ζεύγος γεωγραφικών συντεταγμένων όπου υπάγεται και που μπορεί να φανεί πολύ χρήσιμη στις εφαρμογές μας. Ο στόχος είναι να δοθεί μια λίστα με τις υποψήφιες θέσεις που είναι αρκετά κοντά στη γεωγραφική τοποθεσία, για την οποία έχει ζητηθεί να πραγματοποιηθεί η διαδικασία του geocoding. Κάτι πολύ σημαντικό που πρέπει να αναφέρουμε είναι ότι το geocoding μετατρέπει συνηθισμένες εγγραφές δεδομένων που περιέχουν κάποια διεύθυνση, σε γεωγραφικό αντικείμενο που μπορεί να προβληθεί στον χάρτη. Για να γίνει αυτή η διαδικασία απαιτείται να γίνει το ταίριασμα μιας διεύθυνσης στόχου ή της ονομασίας μιας περιοχής, με τα αρχεία μιας βάσης δεδομένων η οποία περιέχει διευθύνσεις και ονόματα σημείων. Το αποτέλεσμα αυτού του ταιριάσματος είναι η επιστροφή των συντεταγμένων που έχουν σχέση με τη διεύθυνση στόχο.

Συμπληρώνοντας όσα είπαμε πιο πάνω για τη δομή μιας εφαρμογής Android, επιβάλλεται να αναφέρω ότι για να μπορέσουμε να πραγματοποιήσουμε επιτυχώς τη διαδικασία του geocoding απαιτείται να ορίσουμε το κατάλληλο <users-permission> tag για τα δικαιώματα πρόσβασης στο αρχείο AndroidManifest.xml.

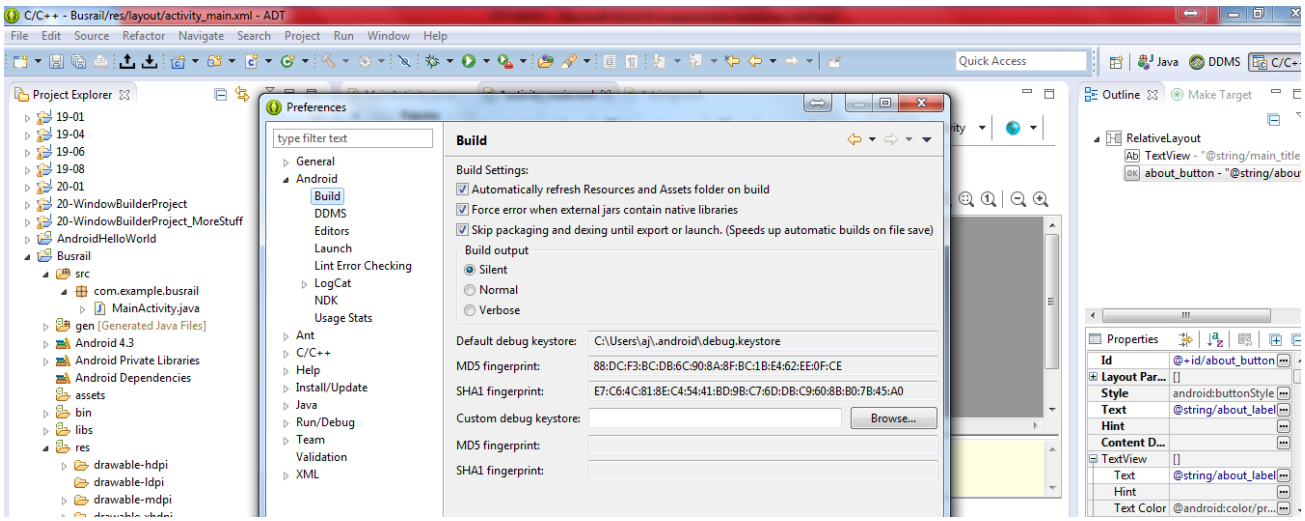
Πέρα από το geocoding θα αναφέρουμε και την αντίστροφη διαδικασία, την αντιστοιχία δηλαδή ενός ζεύγους γεωγραφικών συντεταγμένων σε μία διεύθυνση. Αυτό ονομάζεται *reverse geocoding* και πρόκειται για μία εξίσου χρήσιμη λειτουργία για τις εφαρμογές. Στο reverse geocoding η διεύθυνση μπορεί να βρεθεί με βάση τις συντεταγμένες της τοποθεσίας του αντικειμένου. Η υπηρεσία βασιζόμενη στη γεωγραφική θέση του χρήστη στέλνει query στην υπηρεσία θέσης για να μάθει για τις γεωγραφικές τοποθεσίες των στόχων της.

Αφού εξηγήσαμε τον στόχο της εφαρμογής μας πρέπει να συμπληρώσω ότι πέρα από το να βρίσκει ποια στάση είναι πιο κοντά στον χρήστη της συσκευής, θα μπορούσε να εμφανίζεται και στον χάρτη. Όπως ειπώθηκε και στην εισαγωγή λοιπόν, μπορούμε να ενσωματώσουμε την τεχνολογία Google Maps στις εφαρμογές μας είτε χρησιμοποιώντας το αντίστοιχο στοιχείο UI που είναι το Mapview, είτε χρησιμοποιώντας απευθείας το API. Για να λειτουργήσει σωστά το Google Maps API απαιτείται η χρήση ενός έγκυρου κλειδιού το οποίο μπορούμε να προμηθευτούμε από την Google. Η διαδικασία ενός έγκυρου κλειδιού περιγράφεται λεπτομερώς στη σελίδα που βρίσκεται στη διεύθυνση:

- **https://developers.google.com/maps/documentation/android/start#obtain_a_google_maps_api_key**

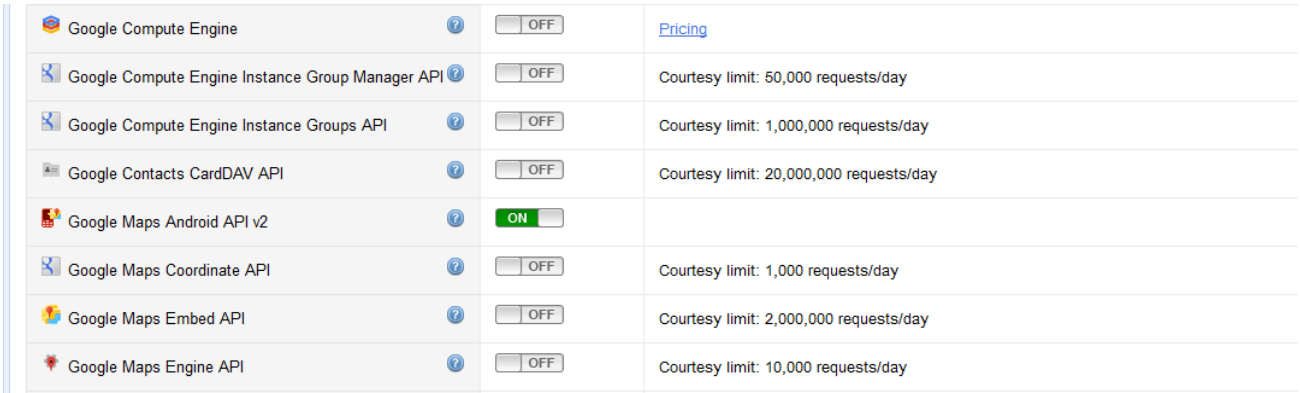
Για να ολοκληρωθεί επιτυχώς η διαδικασία, θα πρέπει πρώτα να δημιουργήσουμε ένα MD5 πιστοποιητικό αποτύπωμα για το κλειδί που χρησιμοποιούμε για την υπογραφή των εφαρμογών μας. Ευτυχώς για μας, το Eclipse έχει δημιουργήσει αυτόματα και χρησιμοποιεί ένα default κλειδί, ώστε να μπορούμε να εκτελούμε και να δοκιμάσουμε τις εφαρμογές μας κατά την υλοποίησή τους στο περιβάλλον. Μπορούμε να βρούμε σε ποια τοποθεσία βρίσκεται αυτό το default κλειδί στον υπολογιστή μας επιλέγοντας **Window -> Preferences** μέσα από το Eclipse. Στο παράθυρο που θα ανοίξει αναπτύσσουμε την επιλογή Android και επιλέγουμε το Build. Στην εικόνα παρακάτω

βλέπουμε στο πεδίο Default debug keystore τη διαδρομή στον υπολογιστή μας όπου βρίσκεται το default κλειδί που χρησιμοποιεί το Eclipse, το MD5 fingerprint του κλειδιού, αλλά και το SHA1 fingerprint. Αυτό το τονίζουμε καθώς είναι αυτό το οποίο ζητείται από την Google πλέον για τη δημιουργία του API κλειδιού.



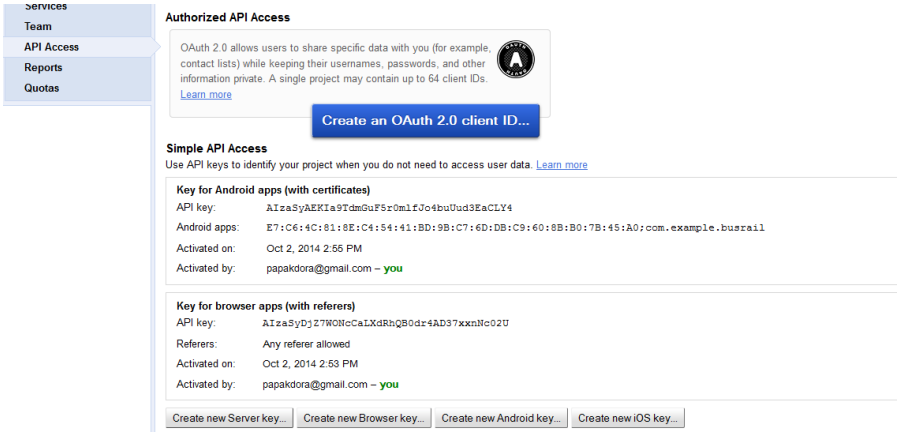
Εικόνα 3: Το SHA1 fingerprint του project

Στη συνέχεια ενεργοποιούμε την τωρινή έκδοση που χρησιμοποιείται για το Android, το Google Maps Android API v2.



Εικόνα 4:Ενεργοποίηση των Google Maps Android API v2

Στην καρτέλα API Access του API Project δημιουργώ το νέο κλειδί λοιπόν, χρησιμοποιώντας το SHA1 fingerprint που είδαμε πιο πάνω.



Εικόνα 5: Δημιουργία API κλειδιού

Τέλος, το μόνο που μένει να κάνουμε είναι το να δηλώσουμε το API κλειδί μας στο αρχείο AndroidManifest.xml, ώστε να υπάρχουν όλα τα δικαιώματα πρόσβασης.

2.5 ΣΧΕΔΙΑΣΗ ΚΑΙ ΜΕΛΕΤΗ ΕΦΑΡΜΟΓΗΣ

Αφού είδαμε λοιπόν και τον τρόπο με τον οποίο θα εμφανίζονται οι χάρτες στην εφαρμογή μας σε περίπτωση που το θέλαμε, μένει να κατανοήσουμε τον τρόπο, με τον οποίο θα γίνεται η επιλογή της στάσης που είναι πιο κοντά στον χρήστη.

Για να επιτύχουμε τον συγκεκριμένο σκοπό θα δημιουργήσουμε μία βάση δεδομένων η οποία θα περιέχει έναν πίνακα. Στην αρχή θα ξεκινήσουμε με μία ολοκληρωμένη γραμμή, σε μελλοντική επέκταση θα μπορούν να προστεθούν και άλλες. Σε αυτό το σημείο πρέπει να αναφέρω το πώς θα λειτουργεί η βάση δεδομένων στην εφαρμογή μου. Η βάση αυτή δημιουργήθηκε και βρίσκεται σε έναν web server με domain www.somewherewarm.net . Στο κομμάτι της υλοποίησης θα αναφερθώ και στον τρόπο με τον οποίο επικοινωνεί με την εφαρμογή μου. Το να ανεβάσουμε μια βάση δεδομένων σε έναν web server ίσως δεν είναι πάντα η ιδανική λύση. Υπάρχουν τα θετικά και τα αρνητικά της συγκεκριμένης επιλογής. Το πιο σημαντικό πλεονέκτημα είναι ότι σε μία εμπορική εφαρμογή που πρέπει να λειτουργεί σε real-time μπορεί να παρέχει δυναμική λειτουργία στη βάση δεδομένων. Για να το καταλάβουμε καλύτερα αυτό ακολουθεί ένα παράδειγμα ακριβώς στον τύπο της εφαρμογής που θα πραγματοποιήσω. Στην εφαρμογή μου λοιπόν που θέλω να βρίσκει την κοντινότερη στάση στον χρήστη, αν μία στάση δεν λειτουργεί προσωρινά λόγω έργων π.χ. (καθόλου ασυνήθιστο…) ενημερώνεται η βάση από τον server και το βλέπουν όλοι οι χρήστες της εφαρμογής κανονικά. Σε περίπτωση που η βάση βρισκόταν μέσα στην εφαρμογή, θα έπρεπε ο προγραμματιστής να μπει και να αλλάξει τον κώδικα που έχει γράψει και αφού τελειώσει, να ανεβάσει μια βελτιωμένη έκδοσή της στο Google Play Store ώστε να την αναβαθμίσουν οι χρήστες της. Πολύ πιο χρονοβόρο και δύσκολο σαν διαδικασία. Το μόνο αρνητικό της επιλογής που έχω κάνει είναι ότι η εφαρμογή δεν λειτουργεί σωστά χωρίς σύνδεση στο Internet. Παρόλα αυτά θεωρώ ότι είναι ελάχιστες οι τοποθεσίες στην Ελλάδα που δεν υπάρχει κάλυψη στο δίκτυο (μέσω 3G

ή και 4G εσχάτως) ή δεν υπάρχει κάποιο ασύρματο δίκτυο. Ακόμα και σε αυτή την περίπτωση όμως θα δούμε παρακάτω ότι έχω χρησιμοποιήσει έναν πολύ ενδιαφέρον τρόπο ώστε να εισάγω δεδομένα στην εφαρμογή.

Για την κατασκευή αυτής της βάσης δεδομένων θα χρησιμοποιηθεί η MySQL. Στις στήλες του πίνακα θα έχουμε σαν id του πίνακα τον αριθμό της στάσης ξεκινώντας από το 1. Στη δεύτερη στήλη του πίνακα θα καταχωρηθεί το όνομα της στάσης. Στην τρίτη στήλη θα εμφανίζεται η γραμμή στην οποία ανήκει η εκάστοτε στάση. Στην τέταρτη και πέμπτη στήλη θα καταχωρηθούν το γεωγραφικό μήκος και το γεωγραφικό πλάτος της κάθε στάσης. Οπότε ο πίνακας που θα παραχθεί θα είναι αυτής της μορφής:

id	Name	Grammi	Latitude	Longitude
1	Afetiria	622	37.982197	23.775014
2	Grammou	622	37.982476	23.772590
3	Nosok. Paidwn	622	37.983896	23.768467
4	Laiko	622	37.983194	23.765184

Όπως γίνεται εύκολα αντιληπτό ο λόγος για τον οποίο καταχωρούμε τα ζεύγη των συντεταγμένων του κάθε σημείου, είναι για να τα συγκρίνουμε με το ζεύγος συντεταγμένων του χρήστη της εφαρμογής. Όταν δηλαδή θα ανοίγει την εφαρμογή ο χρήστης, μέσω του GPS του κινητού του, χρησιμοποιώντας τη διαδικασία που είδαμε πιο πάνω θα δίνει το ζεύγος συντεταγμένων της θέσης του στην εφαρμογή.

Για τη σύγκριση αυτή λοιπόν θα χρειαστεί να χρησιμοποιήσουμε έναν μαθηματικό τύπο που θα μας δίνει την απόσταση που θα έχει το κάθε σημείο από τη θέση του χρήστη. Για αυτόν τον τύπο θα χρειαστεί να ανατρέξουμε στις βασικές αρχές της γεωμετρίας. Οπότε έχουμε τα εξής:

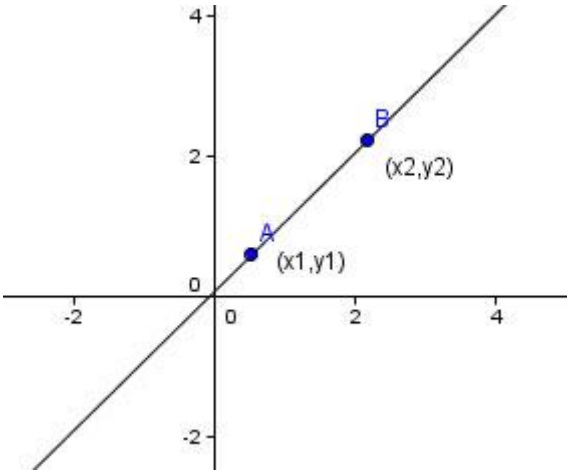
Από την Ευκλείδεια Γεωμετρία γνωρίζουμε ότι η απόσταση μεταξύ δύο σημείων (x1) και (x2) είναι το μήκος του ευθύγραμμου τμήματος που τα συνδέει:

$$d = \sqrt{(\Delta x)^2} = \sqrt{(x_2 - x_1)^2}.$$

Ο τύπος της απόστασης στο Καρτεσιανό σύστημα συντεταγμένων βασίζεται στο Πυθαγόρειο Θεώρημα. Αν (x1, y1) και (x2, y2) είναι σημεία του επιπέδου, τότε η απόσταση μεταξύ τους, που ονομάζεται και ευκλείδεια απόσταση, δίνεται από τον τύπο:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Το βλέπουμε και σχηματικά:



Εικόνα 6: Παράδειγμα διαφοράς απόστασης δύο σημείων

Οπότε στην Αναλυτική Γεωμετρία η απόσταση δύο σημείων που ανήκουν στο Καρτεσιανό σύστημα συντεταγμένων μπορεί να βρεθεί χρησιμοποιώντας τον τύπο της απόστασης. Η απόσταση μεταξύ των σημείων (x1, y1) και (x2, y2) δίνεται από:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

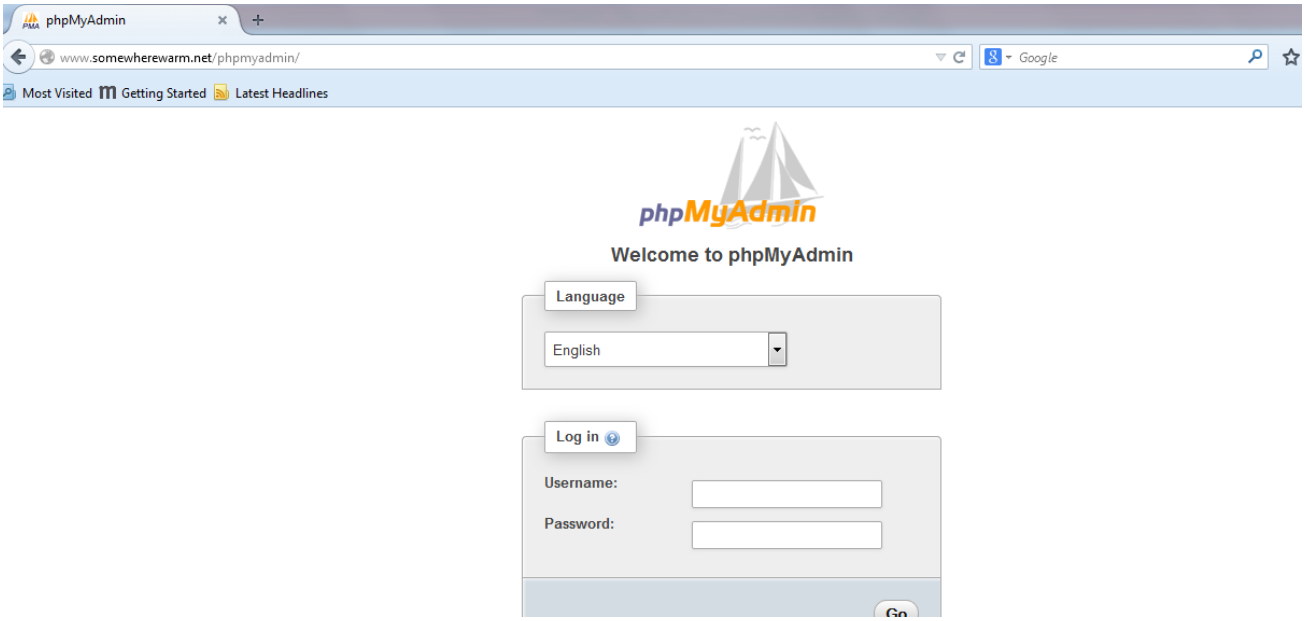
Αναλύσαμε σε πρώτη φάση ό, τι χρειάζεται να γνωρίζουμε σε θεωρητικό επίπεδο για να προχωρήσουμε στην υλοποίηση της εφαρμογής. Τώρα θα περάσουμε στο δεύτερο κομμάτι, και το πιο ενδιαφέρον της εργασίας, όπου θα αναλυθούν όλες οι κινήσεις βήμα-βήμα.

ΚΕΦΑΛΑΙΟ 3 : ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

3.1 ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ (SERVER)

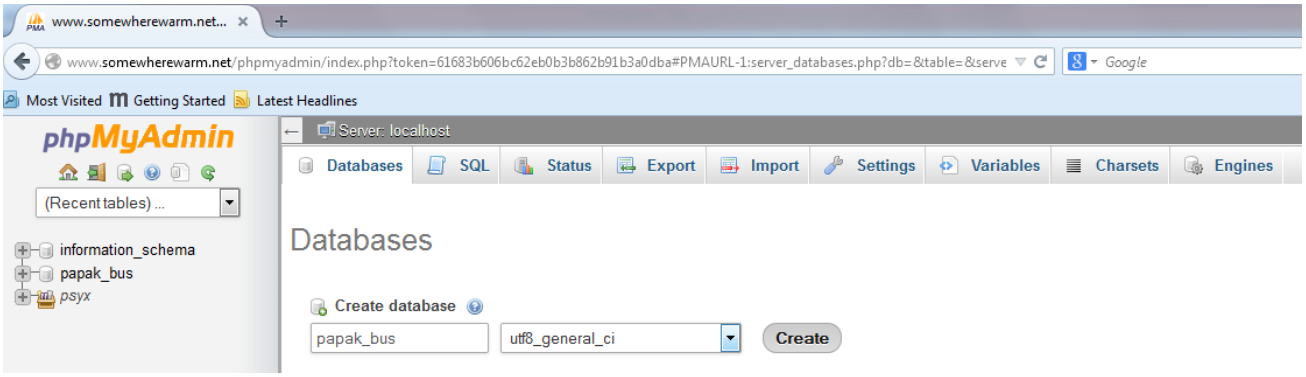
Σε πρώτη φάση θα δούμε τον τρόπο, με τον οποίο κατασκευάστηκε η βάση δεδομένων, που χρησιμοποιεί η εφαρμογή μου. Όπως προαναφέρθηκε, βρίσκεται στον web server www.somewherewarm.net οπότε για τη δημιουργία της βάσης και του πίνακα που θα περιέχει αυτή,

εισάγουμε σαν προορισμό στον Browser τη διεύθυνση www.somewherewarm.net/phpmyadmin και εισάγουμε το username και password που χρησιμοποιεί ο server στον οποίο φιλοξενούμαστε.



