ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής υπολογισμού σημείων ενδιαφέροντος σε περιβάλλον Android

Τζαβάρας Νικολάου Σπύρος

Επιβλέποντες: Αθανασία Αλωνιστιώτη, Επίκουρος καθηγητής

ΑΘΗΝΑ

Ιούνιος 2015

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής υπολογισμού σημείων ενδιαφέροντος σε περιβάλλον Android

Τζαβάρας Νικολάου Σπύρος
Α.Μ.: 1115200700124

Επιβλέποντες: Αθανασία Αλωνιστιώτη, Επίκουρος καθηγητής

ΠΕΡΙΛΗΨΗ

Στα πλαίσια της πτυχιακής εργασίας αναπτύχθηκε μια εφαρμογή η οποία λειτουργεί σε συσκευές με λειτουργικό σύστημα Android. Στόχος της εφαρμογής ήταν η εξαγωγή δεδομένων και ο υπολογισμός σημείων ενδιαφέροντος (Points of Interest), με βάση στίγματα που προέκυπταν από τον χρήστη της εφαρμογής, ή από όλους τους χρήστες μαζί (stay points και stay regions αντίστοιχα).
Για την εξαγωγή των σημείων ενδιαφερόντων και για τη λήψη στιγμάτων από το χρήστη, είναι απαραίτητη η λειτουργία GPS στην Android συσκευή. Η λήψη στιγμάτων μπορεί να υλοποιηθεί και με σύνδεση σε ασύρματο δίκτυο, αλλά προτιμήθηκε το GPS γιατί δίνει πιο αξιόπιστα και ακριβή αποτελέσματα.
Η παρούσα εργασία έχει αναπτυχθεί σε μοντέλο πελάτη – εξυπηρέτη (client-server architecture).Μόλις ο χρήστης της εφαρμογής επιλέξει να σταματήσει τη λειτουργία του monitoring, όπου εξάγονται τα στίγματα, το σύνολο των στιγμάτων αυτών αποστέλλεται στον εξυπηρέτη. Ο εξυπηρέτης από τη μεριά του επεξεργάζεται τα στίγματα αυτά, παράγει ένα σύνολο σημείων διαμονής(stay points) βάσει ενός αλγορίθμου, και στη συνέχεια αφού ομαδοποιεί τα δεδομένα αυτά, εξάγει τα σημεία ενδιαφέροντος(stay regions), τα οποία αφορούν όλους τους χρήστες της εφαρμογής.
Αξίζει να αναφερθεί ότι έχει αναπτυχθεί ένα web service (restful) για την επικοινωνία πελάτη- εξυπηρέτη μέσω του πρωτοκόλου http. Τέλος, υπάρχει μία βάση δεδομένων και στη μεριά του εξυπηρέτη, αλλά και στη μεριά του πελάτη, έτσι ώστε όταν η επικοινωνία με τον εξυπηρέτη δεν είναι εφικτή , να 'αντλεί' τα δεδομένα που ήδη του έχουν σταλεί από την τοπική βάση.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Εξαγωγή σημείων ενδιαφέροντος σε Android περιβάλλον
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Android, GPS, σημεία ενδιαφέροντος, web service

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω τον μεταπτυχιακό φοιτητή Νίκο Μπομπέτση για την καθοδήγησή του και τις χρήσιμες συμβουλές που μου παρείχε.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ.....σελ. 9
1.ANDROID.....σελ. 10
1.1.Εισαγωγή.....σελ. 10
1.2.Κυκλοφορίες(releases).....σελ. 11
1.3.Στοίβα Λογισμικού Android και Dalvik Virtual Machine.....σελ. 12
1.3.1. Στοίβα Λογισμικού.....σελ. 13
1.3.2. Dalvik Virtual Machine.....σελ. 14
1.4.Προγραμματισμός σε Android.....σελ. 15
1.4.1. Απαραίτητες γνώσεις και εργαλεία.....σελ. 15
1.4.2. Manifest File.....σελ. 17
1.4.3. Activities.....σελ. 18
1.4.4. Fragments.....σελ. 21
1.4.5. Άλλα components.....σελ. 24
2.WEB SERVICE.....σελ. 25
3. Η ΕΦΑΡΜΟΓΗ HUMAN POINTS OF INTEREST.....σελ. 27
3.1.Εισαγωγή.....σελ. 27
3.2.Περιβάλλον υλοποίησης της εφαρμογής.....σελ. 28
3.3.Client Side.....σελ. 28
3.3.1. Γραφικό και λειτουργικότητα εφαρμογής.....σελ. 29
3.3.2. Web Service.....σελ. 35
3.3.3. Βάση Δεδομένων.....σελ. 37
3.3.4. Θέματα Προγραμματισμού.....σελ. 38
3.4.Server Side.....σελ. 41
3.4.1. Web Service.....σελ. 41
3.4.2. Λειτουργικότητα Server.....σελ. 44
3.4.3. Βάση Δεδομένων.....σελ. 47
3.4.4. Θέματα Προγραμματισμού.....σελ. 49
3.5.Πειραματική Εκτέλεση Εφαρμογής.....σελ. 51

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1: Android Logo.....σελ. 10
Εικόνα 2: Αλλαγές στο User Interface, Android 1.5.....σελ. 11
Εικόνα 3: Android Software Stack.....σελ. 13
Εικόνα 4: Java VM vs Dalvik VM.....σελ. 14
Εικόνα 5: Εμφανιζόμενο μήνυμα κατά τη σύνδεση android συσκευής με υπολογιστή για έλεγχο σφαλμάτων USB.....σελ. 16
Εικόνα 6: Παράδειγμα ενός AndroidManifest.xml αρχείου.....σελ. 17
Εικόνα 7: Κύκλος ζωής ενός Activity.....σελ. 19
Εικόνα 8: layout αρχείο σε design mode.....σελ. 21
Εικόνα 9: layout αρχείο σε text mode.....σελ. 21
Εικόνα 10: Κύκλος ζωής ενός Fragment.....σελ. 22
Εικόνα 11: User Interfaces από Fragments σε κινητό και tablet αντίστοιχα.....σελ. 22
Εικόνα 12: Toast.....σελ. 24
Εικόνα 13: AlertDialog.....σελ. 24
Εικόνα 14: Επικοινωνία τερματικών μέσω Internet.....σελ. 25
Εικόνα 15: Client-Server model.....σελ. 25
Εικόνα 16: Εξαγωγή stay points και stay regions(Human Points Of Interest).....σελ. 28
Εικόνα 17: Ιεραρχία packages και κλάσεων.....σελ. 29
Εικόνα 18: Ιεραρχία layout αρχείων.....σελ. 29
Εικόνα 19: Μέρος του AndroidManifest.xml αρχείου.....σελ. 29
Εικόνα 20: Αρχική σελίδα εφαρμογής.....σελ. 30
Εικόνα 21: Αρχική σελίδα με επιλεγμένη την απομνημόνευση στοιχείων.....σελ. 30
Εικόνα 22: Προτροπή στο χρήστη να επιλέξει άλλο username.....σελ. 31
Εικόνα 23: MapsActivity μετά τη σύνδεση του χρήστη.....σελ. 31
Εικόνα 24: Navigation Drawer μενού.....σελ. 32
Εικόνα 25: Εμφάνιση καινούριου σημείου από τη λειτουργία monitoring.....σελ. 33
Εικόνα 26: Δημιουργία trajectory στο χάρτη της εφαρμογής.....σελ. 33
Εικόνα 27: Αναπαράσταση stay point.....σελ. 34
Εικόνα 28: Αναπαράσταση stay region.....σελ. 34
Εικόνα 29: Προτροπή χρήστη να ενεργοποιήσει το GPS για το 'Start Monitoring'.....σελ. 35
Εικόνα 30: Υβριδική όψη χάρτη.....σελ. 35
Εικόνα 31: Εισαγωγή SHA-1 και package.....σελ. 39
Εικόνα 32: Δομή packages και κλάσεων του server.....σελ. 41
Εικόνα 33: Libraries που χρησιμοποιήθηκαν και web.xml αρχείο.....σελ. 41
Εικόνα 34: Δημιουργία ενός Servlet Container που παρέχεται από το Jersey.....σελ. 42
Εικόνα 35: Ενδεικτική κλάση server.....σελ. 42
Εικόνα 36: Αίτημα σύνδεσης με server.....σελ. 42
Εικόνα 37: Μέθοδος που διαβάζει το JSON από τον client.....σελ. 43
Εικόνα 38: Μέθοδος που δημιουργεί το JSONArray των stay points που θα σταλούν στον πελάτη.....σελ. 44
Εικόνα 39: Μέθοδος που δημιουργεί το JSONArray των stay regions που θα σταλούν στον πελάτη.....σελ. 44
Εικόνα 40: Μέθοδος που εκτελείται μετά από αίτημα του χρήστη για εγγραφή.....σελ. 45
Εικόνα 41: Μέθοδος που εκτελείται μόλις ο server λάβει από έναν χρήστη l. points..σελ. 45
Εικόνα 42: Συνέχεια μεθόδου εικόνας 41.....σελ. 46
Εικόνα 43: Μέθοδος sendStayPoints.....σελ. 46
Εικόνα 44: Μέθοδος createPOIDB.....σελ. 47
Εικόνα 45: Tables της βάσης δεδομένων POIDB.....σελ. 48
Εικόνα 46: Μέθοδος findStayPoints σε ψευδογλώσσα.....σελ. 49

Εικόνα 47: Μέθοδος υπολογισμού απόστασης 2 σημείων.....σελ. 50
Εικόνα 48: Υλοποίηση αλγορίθμου DBScan.....σελ. 50
Εικόνα 49: Location Point που βρέθηκε κατά τη λειτουργία Monitoring.....σελ. 51
Εικόνα 50: Trajectory που έχει σχηματιστεί από 4 location points.....σελ. 51
Εικόνα 51: Logcat server, μόλις λάβει τα location points.....σελ. 51
Εικόνα 52: Stay point που προέκυψε.....σελ. 52
Εικόνα 53: Stay region μετά την εφαρμογή του DBScan.....σελ. 52

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1: Πίνακας εκδόσεων Android μετά την έκδοση 1.5 έως Ιούνιο 2015.....σελ. 12

ΠΡΟΛΟΓΟΣ

Η παρούσα εργασία διενεργήθηκε ως πτυχιακή εργασία για το τμήμα Πληροφορικής και Τηλεπικοινωνιών του Καποδιστριακού Πανεπιστημίου Αθηνών-ΕΚΠΑ.