Εικόνα 7: Είσοδος στον web server

Δημιουργούμε τη βάση με την οποία θα εργαστούμε με τον εξής τρόπο:

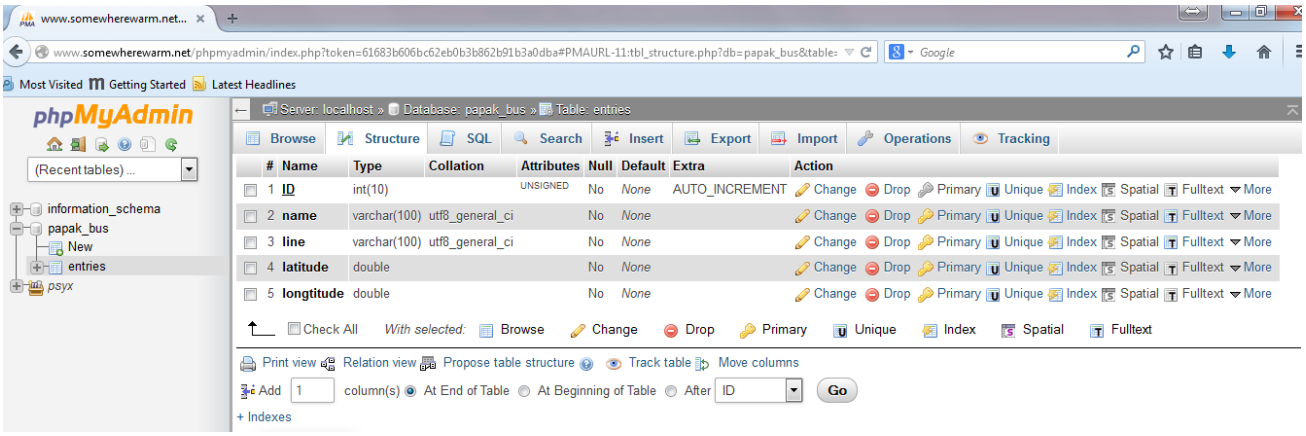


Εικόνα 8: Δημιουργία της βάσης papak_bus

Αφού δημιουργήσουμε τη βάση λοιπόν, προχωράμε στη δημιουργία του πίνακα δίνοντας το όνομά του και τον αριθμό των στηλών που θα έχει:

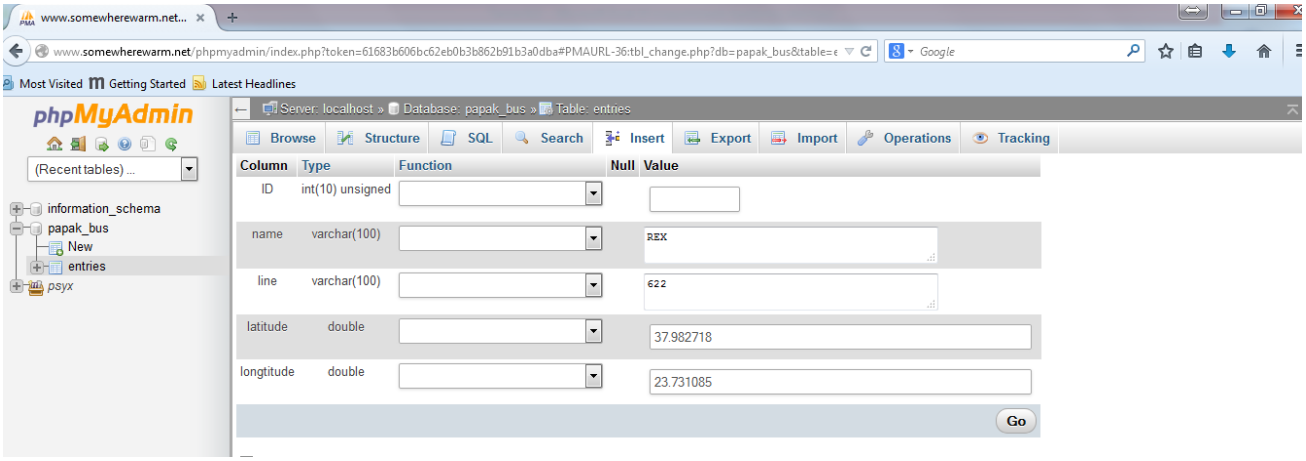
Εικόνα 9: Δημιουργία πίνακα

Στη συνέχεια δηλώνουμε τις στήλες του πίνακα. Την πρώτη την οποία χρησιμοποιούμε και σαν primary την ονομάζουμε ID και είναι τύπου int (ακέραιος) με την επιλογή A_I (Auto Increment) τσεκαρισμένη για να γεμίζει μόνη της μετά από κάθε καταχώρησή μας. Τη δεύτερη την ονομάζουμε name, είναι το όνομα της κάθε στάσης, και είναι τύπου VARCHAR με μέγεθος (100). Η τρίτη στήλη είναι ο αριθμός της γραμμής και λέγεται line. Ορίζεται και αυτή σαν VARCHAR (100). Η τέταρτη και πέμπτη στήλη είναι το γεωγραφικό μήκος και πλάτος αντίστοιχα. Ορίζονται σαν double και είναι οι συντεταγμένες της κάθε στάσης. Τελειώνοντας, βλέπουμε τη δομή του πίνακα που δημιουργήσαμε:



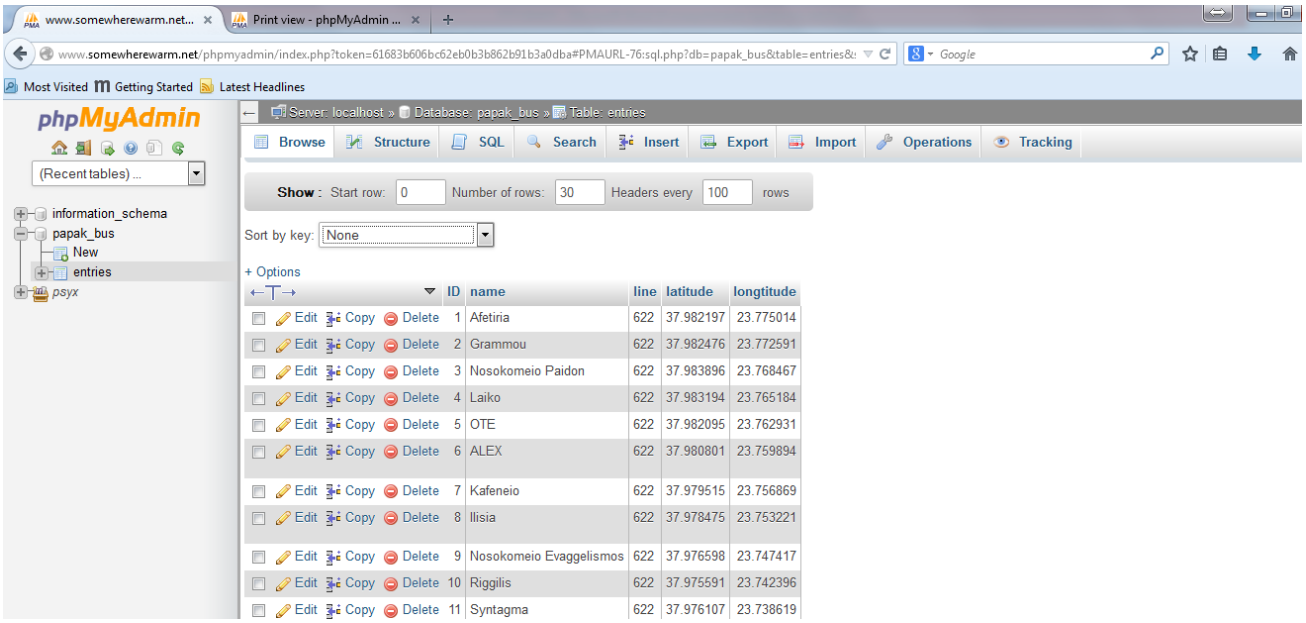
Εικόνα 10: Δομή πίνακα entries

Πάμε τώρα να δούμε πως καταχωρούμε τα στοιχεία στον πίνακα:



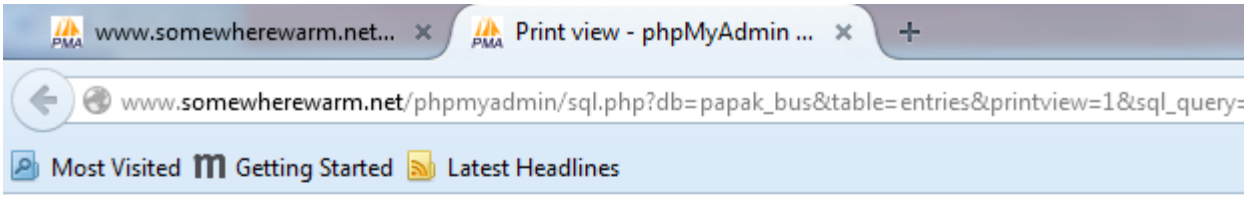
Εικόνα 11: Καταχώρηση Δεδομένων στον πίνακα

Αυτό που κάνουμε είναι να γεμίζουμε τα value της κάθε στήλης με τα ανάλογα δεδομένα. Π.χ. εδώ έχω βάλει στο όνομα της στάσης το REX, στη γραμμή 622 και τις συντεταγμένες της συγκεκριμένης στάσης. Τις συντεταγμένες που χρησιμοποιώ τις έχω βρει από το google maps αναζητώντας το γεωγραφικό σημείο για κάθε στάση ξεχωριστά. Με ανάλογο τρόπο γεμίζουμε ολόκληρο τον πίνακα. Τελειώνοντας με την καταχώρηση των δεδομένων μπορούμε να δούμε τα δεδομένα του πίνακα από το tab Browse:



Εικόνα 12: Περιήγηση στον πίνακα entries

Επίσης μπορούμε να δούμε όλον τον πίνακα κάνοντας ένα Print view (with full texts):



SQL result

Host: localhost
Database: papak_bus
Generation Time: Oct 06, 2014 at 05:09 PM
Generated by: phpMyAdmin 4.0.10deb1 / MySQL 5.5.38-0ubuntu0.14.04.1
SQL query: SELECT * FROM `entries`;
Rows: 32

ID	name	line	latitude	longtitude
1	Afetiria	622	37.982197	23.775014
2	Grammou	622	37.982476	23.772591
3	Nosokomeio Paidon	622	37.983896	23.768467
4	Laiko	622	37.983194	23.765184
5	OTE	622	37.982095	23.762931
6	ALEX	622	37.980801	23.759894
7	Kafeneio	622	37.979515	23.756869
8	Ilisia	622	37.978475	23.753221
9	Nosokomeio Evaggelismos	622	37.976598	23.747417
10	Riggilis	622	37.975591	23.742396
11	Syntagma	622	37.976107	23.738619
12	Akadhmia Athinwn	622	38.980012	23.733434
13	REX	622	37.982718	23.731085
14	Plateia Lavriou	622	37.985653	23.728478
15	Polytexneio	622	37.988519	23.730076
16	Mouseio	622	37.990371	23.731181
17	OTE Pediou Areos	622	37.993998	23.731997
18	Aggelopoulou	622	37.997084	23.732801

19	Kefallinias	622	37.998725	23.733123
20	Kallifrona	622	38.002935	23.735559
21	Lyssiatreio	622	38.006113	23.736095
22	IKA Patision	622	38.007973	23.736095
23	Koliatsou	622	38.009309	23.736234
24	Taigetou	622	38.009495	23.739711
25	Ipeirou	622	38.009723	23.744345
26	Sxoleia	622	38.008379	23.746212
27	1h Ag. Glykerias	622	38.010181	23.748046
28	2h Ag. Glykerias	622	38.010814	23.750181
29	Agora	622	38.011583	23.752617
30	Ydras	622	38.011143	23.756061
31	IKA	622	38.009385	23.758421
32	Terma 622	622	38.008328	23.758925

Print

Εικόνα 13: Print view του πίνακα entries

Έτσι λοιπόν έχουμε τελειώσει με τη δημιουργία της βάσης και του πίνακα μας. Πάμε να δούμε λοιπόν τα αρχεία PHP τα οποία έχουν τον έλεγχο της βάσης αυτής.

Έχω δημιουργήσει ένα αρχείο το οποίο λέγεται core.php και περιέχει βασικές συναρτήσεις για την προσπέλαση και επεξεργασία των δεδομένων της βάσης μας.

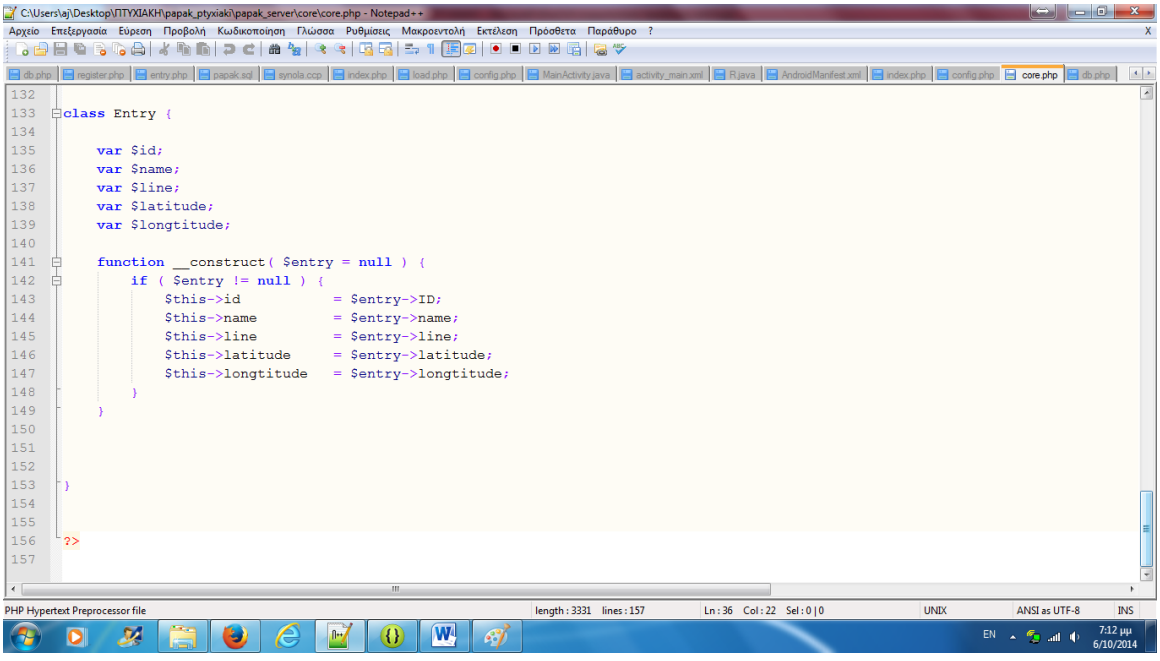
Ακολουθεί η συνάρτηση που έχουμε ορίσει για να διαβάζει τις τιμές από τον πίνακα entries και να τις μεταφέρει σε μία global μεταβλητή.

```php
<?php

function get_entry( $id ) {
    global $db;
    $entry = $db->get_row( $db->prepare( "SELECT * FROM entries WHERE ID = %d LIMIT 1", $id ) );
    $_entry = new Entry( $entry );
    $GLOBALS['entry'] = $_entry;
    return $_entry;
}
```

Η παρακάτω συνάρτηση που έχουμε ορίσει χρησιμοποιείται για να μπαίνει στον πίνακα entries που έχουμε ήδη δημιουργήσει και να τραβάει τις τιμές που παίρνει από τη στήλη line (τη στήλη που είναι το όνομα της γραμμής δηλαδή) και να τις τοποθετεί σε ένα νέο πίνακα.

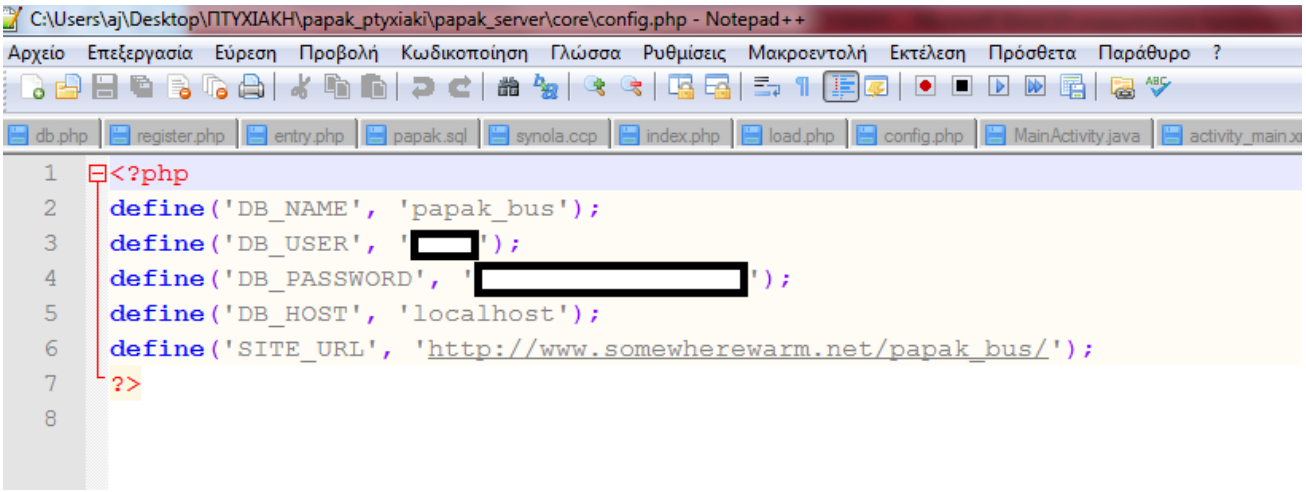
```php
function get_entries_by_line( $value, $max = 1000 ) {
    global $db;

    $num_entries = 0;

    if ( empty( $value ) ) {
        $num_entries = $db->query( $db->prepare( "SELECT * FROM entries LIMIT %d", $max ) );
    } else {
        $num_entries = $db->query( $db->prepare( "SELECT * FROM entries WHERE line = %s LIMIT %d", $value, $max ) );
    }

    $entries = array();

    for ( $i = 0; $i < $num_entries; $i++ ) {
        $entries[] = new Entry( $db->last_result[$i] );
    }

    return $entries;
}
```

Επίσης έχουμε δημιουργήσει μια κλάση στην οποία αποκτούμε τα δεδομένα από τη μεταβλητή entry (από την πρώτη συνάρτηση που είχαμε ορίσει) και σε περίπτωση που δεν είναι μηδενική η τιμή, αποθηκεύει τις τιμές από τα αντίστοιχα πεδία του πίνακα στις μεταβλητές (αυτές φέρουν το όνομα των στηλών), ώστε να είναι επεξεργάσιμες αργότερα.

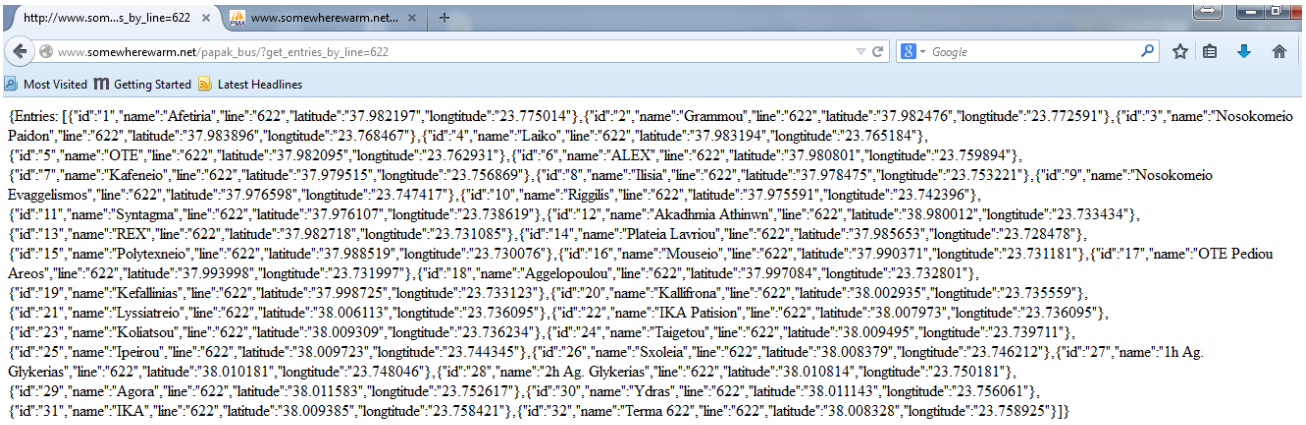


Πέρα από το αρχείο core.php υπάρχει ένα αρχείο config.php με το οποίο ουσιαστικά καταχωρούμε τα στοιχεία της βάσης δεδομένων ώστε να είναι προσβάσιμα. Τα στοιχεία που έχουμε δώσει είναι

το όνομα της βάσης δεδομένων μας, το όνομα χρήστη, τον κωδικό (αυτά που είχαμε εισάγει στην αρχή όταν δημιουργούσαμε τη βάση και τον πίνακα), το localhost και τέλος το url της βάσης μας.



Τώρα που είδαμε κάποιες συναρτήσεις που έχουμε ορίσει κάνουμε έναν τυπικό έλεγχο στον πίνακα που έχουμε φτιάξει με ένα request στον web server κάνοντας χρήση της get_entries_by_line με σκοπό να τραβήξουμε τις στάσεις της γραμμής 622. Αυτό το κάνουμε για να δούμε ότι έχουν καταχωρηθεί όλα σωστά (χωρίς κενά). Δίνουμε στο web browser μας σαν προορισμό το: www.somewherewarm.net/papak_bus/?get_entries_by_line=622 και έχουμε:



Εικόνα 14: Περιεχόμενα πίνακα entries με χρήση της get_entries_by_line

Ένα εξίσου σημαντικό αρχείο παρόλα αυτά είναι αυτό που ακολουθεί. Στο αρχείο db.php έχουμε αρκετές συναρτήσεις οι οποίες χρησιμοποιούνται διαρκώς. Μία από αυτές είναι η συνάρτηση connect, η οποία είναι αυτή η οποία παίρνει σαν ορίσματα τα στοιχεία που δηλώσαμε στο αρχείο

config.php και πραγματοποιεί τη σύνδεση με τον server. Σε περίπτωση μη σύνδεσης υπάρχει ένα μήνυμα λάθους.

```
93      function connect() {
94
95          $this->dbh = @mysql_connect( $this->dbhost, $this->dbuser, $this->dbpassword, true );
96
97          if ( ! $this->dbh ) {
98
99              echo 'Could not connect to server!</br>';
100             return;
101         }
102
103         $this->set_charset( $this->dbh );
104         $this->ready = true;
105         $this->select( $this->dbname, $this->dbh );
106     }
```

Η συνάρτηση select έχει παρόμοια λειτουργία με την connect. Απλά η συγκεκριμένη συνάρτηση ελέγχει αν έχει πραγματοποιηθεί η σύνδεση με τη βάση δεδομένων. Και αυτή σε περίπτωση λάθους κατά τη διάρκεια της σύνδεσης εκτυπώνει ένα μήνυμα για το σφάλμα.

```
function select( $db, $dbh = null ) {
    if ( is_null($dbh) )
        $dbh = $this->dbh;

    if ( ! @mysql_select_db( $db, $dbh ) ) {
        $this->ready = false;
        echo 'Could not connect to database!</br>';
        return;
    }

}
```

Στο αρχείο υπάρχουν και δύο μικρές συναρτήσεις οι οποίες φαίνονται ασήμαντες, όμως στην πραγματικότητα είναι αρκετά σημαντικές. Η has_cap και η db_version. Η db_version μας επιστρέφει την έκδοση του database server που χρησιμοποιούμε την παρούσα χρονική στιγμή. Η has_cap παίρνει το αποτέλεσμα της db_version και προσαρμόζει το σετ χαρακτήρων για την επικοινωνία με τον database server (είτε στέλνουμε δεδομένα, είτε παίρνουμε) στο default.

```php
    function has_cap( $db_cap ) {
        $version = $this->db_version();

        switch ( strtolower( $db_cap ) ) {
            case 'collation' :    // @since 2.5.0
            case 'group_concat' : // @since 2.7
            case 'subqueries' :   // @since 2.7
                return version_compare( $version, '4.1', '>=' );
            case 'set_charset' :
                return version_compare($version, '5.0.7', '>=');
        };

        return false;
    }


    function db_version() {
        return preg_replace( '/[^0-9.].*/', '', mysql_get_server_info( $this->dbh ) );
    }
```

Επίσης μια πολύ βασική συνάρτηση είναι η insert, η οποία μας εξυπηρετεί στο να εισάγουμε δεδομένα στον εκάστοτε πίνακα.

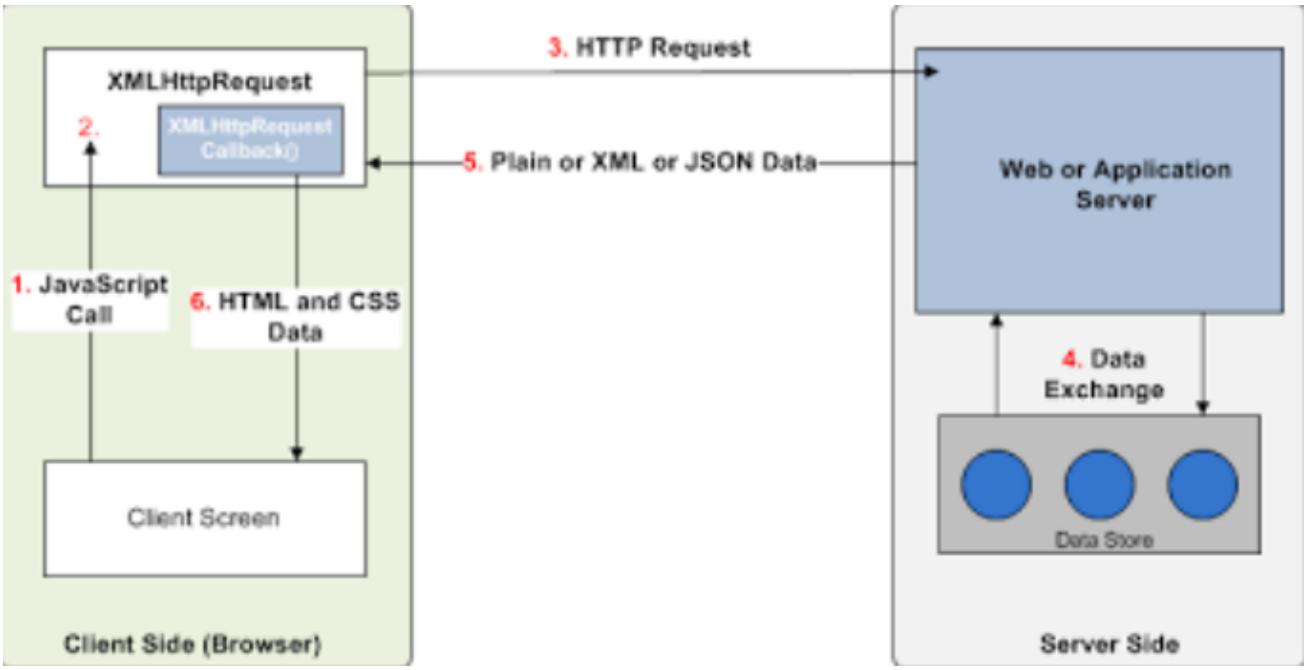
```php
function insert( $table, $data, $format = null, $type = 'INSERT' ) {
    if ( ! in_array( strtoupper( $type ), array( 'REPLACE', 'INSERT' ) ) )
        return false;
    $formats = $format = (array) $format;
    $fields = array_keys( $data );
    $formatted_fields = array();
    foreach ( $fields as $field ) {
        if ( !empty( $format ) )
            $form = ( $form = array_shift( $formats ) ) ? $form : $format[0];
        elseif ( isset( $this->field_types[$field] ) )
            $form = $this->field_types[$field];
        else
            $form = '%s';
        $formatted_fields[] = $form;
    }
    $sql = "{$type} INTO `$table` (`" . implode( '`,`', $fields ) . "`) VALUES ('" . implode( "','", $formatted_fields ) . "')";
    return $this->query( $this->prepare( $sql, $data ) );
}

```

Το τέταρτο αρχείο load.php είναι ένα πιο απλό αρχείο σε σχέση με τα υπόλοιπα τρία. Δεν περιέχει κάποια συνάρτηση αλλά περιέχει την εντολή require_once για τα άλλα τρία αρχεία. Το require_once είναι ταυτόσημο με το require. Η μόνη διαφορά είναι ότι η PHP θα ελέγξει αν τα αρχεία έχουν περιληφθεί ήδη και , αν δεν έχουν περιληφθεί, τότε θα τα απαιτήσει πάλι.

```php
<?php
require_once('config.php');
require_once('db.php');
require_once('core.php');
?>
```

Ανακεφαλαιώνοντας, αυτά τα τέσσερα αρχεία είναι αυτά που μας προσφέρουν όλα τα απαραίτητα εργαλεία για την προσπέλαση και επεξεργασία των δεδομένων της βάσης μας.

Ανέφερα παραπάνω σε ανύποπτο χρόνο μια λειτουργία χωρίς να εξηγήσω τη μέθοδο, με την οποία την πραγματοποίησα. Αυτή η μέθοδος που θα αναλύσω παρακάτω είναι η γέφυρα της βάσης δεδομένων που βρίσκεται στον web server που με φιλοξενεί, με την android εφαρμογή που θα υλοποιήσω. Μέχρι τώρα όλα τα αρχεία που έχω χρησιμοποιήσει είναι αρχεία PHP, εγώ αυτό που θέλω είναι να μετατρέψω τα δεδομένα της βάσης μου, σε δεδομένα τα οποία μπορούν να επεξεργαστούν από το Eclipse, δηλαδή σε Java. Πρέπει δηλαδή να εξάγω τα δεδομένα από τη βάση μου σε τέτοια μορφή που θα είναι διαχειρίσιμη από το Eclipse. Η διαδικασία ξεκινάει με ένα http request του client (ο browser που χρησιμοποιούμε) στον server (ο server που βρίσκεται η βάση δεδομένων μας). Αυτό το http request γίνεται με τη μέθοδο GET, με την οποία ζητάμε δεδομένα από μία συγκεκριμένη πηγή. Ο server βρίσκει τα δεδομένα που του έχουμε ζητήσει από τη βάση και μπαίνει στη διαδικασία να μας τα επιστρέψει. Τα δεδομένα που θα μας επιστρέψει λοιπόν θα είναι είτε σε XML, είτε σε JSON data (JavaScript Object Notation). Τα JSON data είναι μια πολύ ελαφριά, βασισμένη σε κείμενο, μορφή ανταλλαγής δεδομένων ανάμεσα στον client και στον server. Σχηματικά όλο αυτό που περιέγραψα πιο πάνω δίνεται ως εξής:



Εικόνα 15: Επικοινωνία εφαρμογής με τον web server

Αφού εξηγήσαμε τη μέθοδο λοιπόν, με την οποία θα γίνει η παραλαβή των δεδομένων από τη βάση δεδομένων στον web server, ας δούμε και το αρχείο που χρησιμοποιούμε για να κάνουμε το HTTP Request:

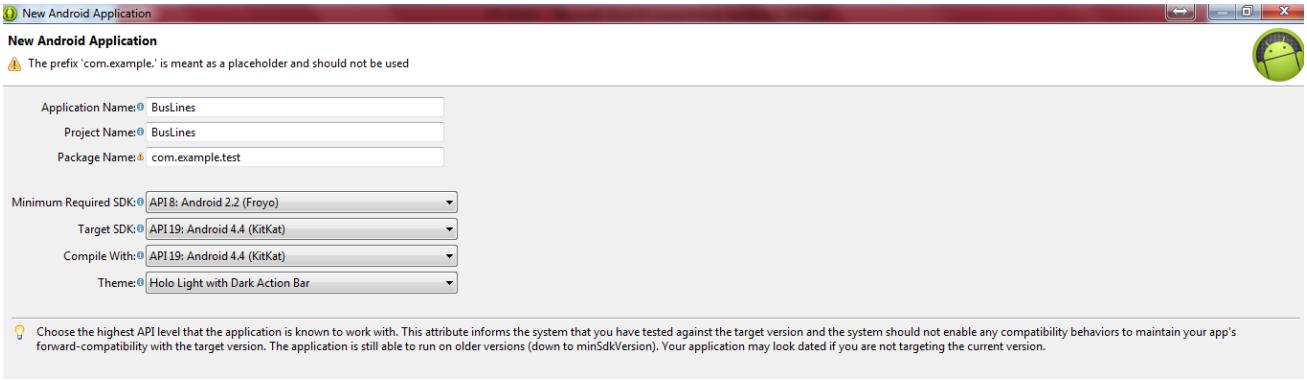
```php
<?php
require_once('./core/load.php');


if ( isset( $_REQUEST[ 'get_entries_by_line' ] ) ) {

    $line = $_REQUEST[ 'get_entries_by_line' ];

    $entries = get_entries_by_line( $line );

    echo '{Entries: ', json_encode( $entries ), '}'; die;
}


?>

```

Πάλι στην έναρξη χρησιμοποιούμε την require_once και το αρχείο που κάνουμε require είναι το load.php το οποίο περιέχει τα require_once για τα άλλα τρία προηγούμενα PHP αρχεία. Στη συνέχεια ορίζουμε το είδος του request. Εδώ χρησιμοποιούμε την get_entries_by_line και υποδεικνύουμε ότι τα δεδομένα που θα εξαχθούν από τον πίνακα θα είναι φιλτραρισμένα με τον αριθμό που θα βρίσκεται στη στήλη της γραμμής (δηλαδή να πάρει όλα τα δεδομένα που έχουν τον αριθμό 622, για παράδειγμα, στη στήλη line του πίνακα). Τέλος, γίνεται χρήση όπως βλέπουμε της συνάρτησης json_encode(). Αυτή η συνάρτηση χρησιμοποιείται για το serialization ενός αντικειμένου που να υποστηρίζει JSON. Serialization στην επιστήμη των υπολογιστών είναι η μετάφραση δομών δεδομένων και ορισμών αντικειμένων σε μορφές, οι οποίες μπορούν να αποθηκευτούν και να μορφοποιηθούν αργότερα από το συγκεκριμένο ή από κάποιο άλλο υπολογιστικό περιβάλλον.

3.2 ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ (CLIENT)

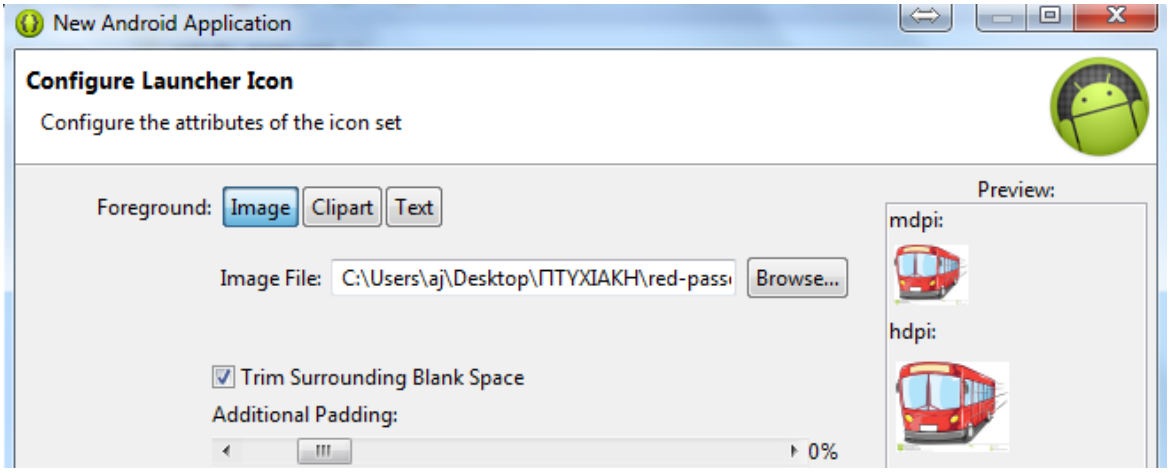
Σε αυτό το σημείο τελειώσαμε με το κομμάτι της υλοποίησης της εφαρμογής που αφορά τον server. Προχωράμε στο δεύτερο και εξίσου ενδιαφέρον κομμάτι της υλοποίησης, που έχει να κάνει με την υλοποίηση της android εφαρμογής στο Eclipse.

Ξεκινάμε δημιουργώντας ένα νέο Android Application Project με το όνομα της εφαρμογής να είναι BusLines και με όνομα πακέτου com.example.test



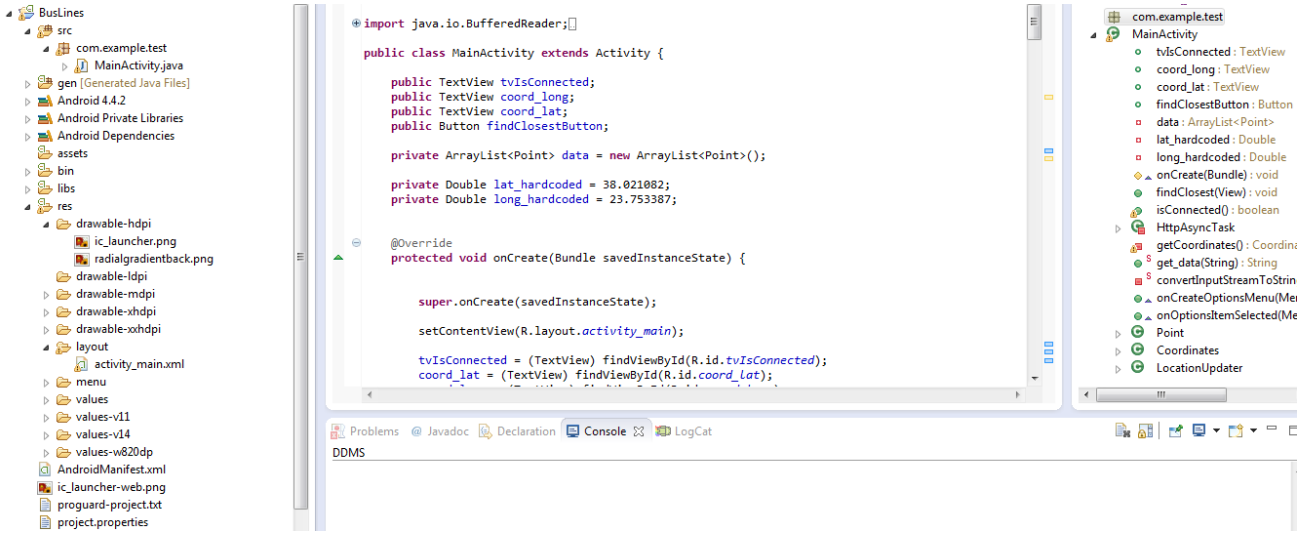
Εικόνα 16: Δημιουργία ενός νέου Android project

Στη συνέχεια προσθέτουμε το εικονίδιο που θέλουμε να έχει η εφαρμογή μας όταν εμφανίζεται στην οθόνη εφαρμογών του κινητού που τη χρησιμοποιεί.



Εικόνα 17:Εικονίδιο του Android project

Η μορφή της εφαρμογής μας είναι και η default μορφή μιας Android εφαρμογής.



Εικόνα 18: Δομή της εφαρμογής BusLines

Στον φάκελο src υπάρχει το MainActivity.java (μέσα στο πακέτο com.example.test) που είναι και το αρχείο που περιέχει τον πηγαίο κώδικα. Στον φάκελο res/drawable-hdpi βρίσκονται οι εικόνες που χρησιμοποιεί η εφαρμογή, η πρώτη είναι η εικόνα που επιλέξαμε πιο πάνω σαν εικονίδιο της εφαρμογής μας και η δεύτερη είναι η εικόνα που χρησιμοποιείται σαν background στην εφαρμογή. Γενικά οποιαδήποτε άλλη εικόνα θέλαμε να χρησιμοποιήσουμε θα έπρεπε να την κάνουμε Import στον συγκεκριμένο φάκελο ώστε να μπορεί να τη βλέπει η εφαρμογή μας και να τη χρησιμοποιεί. Στο φάκελο res/layout υπάρχει το αρχείο activity_main.xml. Ας δούμε αναλυτικά τι κάνει το κάθε κομμάτι του.

Το RelativeLayout είναι ο τύπος του Layout που χρησιμοποιούμε. Είναι υπεύθυνο για το πως θα κατανέμεται ο χώρος στην οθόνη. Περιέχει μια ομάδα views τα οποία εμφανίζονται σε σχετικές θέσεις κατά τη διάρκεια της λειτουργίας της εφαρμογής. Η κατανομή των views γίνεται είτε βάσει της μεταξύ τους σχέσης (το ένα αριστερά από το άλλο για παράδειγμα), είτε βάσει της συνολικής περιοχής της οθόνης (κάτω στην οθόνη ή στο κέντρο της). Στην τελευταία σειρά του RelativeLayout υπάρχει η εντολή με την οποία υποδεικνύουμε ποια εικόνα θα χρησιμοποιήσουμε σαν background image στην εφαρμογή μας με την ακριβή τοποθεσία και ονομασία της (χωρίς την κατάληξη του αρχείου όμως).

```
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.test.MainActivity"
    android:background="@drawable/radialgradientback">
```

Το στοιχείο TextView είναι αυτό που ορίζει ακριβώς το όνομά του. Είναι εκείνο το στοιχείο λοιπόν που είναι υπεύθυνο για την εμφάνιση ενός πεδίου που περιέχει κείμενο. Στην πρώτη εντολή βλέπουμε τη σύνδεση του πεδίου με το τι θα εμφανίζει ακριβώς (στη συγκεκριμένη περίπτωση δηλαδή θα εμφανίζει το κείμενο που περιέχει η μεταβλητή *tvIsConnected*). Ακριβώς παρακάτω ορίζουμε το μέγεθος του πεδίου (με την εντολή *wrap_content* του υποδεικνύουμε να έχει όσο μέγεθος χρειάζεται ώστε να καλύπτει το περιεχόμενό του) και τη στοίχιση που θα έχει στην οθόνη.

```
<TextView
    android:id="@+id/tvIsConnected"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:textSize="18sp"
    android:layout_marginBottom="5sp" />
```

Στο δεύτερο Textview έχουμε σαν σύνδεση τη μεταβλητή *coord_long*. Στην τελευταία εντολή πριν κλείσει αυτό το Textview βλέπουμε ότι το τοποθετούμε στο κάτω μέρος της οθόνης (από τη στιγμή που το Boolean της εντολής `android:layout_alignParentBottom` είναι *true*).

```
    <TextView
        android:id="@+id/coord_long"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" />
```

Στο τρίτο και τελευταίο Textview έχουμε σαν σύνδεση τη μεταβλητή *coord_lat*. Στις δύο τελευταίες σειρές βλέπουμε μια εντολή που τοποθετεί το παρόν textview κάτω από αυτό που συνδέεται με τη μεταβλητή *coord_long* και παράλληλα τα στοιχίζει στην αριστερή πλευρά της οθόνης.

```
    <TextView
        android:id="@+id/coord_lat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/coord_long"
        android:layout_alignLeft="@+id/coord_long" />
```

Στο τελευταίο κομμάτι του XML αρχείου δηλώνουμε την ύπαρξη ενός κουμπιού. Δεν δηλώνουμε από εδώ το τι θα κάνει αυτό το κουμπί, το μόνο που κάνουμε είναι να το συνδέσουμε με τη μεταβλητή, που θα αναγνωρίζει. Αυτή είναι η *findClosestButton*. Και εδώ, όπως και στα textviews που ορίσαμε παραπάνω, οι εντολές που ακολουθούν έχουν να κάνουν με την ακριβή διάταξη του κουμπιού στην οθόνη της εφαρμογής μας. Συγκεκριμένα παρατηρούμε ότι στοιχίζεται κάτω ακριβώς από το πρώτο textview, οριζόντια και στο κέντρο της οθόνης. Με την εντολή `android:text` ορίζουμε το κείμενο που θα εμφανίζεται στο χρήστη βλέποντας το κουμπί. Οι δύο τελευταίες εντολές αφορούν την εμφάνιση του κουμπιού. Αυτό που έχουμε ορίσει είναι ότι όταν θα ξεκινάει η εφαρμογή μου και μέχρι να γίνει η σύνδεση της βάσης δεδομένων με την εφαρμογή,

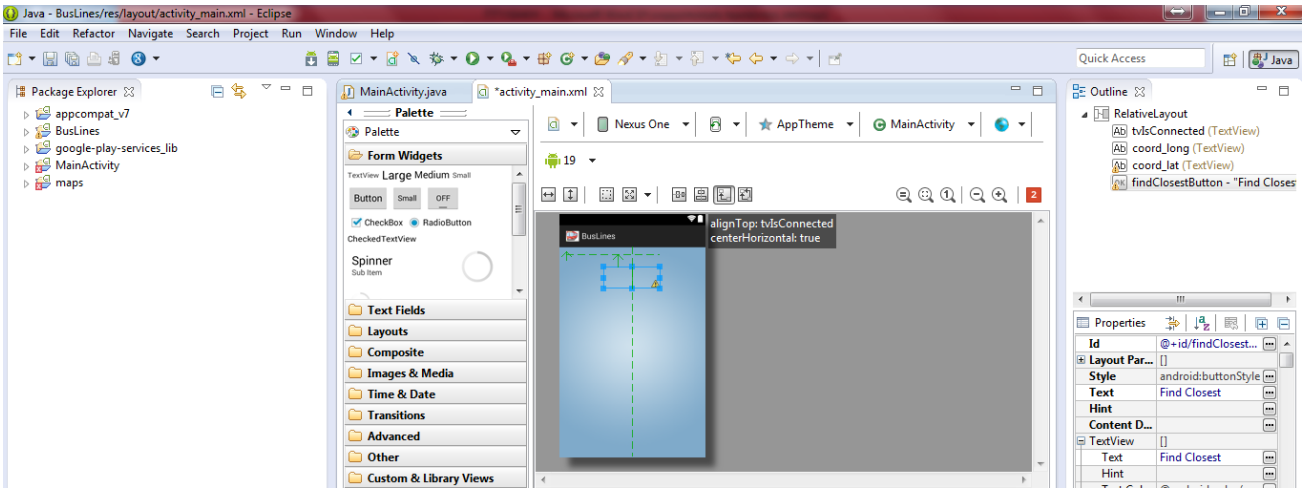
αυτό το κουμπί δεν θα είναι ορατό. Κάτι απολύτως λογικό, καθώς χωρίς να έχουμε τα δεδομένα της βάσης οποιαδήποτε διαδικασία της εφαρμογής είναι άσκοπη.

```
<Button
    android:id="@+id/findClosestButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/tvIsConnected"
    android:layout_centerHorizontal="true"
    android:text="Find Closest"
    android:onClick="findClosest"
    android:visibility="invisible" />
```

Στην τελευταία εντολή κλείνουμε το RelativeLayout που ανοίξαμε στην αρχή του αρχείου οπότε και ενημερώνουμε την εφαρμογή μας ότι δεν υπάρχει κάτι άλλο που πρέπει να εμφανιστεί στην οθόνη του χρήστη.

```
</RelativeLayout>
```

Όπως αναφέραμε και στην εισαγωγή που κάναμε για τις τεχνολογίες που θα χρησιμοποιηθούν στη συγκεκριμένη διπλωματική εργασία, το eclipse μας δίνει τη δυνατότητα να τεστάρουμε το γραφικό περιβάλλον ακόμη και χωρίς να τρέξουμε τον κώδικα. Αυτό γίνεται με το εργαλείο Graphical Layout, στο οποίο μπορούμε να ορίσουμε εμείς ό,τι θέλουμε (ακόμα και να δημιουργήσουμε ένα νέο στοιχείο) με ένα απλό drug'n'drop. Οτιδήποτε κάνουμε εμείς στο Graphical Layout μεταφέρεται απευθείας στο αρχείο activity_main.xml συμπληρώνοντας και τον ανάλογο κώδικα κάθε φορά. Παραθέτω ακριβώς παρακάτω μια εικόνα για το τι θα μου εμφανίζει η εφαρμογή σε αυτό το στάδιο.



Εικόνα 19: Graphical Layout στο Eclipse

Τελειώσαμε λοιπόν με το αρχείο που είναι υπεύθυνο για την εμφάνιση και την όψη της εφαρμογής μας. Ένα άλλο πολύ σημαντικό αρχείο πριν φτάσουμε στην ανάλυση του πηγαίου κώδικα είναι το αρχείο AndroidManifest.xml. Το αρχείο Manifest του Android είναι ένα ειδικά μορφοποιημένο αρχείο XML το οποίο μπορούμε να επεξεργαστούμε χειροκίνητα. Το αρχείο αυτό περιλαμβάνει μία ετικέτα <manifest> με μία μόνο ετικέτα <application>. Ας δούμε παρακάτω το αρχείο που χρησιμοποιεί η εφαρμογή μου και θα εξηγήσω κομμάτι-κομμάτι τον κώδικα.