1. Android

1.1 Εισαγωγή

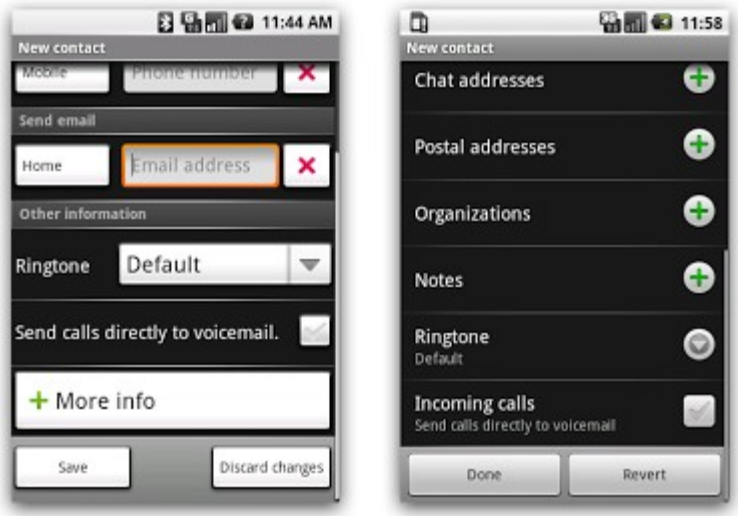
Τα τελευταία χρόνια η χρήση των επονομαζόμενων έξυπνων συσκευών(smartphones), είτε αυτές είναι κινητά είτε tablets, έχει αυξηθεί σημαντικά. Κάθε έξυπνη συσκευή έχει κάποιο λειτουργικό σύστημα πάνω στο οποίο εκτελούνται όλες οι εφαρμογές της. Τα πιο γνωστά λειτουργικά συστήματα για έξυπνες συσκευές αυτή την στιγμή είναι τα Android, iOS και windows phone. Οι κατασκευαστές των έξυπνων συσκευών αυξάνουν συνεχώς τις συνολικές τους δυνατότητες, με αποτέλεσμα η δυνατότητα χρήσης όλο και πιο απαιτητικού λογισμικού από τους χρήστες. Πλέον υπάρχουν κινητά που συνδέονται στο διαδίκτυο, λειτουργούν ως GPS, καλούν αυτόματα ταξί. Όλες αυτές οι λειτουργίες είναι εφικτές με την χρήση εφαρμογών που έχουν κατασκευαστεί ειδικά για τις έξυπνες συσκευές και ονομάζονται εφαρμογές κινητών (mobile applications).Τα mobile applications είναι συνήθως διαθέσιμα μέσω πλατφορμών διανομής εφαρμογών, που άρχισαν να εμφανίζονται το 2008 και συνήθως λειτουργούν από τον ιδιοκτήτη του λειτουργικού συστήματος της συσκευής , όπως το Apple App Store, Google Play Store και το Windows Phone Store. Ορισμένες εφαρμογές είναι δωρεάν, ενώ άλλες πρέπει να αγοραστούν.
Το Android λοιπόν, το οποίο τρέχει τον πυρήνα του λειτουργικού συστήματος Linux, και αρχικά αναπτύχθηκε από τη google, είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα smartphones και τα tablets, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις(Android TV) και αυτοκίνητα(Android Auto) . Αξίζει να αναφερθεί ότι είναι το πιο ευρέως διαδεδομένο λογισμικό. Ενδεικτικά οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS μαζί.
Σχετικά με την άδεια χρήσης του Android, ο κώδικας του είναι ανοιχτός σε όλους, ωστόσο υπάρχουνε και κάποια ιδιόκτητα μέρη.

Εικόνα 1: Android Logo

1.2 Κυκλοφορίες (releases)

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιρειών, εταιρειών λογισμικού και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας.
Η εταιρεία η οποία δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android ήταν η google, υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.
Όπως σε κάθε λειτουργικό σύστημα, έτσι και στο Android, κρίνεται επιτακτική η ανάγκη για κυκλοφορίες νέων εκδόσεων, οι οποίες περιέχουν καινούρια πρότυπα, βελτιώνουν κάποια άλλα που υφίστανται ήδη ή ακόμα και διορθώνουν κάποια λάθη (κοινώς bugs) που βρίσκονται στις προηγούμενες εκδόσεις. Είναι εύκολα αντιληπτό, ότι η ανάγκη των χρηστών του Android για καινούριες ή καλύτερες παροχές, είναι άρρηκτα συνδεδεμένη με την κυκλοφορία νέων εκδόσεων.
Στις 30 Απριλίου 2009 κυκλοφόρησε η επίσημη ενημέρωση έκδοσης 1.5 για το Android. Αυτή αποτελείτο από πολλά νέα χαρακτηριστικά, μερικά από τα οποία παρατίθενται παρακάτω:

- Δυνατότητα καταγραφής κινούμενης εικόνας με τη χρήση της αντίστοιχης λειτουργίας του τηλεφώνου
- Επανασχεδιασμένο λογισμικό πληκτρολογίου με λειτουργία αυτόματης συμπλήρωσης κειμένου
- Νέα widgets και φάκελοι που μπορούν να τοποθετηθούν στην επιφάνεια εργασίας
- Διευρυμένη λειτουργία αντιγραφής / επικόλλησης για να περιλαμβάνει δικτυακές διευθύνσεις
- Εφέ αλλαγής οθονών και μενού
- Σημαντικές βελτιώσεις στο γραφικό περιβάλλον



Εικόνα 2: Αλλαγές στο User Interface, Android 1.5

Παρακάτω παρατίθεται ένας πίνακας, ο οποίος περιλαμβάνει τις κυκλοφορίες του Android μετά την έκδοση 1.5, έως και σήμερα (Ιούνιο 2015). Είναι αξιοσημείωτο το γεγονός ότι τα ονόματα των εκδόσεων προέρχονται από νόστιμες λιχουδιές, ενώ ταυτόχρονα είναι ταξινομημένες αλφαβητικά.

Πίνακας 1: Πίνακας εκδόσεων Android μετά την έκδοση 1.5 έως Ιούνιο 2015

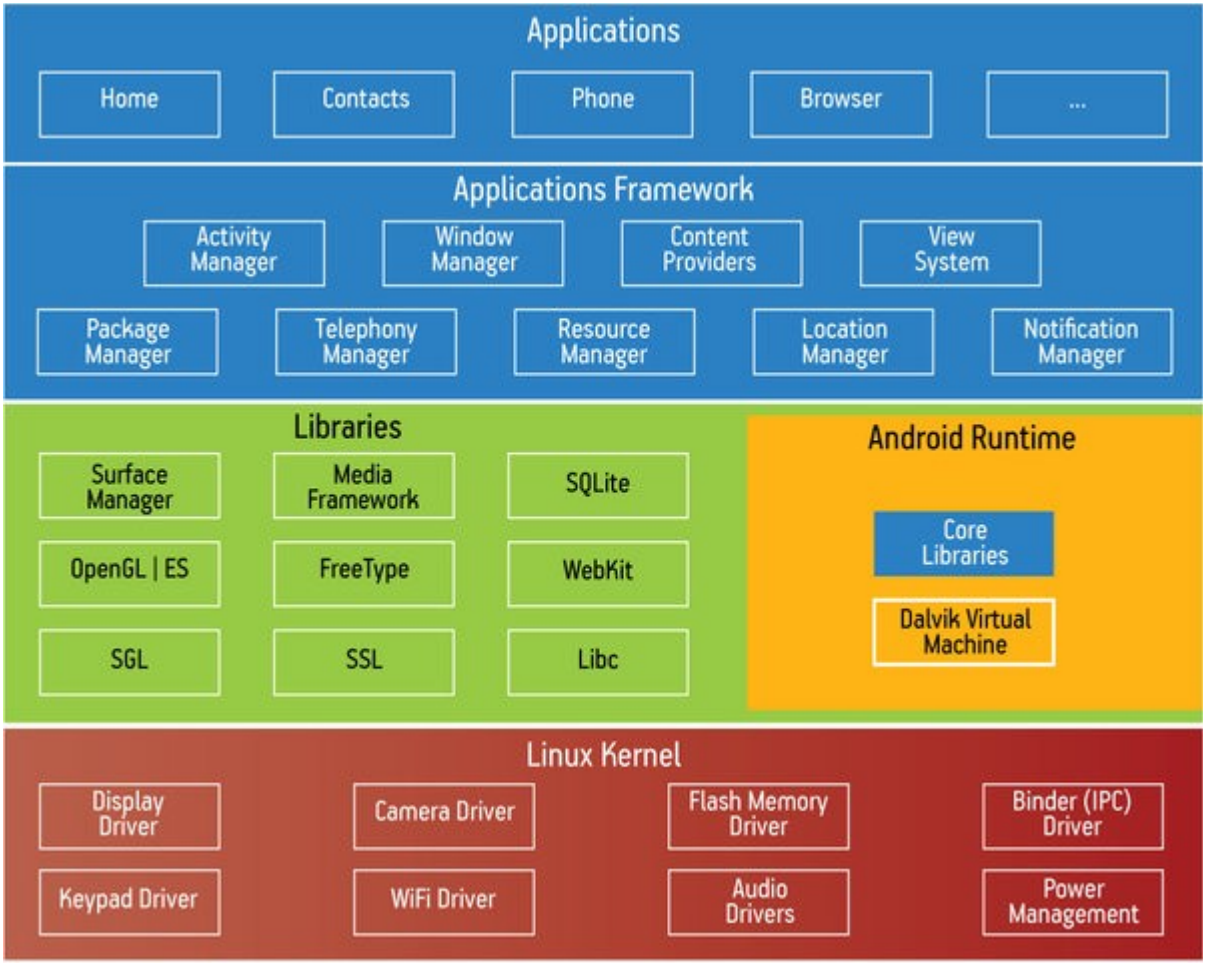
Έκδοση	**Κωδική ονομασία**	**Ημερομηνία**	**API level**
1.6	*Donut*	15 Σεπτεμβρίου 2009	4
2.0-2.1	*Eclair*	26 Οκτωβρίου 2009	7
2.2	*Froyo*	20 Μαΐου 2010	8
2.3.3-2.3.7	*Gingerbread*	9 Φεβρουαρίου 2011	10
3.2	*Honeycomb*	15 Ιουλίου 2011	13
4.0.3 - 4.0.4	*Ice Cream Sandwich*	16 Δεκεμβρίου 2011	15
4.1.x	Jelly Bean	9 Ιουλίου 2012	16
4.2.x	Jelly Bean	13 Νοεμβρίου 2012	17
4.3	Jelly Bean	24 Ιουλίου 2013	18
4.4	*KitKat*	31 Οκτωβρίου 2013	19
5.0	Lollipop	3 Νοεμβρίου του 2014	21
5.1	Lollipop	9 Μαρτίου 2015	22

1.3 Στοίβα λογισμικού Android και Dalvik Virtual Machine

Σε αυτό το κεφάλαιο γίνεται μία σύντομη ανάλυση της δομής και ιεραρχίας της στοίβας του Android, καθώς και το ρόλο του Dalvik Virtual Machine. Για να κατανοήσει κάποιος αυτή την ανάλυση πρέπει να κατέχει βασικές γνώσεις προγραμματισμού , αλλά ιδιαίτερα γνώσεις αντικειμενοστραφούς προγραμματισμού.
Το Android είναι μια “στοίβα” λογισμικού για φορητές συσκευές η οποία περιλαμβάνει το λειτουργικό σύστημα, τις εφαρμογές καθώς και το επίπεδο λογισμικού (middleware) που διασυνδέει το λειτουργικό σύστημα με τις εφαρμογές.

1.3.1 Στοίβα λογισμικού

Για να κατανοήσει κανείς τον τρόπο με τον οποίο λειτουργεί προγραμματιστικά μια android εφαρμογή πρέπει πρώτα να μελετήσει τη στοίβα λογισμικού του Android, γνωστη και ως Android Software Stack. Παρακάτω παρατίθεται μία σύνοψη αυτής της στοίβας, η οποία μεταξύ άλλων περιλαμβάνει διάφορες βιβλιοθήκες C/C++ , όπως και διάφορα framework APIs, τα οποία δίνουν τη δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν διάφορα χαρακτηριστικά της Android συσκευής. Το Android βασίζεται στην έκδοση 2.6 του πυρήνα του Linux για να υλοποιήσει υπηρεσίες όπως ασφάλεια, διαχείριση μνήμης, διαχείριση διεργασιών και δικτύου. Επίσης, ο πυρήνας Linux είναι υπεύθυνος για την επικοινωνία της φορητής συσκευής με τα υπόλοιπα επίπεδα της στοίβας λογισμικού.

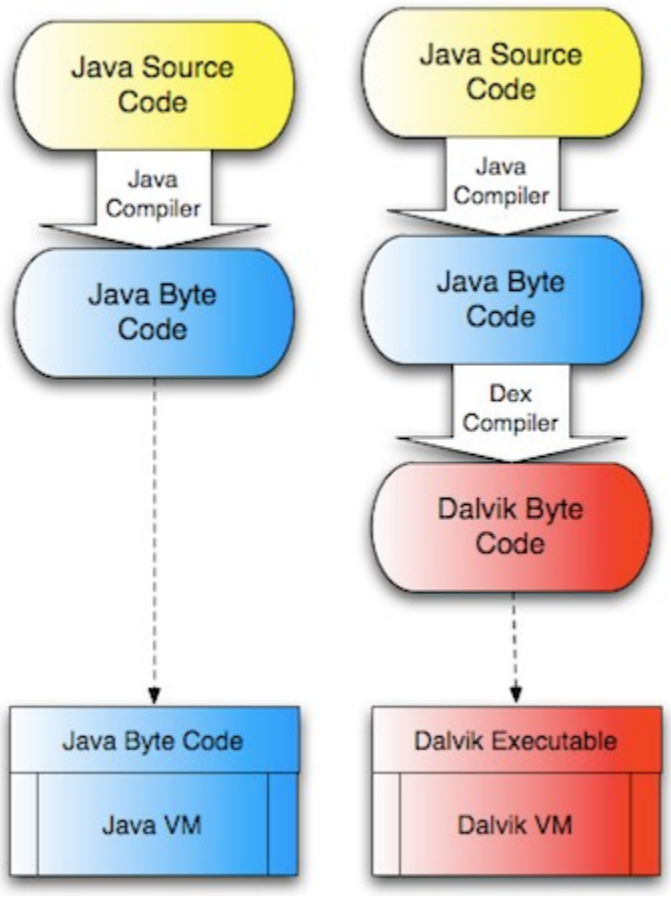


Εικόνα 3: Android Software Stack

Από το παραπάνω σχήμα διακρίνουμε ότι στη βάση της στοίβας βρίσκεται ο πυρήνας του Linux, στον οποίο περιλαμβάνονται όλα τα hardware drivers, αλλά και η διαχείριση μνήμης και δικτύου. Στην κορυφή της βάσης βρίσκονται όλες οι εφαρμογές που αλληλεπιδρούν με το χρήστη.

1.3.2 Dalvik Virtual Machine

Κάτι ακόμα που πρέπει να γίνει πλήρως κατανοητό είναι η λειτουργία και η χρησιμότητα του Dalvik Virtual Machine (εικονική μηχανή Dalvik), το οποίο είναι ένα ελεύθερο λογισμικό που δημιουργήθηκε από τον Dan Bornstein. Το Dalvik χαρακτηρίζεται με μεγαλύτερη ακρίβεια ως ένα process virtual machine στο λειτουργικό σύστημα Android, το οποίο είναι υπεύθυνο για την εκτέλεση εφαρμογών που έχουν δημιουργηθεί για αυτό το λειτουργικό και έτσι αποτελεί αναπόσπαστο κομμάτι της στοίβας λογισμικού του Android.
Ας δούμε όμως πώς λειτουργεί το Dalvik. Κάθε Android εφαρμογή εκτελεί τη δική της διεργασία, με το δικό της στιγμιότυπο του Dalvik VM, το οποίο επωφελείται από το λειτουργικό σύστημα Linux για απομόνωση ασφάλειας. Η εφαρμογή, η οποία είναι (συνήθως) γραμμένη σε γλώσσα Java, μεταγλωττίζεται σε bytecode για το Java Virtual Machine (γνωστό και ως JVM). Αυτά τα bytecode μεταφράζονται εν συνεχεία σε bytecode για το Dalvik και αποθηκεύονται σε .dex και .odex αρχεία.
Τέλος, αξίζει να σημειωθεί ότι το Dalvik VM είναι βασισμένο στον πυρήνα του Linux OS για θέματα όπως threading και διαχείριση μνήμης χαμηλού επιπέδου.



Εικόνα 4 : Java VM vs Dalvik VM

Να αναφέρουμε, ότι από την έκδοση 5.0 “Lollipop” ένα άλλο περιβάλλον εκτέλεσης, το ART αντικατέστησε πλήρως το Dalvik Virtual Machine.

1.4 Προγραμματισμός σε Android

1.4.1 Απαραίτητες γνώσεις και εργαλεία

Όπως αναφέρθηκε και στην προηγούμενη ενότητα, ορισμένες γνώσεις είναι απαραίτητες προκειμένου κάποιος να μπορέσει να αναπτύξει μία εφαρμογή για Android συσκεύη. Πιο συγκεκριμένα, γνώσεις προγραμματισμού σε γλώσσα Java είναι αναγκαία, όπως επίσης και γενικότερες γνώσεις πάνω στον αντικειμενοστραφή προγραμματισμό. Έννοιες όπως ενθυλάκωση ή κληρονομικότητα πρέπει να είναι απόλυτα σαφείς στον προγραμματιστή.

Java

Για την υλοποίηση μιας android εφαρμογής είναι απαραίτητη η εγκατάσταση της Java στον υπολογιστή του προγραμματιστή, καθώς τα περισσότερα από τα αρχεία της εφαρμογής περιέχουν κώδικα σε γλώσσα προγραμματισμού java. Ενδεικτικά , η εγκατάσταση της java σε έναν υπολογιστή με λειτουργικό σύστημα linux, πραγματοποιείται με την εξής εντολή:

sudo apt-get install sun-java6-jdk

Ολοκληρωμένο Περιβάλλον Ανάπτυξης (IDE)

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment) είναι μία σουίτα λογισμικού που βοηθάει στην ανάπτυξη προγραμμάτων υπολογιστή. Ένα IDE περιλαμβάνει συνήθως όλα τα απαραίτητα εργαλεία για την ανάπτυξη ενός προγράμματος, όπως επεξεργαστή πηγαίου κώδικα, μεταγλωττιστή, εργαλεία αυτόματης παραγωγής κώδικα, συνδέτη, αποσφαλματωτή και σύστημα ελέγχου εκδόσεων.
Το πιο διαδεδομένο IDE για προγραμματισμό android εφαρμογών είναι το Android Studio.
Το Android Studio είναι ένα ελεύθερο λογισμικό (open source) και διατίθεται εδώ και μόλις δύο χρόνια υπό τους όρους της Apache License. Έχει δημιουργηθεί αποκλειστικά για προγραμματισμό android εφαρμογών και είναι διαθέσιμο προς λήψη για όλα τα λειτουργικά συστήματα.
Ένα άλλο ευρέως διαδεδομένο IDE είναι το Eclipse, το οποίο ωστόσο δεν έχει δημιουργηθεί αποκλειστικά για ανάπτυξη android εφαρμογών. Αντιθέτως, με την χρήση του κατάλληλου plugin, το Eclipse μπορεί να υποστηρίξει πάρα πολλές γλώσσες προγραμματισμού, όπως C, C++ , Perl, Python, Javascript, Ruby κα. Το plugin που είναι απαραίτητο για προγραμματισμό σε android είναι το Eclipse ADT ή αλλιώς Eclipse Android Development Tools, το οποίο μπορεί να εγκατασταθεί ακολουθώντας τη διαδικασία στον παρακάω σύνδεσμο :

http://developer.android.com/sdk/installing/installing-adt.html

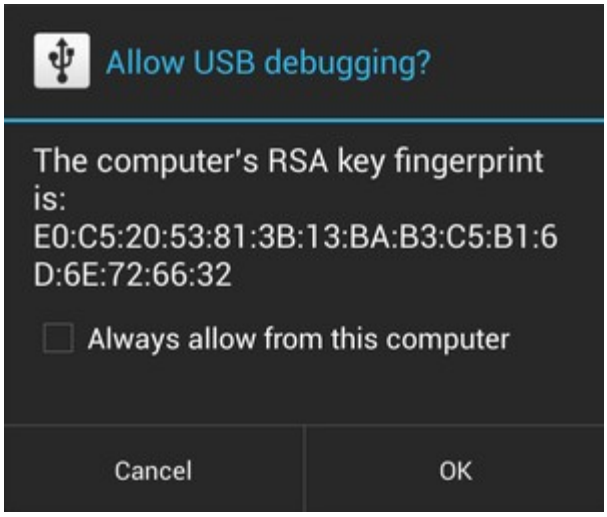
Android SDK

Το Android SDK package περιλαμβάνει τα βασικά εργαλεία μέσω των οποίων μπορεί να εγκαταστήσει και τα υπόλοιπα εργαλεία μέσω διαδικτύου. Μετά την ολοκλήρωση του κατεβάσματος του Android SDK, η αποσυμπίεση του αρχείου πρέπει να γίνει σε κάποια ασφαλή περιοχή του υπολογιστή. Μετά την αποσυμπίεση του αρχείου δημιουργείται ένας

κατάλογος στον υπολογιστή με το όνομα android-sdk-<machine-platform>, ο οποίος περιέχει τα αρχεία του Android SDK. Το μονοπάτι με τα αποσυμπιεσμένα αρχεία χρειάζεται στη συνέχεια για να χρησιμοποιηθεί κάποιο από τα εργαλεία του SDK, ή για την εγκατάσταση του ADT plugin σε περίπτωση που χρησιμοποιείται το Eclipse IDE.

Android Συσκευή ή AVD Manager

Είναι προφανές ότι είναι αναγκαίο για τον προγραμματιστή να βλέπει στην πράξη τη λειτουργικότητα και την συμπεριφορά της εφαρμογής που αναπτύσσει. Αυτό είναι εφικτό ακόμα και εάν δεν υπάρχει διαθέσιμη συσκευή με android λειτουργικό σύστημα, με τη χρήση του AVD Manager. Το τελευταίο παρέχει μία γραφική διεπαφή για το χρήστη, στην οποία υπάρχει η δυνατότητα δημιουργίας και διαχείρισης AVDs, τα αρχικά του οποίου σημαίνουν Android Virtual Device, δηλαδή εικονική συσκευή android. Κατα τη δημιουργία λοιπόν μιας εικονικής συσκευής, ο χρήστης μπορεί να προσαρμόσει τη συσκευή του σύμφωνα με τις ανάγκες του. Μπορεί να διαλέξει για παράδειγμα κάποιο συγκεκριμένο μοντέλο της αγοράς για προσομοίωση, να προσαρμόσει το μέγεθος της οθόνης, την ανάλυση της οθόνης, τη RAM της συσκευής, ή ακόμα και το μέγεθος της εικονικής SD κάρτας. Τα AVDs απαιτούνται από το Android Emulator, ή αλλιώς προσομοιωτή Android, ο οποίος περιλαμβάνεται στο Android-SDK και είναι ουσιαστικά αυτός ο οποίος επιτρέπει τη δοκιμή των εφαρμογών χωρίς την ύπαρξη μιας φυσικής συσκευής android.
Φυσικά, είναι εφικτό για τον προγραμματιστή να ελέγχει τη λειτουργικότητα της εφαρμογής που αναπτύσσει και σε μία φυσική συσκευή με λειτουργικό android. Το μόνο που χρειάζεται να κάνει είναι να επιλέξει το 'Allow USB Debugging', ή στα ελληνικά "Ελεγχος σφαλμάτων USB', και φυσικά να συνδέσει την συσκεύη με καλώδιο USB στον υπολογιστή στον οποίο παράγεται ο κώδικας στο IDE το οποίο χρησιμοποιείται. Η συνηθισμένη διαδρομή που ακολουθείται σε μία android συσκευή για να ενεργοποιηθεί ο έλεγχος σφαλμάτων USB είναι :
Ρυθμίσεις → Εφαρμογές → Επιλογές Προγραμματιστή → Έλεγχος σφαλμάτων USB.
Ενδέχεται η διαδρομή αυτή να διαφέρει απο συσκευή σε συσκευή, ανάλογα κυρίως με την έκδοση android που είναι εγκατεστημένη στη συσκευή. Τέλος, να αναφερθεί ότι όταν συνδέεται μία συσκευή android στον υπολογιστή για έλεγχο σφαλμάτων, ζητείται εκ νέου άδεια για το συγκεκριμένο λόγο.