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

Σε αυτές τις τρεις εντολές που ακολουθούν βλέπουμε ότι η εφαρμογή χρησιμοποιεί το όνομα πακέτου com.example.test, ότι η έκδοση της είναι η 1.0 και ότι ο κωδικός έκδοσης της εφαρμογής μου είναι 1.

```
    package="com.example.test"
    android:versionCode="1"
    android:versionName="1.0" >
```

Οι εντολές που περιλαμβάνονται στο <uses-sdk> είναι για να ορίσουν ότι η εφαρμογή λειτουργεί στο επίπεδο API 19 δηλαδή στην έκδοση 4.4 του Android που ονομάζεται KITKAT. Εκφράζει δηλαδή την συμβατότητα με μία ή περισσότερες εκδόσεις της πλατφόρμας Android.

```
    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="19" />
```

Τα <uses-permission> είναι τα δικαιώματα που απαιτεί η εφαρμογή για να εκτελεστεί, είναι δηλαδή οι άδειες που πρέπει να παρέχονται στην εφαρμογή μας για να λειτουργήσει σωστά. Οι άδειες αυτές δίνονται από το χρήστη, όταν η εφαρμογή εγκαθίσταται στη συσκευή του και όχι κατά τη διάρκεια της λειτουργίας της. Το πρώτο permission που ζητάμε είναι να έχει η εφαρμογή μας σύνδεση στο Internet, καθώς η βάση δεδομένων μας βρίσκεται σε έναν remote server. Οπότε είναι απαραίτητο για την αλληλεπίδραση και την επικοινωνία με τον server. Το δεύτερο permission είναι η αίτηση που κάνει η εφαρμογή για να έχει πρόσβαση σε πληροφορίες για τα δίκτυα. Εξίσου σημαντική με την πρώτη, καθώς με αυτήν την άδεια μπορεί η εφαρμογή να ελέγξει αν έχει πρόσβαση στο Διαδίκτυο είτε μέσω WiFi, είτε μέσω 3G. Το τρίτο permission είναι η άδεια που ζητάει η εφαρμογή για να έχει πρόσβαση στην ακριβή τοποθεσία από το hardware που χρησιμοποιείται για τις υπηρεσίες θέσης, το GPS και το Wifi. Το τέταρτο και τελευταίο permission είναι η αίτηση για την πρόσβαση στην κατά προσέγγιση θέση του χρήστη από πηγές τοποθεσίας δικτύου, όπως οι κεραίες κινητής τηλεφωνίας ή το Wifi.

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

Στην ετικέτα <application> ορίζουμε κάποιες ρυθμίσεις τις εφαρμογής. Η πρώτη είναι ότι επιτρέπουμε στην εφαρμογή να συμμετέχει στη δημιουργία αντιγράφων και αποκατάσταση των

υποδομών. Η δεύτερη είναι το εικονίδιο που χρησιμοποιεί η εφαρμογή, η τρίτη το φιλικό όνομα της εφαρμογής και τέλος το θέμα της.

```
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

Μέσα στο <activity> πρέπει να υπάρχει κάθε δραστηριότητα που διενεργείται στην εφαρμογή. Αν δεν ορίσουμε την ετικέτα <activity> για μια Activity, δεν θα εκτελεστεί. Εδώ λοιπόν έχω ορίσει την MainActivity που χρησιμοποιώ, το όνομα της εφαρμογής που υπάρχει στο string και τέλος έχω ορίσει το ότι, όταν θα ξεκινάει η εφαρμογή και θα μπαίνει στην κύρια οθόνη, θα καλύπτει όλη την οθόνη, χωρίς τον τίτλο της.

```
        <activity
            android:name=".MainActivity"
             android:label="@string/app_name"
             android:theme="@android:style/Theme.NoTitleBar">
```

Το λειτουργικό σύστημα Android χρησιμοποιεί φίλτρα προθέσεων για την ανάλυση των έμμεσων προθέσεων, δηλαδή προθέσεων οι οποίες δεν καθορίζουν την Activity που θέλουν να ξεκινήσει. Η ετικέτα <intent-filter> καθορίζει τους τύπους των προθέσεων που μια Activity, μια Service ή μια Broadcast receiver μπορεί να απαντήσει. Το φίλτρο προθέσεων καθορίζει τις δυνατότητες του περιεχομένου της (τι μπορεί να κάνει μια Activity ή μια Service και σε τι τύπους εκπομπών μπορεί να ανταποκριθεί ένας δέκτης). Εδώ έχω προσθέσει το όνομα της πράξης MAIN με το όνομα κατηγορίας LAUNCHER.

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

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

Στο συγκεκριμένο σημείο τελειώνει το αρχείο AndroidManifest.

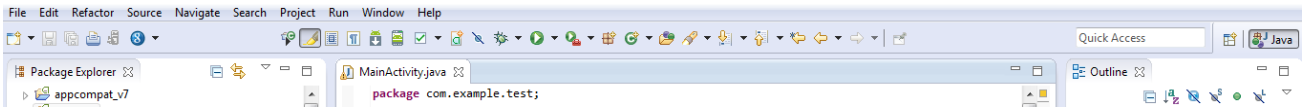
```
</manifest>
```

Πριν προχωρήσουμε στο τελευταίο κομμάτι της υλοποίησης που είναι ο πηγαίος κώδικας, θα κάνω μια παρένθεση για να παραθέσω ένα μικρό αρχείο με path res/values/strings.xml, στο οποίο υπάρχει το όνομα της εφαρμογής (*app_name*) που είδαμε πιο πάνω:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">BusLines</string>
    <string name="action_settings">Settings</string>

</resources>
```

Προχωράμε λοιπόν έπειτα από αυτή τη μικρή παρένθεση στον πηγαίο κώδικα της εφαρμογής μου. Εξηγώντας αυτόν τον κώδικα κομμάτι-κομμάτι, όπως έκανα και σε όλα τα παραπάνω στάδια, θα επιδιώξω να καλύψω όλα τα αναπάντητα ερωτήματα. Στον πηγαίο κώδικα ουσιαστικά βρίσκεται ο πυρήνας της εφαρμογής. Τι διεργασία θα εκτελεστεί, με ποιά σειρά θα εκτελεστεί, τι χρειάζεται για να εκτελεστεί η συγκεκριμένη διεργασία και άλλα πολλά θα τα δούμε όλα εδώ. Πάμε λοιπόν να ξεκινήσουμε την περιήγηση στον πυρήνα της εφαρμογής μου.

Αρχικά δηλώνεται το πακέτο, στο οποίο είναι αποθηκευμένος ο πηγαίος κώδικας:



Στη συνέχεια ακολουθούν τα import statements της Java. Αυτά είναι ο ορισμός του μέλους ενός πακέτου στην εφαρμογή, με σκοπό όταν θα θέλουμε να χρησιμοποιήσουμε το συγκεκριμένο στοιχείο να μην χρειάζεται να αναφέρουμε το πλήρες όνομα του στοιχείου αυτού. Για παράδειγμα αν ήθελα να χρησιμοποιήσω την κλάση Rectangle από το πακέτο graphics θα έπρεπε να αναφερθώ σε αυτή ως

```
graphics.Rectangle myRect = new graphics.Rectangle();
```

Ενώ μπορώ να συμπληρώσω στην αρχή του πηγαίου κώδικα, μετά τον ορισμό του πακέτου, την εξής δήλωση:

```
import graphics.Rectangle;
```

και να αναφέρομαι στη συγκεκριμένη κλάση μόνο με το όνομά της, δηλαδή:

```
Rectangle myRectangle = new Rectangle();
```

Σε περίπτωση που ήθελα να εισάγω όλο το πακέτο θα χρησιμοποιούσα τη δήλωση:

```
import graphics.*;
```

Οπότε κοιτώντας τις δηλώσεις των import statements και μόνο μπορούμε να καταλάβουμε τις κλάσεις που χρησιμοποιούμε.

Τα παρακάτω imports αφορούν το άνοιγμα του αρχείου κειμένου που θα χρησιμοποιήσουμε και την ασφαλή διαδικασία της ανάγνωσής του.

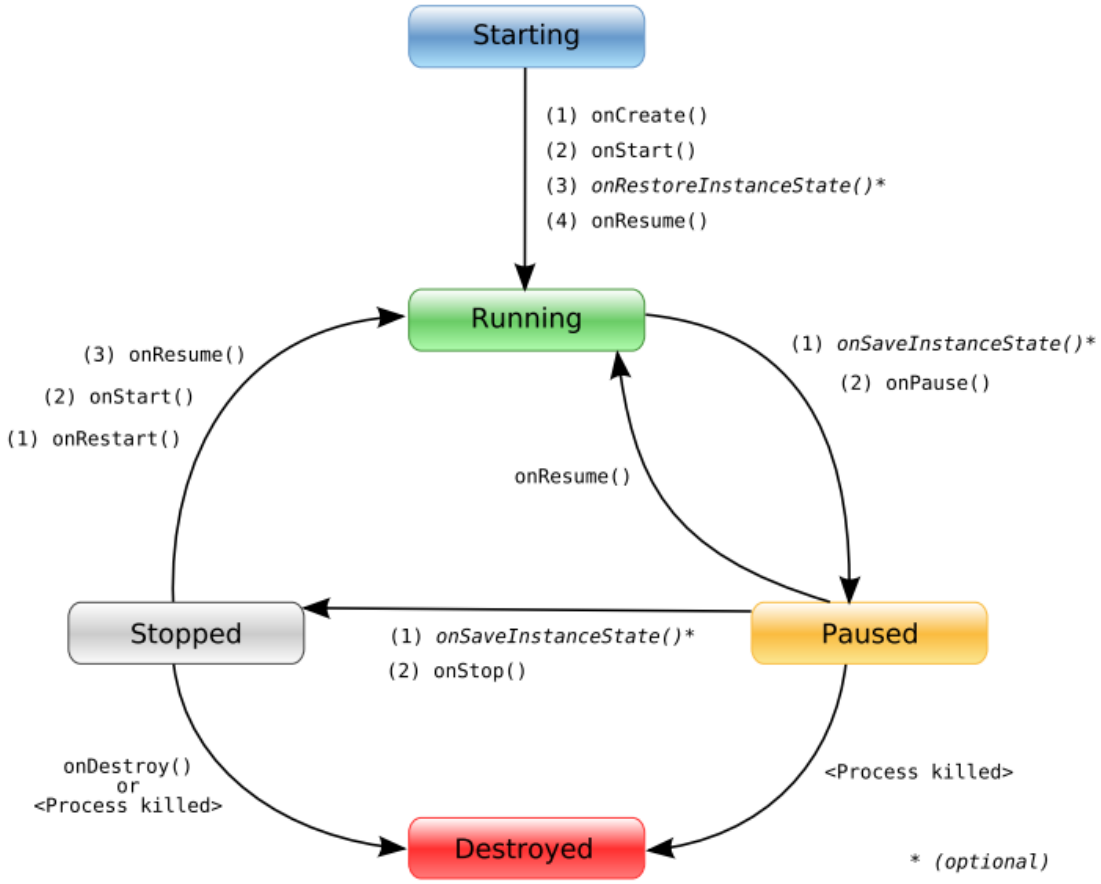
```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
```

Αυτά που ακολουθούν έχουν να κάνουν με τη επικοινωνία με τον web server που χρειάζεται να έχει η εφαρμογή μου και την ανάκτηση του αρχείου JSON με τη μορφή που θέλουμε, ώστε να μπορούμε να χρησιμοποιήσουμε τα δεδομένα του.

```
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

Το παρακάτω import statement είναι από τα πιο σημαντικά, αν όχι το πιο σημαντικό. Αφορά το βασικό Activity του project, αυτό δηλαδή που είναι υπεύθυνο για να προβάλλει τη βασική οθόνη της εφαρμογής. Γενικά, κάθε οθόνη διάδρασης με το χρήστη, είναι συνδεδεμένη με ένα Activity, κάτι που σημαίνει στην πράξη ότι κάθε εφαρμογή θα πρέπει να έχει τουλάχιστον ένα Activity. Σε περίπτωση που ο προγραμματιστής θέλει να προσθέσει άλλη μία οθόνη διάδρασης θα πρέπει να δημιουργήσει ένα νέο Activity και να το δηλώσει στην αντίστοιχη διασύνδεση.
Αξίζει να δούμε πως ένα Activity κληρονομεί από την κλάση android.app.Activity του SDK. Αυτή η κλάση περιέχει μεθόδους που μπορούν να υπερκαλυφθούν (Override) έτσι ώστε να γράψουμε έναν κώδικα που θα εκτελεστεί ως αντίδραση σε κάποια κατάσταση του κύκλου ζωής, στην οποία έχει περιέλθει το Activity της εφαρμογής.

Αφού αναφερθήκαμε στον κύκλο ζωής ενός Activity ας δούμε τα πολύ βασικά χαρακτηριστικά του. Κατά τη διάρκεια του κύκλου ζωής λοιπόν, το Activity είναι δυνατόν να περιέλθει σε διάφορες καταστάσεις. Ο προγραμματιστής δεν έχει τον έλεγχο του σε ποιά κατάσταση είναι το πρόγραμμα, αυτό διαχειρίζεται από το σύστημα. Παρόλα αυτά ο προγραμματιστής ενημερώνεται όταν μια κατάσταση είναι να αλλάξει με τη χρήση των onXX() μεθόδων. Ας το δούμε σχηματικά για να αναλύσουμε τις πιο βασικές από αυτές τις μεθόδους.



Εικόνα 20: Η ζωή και ο κύκλος ενός Activity

- onCreate(bundle): καλείται, όταν ξεκινάει το Activity και μπορούμε να το χρησιμοποιήσουμε για μια αρχικοποίηση που γίνεται μια και μοναδική φορά. Παίρνει μια παράμετρο που είναι είτε το null, είτε κάποιες πληροφορίες κατάστασης που έχουν σωθεί προηγουμένως στην onSaveInstanceState() μέθοδο.
- onStart(): υποδεικνύει ότι το Activity είναι έτοιμο να εμφανιστεί στο χρήστη.
- onResume(): καλείται όταν το Activity αρχίζει να αλληλεπιδρά με το χρήστη.
- onPause(): τρέχει όταν το Activity είναι έτοιμο να μπει σε background λειτουργία.
- onStop(): καλείται όταν το Activity δεν είναι ορατό πια στο χρήστη και δεν θα χρειαστεί για λίγο. Αν δεν υπάρχει πολλή μνήμη στη συσκευή καλύτερα να μην καλείται ποτέ.
- onRestart(): καλείται για να επανεκκινήσει η εφαρμογή από την προηγούμενη κατάσταση.
- onDestroy(): καλείται λίγο πριν το Activity τερματιστεί.
- onSaveInstanceState(Bundle): το Android καλεί αυτή τη μέθοδο για να επιτρέψει στο Activity να αποθηκεύσει την παρούσα κατάσταση.
- onRestoreInstanceState(Bundle): καλείται όταν η το Activity επανέρχεται από την προηγούμενη κατάσταση.

Αυτές είναι οι βασικές μέθοδοι που φαίνονται και στην εικόνα παραπάνω και καθορίζουν τον κύκλο ζωής ενός Activity. Παρακάτω υπάρχουν και κάποιες άλλες μέθοδοι που χρησιμοποιώ στην εφαρμογή μου.

- onPostExecute(Result): καλείται όταν τελειώσει μια background διαδικασία υπολογισμού. Το αποτέλεσμα αυτής της διαδικασίας υπολογισμού καταχωρείται ως παράμετρος σε αυτό το βήμα.
- onCreateOptionsMenu(Menu menu): καλείται, όταν αρχικοποιούνται τα περιεχόμενα του στάνταρ μενού επιλογών του Activity.
- onOptionsItemSelected(MenuItem item): καλείται, όταν επιλεχθεί ένα αντικείμενο από το μενού επιλογών.
- onLocationChanged(Location location): καλείται, όταν αλλάζει η τοποθεσία.
- onProviderDisabled(String provider): καλείται, όταν ο πάροχος απενεργοποιείται από το χρήστη.
- onProviderEnabled(String provider): καλείται, όταν ο πάροχος ενεργοποιείται από το χρήστη.
- onStatusChanged(String provider, int status, Bundle extras): καλείται όταν αλλάζει το status του πάροχου.

Οι τέσσερεις τελευταίες μέθοδοι χρησιμοποιούνται στην εξακρίβωση της τοποθεσίας του χρήστη.

```
import android.app.Activity;
import android.content.Context;
```

Τα ακόλουθα import χρησιμοποιούνται για την κλάση AsynchTask που θα χρησιμοποιήσουμε στην εφαρμογή. Αυτή η κλάση επιτρέπει να διενεργούνται background λειτουργίες και να δημοσιοποιούνται τα αποτελέσματά τους στο χρήστη χωρίς να χρειάζεται να ασχοληθεί με Handlers.

```
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
```

Τα παρακάτω, αφορούν το οπτικό κομμάτι της εφαρμογής (το μενού, το textview, το κουμπί).

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

Αυτό το import έχει να κάνει με την κλάση Toast που δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν εύκολα και γρήγορα μηνύματα ενημέρωσης προς τους χρήστες. Ένα Toast μήνυμα αναδύεται στην οθόνη και προβάλλει μόνο κείμενο για μικρό χρονικό διάστημα, πριν τελικά εξαφανιστεί με ένα fading εφέ. Κατά τη διάρκεια της προβολής του ο χρήστης μπορεί να συνεχίσει να αλληλεπιδρά κανονικά με την εφαρμογή, ενώ το ίδιο δεν μπορεί να λάβει κανένα είδος διάδρασης.

```
import android.widget.Toast;
```

Η κλάση Location αναπαριστά μία γεωγραφική τοποθεσία τη δεδομένη χρονική στιγμή. Μια τοποθεσία μπορεί να ορίζεται από το γεωγραφικό μήκος, το γεωγραφικό πλάτος, timestamp και άλλες πληροφορίες όπως η ταχύτητα.

```
import android.location.Location;
```

Η κλάση LocationManager είναι η κεντρική κλάση που μας παρέχει πρόσβαση στις υπηρεσίες τοποθεσίας. Η κλάση LocationListener είναι ένα Interface που παρέχει ενημερώσεις μέσω του LocationManager όταν η θέση της συσκευής αλλάζει.

```
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
```

Τώρα που αναλύσαμε όλα τα import statements που περιλαμβάνει η εφαρμογή μου, και είδαμε σε τι αναφέρεται το καθένα, προχωράμε στην επεξήγηση του πηγαίου κώδικα.

Η εντολή extends που βλέπουμε πιο κάτω έχει να κάνει με την κληρονομικότητα στην Java. Ουσιαστικά επιτρέπει στη subclass να κληρονομήσει (και να χρησιμοποιήσει προφανώς) όλες τις public και protected μεταβλητές και μεθόδους της superclass.

```
public class MainActivity extends Activity {
```

Ακολουθεί η αρχικοποίηση και η δήλωση των μεταβλητών. Οι λέξεις public και private υποδηλώνουν την προσβασιμότητα των συγκεκριμένων μεταβλητών. Οι πρώτες τρεις μεταβλητές είναι τα Textviews που έχουμε ορίσει να βρίσκονται στην οθόνη μας. Το πρώτο μας πληροφορεί ότι έχουμε συνδεθεί με τη βάση δεδομένων. Το δεύτερο και το τρίτο εμφανίζουν στην οθόνη το γεωγραφικό πλάτος και μήκος αντίστοιχα, όταν αλλάξουν οι συντεταγμένες. Η τελευταία μεταβλητή είναι για το κουμπί που θα υπάρχει στην οθόνη της εφαρμογής.

```
    public TextView tvIsConnected;
    public TextView coord_long;
    public TextView coord_lat;
    public Button findClosestButton;
```

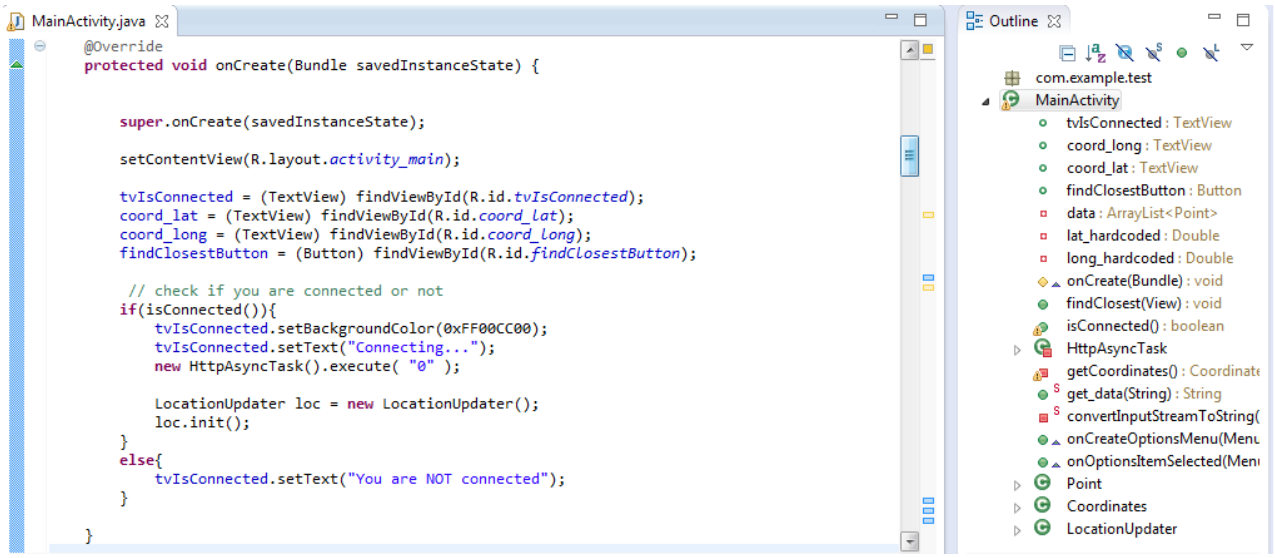
Στην παρακάτω δήλωση αρχικοποιούμε έναν πίνακα που περιέχει αντικείμενα που προέρχονται από την κλάση Point (την οποία θα δούμε πιο κάτω τι περιέχει) και τα εισάγει σε μια μεταβλητή που ονομάζεται data.