Εικόνα 5: Εμφανιζόμενο μήνυμα κατά τη σύνδεση android συσκευής με υπολογιστή για έλεγχο σφαλμάτων USB

1.4.2 Manifest File

Ας εμβαθύνουμε σιγά σιγά στον τρόπο με τον οποίο μπορεί κάποιος να προγραμματίσει σε android, όπως επίσης και να δούμε πώς είναι δομημένη μία android εφαρμογή στο σύνολό της.
Αναμφισβήτητα, δε θα μπορούσαμε να ξεκινήσουμε με κάτι διαφορετικό από την περιγραφή του manifest file. Κάθε εφαρμογή πρέπει να περιλαμβάνει ένα τέτοιο αρχείο, με ακριβές όνομα AndroidManifest.xml , στο root κατάλογο της εφαρμογής. Το συγκεκριμένο αρχείο περιλαμβάνει χρήσιμες πληροφορίες σχετικά με την εφαρμογή, καθώς και με τις απαραίτητες προδιαγραφές που πρέπει να έχει ένα σύστημα android προκειμένου να μπορεί να εκτελέσει όλο τον κώδικα της εφαρμογής. Μερικές από τις λειτουργίες του manifest file είναι:

- Ονομάζει το πακέτο java για την εφαρμογή. Το όνομα του πακέτου χρησιμεύει ως μοναδικό χαρακτηριστικό για την εφαρμογή.
- Περιγράφει τα στοιχεία της εφαρμογής, όπως τα activities, broadcast receivers κα, για τα οποία θα γίνει αναφορά στις επόμενες ενότητες.
- Καθορίζει ποιες διαδικασίες θα φιλοξενήσουν τα στοιχεία της εφαρμογής.
- Δηλώνει τα δικαιώματα τα οποία πρέπει να έχει η εφαρμογή προκειμένου να μπορεί να αποκτήσει πρόσβαση σε προστατευόμενα μέρη του API έτσι ώστε να μπορεί να αλληλεπιδρά και με άλλες εφαρμογές.
- Δηλώνει το ελάχιστο επίπεδο του API του Android που απαιτεί η εφαρμογή
- Παραθέτει τις βιβλιοθήκες με τις οποίεσ συνδέεται η εφαρμογή.

Παρακάτω παρατίθεται ένα παράδειγμα ενός AndroidManifest.xml αρχείου. Στη συνέχεια θα αναλύσουμε εκτενέστερα μερικά από τα σημεία του κώδικα.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tapittest_libtest"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="4"
        android:targetSdkVersion="15" />

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

    <!-- Optional permissions to enable ad geotargeting
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
    -->

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.tapit.adview.AdActivity"
            android:configChanges="keyboard|keyboardHidden|orientation" />
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Εικόνα 6: Παράδειγμα ενός AndroidManifest.xml αρχείου

android:minSdkVersion="4"

Σε αυτό το σημείο δηλώνεται ότι το ελάχιστο επίπεδο API που απαιτείται από την εφαρμογή είναι το 4, δηλαδή η έκδοση 1.6.

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

Εδώ δηλώνεται ότι για την ομαλή λειτουργία της εφαρμογής απαιτείται πρόσβαση στο internet. Να σημειωθεί ότι η άδεια χορηγείται από το χρήστη κατά την εγκατάσταση της εφαρμογής, και όχι κατά την εκτέλεσή της.

android:label="@string/app_name"

Σε αυτή τη γραμμή κώδικα, δηλώνεται το όνομα της εφαρμογής. Το "@string" σημαίνει ότι το όνομα της εφαρμογής είναι αποθηκευμένο στο αρχείο strings.xml, το οποίο συνήθως βρίσκεται μέσα στον φάκελο values. Το όνομα της εφαρμογής έχει την ετικέτα "app_name".

<activity android:name = ".MainActivity" . . . . . </activity>

Σε αυτό το σημείο δηλώνεται ότι η εφαρμογή περιλαμβάνει ένα Activity το οποίο ονομάζεται MainActivity.

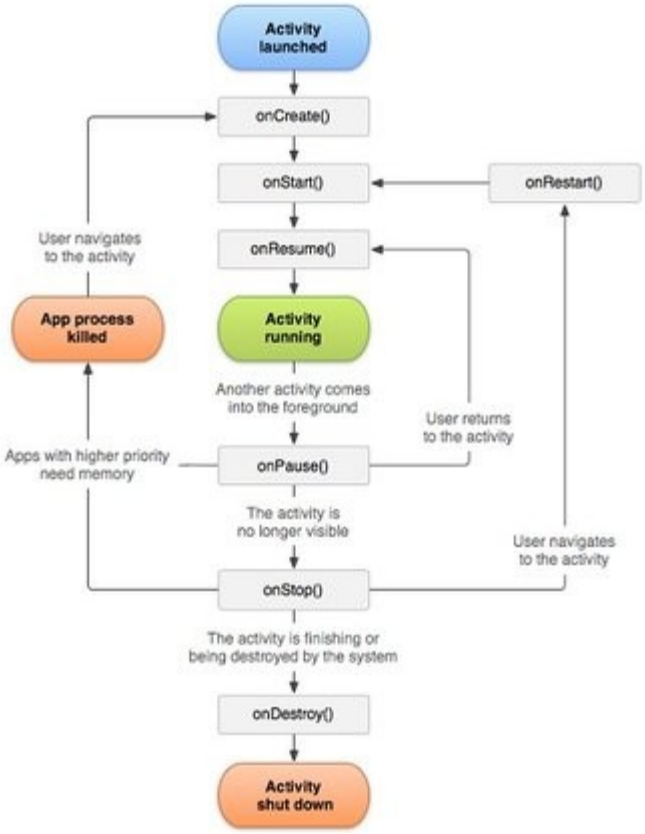
1.4.3 Activities

Θα μελετήσουμε πολλά μέρη μιας android εφαρμογής , γνωστά και ως components (έτσι θα αναφέρονται από εδώ και στο εξής) , ωστόσο το πιο σημαντικό από αυτά είναι το Activity.
Ένα activity (δραστηριότητα) παρέχει μία οθόνη στο χρήστη έτσι ώστε αυτός να είναι σε θέση να αλληλεπιδρά με αυτήν, με σκοπό να κάνει κάποια ενέργεια, όπως να τραβήξει μία φωτογραφία, να στείλει ένα email κλπ. Κάθε activity περιέχει ένα παράθυρο το οποίο περιέχει μία γραφική διεπαφή για το χρήστη και συνήθως 'γεμίζει' όλο το παράθυρο της android συσκευής.
Μία εφαρμογή αποτελείται συνήθως από πολλά activities τα οποία είναι συνδεδεμένα μεταξύ τους, ενώ ένα από αυτά παρουσιάζεται στο χρήστη κατά την εκκίνηση της εφαρμογής. Κάθε φορά που ξεκινάει ένα καινούριο activity, το παλιό σταματάει μεν, αλλά διατηρείται σε μία στοίβα του συστήματος, γνωστή και ως back stack. Αυτή η στοίβα, εξ'ορισμού άλλωστε, υιοθετεί το πρότυπο last in first out. Έτσι , όταν ο χρήστης επιλέξει να πατήσει για παράδειγμα το κουμπί 'back' στη συσκευή του, το activity με το οποίο αλληλεπιδρούσε αφαιρείται από τη στοίβα, καταστρέφεται, και το προηγούμενο activity εμφανίζεται στην οθόνη του.
Για να ξεκινήσει ο προγραμματιστής ένα καινούριο activity, πρέπει να χρησιμοποιήσει τη μέθοδο startActivity() και να της δώσει σαν όρισμα ένα Intent, το οποίο περιγράφει ουσιαστικά το activity προς εκκίνηση. Έτσι προγραμματιστικά έχουμε :

```
Intent intent = new Intent(this,ExampleActivity.class);
startActivity(intent);
```

i. Κύκλος ζωής ενός Activity

Για να μπορέσει κανείς να καταλάβει τη συμπεριφορά και τη λειτουργία ενός Activity, κρίνεται αναγκαία η επεξήγηση και κατανόηση του παρακάτω σχήματος, που μας δείχνει τον κύκλο ζωής ενός Activity.



Εικόνα 7: Κύκλος ζωής ενός Activity

Ας εξηγήσουμε τώρα τις μεθόδους που πρέπει να υλοποιεί ένα Activity:

onCreate()

Η μέθοδος αυτή καλείται όταν το activity δημιουργείται πρώτη φορά. Ενέργειες όπως το στήσιμο του γραφικού και η δημιουργία λιστών πρέπει να γίνονται σε αυτή τη μέθοδο, η οποία πάντα ακολουθείται από τη μέθοδο onStart(). Να αναφερθεί ότι η μέθοδος αυτή παρέχει και ένα αντικείμενο Bundle, το οποίο περιλαμβάνει την προηγούμενη κατάσταση του Activity, εάν υπάρχει.

onRestart()

Αυτή η μέθοδος καλείται αφού το Activity έχει διακοπεί. Και αυτή η μέθοδος ακολουθείται πάντα από την onStart.

onStart()

Καλείται όταν το Activity γίνει ορατό στο χρήστη. Ακολουθείται από την onResume() ή την onStop(), αναλόγως με το αν το Activity κρυφτεί ή όχι.

onResume()

Η συγκεκριμένη μέθοδος καλείται όταν ο χρήστης ξεκινά να αλληλεπιδρά με την οθόνη. Σε αυτό το σημείο το Activity είναι στην κορυφή της στοίβας , ενώ η μέθοδος αυτή ακολουθείται πάντα από την onPause.

onPause()

Εκτελείται όταν το σύστημα πρόκειται να επαναφέρει στο παρασκήνιο ένα προηγούμενο Activity. Η υλοποίηση αυτής της μεθόδου πρέπει να είναι σύντομη, για να μην μπλοκάρεται η επόμενη δραστηριότητα. Η onResume ή η onStop μέθοδος καλείται μετά την onPause.

onStop()

Καλείται όταν το Activity δεν είναι πλέον ορατό στο χρήστη, καθώς κάποιο άλλο έχει πάρει τη θέση του. Ακολουθείται από την onRestart ή την onDestroy, αναλόγως εάν το Activity πρόκειται να καταστραφεί ή όχι.

onDestroy

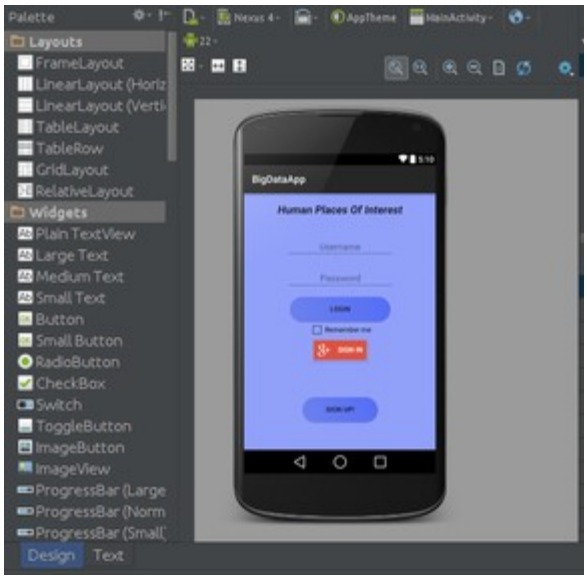
Είναι η τελευταία μέθοδος που καλείται πριν καταστραφεί το Activity.

ii. Activities και XML αρχεία

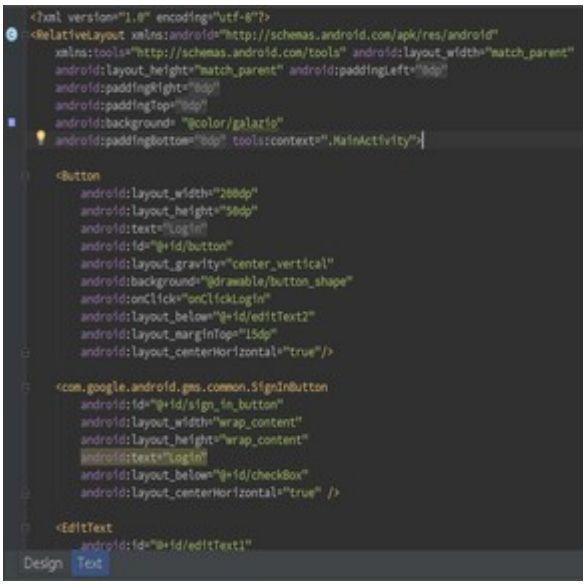
Στην προηγούμενη ενότητα έγινε αναφορά στις μεθόδους που καθορίζουν το πως συμπεριφέρεται ένα Activity. Σε αυτό το σημείο θα δούμε το πώς δημιουργείται μία διεπαφή για το χρήστη, γνωστό και ως User Interface. Αυτό που καθορίζει το γραφικό κομμάτι της εφαρμογής είναι το layout, ή αλλιώς σχέδιο. Το layout μπορεί να διαμορφωθεί είτε με χρήση xml αρχείων, είτε αποκλειστικά προγραμματιστικά. Εδώ θα γίνει σχετική ανάλυση για τον πρώτο τρόπο, καθώς είναι πιο διαδεδομένος αλλά και έχει ορισμένα πλεονεκτήματα έναντι του δεύτερου. Το κυριότερο από αυτά είναι ότι με τη χρήση xml αρχείων είναι ξεκάθαρα για τον προγραμματιστή τα σημεία του κώδικα που αφορούν το γραφικό κομμάτι της εφαρμογής και αυτά που είναι συνδεδεμένα με τη λειτουργικότητά της. Η xml γλώσσα δεν είναι ιδιαίτερα δύσκολη, ενώ εφόσον έχει δημιουργηθεί το layout που μας ενδιαφέρει μπορούμε πάρα πολύ εύκολα να το 'φορτώσουμε' στο Activity χρησιμοποιώντας ουσιαστικά μία γραμμή κώδικα. Όπως αναφέρθηκε στην προηγούμενη ενότητα, η φόρτωση του xml αρχείου πρέπει να γίνει στην onCreate μέθοδο του Activity.
Έστω λοιπόν ότι θέλουμε να φορτώσουμε σε ένα Activity το main_layout.xml αρχείο. Ο κώδικας που θα χρησιμοποιήσουμε είναι:

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);
}
```

Να σημειωθεί ότι τα xml αρχεία που είναι συνδεδεμένα με activities τοποθετούνται συνήθως στον φάκελο res-->layout της εφαρμογής.
Επιπρόσθετα, να προσθέσουμε ότι τα IDEs δίνουν τη δυνατότητα να τροποποιήσουμε ένα xml αρχείο που αφορά ένα Activity, με δύο τρόπους. Είτε με καθαρό κώδικα xml, είτε με μία προσομοίωση μιας android συσκευής.



Εικόνα 8: layout αρχείο σε design mode



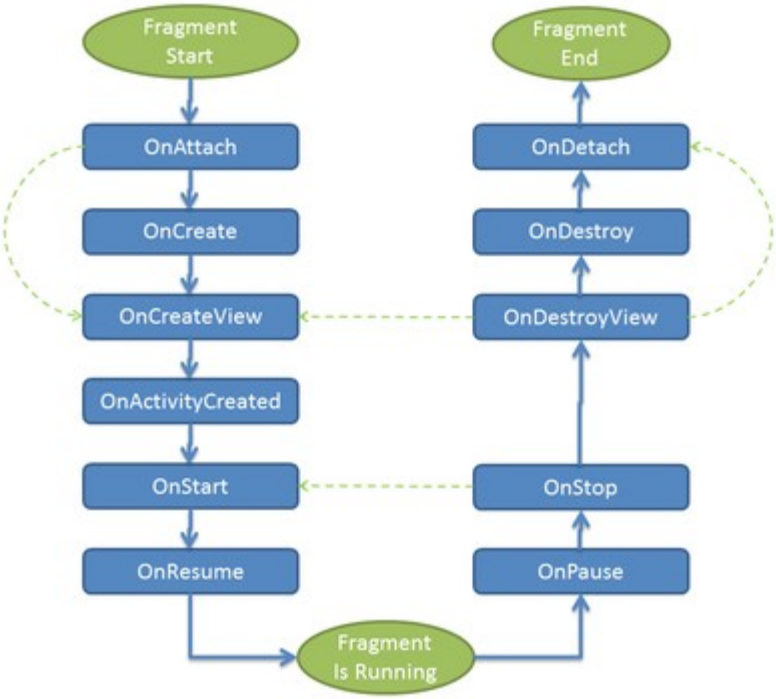
Εικόνα 9: layout αρχείο σε text mode

Στις εικόνες 7 και 8 διακρίνουμε αυτό που αναφέρθηκε παραπάνω. Στο κάτω αριστερά μέρος των εικόνων, διακρίνεται το γεγονός ότι ο προγραμματιστής μπορεί να διαλέξει ανάμεσα σε design mode και text mode. Στο design mode δίνονται πολλά έτοιμα modules στον προγραμματιστή. Με ένα απλό drag' n' drop, σύρσιμο του αντικειμένου δηλαδή πάνω στην οθόνη του προσομοιωτή, modules όπως κουμπιά ή κείμενα είναι έτοιμα προς χρήση. Όταν γίνει κάτι τέτοιο, ο κώδικας xml παράγεται αυτόματα και είναι διαθέσιμος στο text mode.

1.4.4 Fragments

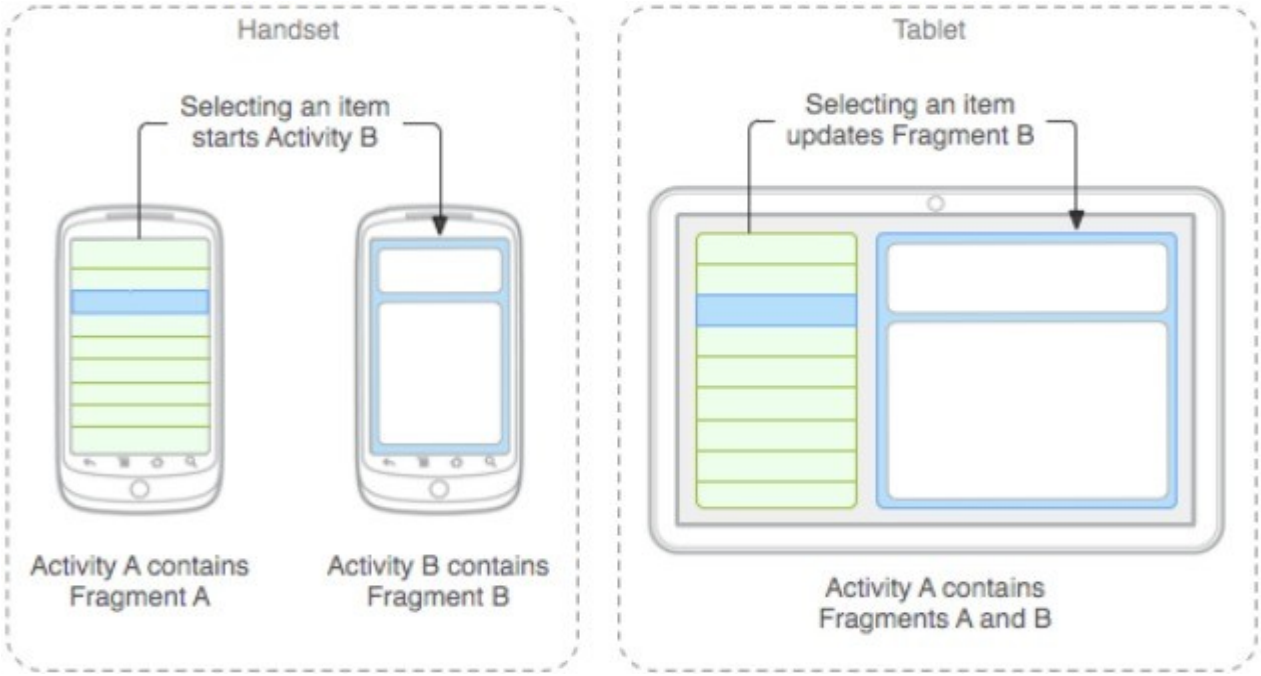
i. Κύκλος ζωής και χρησιμότητα ενός Fragment

Κρίνεται αναγκαίο να κατανοήσουμε τις διαφορές μεταξύ ενός Fragment και ενός Activity.
Θα μπορούσαμε να χαρακτηρίσουμε το Fragment ως ένα sub-Activity, υποδραστηριότητα δηλαδή, η οποία μπορεί να επαναχρησιμοποιηθεί σε πολλά activities. Ουσιαστικά το Fragment αντιπροσωπεύει τη συμπεριφορά ή ένα μέρος της διεπαφής του χρήστη σε ένα Activity. Μπορούν να χρησιμοποιηθούν πολλά Fragments μαζί σε ένα Activity , προκειμένου να δημιουργηθεί μία διεπαφή χρήστη με πολλαπλά παράθυρα. Να σημειωθεί ότι το Fragment έχει το δικό του κύκλο ζωής και είναι πάντα ενσωματωμένο σε ένα Activity, με το οποίο είναι άρρηκτα συνδεδεμένο. Όταν για παράδειγμα ένα Activity σταματήσει, τότε σταματάνε και όλα τα Fragments σε αυτό, ενώ εάν το Activity καταστραφεί, καταστρέφονται και τα Fragments. Παρακάτω παρατίθεται μία εικόνα η οποία μας δείχνει τον κύκλο ζωής ενός Fragment.



Εικόνα 10: Κύκλος ζωής ενός Fragment

Τα Fragments παρέχουν περισσότερο δυναμικές και ευέλικτες διεπαφές χρηστών, ειδικότερα σε μεγάλες οθόνες, όπως έχουν τα tablets. Επειδή τα Fragments έχουν δικό τους σχέδιο, συμπεριφορά και κύκλο ζωής, μπορούν να συμπεριληφθούν σε πολλαπλά Activities, και έτσι δε χρειάζεται να σχεδιάζονται για να διαχειρίζονται για παράδειγμα από άλλα Fragments. Αυτό είναι αρκετά σημαντικό καθώς μπορούν να προσαρμοστούν για διαφορετικά μεγέθη οθονών.



Εικόνα 11: User Interfaces από Fragments σε κινητό και tablet αντίστοιχα

ii. Δημιουργώντας ένα Fragment

Ένα Fragment πρέπει να υλοποιεί τουλάχιστον τις μεθόδους onCreate, onCreateView και onPause, ενώ μπορεί να υλοποιεί και άλλες όπως τις onPause ή onStart. Επίσης να αναφερθεί ότι παρέχονται έτοιμες κλάσεις για διαφορετικούς σκοπούς και χρήση, όπως:

- DialogFragment: Εμφανίζει έναν διάλογο.
- ListFragment: Εμφανίζει μία λίστα από αντικείμενα που διαχειρίζονται από έναν προσρμογέα (adapter).
- PreferenceFragment: Εμφανίζει μία ιεραρχία από Preference αντικείμενα σα μία λίστα.

Μία ενδεικτική κλάση Fragment που φορτώνει το αρχείο example_fragment.xml είναι:

```
public static class ExampleFragment extends Fragment {
        @Override
        public View onCreateView(Layout inflater, ViewGroup container,
                                        Bundle savedInstanceState) {
                return inflater.inflate(R.layout.example_fragment,container,false);
        }
}
```

Ένα Fragment μπορεί να προστεθεί σε ένα Activity με δύο τρόπους:

Στο layout αρχείο του Activity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width=”match_parent”
              android:layout_height=”match_parent”>
              <fragment android:name="com.example.news.ArticleListFragment"
                 android:id="@+id/list"
                 android:layout_weight="1"
                 android:layout_width="0dp"
                 android:layout_height="match_parent" />
</LinearLayout>
```

Προγραμματιστικά

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
// now we add the fragment
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

1.4.5 Άλλα components

Προφανώς υπάρχουνε πάρα πολλά components ('συστατικά' στα ελληνικά) που προσφέρονται για προγραμματισμό σε Android, ωστόσο σε αυτή την ενότητα θα γίνει μία αναφορά στα πιο σημαντικά από αυτά.