```
    private ArrayList<Point> data = new ArrayList<Point>();
```

Οι δύο παρακάτω μεταβλητές είναι οι μεταβλητές που τις έχουμε περάσει χειροκίνητα και είναι αυτές που παίρνει σαν default η εφαρμογή μου, όταν εκκινεί. Είναι ένα ζεύγος συντεταγμένων που αντιστοιχεί σε μια γεωγραφική θέση στο Γαλάτσι. Ο λόγος που όρισα αυτές τις συντεταγμένες είναι ότι ήθελα να δοκιμάσω το αποτέλεσμα της εφαρμογής και να δω αν όντως αντιστοιχεί σε αυτό το σημείο η κοντινότερη στάση που θα έβγαζε ως αποτέλεσμα. Πέρα από αυτό όμως, πρέπει να υπάρχουν κάποιες default συντεταγμένες για να λειτουργήσει η εφαρμογή.

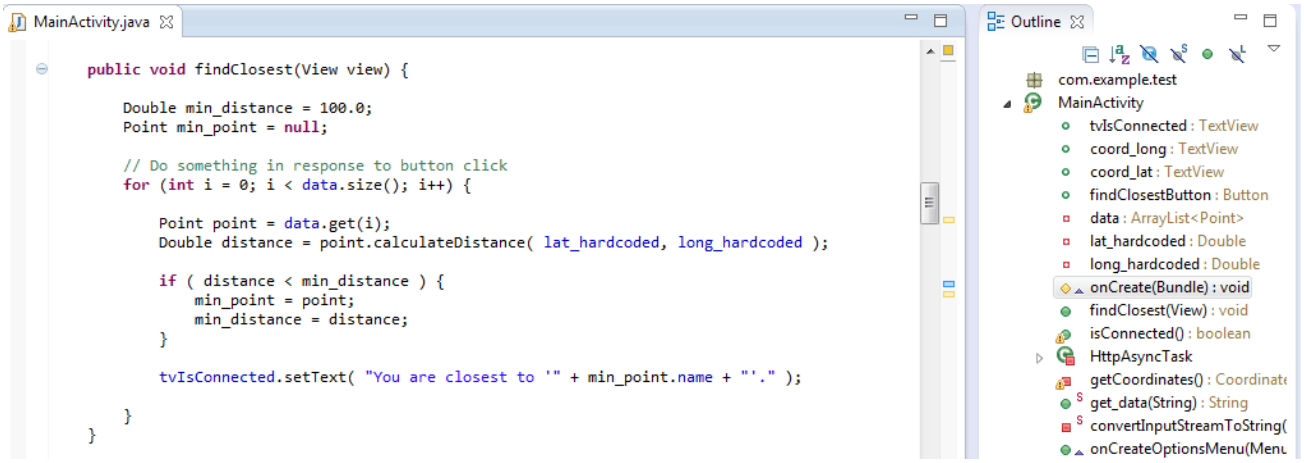
```
private Double lat_hardcoded = 38.021082;
private Double long_hardcoded = 23.753387;
```

Προχωράμε λοιπόν στο πρώτο κομμάτι του κώδικα:



Εδώ λοιπόν βλέπουμε τις οδηγίες που δίνουμε στην εφαρμογή για το πώς θα ενεργεί κατά την εκκίνησή της. Όπως είδαμε και παραπάνω αναλύοντας τις βασικές μεθόδους αυτό που κάνουμε είναι όταν εκκινεί το Activity να παίρνουμε σαν παράμετρο την κατάσταση που έχει σωθεί. Επειδή η μέθοδος onCreate() καλείται είτε το σύστημα δημιουργεί ένα νέο παράδειγμα του Activity είτε στην αναδημιουργία ενός προηγούμενου , θα πρέπει να ελέγχουμε αν η κατάσταση Bundle είναι null. Αν είναι μηδενική , τότε το σύστημα δημιουργεί ένα νέο παράδειγμα του Activity, αντί της αποκατάστασης ενός προηγούμενου, που καταστράφηκε. Με την εντολή setContentView ορίζουμε το περιεχόμενο του Activity σε μια συγκεκριμένη μορφή. Αυτή η μορφή έχει προκύψει από την ιεραρχία του Activity που έχει συμπληρωθεί από τον προγραμματιστή στο αρχείο activity_main.xml. Στις επόμενες τέσσερεις εντολές βλέπουμε ότι χρησιμοποιούμε την εντολή findViewById. Αυτή η εντολή ουσιαστικά συνδέει τις μεταβλητές που είχαμε ορίσει στην αρχή (τα textviews και το button) με τη μορφή, που θα εμφανίζονται χρησιμοποιώντας το id, που τους είχαμε ορίσει στο activity_main.xml.
Με τη χρήση της isConnected ελέγχουμε αν υπάρχει συνδεσιμότητα σε δίκτυο και αν είναι δυνατόν να ξεκινήσει η σύνδεση για ανταλλαγή δεδομένων. Σε αυτό το if ορίζουμε ότι αν υπάρχει σύνδεση σε δίκτυο, στο textview tvIsConnected να εμφανίζεται το μήνυμα Connecting… σε πράσινο φόντο και να εκτελείται μια Asynchronous Task. Αν δεν υπάρχει σύνδεση στο δίκτυο να εμφανίζεται στο textview tvIsConnected το κείμενο You are NOT connected. Για τις δύο γραμμές που αφορούν την κλάση LocationUpdater θα μιλήσω εκτενέστερα πιο κάτω, για την ώρα ας αρκεστούμε στο ότι, εάν υπάρξει αλλαγή της θέσης του χρήστη, ενημερώνεται η εφαρμογή.

Συνεχίζουμε με την επόμενη μέθοδο.

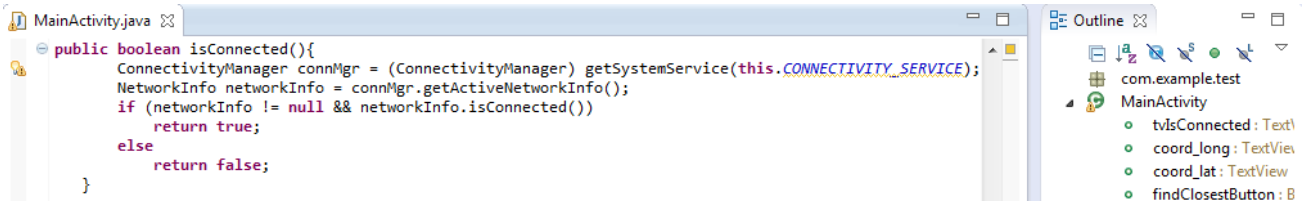


Σε αυτή την κλάση ξεκινάμε με τη δήλωση δύο μεταβλητών, Ενός δεκαδικού αριθμού που αρχικοποιείται με την τιμή 100.0 και τη χρησιμοποιούμε για να ορίσουμε τη μικρότερη απόσταση και αρχικοποιούμε μια μεταβλητή min_point που συνδέεται με την κλάση Point που θα δούμε πιο κάτω. Η ενέργεια που ορίζουμε στην παρούσα κλάση είναι στο τι θα γίνεται, όταν ο χρήστης θα πατάει το κουμπί, που έχει η εφαρμογή. Αφού ο χρήστης λοιπόν πατήσει το κουμπί, στο for που ακολουθεί θα γίνει προσπέλαση στα δεδομένα της μεταβλητής data και θα καταχωρηθεί στη μεταβλητή distance, που θα έχει μορφή δεκαδικού αριθμού, το αποτέλεσμα της συνάρτησης calculateDistance (η οποία έχει ως ορίσματα τις μεταβλητές lat_hardcoded, long_hardcoded και ανήκει στην κλάση Point).

Κατά την προσπέλαση των αρχείων (κατά τη διάρκεια του for δηλαδή) υπάρχει μια σύγκριση η οποία αυτό που κάνει είναι να συγκρίνει τις τιμές των μεταβλητών distance και min_distance. Εφόσον η min_distance είναι μεγαλύτερη της distance τότε καταχωρείται στην τιμή min_point η τιμή της point και στην τιμή της min_distance η τιμή της distance. Στο τέλος της σύγκρισης αυτής στο textview tvIsConnected εμφανίζεται το κείμενο You are closest to και ακολουθεί η τιμή της μεταβλητής min_point.name .

Καταλαβαίνω ότι δεν είναι τόσο ξεκάθαρο το ποιες είναι οι τιμές των μεταβλητών και τι κάνει η κάθε μία αλλά στο τέλος της επεξήγησης όλα τα σκοτεινά σημεία θα διαλευκανθούν.

Το επόμενο κομμάτι του αρχείου MainActivity.java είναι η δημιουργία του Boolean isConnected():



Η κλάση ConnectivityManager είναι εκείνη, η οποία αποκρίνεται σε ερωτήσεις που γίνονται σχετικά με την κατάσταση της συνδεσιμότητας σε δίκτυο. Επίσης ειδοποιεί τις εφαρμογές όταν η συνδεσιμότητα σε δίκτυο αλλάξει. Οι πρωταρχικές ευθύνες αυτής της κλάσης είναι:

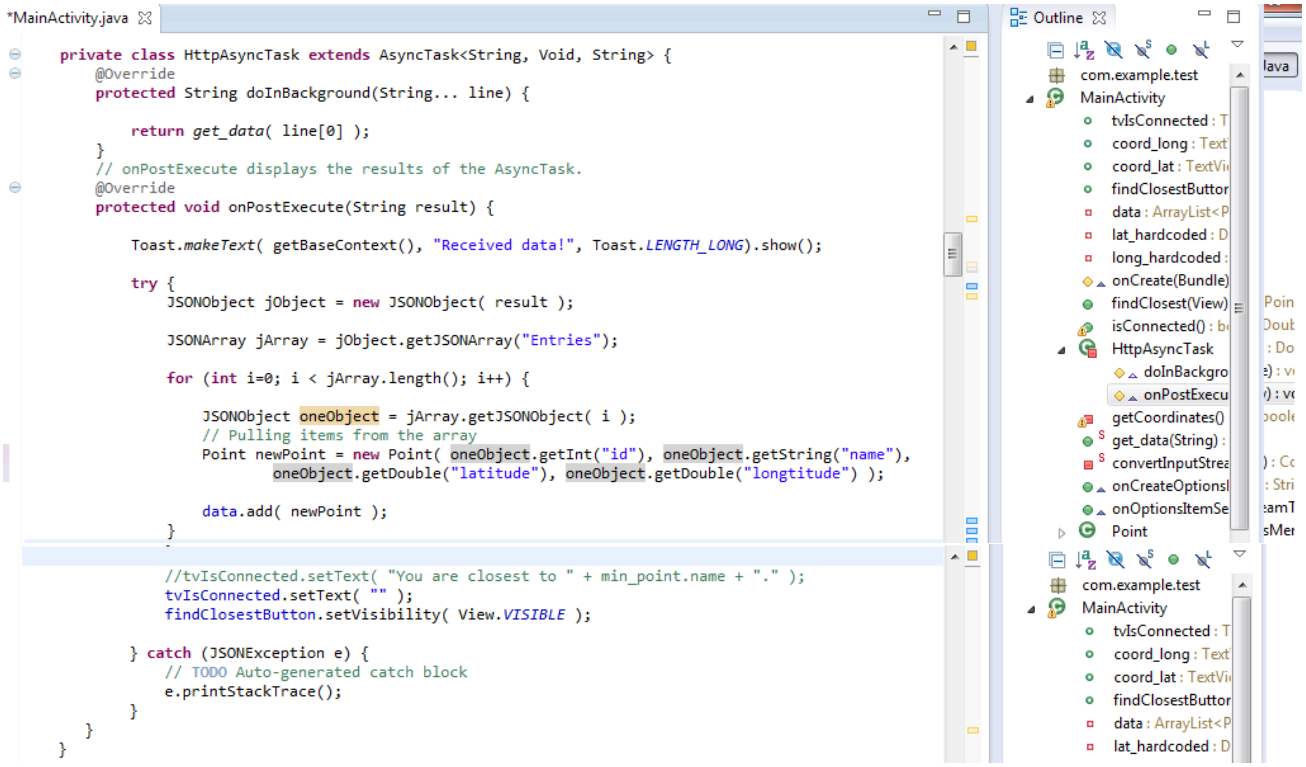
1. Να εμφανίζει στην οθόνη τις συνδέσεις δικτύου (Wi-fi, GPRS, UMTS, κ.ά)
2. Να στέλνει Broadcast intents, όταν αλλάζει η συνδεσιμότητα στο δίκτυο
3. Προσπάθεια για σύνδεση σε άλλο δίκτυο, όταν η συνδεσιμότητα με το παρόν δίκτυο χαθεί
4. Να παρέχει ένα API στην εφαρμογή για να ελέγχει την coarse-grained ή fine-grained κατάσταση των διαθέσιμων δικτύων.

Με την εντολή getSystemService αναθέτουμε σε έναν ConnectivityManager να διαχειριστεί τις συνδέσεις δικτύου.

Με την κλάση NetworkInfo περιγράφεται το status από το interface ενός δικτύου. Με τη μέθοδο getActiveNetworkInfo μας επιστρέφονται λεπτομέρειες για το ενεργό default δίκτυο δεδομένων. Όταν είναι συνδεδεμένο αυτό το δίκτυο, είναι η default διαδρομή για την ανάδραση με τα υπόλοιπα δίκτυα. Αν η τιμή της είναι null, τότε σημαίνει ότι δεν υπάρχει default δίκτυο.

Μετά από αυτή την επεξήγηση παρατηρούμε ότι, αν η τιμή του networkinfo δεν είναι null ΚΑΙ υπάρχει συνδεσιμότητα σε δίκτυο, τότε επιστρέφει την τιμή true, σε αντίθετη περίπτωση επιστρέφει την τιμή false.

Η παρακάτω κλάση είναι αρκετά σημαντική καθώς σε αυτήν περιγράφεται η λειτουργία της Asynchronous Task και η διαδικασία με την οποία επικοινωνεί το eclipse με τη βάση δεδομένων.




```
private class HttpAsyncTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... line) {

        return get_data( line[0] );
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {

        Toast.makeText( getBaseContext(), "Received data!", Toast.LENGTH_LONG).show();

        try {
            JSONObject jObject = new JSONObject( result );

            JSONArray jArray = jObject.getJSONArray("Entries");

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

                JSONObject oneObject = jArray.getJSONObject( i );
                // Pulling items from the array
                Point newPoint = new Point( oneObject.getInt("id"), oneObject.getString("name"),
                        oneObject.getDouble("latitude"), oneObject.getDouble("longtitude") );

                data.add( newPoint );
            }

            //tvIsConnected.setText( "You are closest to " + min_point.name + "." );
            tvIsConnected.setText( "" );
            findClosestButton.setVisibility( View.VISIBLE );

        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Ξεκινώντας, να αναφέρω τη λειτουργία της μεθόδου doInBackground(). Η μέθοδος αυτή χρησιμοποιείται για να εκτελεστεί ο υπολογισμός ενός background thread. Οι παράμετροι που χρησιμοποιώ εδώ είναι το String και το line. Το αποτέλεσμα είναι το όρισμα της get_data. Η μέθοδος onPostExecute που έχει ως παράμετρο το result εμφανίζεται στην οθόνη μετά την εκτέλεση της doInBackground(). Το result που έχουμε θέσει ως παράμετρο λαμβάνει την τιμή που θα επιστραφεί από την doInBackground(). Στη συγκεκριμένη εφαρμογή μετατρέπει σε String το αποτέλεσμα που θα λάβει από την επικοινωνία με τη βάση δεδομένων, τα στοιχεία του πίνακα δηλαδή.

Η γραμμή που ακολουθεί είναι η δημιουργία του toast (το μικρό και γρήγορο μήνυμα που εμφανίζεται στο χρήστη) που αναφέραμε και πιο πάνω. Με τη μέθοδο makeText δημιουργούμε ένα standard toast που περιέχει ένα μήνυμα κειμένου. Ως παραμέτρους δίνουμε τη μέθοδο getBaseContext, το κείμενο που θα εμφανίζεται στο toast (Received data! στη δική μου εφαρμογή) και τη διάρκεια που θα έχει στην οθόνη.

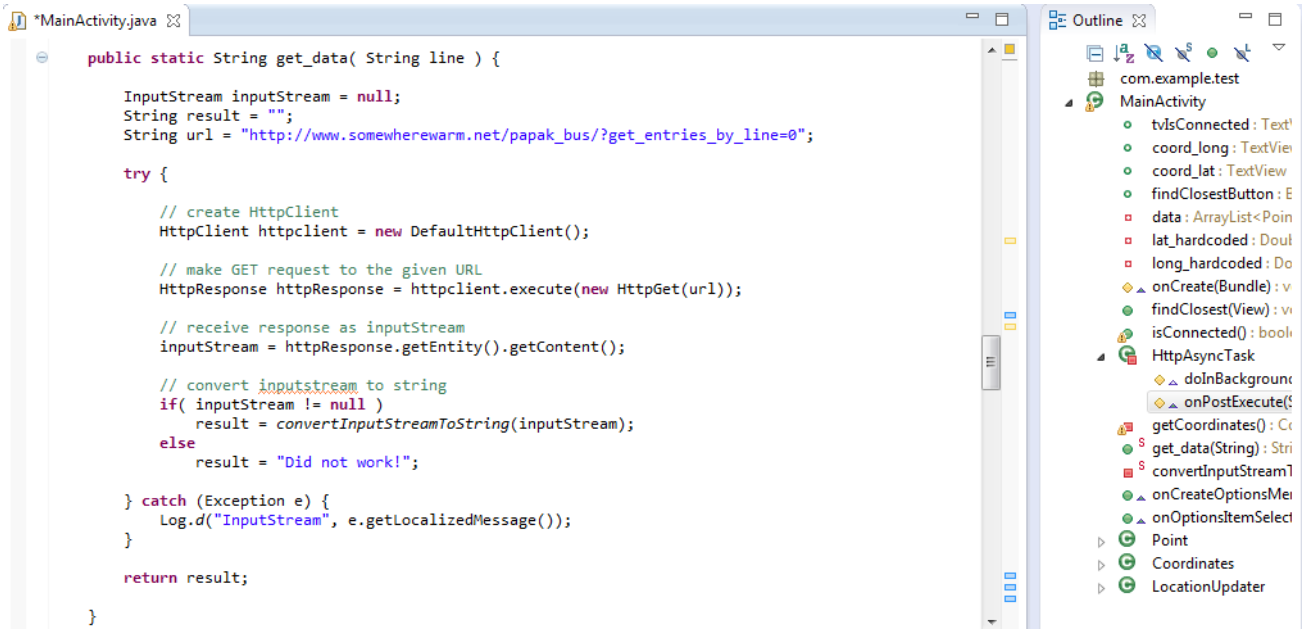
Στη συνέχεια δημιουργούμε ένα νέο json αντικείμενο με τα δεδομένα που έχουν καταχωρηθεί στη μεταβλητή result και είναι μορφής string. Η κλάση JSONOBJECT είναι ένα σετ από ζευγάρια ονομάτων και τιμών. Τα ονόματα είναι μοναδικά non-null strings, ενώ οι τιμές μπορεί να είναι μια μίξη από JSONObjects, JSONArrays με ακέραιους αριθμούς, strings, Booleans κ.ά. Η επόμενη κίνηση για να γίνουν επεξεργάσιμα τα δεδομένα είναι να δημιουργήσουμε έναν πίνακα με τα δεδομένα που προέρχονται από τον πίνακα entries που έχουμε δημιουργήσει στη βάση δεδομένων.

Στο for που ακολουθεί γίνεται η προσπέλαση των δεδομένων με σκοπό να τραβήξουμε τα δεδομένα από τον JSON πίνακα που είχαμε δημιουργήσει πριν λίγο. Τα δεδομένα λαμβάνονται με την ανάλογη μέθοδο που αρμόζει στο καθένα (π.χ. για τη στήλη id επιλέγουμε τη μέθοδο getInt καθώς οι τιμές αυτής της στήλης είναι ακέραιοι αριθμοί) και τοποθετούνται στη μεταβλητή newPoint.

Όταν ολοκληρωθεί αυτή η διαδικασία, στο textview tvIsConnected εμφανίζεται κενό και γίνεται ορατό το κουμπί findClosestButton.

Στο τέλος της κλάσης HttpAsyncTask υπάρχει και η κλάση JSONException που είναι υπεύθυνη να αναδείξει ότι πρόβλημα υπάρχει με το JSON API εμφανίζοντας στην οθόνη ένα exception.

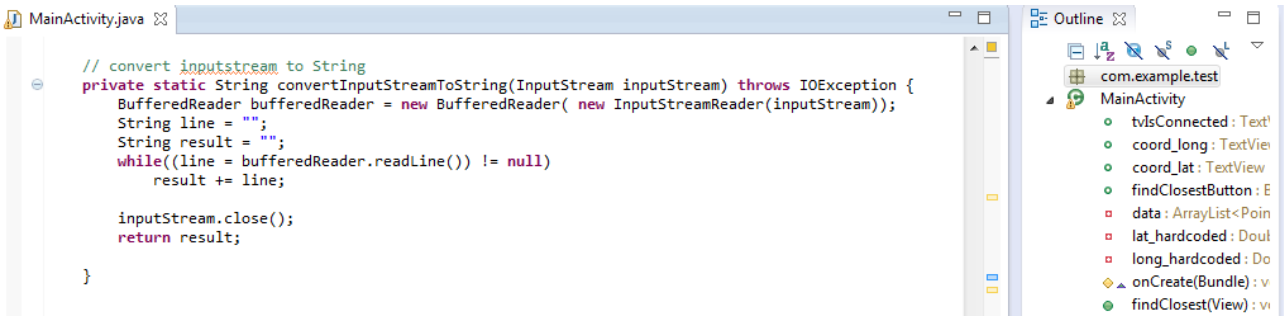
Η επόμενη μέθοδος ονομάζεται String get_data και σε αυτή δηλώνεται ο τρόπος με τον οποίο εγκαθίσταται η επικοινωνία ανάμεσα στο eclipse και στον web server, στον οποίο βρίσκεται η βάση δεδομένων μου.



Η κλάση InputStream είναι μια αναγνώσιμη πηγή από bytes. Πολλοί clients χρησιμοποιούν τα input streams για να διαβάσουν δεδομένα από τα αρχεία συστήματος ή από το δίκτυο, είτε για να διαβάσουν byte πίνακες που βρίσκονται εσωτερικά της μνήμης. Στο string url καταχωρούμε τη διεύθυνση που πρέπει να καταχωρήσουμε στο browser ώστε να γίνει το request, για να λάβουμε τα δεδομένα από τον πίνακα entries βάσει της στήλης που αναγράφεται ο αριθμός της γραμμής. Στο string result αρχικοποιείται δίνοντάς του ένα κενό. Οι εντολές που ακολουθούν μετά το try είναι οι εντολές με τις οποίες συνδέουμε την εφαρμογή με τον server. Η κλάση HttpClient είναι ένα Interface για έναν Http client. Ο Http client ενθυλακώνει αντικείμενα που απαιτούνται για να εκτελεστούν HTTP αιτήματα με την ταυτόχρονη διαχείριση των cookies, και της σύνδεσης

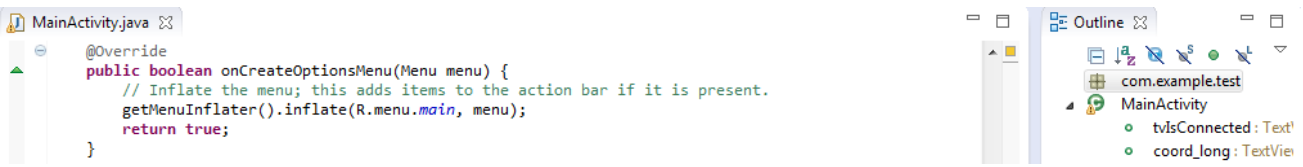
γενικότερα. Χρησιμοποιώντας την κλάση DefaultHttpClient() δημιουργούμε έναν νέο Http client. Στην επόμενη γραμμή δίνουμε ως απάντηση στο HTTP αίτημα την ενέργεια του GET από το URL που έχουμε ορίσει. Στη συνέχεια δηλώνουμε ότι το response που θα πάρουμε,θέλουμε να είναι της μορφής input stream. Στο if που ακολουθεί αυτό που θέλουμε να πετύχουμε είναι να μετατρέψουμε το input stream που έχουμε λάβει σαν response σε string ούτως ώστε να είναι αναγνώσιμο από την εφαρμογή. Οπότε σε περίπτωση που η μεταβλητή inputStream δεν έχει τιμή null καταχωρούμε στη μεταβλητή result το αποτέλεσμα της μετατροπής του inputStream σε String. Σε περίπτωση που το inputStream δεν έχει δεδομένα η διαδικασία μας βγάζει ως αποτέλεσμα το κείμενο Did not work! Τέλος με τη μέθοδο Log.d παράγεται ένα log που αφορά το Debug της διαδικασίας και καταγράφεται στο exception που τη χρησιμοποιεί.

Με την επόμενη μέθοδο πραγματοποιείται η διαδικασία που ανέφερα παραπάνω, η μετατροπή του input stream σε string.



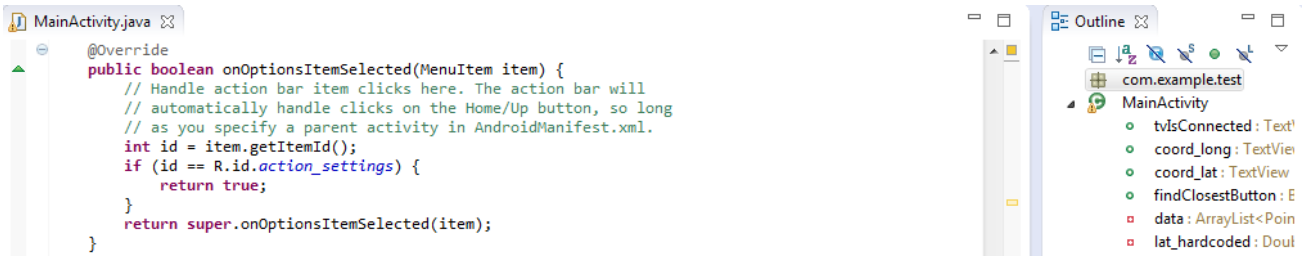
Με την κλάση BufferedReader ενεργοποιούμε έναν υπάρχοντα Reader και αποθηκεύουμε αυτό που του δίνουμε σαν είσοδο. Το μόνο μειονέκτημα της συγκεκριμένης κλάσης είναι ότι χρειάζεται επιπλέον χώρος για να κρατήσει τον buffer και ότι η αντιγραφή γίνεται αφού γεμίσει ο buffer, παρόλα αυτά η επίδοση της εφαρμογής δεν επηρεάζεται καθόλου (ειδικά αν μιλάμε για μία <<ελαφριά >> εφαρμογή, που δεν απαιτεί πολύπλοκους υπολογισμούς δηλαδή). Με τη μέθοδο readLine μας επιστρέφει η επόμενη γραμμή του κειμένου που είναι διαθέσιμο στον Reader. Στο while που ακολουθεί καταχωρούμε στο string result το αποτέλεσμα που παίρνει το string line. Όσο λοιπόν δεν υπάρχει κενό στο κείμενο που προέρχεται από τον Reader καταχωρείται στη μεταβλητή result. Στο τέλος της κλάσης με τη μέθοδο inputStream.close() ουσιαστικά κλείνουμε το stream και η κλάση μας επιστρέφει την τιμή της μεταβλητής result.

Ακολουθούν δύο Boolean τα οποία είναι auto-generated, δηλαδή έχουν συμπληρωθεί από μόνα τους τη στιγμή της δημιουργίας του αρχείου MainActivity.java.