Service

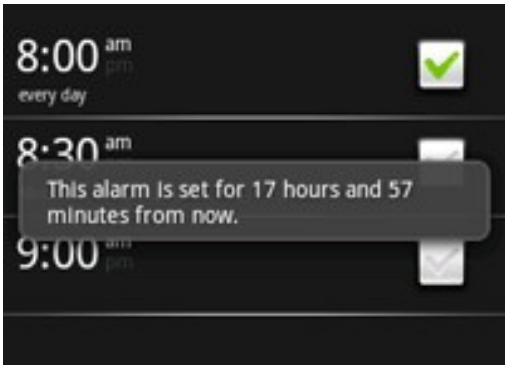
Ένα service είναι ένα στοιχείο της εφαρμογής το οποίο εκτελεί μακροχρόνιες εργασίες στο παρασκήνιο και δεν παρέχει γραφικό περιβάλλον στο χρήστη. Μπορεί να δοθεί εντολή σε ένα service για να ξεκινήσει από ένα άλλο component της εφαρμογής, ενώ θα συνεχίσει να εκτελείται στο παρασκήνιο ακόμα και σε περίπτωση που ο χρήστης μεταβεί σε άλλη εφαρμογή που έχει εγκαταστήσει στη συσκευή του. Να αναφέρουμε ότι είναι δυνατή η επικοινωνία ανάμεσα σε ένα service με ένα άλλο component της εφαρμογής, όπως αόμα και το γεγονός ότι πρέπει να δηλωθεί στο AndroidManifest.xml αρχείο της εφαρμογής.
Ένα ενδεικτικό παράδειγμα είναι ένα service το οποίο ανά τακτά χρονικά διαστήματα βρίσκει τις ακριβείς συντεταγμένες στις οποίες βρίσκεται ο χρήστης.

Broadcast Receiver

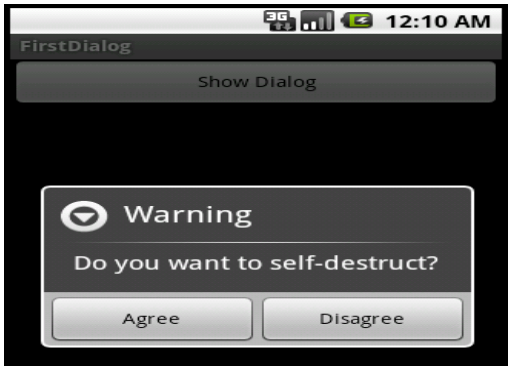
Αυτή η κλάση λαμβάνει ανακοινώσεις broadcast και αντιδρά κατάλληλα. Αυτές οι ανακοινώσεις μπορεί να παράγονται είτε από το λειτουργικό (OS-generated), όπως για παράδειγμα ανακοίνωση για χαμηλή στάθμη μπαταρίας, είτε από το χρήστη (user-generated), όπως για παράδειγμα ανακοίνωση ενεργοποίησης ενός χαρακτηριστικού.
Μόλις ληφθεί κάποια ανακοίνωση, εκτελείται η μέθοδος onReceive. Χαρακτηριστικό παράδειγμα ενός broadcast receiver είναι η ανακοίνωση σύνδεσης σε κάποιο δίκτυο wi-fi.

Notifications

Τα notifications(ειδοποιήσεις) αποσκοπούν στην ενημέρωση του χρήστη για κάποιο συμβάν,κάποια αλλαγή κατάστασης ή κάποια ενέργεια που πρέπει να γίνει. Υπάρχουν 3 είδη ειδοποιήσεων, το Toast, το AlertDialog και το Notification.



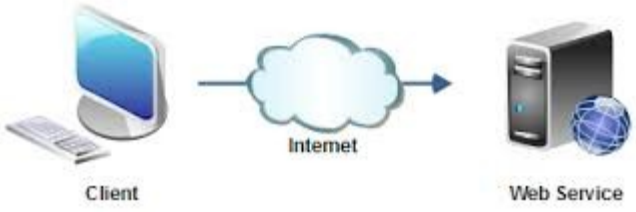
Εικόνα 12: Toast



Εικόνα 13: AlertDialog

2. Web Service

Είναι επιτακτική η ανάγκη να αφιερωθεί ένα τμήμα αυτής της πτυχιακής εργασίας στο τι ακριβώς είναι ένα web service αλλά και στον τρόπο με τον οποίο λειτουργεί, όχι μόνο επειδή χρησιμοποιήθηκε στο προγραμματιστικό μέρος της εργασίας, αλλά επειδή πλέον βρίσκεται στην καθημερινότητα μας. Τα PDAs των εργαζομένων σε εστιατόρια ή καφετέριες, ή ορισμένες πληρωμές μέσω του διαδικτύου αποτελούν κάποια χαρακτηριστικά παραδείγματα. Ηλεκτρονικές διαφημίσεις, ανακάλυψη απάτης, εντοπισμός 'πακέτου' σε μεταφορική εταιρεία, αποτελούν λόγους για μεγάλες και γνωστές εταιρείες να επενδύσουν σε εφαρμογές που χρησιμοποιούν web service.
Ας δούμε όμως τι είναι ένα web service. Πολύ απλά είναι μία μέθοδος, ένας τρόπος επικοινωνίας σε ένα δίκτυο ανάμεσα σε δύο ηλεκτρονικές συσκευές. Το δίκτυο είναι μία συλλογή τερματικών συνδεδεμένων μεταξύ τους με τέτοιο τρόπο ώστε να καθίσταται δυνατή η μεταξύ τους επικοινωνία. Η σύνδεση πραγματοποιείται είτε μέσω καλωδίων είτε ασύρματα, αν και πλέον στην εποχή μας στις περισσότερες των περιπτώσεων γίνεται με τον δεύτερο τρόπο. Το πιο γνωστό δίκτυο είναι το Internet, το οποίο χρησιμοποιεί το πρωτόκολλο http. Συνήθως το service, ή αλλιώς η υπηρεσία που παρέχεται, είναι συνέχεια διαθέσιμο και άμεσα προσπελάσιμο μέσω του πρωτοκόλλου http από μία διεύθυνση διαδικτύου.



Εικόνα 14 : Επικοινωνία τερματικών μέσω Internet

Η παραπάνω εικόνα δείχνει στην απλούστερη μορφή της την επικοινωνία ανάμεσα σε δύο τερματικά μέσω του διαδικτύου. Το μοντέλο που ακολουθείται συνήθως είναι το μοντέλο πελάτη-εξυπηρέτη, γνωστό και ως client-server. Συνήθως οι ενέργειες που γίνονται είναι περισσότερες, ενώ τις πιο πολλές φορές ο server στέλνει πίσω στον πελάτη μία απάντηση.

Εικόνα 15 : Client-Server model

Τα αιτήματα των πελατών, όπως φαίνεται και στην εικόνα, ονομάζονται http requests, ενώ η απάντηση που στέλνει ο server ονομάζεται http response. Εύκολα γίνεται αντιληπτό, ότι ο server πρέπει να είναι σε θέση να εξυπηρετεί ταυτόχρονα παραπάνω από έναν πελάτες, με τρόπο που προγραμματιστικά είναι ασύγχρονος.
Υπάρχουν διάφορες μέθοδοι που χρησιμοποιούνται σε μία επικοινωνία μεταξύ server-client. Οι πιο γνωστές από αυτές είναι οι GET και η POST ενώ υπάρχουν και άλλες όπως η MODIFY και DELETE. Εάν για παράδειγμα ο client θέλει να ανακτήσει δεδομένα από το server τότε στέλνει ένα αίτημα με GET μέθοδο. Πέρα από τη μέθοδο που θα χρησιμοποιήσει ο client υπάρχουν και άλλες παράμετροι που πρέπει να καθοριστούν για τη σωστή επικοινωνία με το server, όπως για παράδειγμα τι τύπου δεδομένα θα παραχθούν (συνήθως είναι xml αρχεία).
Όλα τα παραπάνω πάντως είναι άρρηκτα συνδεδεμένα και με την τεχνολογία που θα χρησιμοποιήσει ο προγραμματιστής για το web service. Παλιότερα η πιο γνωστή ήταν το WSDL, ενώ πλέον ευρέως διαδεδομένο είναι το REST, καθώς είναι αρκετά πιο απλό.
Περισσότερα για το web service, και πιο συγκεκριμένα για το rest, θα δούμε στο 3ο κεφάλαιο, όπου γίνεται εκτενέστερη ανάλυση της εφαρμογής Human Points Of Interest, η οποία έχει βασιστεί σε μοντέλο πελάτη-εξυπηρέτη.

3. Η εφαρμογή Human Points Of Interest

3.1 Εισαγωγή

Η εξέλιξη στον τομέα της τεχνολογίας που αφορά τα κινητά τηλέφωνα είναι κάτι γνωστό σε όλους. Σχεδόν όλοι χρησιμοποιούμε κινητό τηλέφωνο κάθε μέρα, ενώ αυτό πλέον διαθέτει GPS, αισθητήρα wifi και διάφορες άλλες τεχνολογίες, οι οποίες έχουν αποδειχθεί χρήσιμες για τη μελέτη της ανθρώπινης συμπεριφοράς, ειδικά στην εποχή μας όπου η ανάλυση δεδομένων και το λεγόμενο 'Big Data' εξελίσσεται ραγδαία. Πέρα από αυτό, θα χρειαστεί να αναλύσουμε 3 όρους που χρησιμοποιούνται σε αυτήν την πτυχιακή εργασία:

Location Point

Είναι μία μέτρηση η οποία δίνεται από έναν αισθητήρα του κινητού και αφορά την τοποθεσία του χρήστη. Στην εφαρμογή που υλοποιήθηκε δίνεται από το GPS για μεγαλύτερη ακρίβεια στα αποτελέσματα. Ένα Location Point αναπαριστάται από τις συντεταγμένες ενός σημείου καθώς και από την ώρα που βρέθηκε ο χρήστης στο σημείο αυτο. Για παράδειγμα: ([36.5N, 26.5E], [16 : 34 : 57]).

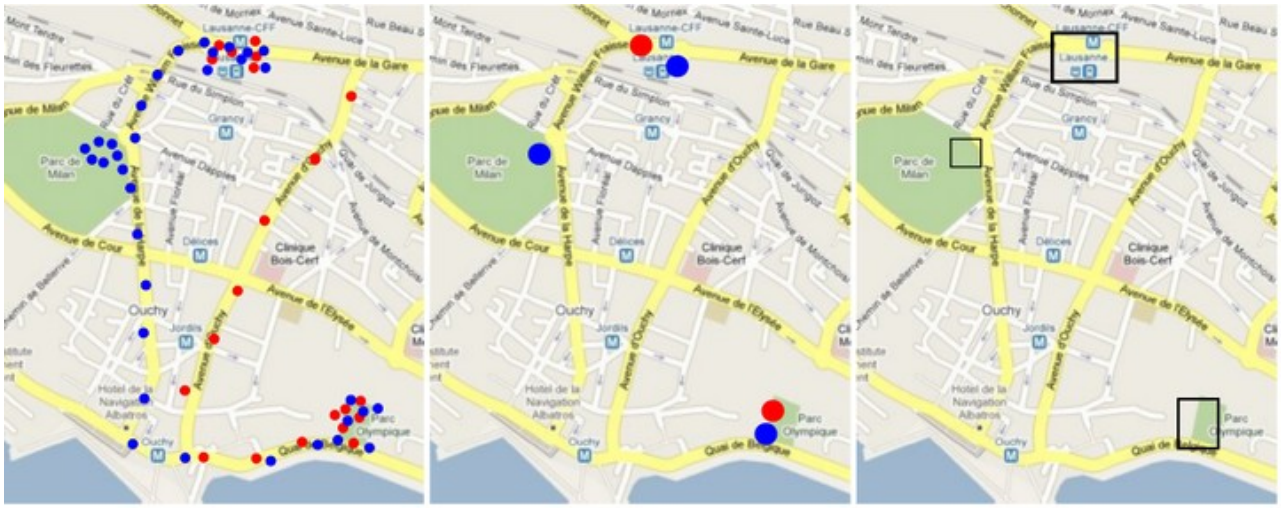
Stay Point

Είναι μία συστάδα από location points, η οποία αναπαριστά μία περιοχή στην οποία ο χρήστης έμεινε για κάποιο χρονικό διάστημα. Αναπαριστάται από το κεντροειδές της συστάδας και από τη χρονική στιγμή που ο χρήστης έφτασε και έφυγε από αυτήν.
Για παράδειγμα: ([46.6N, 6.5E], [16 : 30 : 00], [17 : 54 : 34]).

Stay Region

Είναι μία συστάδα από stay points. Αναπαριστάται από το κεντροειδές της συστάδας και τις μέγιστες και ελάχιστες συντεταγμένες των stay points που ανήκουν σε αυτήν.
Για παράδειγμα: ([46.6N, 6.5E], [46.595N,- 46.599N], [6.498E, 6.502E]).

Να αναφέρουμε ότι σε αυτή την εργασία τα stay regions είναι συνώνυμα με τα Human Points Of Interest, ενώ περισσότερα για τις μεθόδους από τις οποίες προκύπτουν τα stay points και stay regions θα αναφερθούν στο κεφάλαιο 3.3. Κατά την υλοποίηση των αλγορίθμων, λήφθηκαν υπόψη διάφορες παράμετροι, όπως για παράδειγμα το γεγονός ότι η εύρεση δορυφόρου για τη λειτουργια του GPS δεν είναι πάντα εφικτή. Σε 'κλειστά' μέρη η εύρεση δορυφόρου χαρακτηρίζεται από δύσκολη έως ανέφικτη, ενώ προφανώς ο αισθητήρας GPS που διαθέτουν τα κινητά τηλέφωνα δεν έχει την ίδια αποδοτικότητα με έναν επαγγελματικό αισθητήρα GPS.
Επιπρόσθετα, αξίζει να αναφέρουμε ότι για την εξαγωγή των stay regions δε λαμβάνονται υπόψη παράγοντες οι οποίοι είναι άρρηκτα συνδεδεμένοι με το χρήστη, όπως για παράδειγμα η εμπειρία που έχει ο χρήστης στην περιοχή στην οποία κινείται. Είναι προφανές ότι ο χρήστης που είναι κάτοικος Αθήνας, κινείται στο κέντρο της Αθήνας με εντελώς διαφορετικό τρόπο από κάποιον χρήστη που έχει έρθει για τουρισμο.
Παρακάτω παρατίθεται ένα χαρακτηριστικό παράδειγμα εξαγωγής stay points από location points, και stay regions από αυτά τα stay points.



Εικόνα 16 : Εξαγωγή stay points και stay regions(Human Points Of Interest)

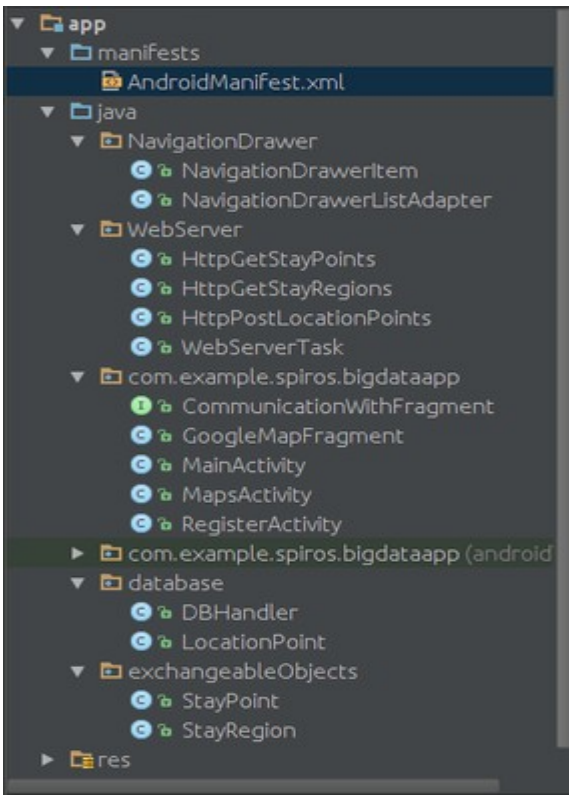
Η παραπάνω εικόνα είναι ένα ενδεικτικό παράδειγμα σχετικά με το πώς προκύπτουν τα stay regions. Αρχικά, στο (a) πλαίσιο, φαίνονται τα location points δύο χρηστών που παράγονται από τη χρήση του GPS. Από αυτά τα location points εξάγονται τα stay points του κάθε χρήστη, ενώ με την ομαδοποίηση των stay points προκύπτουν τα stay regions, τα οποία φορούν όλους τους χρήστες. Στην ενότητα 3.5 παρουσιάζεται μία πραγματική εκτέλεση της εφαρμογής σχετικά με τη διαδικασία που αναφέρθηκε παραπάνω.

3.2 Περιβάλλον υλοποίησης της εφαρμογής

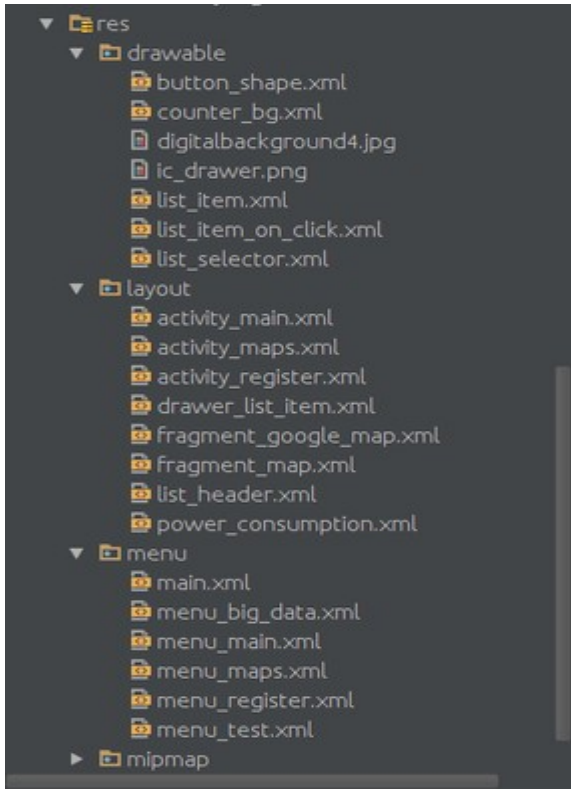
Για την υλοποίηση του πελάτη χρησιμοποιήθηκε το Android Studio 1.35 IDE. Για προγραμματιστικούς λόγους(debugging) και εκτελέσεις της εφαρμογής, χρησιμοποιήθηκε ένα κινητό Samsung Galaxy S3 με λειτουργικό Android 5.0, αλλά και ένα Virtual Device Nexus 5 με λειτουργικό Andoid 5.1. Για την υλοποίηση του server χρησιμοποιήθηκε το Eclipse 4.4 (Luna) IDE και ο Tomcat 7, ενώ για την βάση δεδομένων χρησιμοποιήθηκε η mysql. Το λειτουργικό σύστημα του υπολογιστή στον οποίο αναπτύχθηκε η εφαρμογή είναι Ubuntu 14.04 με εγκατεστημένη την java 7.

3.3 Client Side

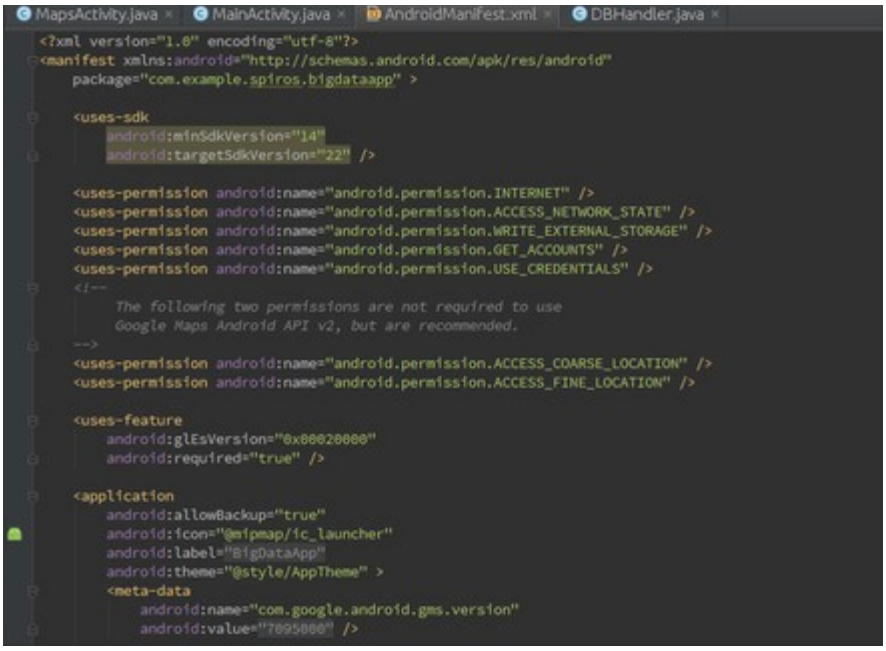
Σε πρώτη φάση παρατίθενται κάποια screenshots, τα οποία δείχνουν τη δομή του project, πώς είναι χωρισμένα τα packages, τις κλάσεις του, τα layouts αλλά και ένα μέρος από το AndroidManifest.xml αρχείο, όπου φαίνονται κυρίως τα permissions που χρειάζεται η εφαρμογή.