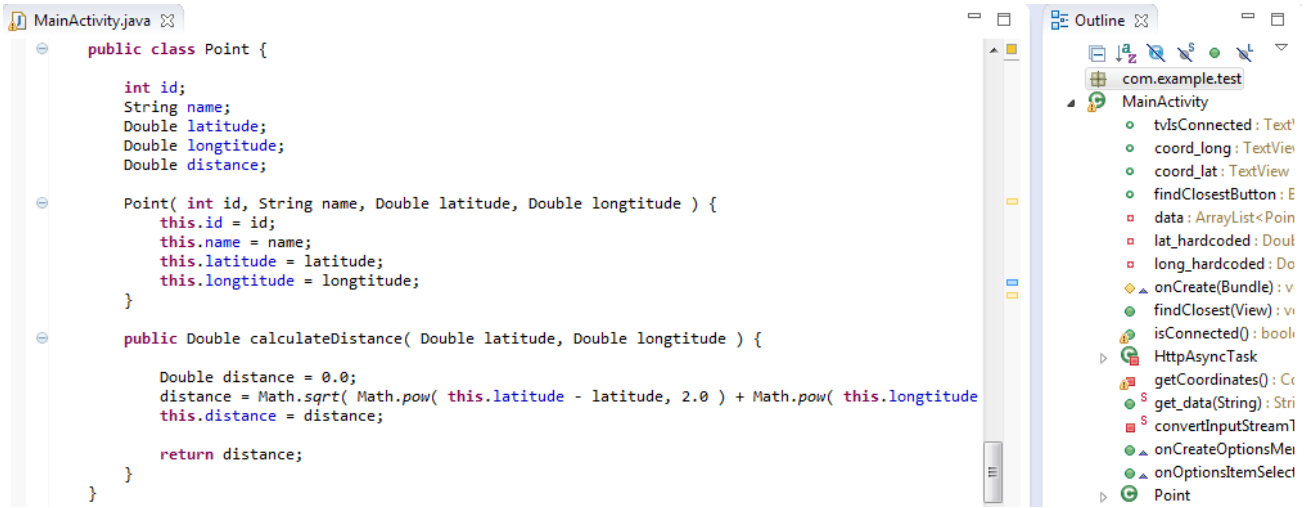
Η λειτουργία του πρώτου Boolean είναι η αρχικοποίηση των περιεχομένων του στάνταρ μενού επιλογών της εφαρμογής. Σε περίπτωση που θα θέλαμε να προσθέσουμε κάποιο αντικείμενο στο στάνταρ μενού, θα πρέπει να το καταγράψουμε αρχικά σε ένα νέο αρχείο xml στο φάκελο res/menu. Η μέθοδος onCreateOptionsMenu() καλείται μόνο μία φορά στην εφαρμογή κατά τη διάρκεια της δημιουργίας του μενού επιλογών.

Προχωράμε να δούμε και το δεύτερο Boolean και να εξηγήσω τη λειτουργία του.



Η μέθοδος του δεύτερου Boolean καλείται κάθε φορά που επιλέγεται ένα αντικείμενο από το μενού επιλογών. Στην ουσία η λειτουργία που έχει είναι να επιστρέφει το μενού επιλογών στην αρχική του κατάσταση όταν επιλέγουμε κάποιο αντικείμενο από το μενού.

Στο αρχείο του πηγαίου κώδικα ακολουθεί μια κλάση στην οποία αναφέρθηκα αρκετές φορές στην επεξήγηση του κώδικα νωρίτερα. Ας τη δούμε λοιπόν για να καταλάβουμε καλύτερα αρκετά από τα παραπάνω.

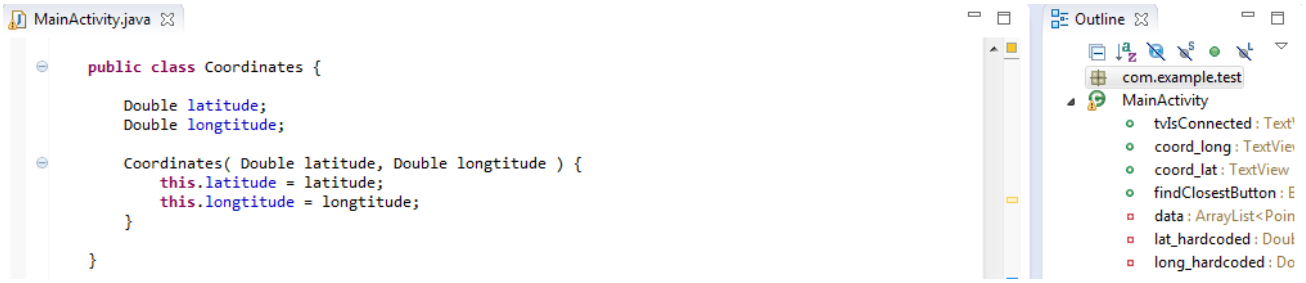


Όπως καταλαβαίνουμε αν ρίξουμε μια πιο προσεκτική ματιά στις μεταβλητές της κλάσης μιλάμε για εκείνη την κλάση που είναι υπεύθυνη για αρκετές από τις μαθηματικές πράξεις που γίνονται στην εφαρμογή μας. Η κλάση λοιπόν ξεκινάει με τη δήλωση πέντε μεταβλητών, ενός ακέραιου αριθμού με όνομα id, ενός string με όνομα name και τριών δεκαδικών αριθμών με τα ονόματα latitude, longitude και distance. Στη συνέχεια αυτό που κάνουμε είναι να καταχωρήσουμε τις τιμές που έχουμε τραβήξει από τον πίνακα entries στις αντίστοιχες μεταβλητές. Τα δεδομένα δηλαδή της κάθε στήλης, στην αντίστοιχη μεταβλητή. Οπότε όλες οι τιμές που βλέπαμε ως τώρα με αντικείμενα της κλάσης Point καταλαβαίνουμε ότι ήταν τα δεδομένα που θέλαμε να επεξεργαστούμε από τον πίνακα entries στη βάση δεδομένων που δημιουργήσαμε.

Στην calculateDistance() λαμβάνει χώρα η μαθηματική πράξη στην οποία είναι βασισμένη όλη η εφαρμογή μου. Αναφέρομαι στον μαθηματικό τύπο που ανέλυσα στη σχεδίαση της εφαρμογής και μου δίνει την απόσταση ανάμεσα σε δύο σημεία με τα αντίστοιχα ζεύγη συντεταγμένων. Εδώ βλέπουμε την αρχικοποίηση της μεταβλητής distance που έχει τη μορφή δεκαδικού αριθμού και αναφέρεται στην απόσταση των δύο σημείων. Ο κώδικας που ακολουθεί είναι στην πραγματικότητα η προγραμματιστική υλοποίηση του μαθηματικού τύπου. Οπότε βλέπουμε ότι είναι μια ρίζα που περιέχει το τετράγωνο των διαφορών των γεωγραφικών μηκών και των γεωγραφικών πλατών. Στο τέλος της κλάσης το αποτέλεσμα είναι ο δεκαδικός αριθμός, που αντιστοιχεί στην απόσταση των δύο σημείων.

Αν ανατρέξουμε στην πιο πάνω findClosest συνάρτηση βλέπουμε ότι, κατά τη διάρκεια της προσπέλασης των δεδομένων, η πράξη που γινόταν ήταν να συγκρίνουμε κάθε φορά την απόσταση των δύο σημείων και αφού βρίσκαμε ποιο σημείο έχει τη μικρότερη απόσταση να εμφανίζουμε το όνομα αυτού του σημείου. Κάτι που είναι και ο αντικειμενικός στόχος αυτής της εφαρμογής.

Προχωράμε στη συνέχεια του αρχείου να δούμε και τις εναπομείνασες κλάσεις.

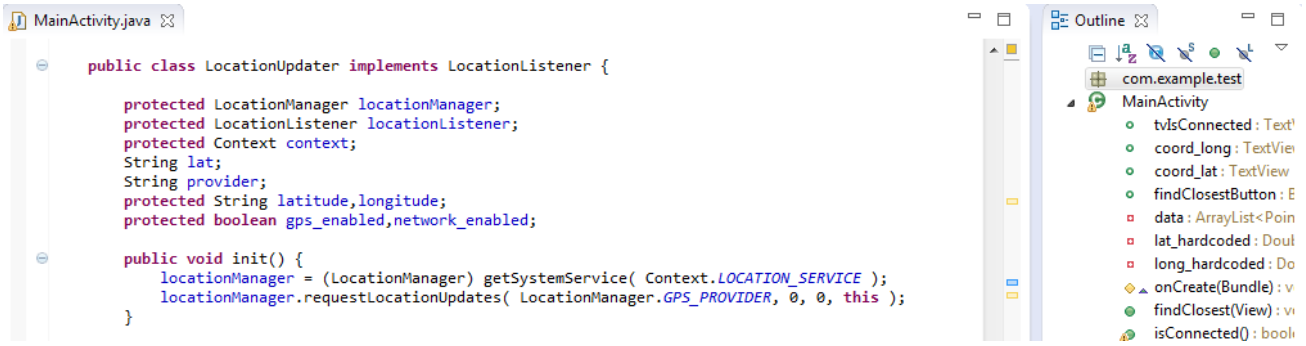


Σε αυτή την κλάση ορίζουμε τη μορφή που θα έχουν οι συντεταγμένες και το όνομα της κάθε μεταβλητής. Οπότε παρατηρούμε ότι το latitude αντιστοιχεί στο γεωγραφικό μήκος και το longitude στο γεωγραφικό πλάτος.

Στο τέλος της επεξήγησης του πηγαίου κώδικα θα δούμε τον τρόπο με τον οποίο δοκιμάζουμε το αποτέλεσμα της εφαρμογής. Με λίγα λόγια αυτό που θα αναφέρω σε αυτό το σημείο είναι ότι

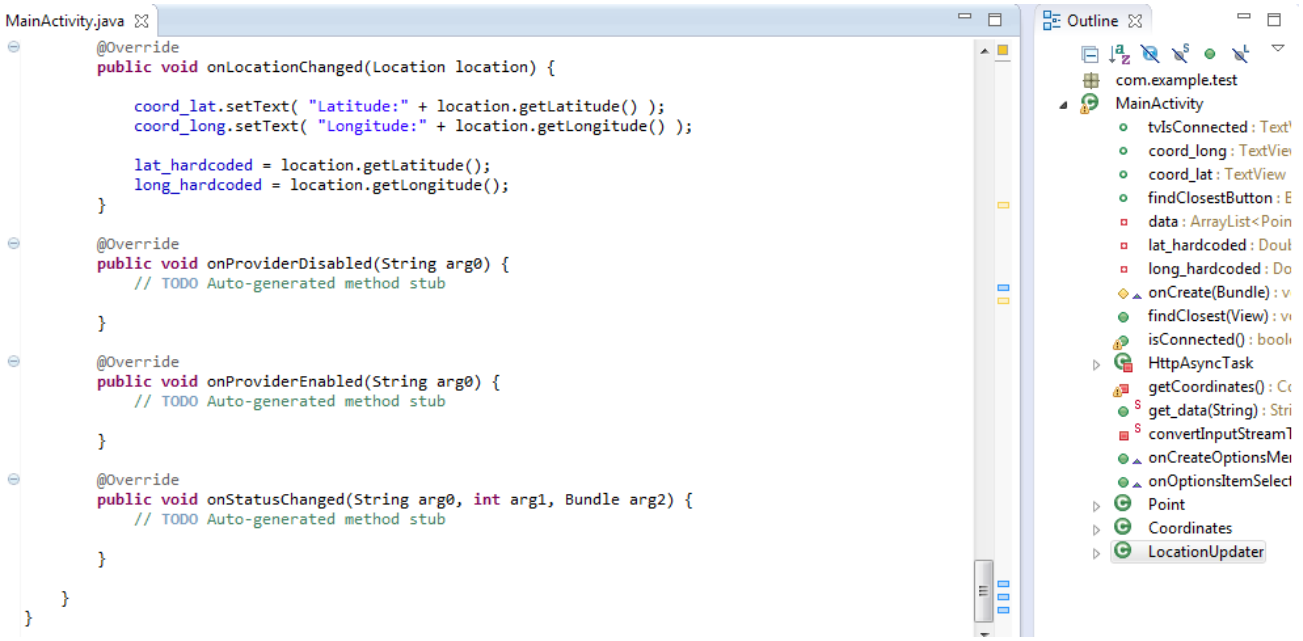
κατασκευάζουμε μια εικονική συσκευή και φορτώνουμε την εφαρμογή σε αυτή σαν να ήταν μια πραγματική smartphone ή tablet συσκευή. Υπάρχει ένα πρόβλημα όμως. Αυτό το πρόβλημα είναι ότι δεν λειτουργούν οι υπηρεσίες βάσει τοποθεσίας όπως θα λειτουργούσαν σε μία κανονική συσκευή, καθώς δεν υπάρχει GPS για να τραβάει τις πραγματικές συντεταγμένες του χρήστη. Οπότε για να δοκιμάσουμε την εφαρμογή θα πρέπει να καταχωρούμε εμείς συνεχώς ζεύγη συντεταγμένων ώστε να γίνονται οι συγκρίσεις με αυτά που είναι καταχωρημένα στη βάση δεδομένων. Δηλαδή να αλλάζουμε διαρκώς τις πρώτες δύο μεταβλητές του κώδικα που έχουμε ορίσει ως ένα σημείο εκκίνησης της εφαρμογής. Αντ' αυτού με την παρακάτω κλάση δίνουμε ως είσοδο σε αυτήν την εικονική συσκευή ζεύγη συντεταγμένων μέσω Telnet. Κάτι πολύ πιο χρήσιμο και εύκολο από το να αλλάζουμε συνεχώς τον κώδικα. Το πώς θα το δούμε κατά τη διάρκεια της δοκιμής της εφαρμογής.

Ας συνεχίσουμε λοιπόν με τα τελευταία κομμάτια του κώδικα.



Η κλάση LocationUpdater θα εφαρμοστεί ως εσωτερική κλάση στην κλάση LocationListener. Ορίζω τις μεταβλητές που θα χρησιμοποιήσουμε, τον locationManager, τον locationListener, το Context, τα string lat και provider και τα προστατευμένα μέλη της κλάσης (τα string latitude, longitude) και τέλος το Boolean gps_enabled και network_enabled. Αναθέτουμε στον πάροχο εντοπισμού θέσης να ελέγχει τις ανανεώσεις της τοποθεσίας του χρήστη. Με τη μέθοδο requestLocationUpdates() ορίζουμε το κάθε πότε θα στέλνονται οι ενημερώσεις. Η πρώτη παράμετρος που λαμβάνει είναι ο τρέχων πάροχος εντοπισμού θέσης, η δεύτερη το ελάχιστο χρονικό διάστημα σε msec που θα μεσολαβεί μεταξύ των ενημερώσεων, η τρίτη η ελάχιστη διαφορά απόστασης σε m και τέλος η τέταρτη ο Listener που θα λαμβάνει αυτές τις ενημερώσεις.

Ας δούμε τη συνέχεια της κλάσης που είναι και το τέλος του αρχείου MainActivity.java:



Με το LocationListener Interface βελτιώνουμε την εφαρμογή ανανεώνοντας ανά τακτά χρονικά διαστήματα τη νέα θέση της συσκευής. Η βασική αλλαγή στην κλάση είναι πως έχει τεθεί να υλοποιεί το interface LocationListener όπως προαναφέρθηκε. Το interface αυτό λαμβάνει ενημερώσεις για τη νέα θέση της συκευής και υλοποιώντας τη μέθοδο onLocationChange(), γράφουμε κώδικα που θα εκτελείται κάθε φορά που λαμβάνει χώρα αυτό το συμβάν. Στη συγκεκριμένη μέθοδο προτείνεται να μη γράφουμε κώδικα που απαιτεί μεγάλο επεξεργαστικό φορτίο, αλλά να περνάμε τα δεδομένα για επεξεργασία σε ένα νέο thread. Εκτός από την onLocationChange(), για να έχουμε συμπαγή κλάση θα πρέπει να υλοποιήσουμε και τις μεθόδους onProviderDisabled(), onProviderEnabled() και onStatusChanged(). Όπως βλέπουμε και στην εικόνα με τον κώδικα αυτές οι μέθοδοι δεν περιέχουν κώδικα στο σώμα τους. Η πρώτη καλείται αυτόματα όταν ο τρέχων πάροχος απενεργοποιηθεί από το χρήστη, η δεύτερη όταν ενεργοποιείται από το χρήστη και τέλος η τρίτη όταν μεταβάλλεται η κατάστασή του. Φυσικά μπορούμε να γράψουμε κώδικα που θα εκτελείται ως αντίδραση σε αυτά τα συμβάντα αν το επιθυμούμε.

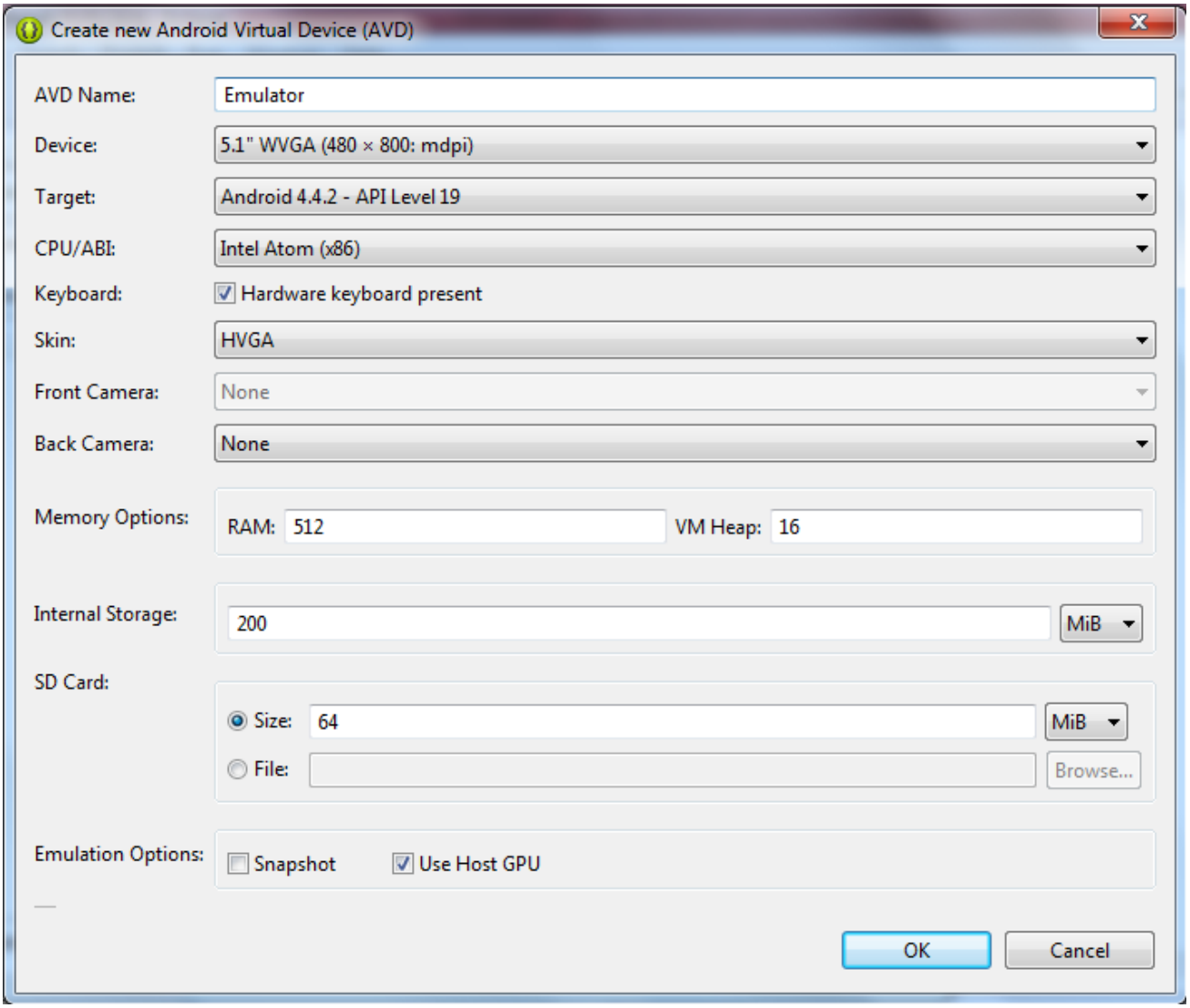
ΚΕΦΑΛΑΙΟ 4: ΔΟΚΙΜΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Σε αυτό το σημείο τελείωσε το κεφάλαιο της επεξήγησης του κώδικα και γενικότερα των αρχείων που χρησιμοποιεί η εφαρμογή μου. Ήρθε η ώρα να δούμε τη λειτουργία της εφαρμογής στην πράξη.

Για αυτό τον λόγο θα δημιουργήσω μια AVD (Android Virtual Device), θα ορίσω δηλαδή την εικονική συσκευή στην οποία θα προβάλλω την εφαρμογή μου. Επιλέγω από το μενού του eclipse το εξής:

Window →Android Virtual Device Manager

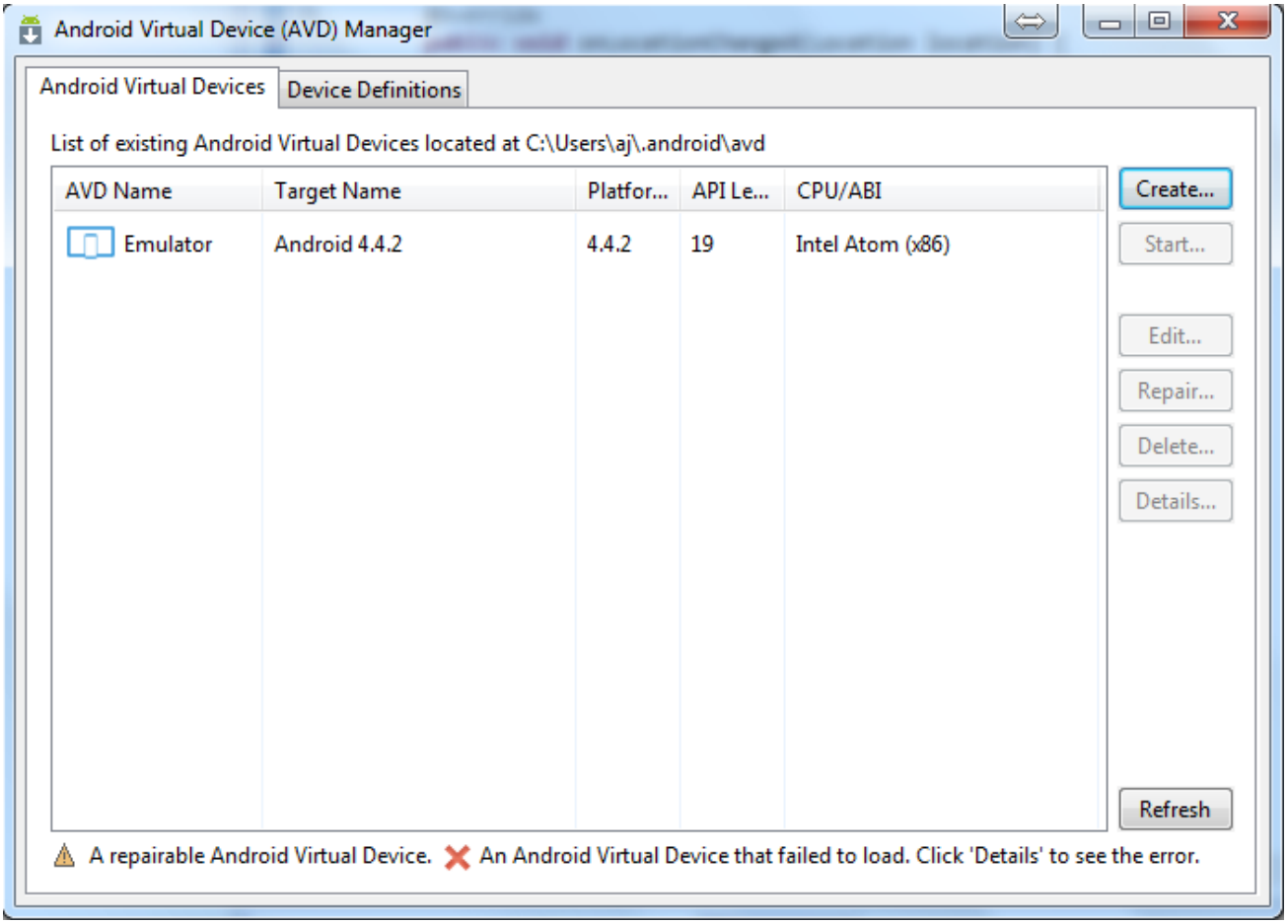
Στην πρώτη οθόνη επιλέγω το **Create** και ρυθμίζω τον AVD.



Εικόνα 21: Δημιουργία Android Virtual Device

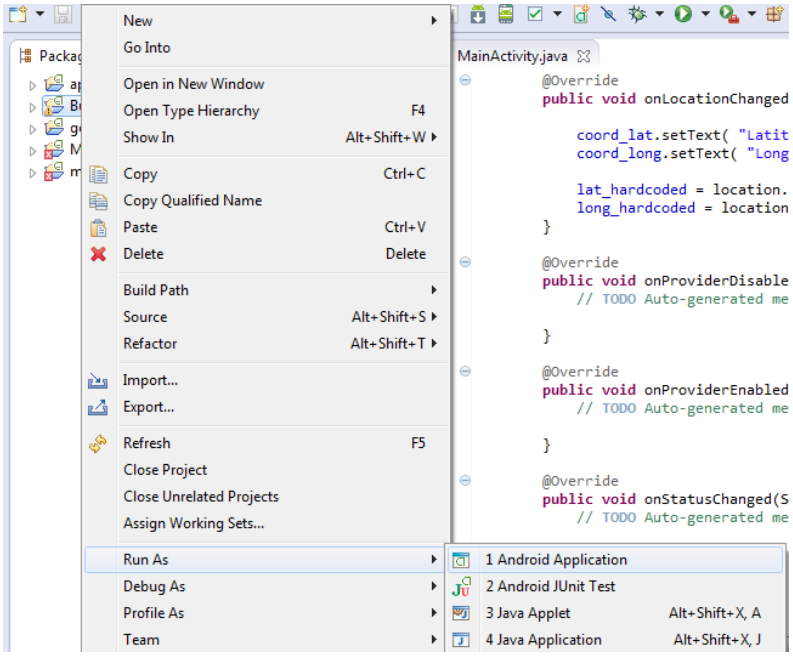
Στο **AVD Name** βάζουμε το όνομα που θέλουμε να έχει η συσκευή, στο **Device** τον τύπο της συσκευής, στο **Target** την έκδοση του λειτουργικού που έχουμε ορίσει εξαρχής στην εφαρμογή, στο **Skin** το πώς θα εμφανίζεται η συσκευή (αν θα έχει πληκτρολόγιο για παράδειγμα) και τέλος στο **SD Card Size** το μέγεθος σε MB που θα έχει η υποστηριζόμενη SD Card.

Τώρα αν ξαναγυρίσουμε στη λίστα με τους AVD θα δούμε ότι υπάρχει αυτός που μόλις δημιουργήσαμε:



Εικόνα 22: Λίστα AVDs

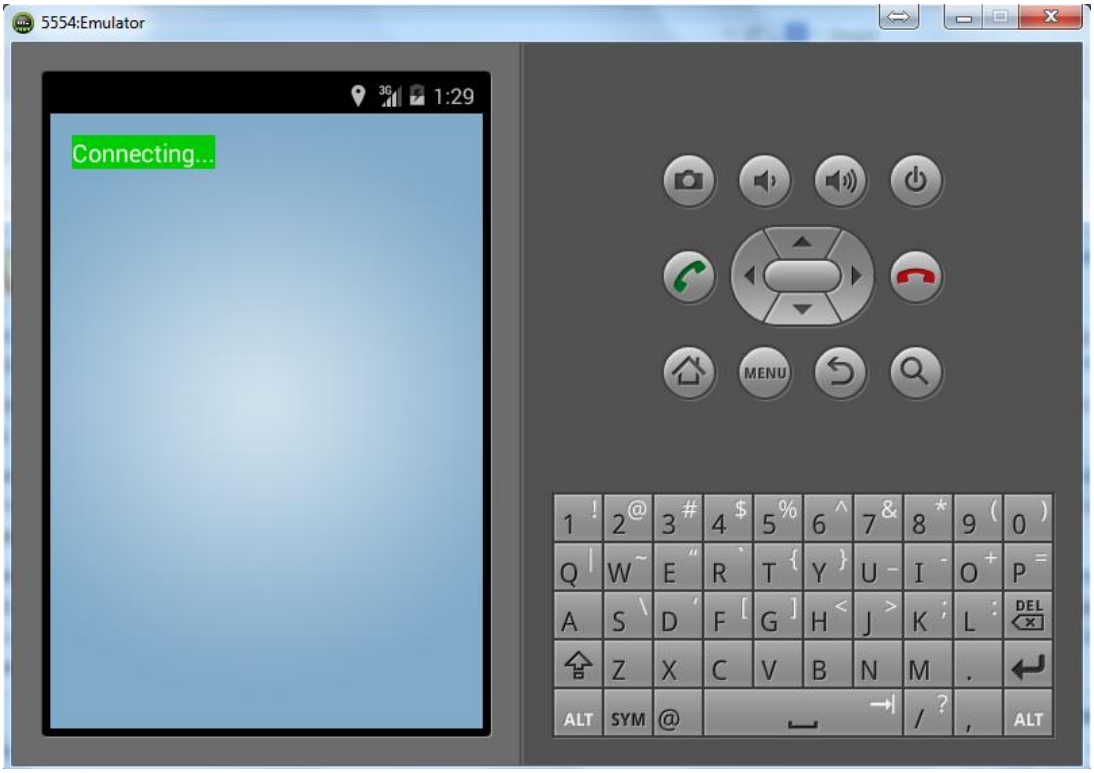
Έτσι είμαστε έτοιμοι να τρέξουμε την εφαρμογή μας και να δούμε τα αποτελέσματα. Για να το κάνουμε αυτό πάμε στον Package Explorer του eclipse και κάνουμε δεξί κλικ στο project και επιλέγουμε **Run As → Android Application**.



Σε αυτή τη φάση θα εκκινήσει ο AVD που έχουμε ορίσει.

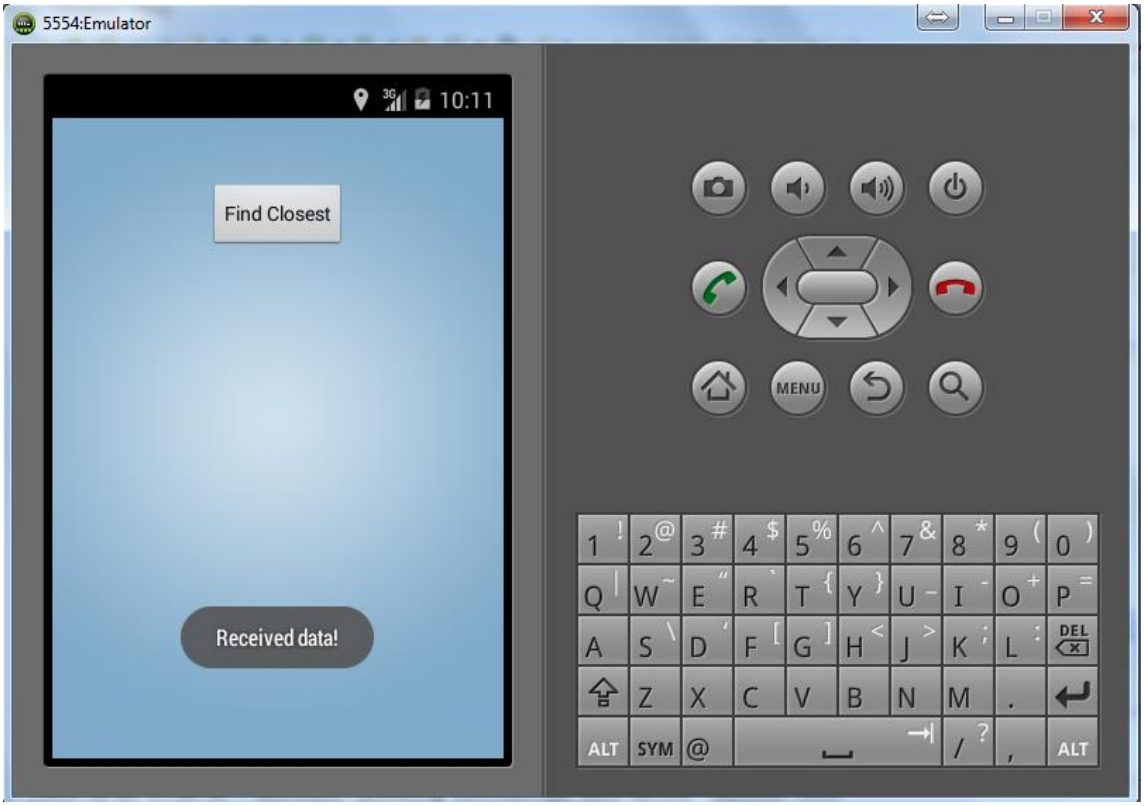
Εικόνα 23: Εκκίνηση AVD

Πηγαίνοντας από το μενού στις εφαρμογές της συσκευής κάνουμε κλικ στο εικονίδιο της εφαρμογής για να ανοίξει. Ανοίγοντας την εφαρμογή εμφανίζεται πάνω αριστερά το μήνυμα που έχουμε ορίσει για την περίοδο που συνδέεται με τον server. Παρατηρούμε ότι δεν φαίνεται το κουμπί.



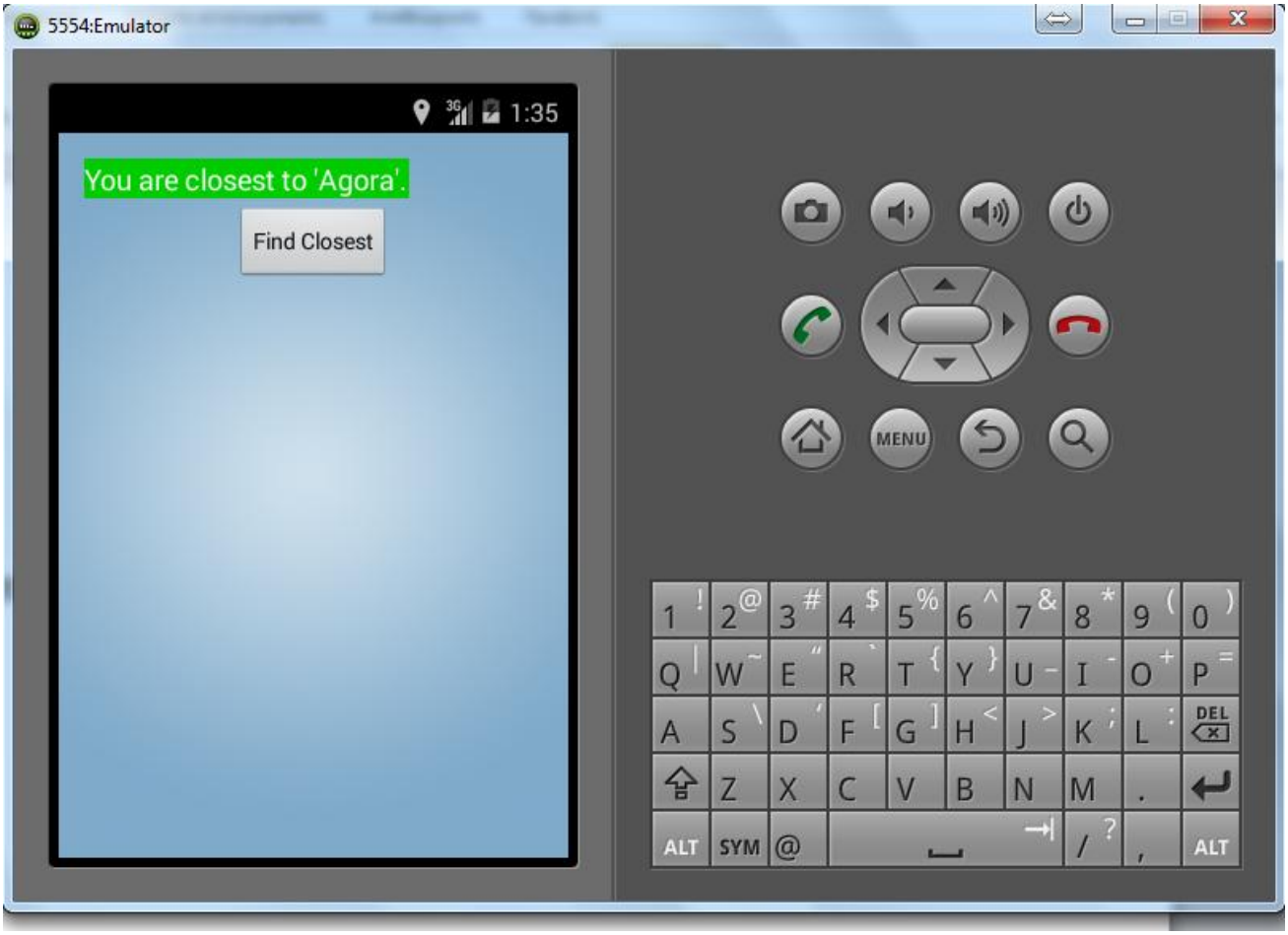
Εικόνα 24: Προσπάθεια σύνδεσης της εφαρμογής με τον web server

Εδώ βλέπουμε το Toast που έχουμε ορίσει να βγαίνει αφού γίνει η σύνδεση και λάβουμε τα δεδομένα από τη βάση. Σε αντίθεση με την προηγούμενη εικόνα εδώ, αφού έγινε η σύνδεση, το κουμπί εμφανίστηκε.



Εικόνα 25: Το μήνυμα Toast που έχουμε ορίσει όταν τελειώσει η μεταφορά των δεδομένων από τη βάση

Στην αρχή της εφαρμογής χρησιμοποιούμε το ζεύγος συντεταγμένων που έχουμε περάσει χειροκίνητα. Οπότε αν πατήσουμε το κουμπί θα μας βγάλει τη στάση που είναι πιο κοντά σε αυτό το σημείο στο Γαλάτσι.

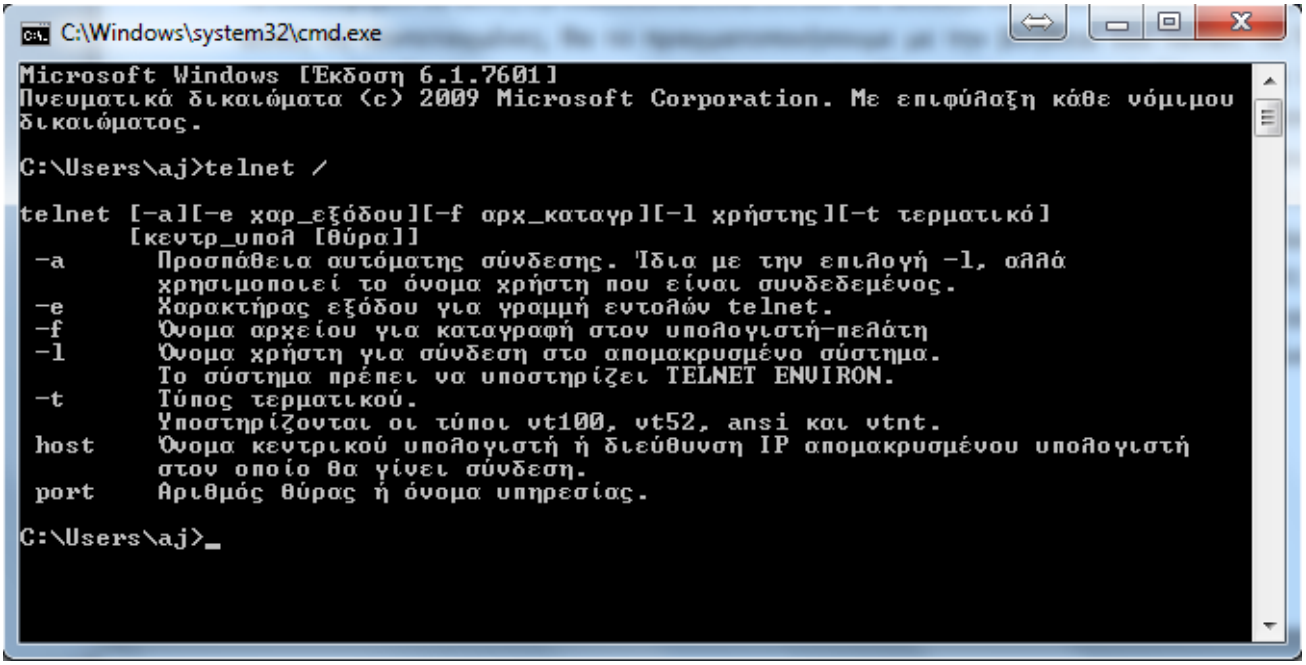


Εικόνα 26: Το αποτέλεσμα της εφαρμογής για το αρχικό ζεύγος συντεταγμένων.

Όντως η συγκεκριμένη στάση είναι η κοντινότερη. Παρόλα αυτά δεν μπορούμε να βασιστούμε μόνο σε αυτό το σημείο. Θα επιλέξουμε άλλα τρία σημεία από το google maps, θα πάρουμε τις ανάλογες συντεταγμένες και θα δούμε ότι το αποτέλεσμα της εφαρμογής είναι σωστό. Θα φροντίσουμε τα σημεία που θα επιλεγούν να είναι ξεκάθαρο σε ποια στάση βρίσκονται πιο κοντά.

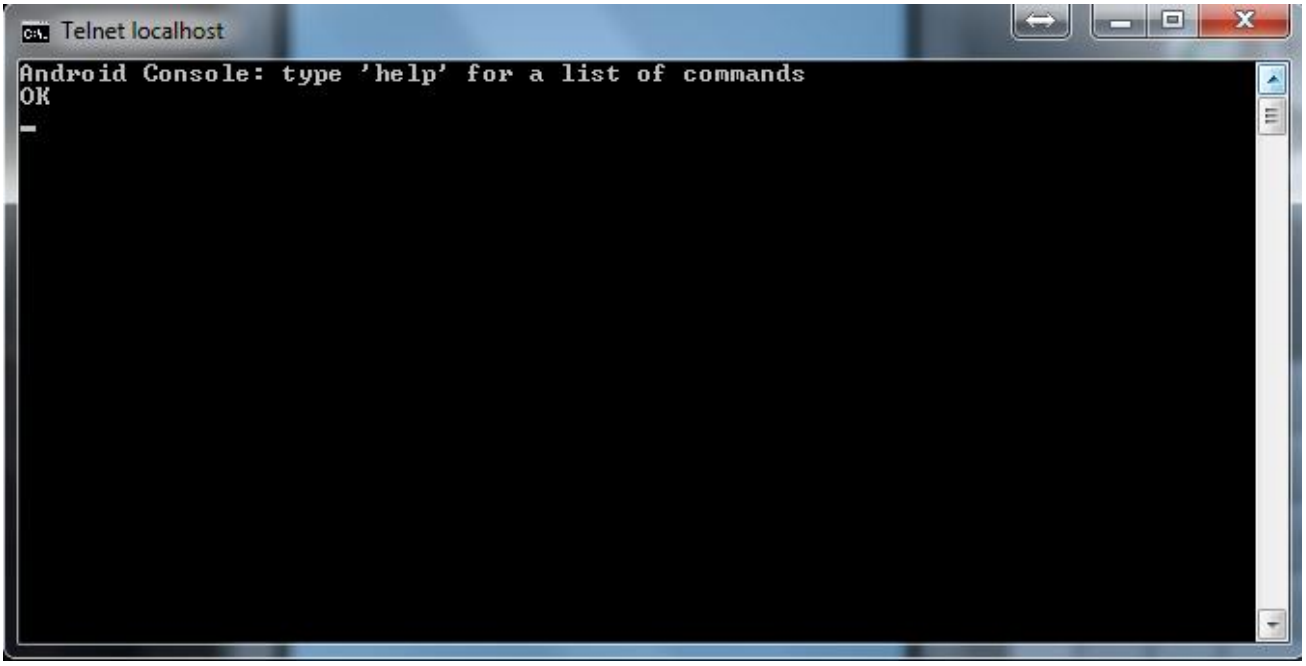
Το ενδιαφέρον σε αυτή τη διαδικασία είναι ότι δεν θα αλλάξω κάτι στον κώδικα για να βάλω χειροκίνητα αυτές τις συντεταγμένες, θα το πραγματοποιήσουμε με τη βοήθεια του Telnet. Το Telnet είναι ένα πρωτόκολλο δικτύου που χρησιμοποιείται στο Διαδίκτυο και σε τοπικά δίκτυα για να παρέχει μια αμφίδρομη κειμενοστραφή επικοινωνία χρησιμοποιώντας μια σύνδεση σε ένα εικονικό τερματικό. Χρησιμοποιούνταν παλαιότερα κατά κόρον για απομακρυσμένη ρύθμιση των παραμέτρων ενός δικτύου, καθώς παρείχε πρόσαβση σε ένα περιβάλλον γραμμής εντολών σε έναν απομακρυσμένο υπολογιστή. Στις μέρες μας ο όρος Telnet αναφέρεται κυρίως στο λογισμικό που υλοποιεί το κομμάτι client του πρωτοκόλλου. Αυτές οι client εφαρμογές είναι διαθέσιμες σχεδόν για όλες τις πλατφόρμες υπολογιστών. Η λειτουργία τους είναι να δημιουργούν μια σύνδεση με το πρωτόκολλο Telnet είτε μέσω γραμμής εντολών από την πλευρά του client (αυτό που θα κάνω και εγώ), είτε με προγραμματιστική διασύνδεση.

Πάμε να δούμε λοιπόν το πώς θα χρησιμοποιήσουμε το Telnet. Για να ανοίξουμε το Telnet στα Windows στο μενού έναρξη γράφουμε cmd για να ανοίξουμε τη γραμμή εντολών. Στη συνέχεια πληκτρολογούμε την εντολή telnet /



Εικόνα 27: Σύνδεση με το Telnet client

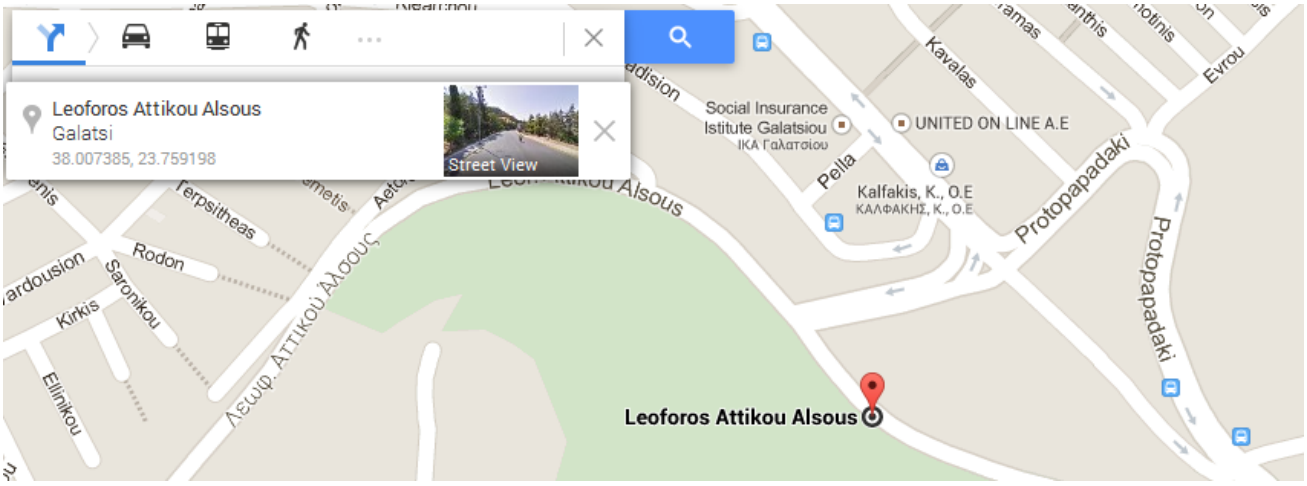
Αφού λοιπόν συνδεθήκαμε με το Telnet επιχειρούμε να συνδεθούμε με τον AVD που έχουμε δημιουργήσει. Για να γίνει αυτό πρέπει να δώσουμε την εντολή telnet localhost 5554



Εικόνα 28: Σύνδεση του Telnet με τον Emulator

Σε αυτή τη φάση έχει εγκατασταθεί η σύνδεση με τον Emulator και μπορούμε να του δώσουμε τις εντολές που επιθυμούμε. Η εντολή που θα χρησιμοποιήσουμε είναι η geo fix. Με αυτήν την εντολή δίνουμε στο GPS του Emulator τις συντεταγμένες που θα χρησιμοποιήσει. Αυτό που πρέπει να προσέξουμε είναι ότι η σειρά στο ζεύγος συντεταγμένων αλλάζει στη διατύπωση της εντολής. Η εντολή δηλαδή συντάσσεται ως εξής: geo fix longitude latitude

Το πρώτο σημείο λοιπόν που θα χρησιμοποιήσουμε είναι ακριβώς δίπλα στο τέρμα της γραμμής 622, τη στάση Τέρμα 622. Ας το δούμε στον χάρτη:

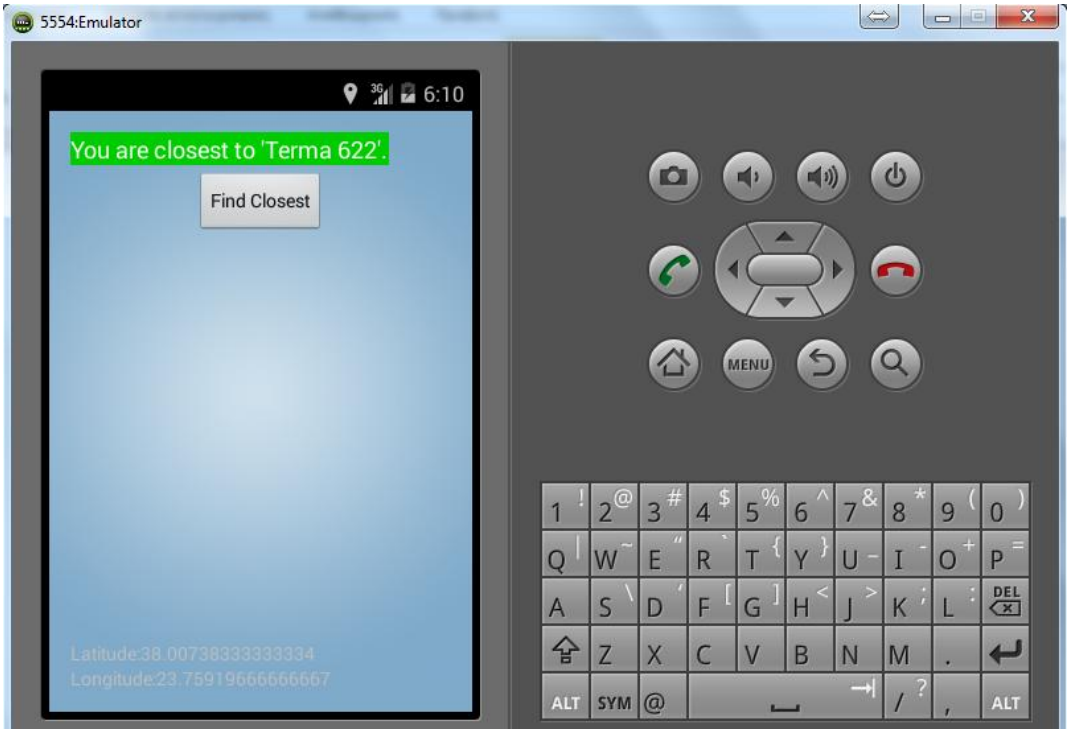


Εικόνα 29: Το πρώτο σημείο που θα δοκιμάσω το αποτέλεσμα της εφαρμογής

Βάζοντας σαν συντεταγμένες αυτές λοιπόν η εφαρμογή θα πρέπει να μου υποδείξει σαν την πιο κοντινή στάση την Τέρμα 622. Η εντολή είναι η παρακάτω

Εικόνα 30: Εισαγωγή συντεταγμένων μέσω Telnet

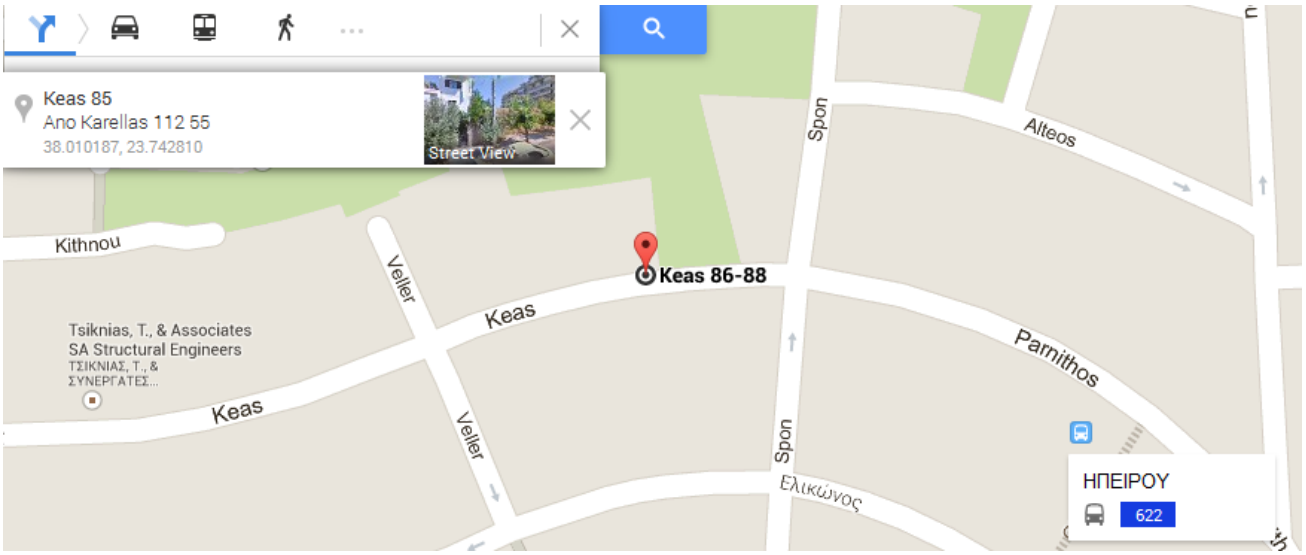
Στην εφαρμογή μας κάνουμε κλικ στο κουμπί Find Closest και παίρνουμε το ακόλουθο αποτέλεσμα.



Εικόνα 31: Το αποτέλεσμα της εφαρμογής για το πρώτο σημείο

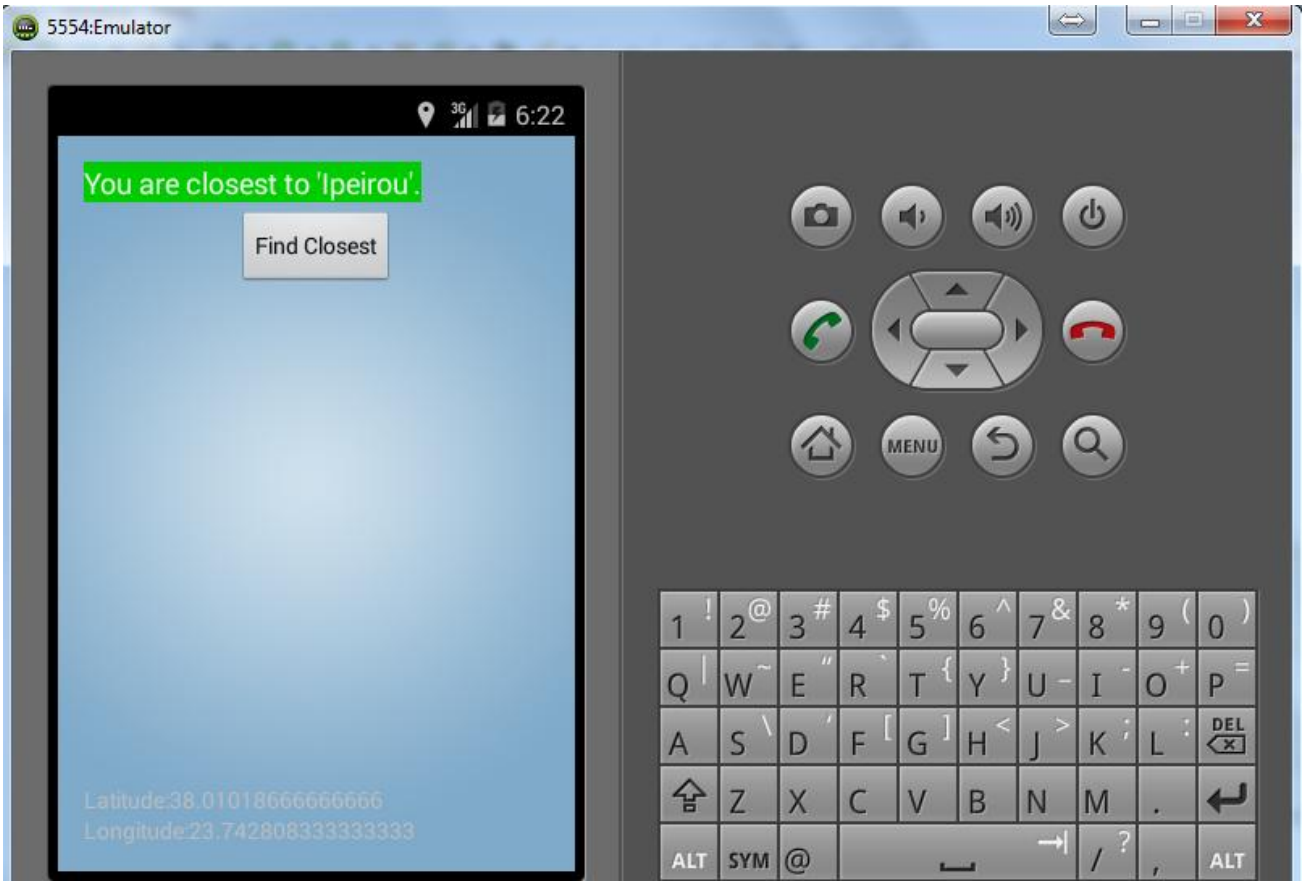
Παρατηρούμε ότι το αποτέλεσμα είναι σωστό. Επίσης κάτι άλλο που παρατηρούμε είναι ότι στο κάτω μέρος της εφαρμογής μας εμφανίστηκαν οι συντεταγμένες που δώσαμε στο Telnet. Αυτό θα συμβαίνει κάθε φορά που αλλάζουμε συντεταγμένες. Σε πραγματική συσκευή θα εμφανιζόταν αυτή η ένδειξη, όταν ο χρήστης θα άλλαζε θέση.

Πάμε να δούμε για το δεύτερο σημείο.



Εικόνα 32: Το δεύτερο σημείο που θα δοκιμάσω το αποτέλεσμα της εφαρμογής

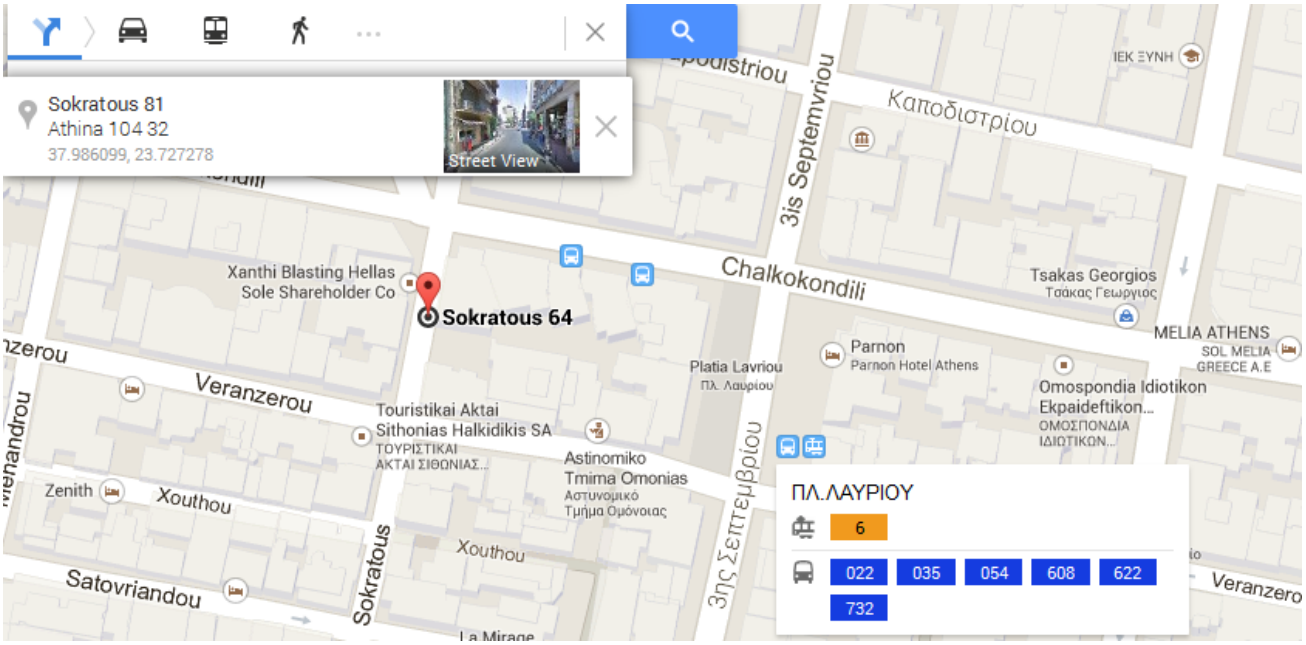
Εισάγουμε τις συντεταγμένες από το Telnet και περιμένουμε σαν αποτέλεσμα τη στάση Ηπείρου.



Εικόνα 33: Το αποτέλεσμα της εφαρμογής για το δεύτερο σημείο

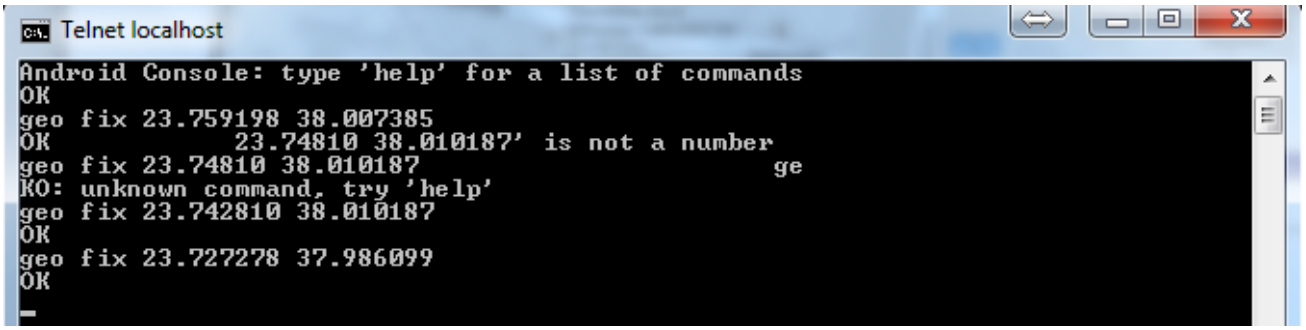
Όντως το αποτέλεσμα και για το δεύτερο σημείο είναι σωστό, καθώς από το σημείο που είδαμε και στον χάρτη πιο πάνω η κοντινότερη στάση είναι η Ηπείρου.

Κάνουμε και την τελευταία δοκιμή για να επιβεβαιωθούμε ότι είναι όλα σωστά. Το τρίτο σημείο είναι αυτό:



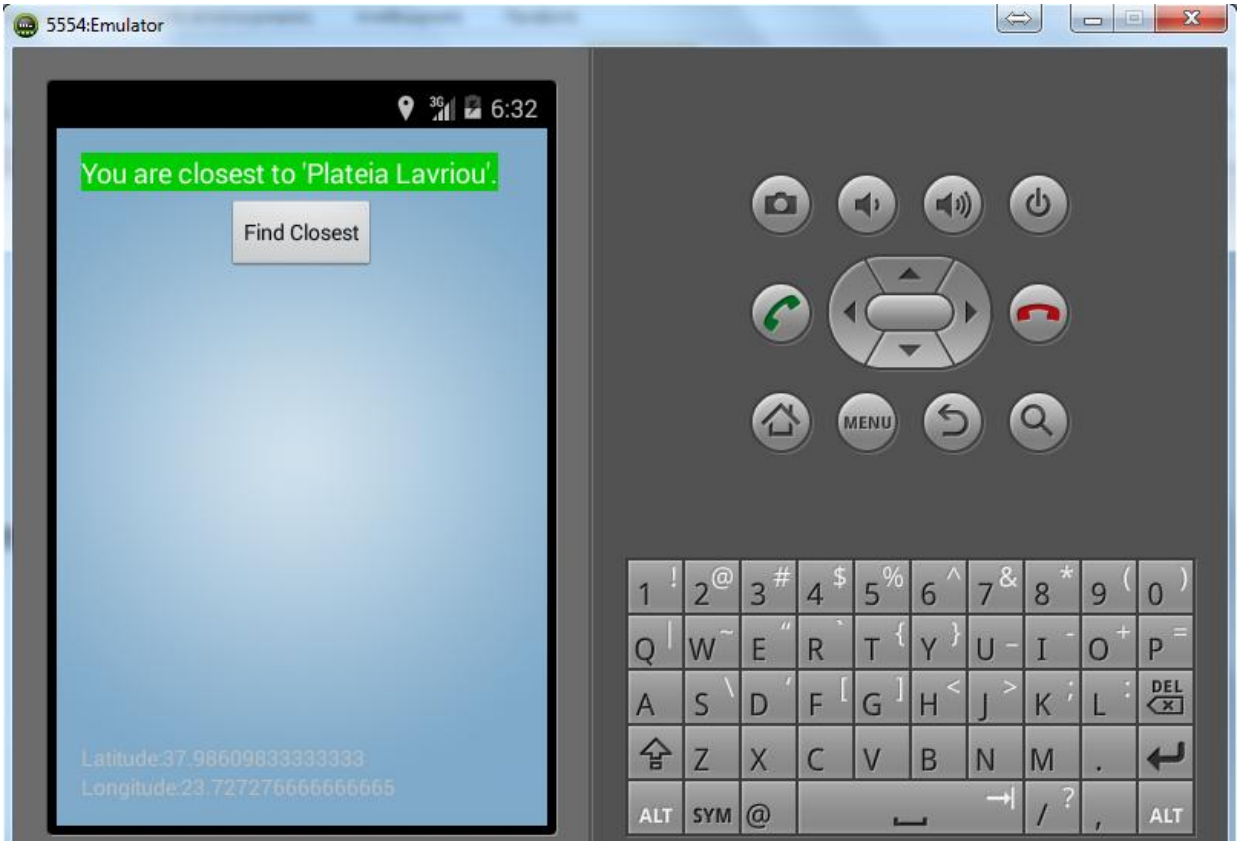
Εικόνα 34: Το τρίτο σημείο που θα δοκιμάσω το αποτέλεσμα της εφαρμογής

Το αποτέλεσμα που περιμένουμε είναι η στάση Πλατεία Λαυρίου δίνοντας την παρακάτω εντολή:



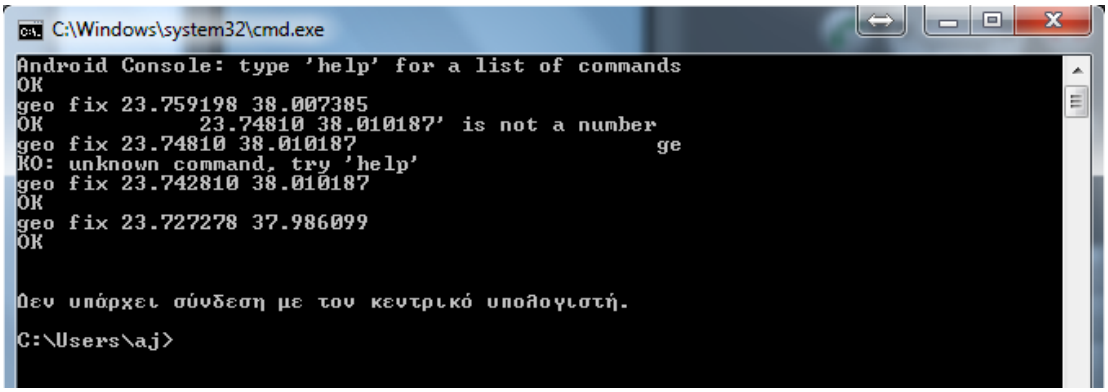
Εικόνα 35: Εισαγωγή συντεταγμένων μέσω της εντολής geo fix στο Telnet

Ξαναπατώντας το κουμπί Find Closest λοιπόν, παίρνουμε το εξής αποτέλεσμα:



Εικόνα 36: Το αποτέλεσμα της εφαρμογής για το τρίτο σημείο

Οπότε είναι σίγουρο πλέον ότι η εφαρμογή λειτουργεί απόλυτα σωστά. Τερματίζοντας τον Emulator, τερματίζει και η σύνδεση Telnet που είχαμε δημιουργήσει.



Εικόνα 37: Κλείσιμο σύνδεσης με τον Emulator

ΕΠΙΛΟΓΟΣ

Η συγκεκριμένη εφαρμογή είναι στην αρχική (Beta) της μορφή. Για να γινόταν εμπορεύσιμη υπάρχουν κάποια πράγματα που θα μπορούσα να προσθέσω. Το πρώτο και πιο σημαντικό από αυτά, είναι να κατασκεύαζα μια βάση δεδομένων η οποία θα ήταν ενσωματωμένη στην εφαρμογή ώστε να μην χρειάζεται να υπάρχει σύνδεση με κάποιο δίκτυο για να τρέξει. Άλλωστε όπως αναφέρθηκε και στη σχεδίαση της εφαρμογής το ιδανικό θα ήταν να αντλεί δεδομένα και από το δίκτυο αλλά και από την υπάρχουσα βάση δεδομένων.

Κάτι ακόμα το οποίο αφορά τόσο το γραφικό κομμάτι όσο και το λειτουργικό, θα ήταν να υπάρχει ένα κουμπί που όταν η εφαρμογή θα μας έδινε το αποτέλεσμα θα εμφάνιζε στην οθόνη τη στάση σε ένα δυναμικό χάρτη. Έτσι ο χρήστης θα μπορούσε να προσανατολιστεί καλύτερα για τον τρόπο με τον οποίο θα μεταβεί στη συγκεκριμένη στάση.

Αυτές είναι οι πιο βασικές μελλοντικές αναβαθμίσεις που θα μπορούσαν να προστεθούν. Πέρα από αυτές, υπάρχουν και άλλες, οι οποίες θα μπορούσαν να αναβαθμίσουν το γραφικό περιβάλλον της εφαρμογής. Ίσως η εισαγωγή κάποιου μενού που θα έδινε και άλλες επιλογές στον χρήστη. Για παράδειγμα, εφόσον η εφαρμογή του εμφάνιζε στο χάρτη την κοντινότερη στάση, να του υπολογίζει μια διαδρομή για το πώς και το πότε θα φτάσει σε αυτήν.

Επίσης κάτι το οποίο θα μπορούσε να γίνει και να δώσει άλλον αέρα στην εφαρμογή είναι να φαίνονται σε real-time τα λεωφορεία με ενσωματωμένους πομπούς ούτως ώστε να υπολογίζει ο χρήστης πόση ώρα του απομένει για να μεταβεί στη στάση και να προλάβει το λεωφορείο. Φυσικά αυτό απαιτεί την ανάλογη μελέτη και τη συνεργασία με τους αρμόδιους φορείς.

Από τη στιγμή που όλο το κομμάτι της εφαρμογής που υλοποίησα είναι ορθό και μας δίνει το σωστό αποτέλεσμα όλα τα υπόλοιπα είναι αναβαθμίσεις που θα την κάνουν πιο φιλική στο χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Barry Burd, Beginning Programming with Java For Dummies, 3rd Edition. Hoboken, NJ: John Wiley & Sons, 2012
2. Donn Felker with Joshua Dobbs, Android Application Development For Dummies. Hoboken, NJ: Wiley Publishing, 2011
3. Bruce Eckel, Thinking in Java, 4th Edition. Prentice Hall, 2006
4. Reto Meier, Professional Android 4 Application Development. Indianapolis, IN: John Wiley & Sons, 2012
5. Jeff Friesen, Learn Java for Android Development. California: Apress, 2013
6. Ed Burnette, Hello, Android Introducing Google's Mobile Development Platform 3rd Edition. United States of America, USA: Pragmatic Programmers, 2010
7. Lauren Darcey, Shane Conder, Ανάπτυξη Εφαρμογών με το Android. Αθήνα: Εκδόσεις Μ. Γκιούρδας, 2011