Εικόνα 17: Ιεραρχία packages και κλάσεων



Εικόνα 18: Ιεραρχία layout αρχείων



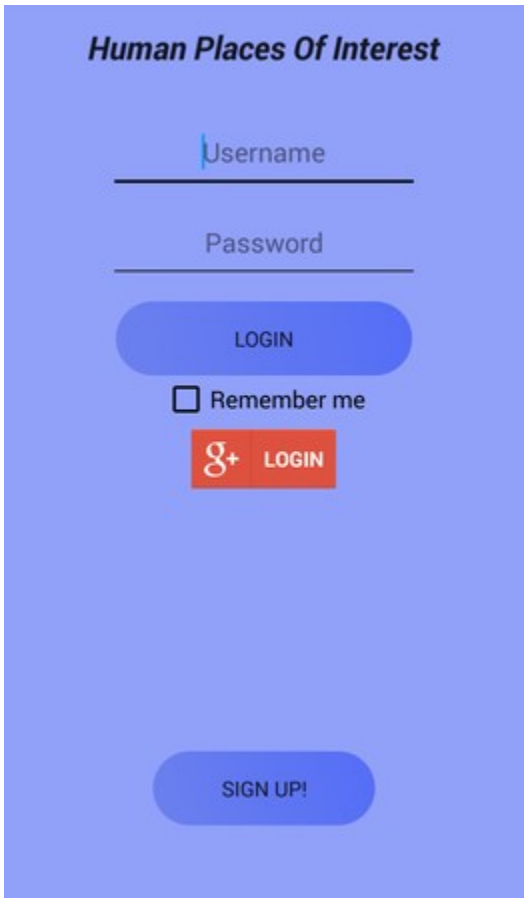
Εικόνα 19: Μέρος του AndroidManifest.xml αρχείου

3.3.1 Γραφικό και λειτουργικότητα εφαρμογής

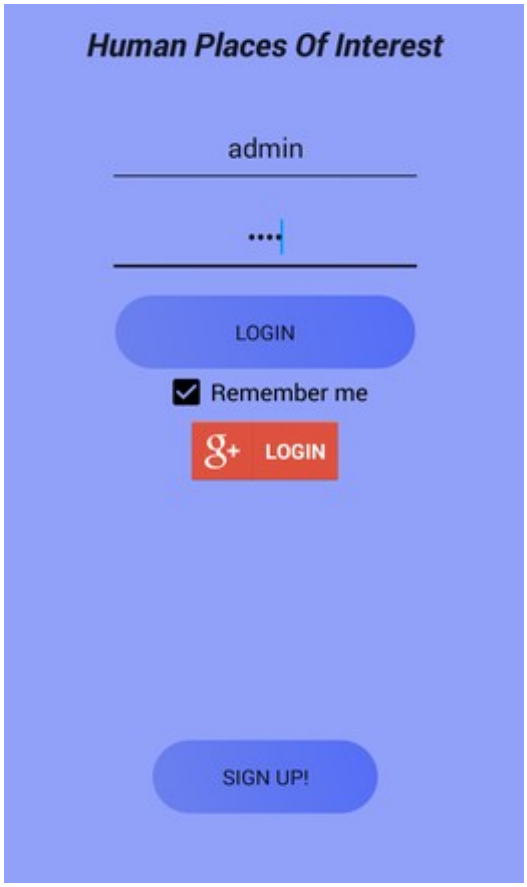
I. Activities

Μόλις ο χρήστης επιλέξει την εκκίνηση της εφαρμογής, στην οθόνη της Android συσκευής του εμφανίζεται το MainActivity. Σε αυτό το σημείο, ο χρήστης μπορεί να επιλέξει να συνδεθεί με την εφαρμογή, χρησιμοποιώντας το όνομα χρήστη και τον κωδικό πρόσβασης που έχει διαλέξει κατά την εγγραφή του, ή να επιλέξει να εγγραφεί εάν δεν το έχει ήδη

κάνει. Ακόμα δίνεται η δυνατότητα στο χρήστη να απομνημονευτούν τα στοιχεία του , μαρκάροντας το πλαίσιο ελέγχου (checkbox) 'Remember me' , έτσι ώστε να μπορεί να αποκτήσει πρόσβαση στην εφαρμογή χωρίς να είναι συνδεδεμένος στο διαδίκτυο.
Σε αυτό το σημείο να αναφερθεί ότι δίνεται η δυνατότητα στο χρήστη να συνδεθεί με την εφαρμογή χρησιμοποιώντας το google λογαριασμό του, περισσότερα όμως για αυτό αναφέρονται στην ενότητα 3.3.4.



Εικόνα 20: Αρχική σελίδα εφαρμογής



Εικόνα 21: Αρχική σελίδα με επιλεγμένη την απομνημόνευση στοιχείων

Εφόσον ο χρήστης επιλέξει την εγγραφή, κάνει κλικ δηλαδή στο κουμπί 'SIGN UP!' , αυτόματα οδηγείται σε ένα καινούριο activity, το RegisterActivity, στο οποίο καλείται να συμπληρώσει τα επιθυμητά username και password. Να αναφερθεί ότι έχουν ακολουθηθεί κανόνες ευχρηστίας , όπως για παράδειγμα υπάρχει σαφής ενημέρωση στο χρήστη εάν επιλέξει όνομα χρήστη το οποίο το έχει ήδη επιλέξει κάποιος άλλος , ή εάν δώσει λανθασμένα στοιχεία πρόσβασης στη διαδικασία σύνδεσης.

Εάν ο χρήστης επιλέξει να συνδεθεί, είτε με όνομα χρήστη και κωδικό πρόσβασης, είτε με το λογαριασμό της google, και συνδεθεί επιτυχώς, τότε ανακατευθύνεται στο MapsActivity, το οποίο ουσιαστικά αποτελεί το κύριο κομμάτι της εφαρμογής.

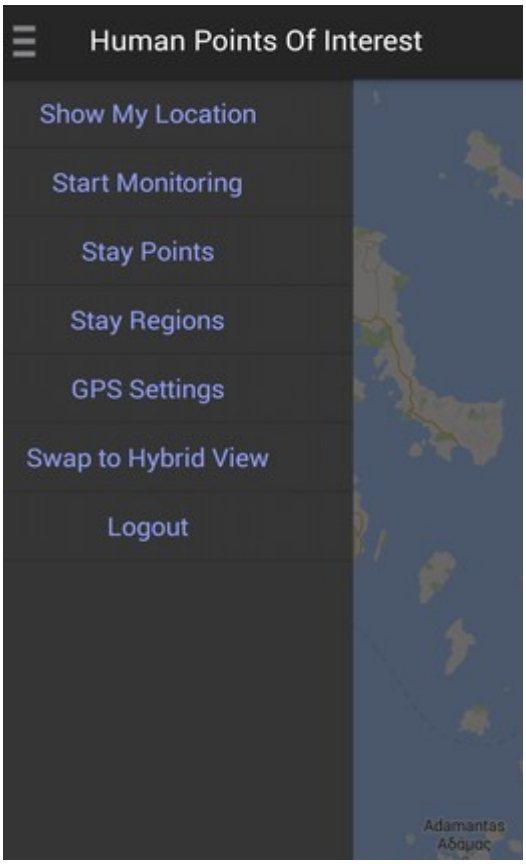
Εικόνα 22: Προτροπή στο χρήστη να επιλέξει άλλο username

Εικόνα 23: MapsActivity μετά τη σύνδεση του χρήστη

II. Κύριο Μενού Εφαρμογής

Για την υλοποίηση του κύριου μενού της εφαρμογής, επιλέχτηκε η σχεδίαση και υλοποίηση ενός Navigation Drawer ή αλλιώς συρταριού πλοήγησης. Το navigation drawer δεν είναι ορατό, αλλά μπορεί να γίνει είτε εάν ο χρήστης σύρει το δαχτυλο του από την αριστερή άκρη της Android συσκευής του προς τη δεξιά, είτε εάν κάνει κλικ στο εικονίδιο στο πάνω αριστερά μέρος της μπάρας της εφαρμογής. Εάν ο χρήστης θέλει να το κάνει πάλι μη ορατό, απλά σέρνει το δάχτυλο του από το δεξί μέρος του Navigation Drawer προς τα αριστερά ή εναλλακτικά κάνει κλικ στο μέρος της οθόνης όπου δεν καταλαμβάνεται από το Navigation Drawer. Να αναφερθεί ότι κάθε φορά που ο χρήστης επιλέξει το επιθυμητό υπομενού, τότε για διευκόλυνση του, το Navigation Drawer αυτόματα 'κρύβεται'.
Το Navigation Drawer μενού της εφαρμογής περιέχει τις εξής επιλογές:

- Show My Location
- Start Monitoring / Stop Monitoring
- Stay Points
- Stay Regions
- GPS Settings
- Swap to Hybrid View / Swap to Normal View
- Logout



Εικόνα 24: Navigation Drawer μενού

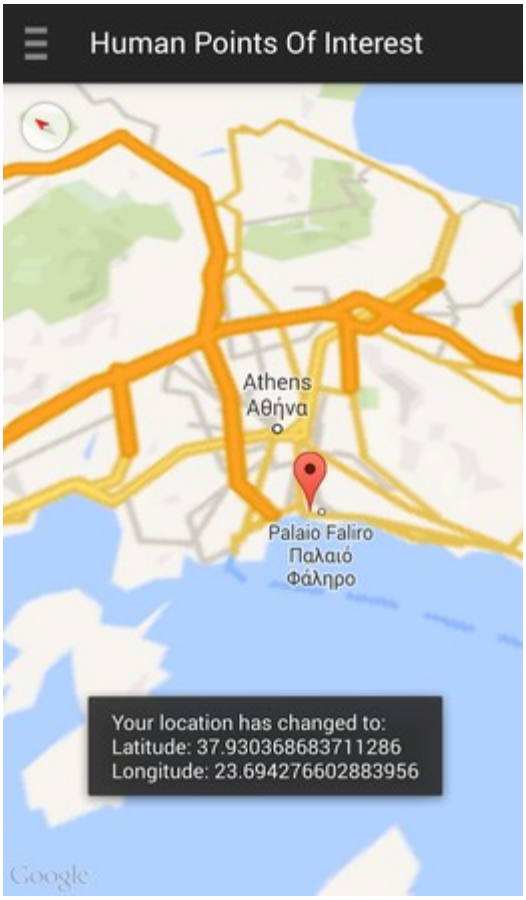
Show My Location

Κάνοντας κλικ σε αυτή την επιλογή, προβάλλεται η τοποθεσία στην οποία βρίσκεται ο χρήστης. Αξίζει να σημειωθεί η εύρεση της τοποθεσίας γίνεται με τη χρήση GPS, εφόσον αυτό είναι ενεργό, για μεγαλύτερη ακρίβεια στο αποτέλεσμα. Εάν το GPS δεν είναι ενεργό τότε η εύρεση της τοποθεσίας γίνεται από το δίκτυο του παρόχου του χρήστη, ενώ σε περίπτωση που κανένα από τα προαναφερθέντα δεν είναι ενεργό, προβάλλεται στο χάρτη η τελευταία 'γνωστή' τοποθεσία του χρήστη(χρήση της μεθόδου getLastKnownLocation).

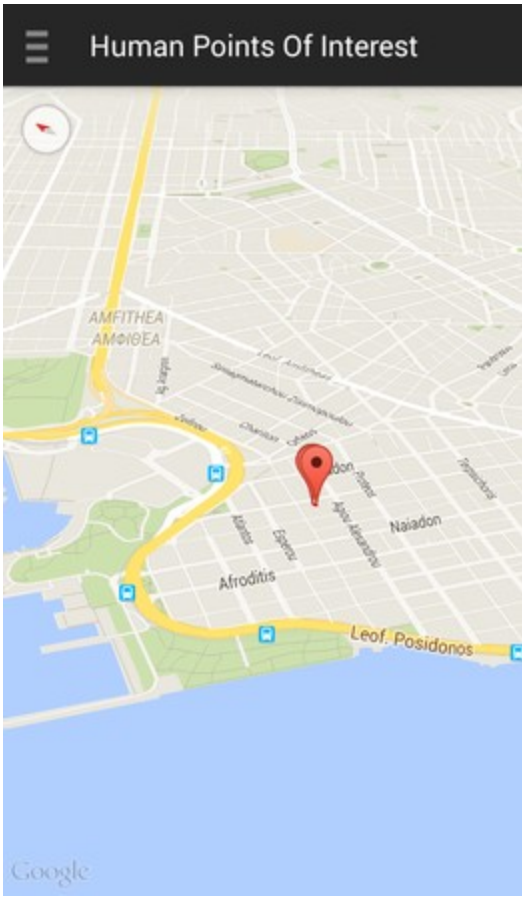
Start Monitoring / Stop Monitoring

Για τη λειτουργία του monitoring, όπου δίνεται ανά τακτά χρονικά και χωρικά διαστήματα η ακριβής τοποθεσία του χρήστη, είναι απαραίτητη η λειτουργία του GPS, για ακριβή αποτελέσματα. Εάν ο χρήστης δεν έχει ενεργοποιημένο το GPS, προτρέπεται από την εφαρμογή να το ενεργοποιήσει. Εάν είναι ήδη ενεργοποιημένο, και προφανώς εφόσον υπάρχει επικοινωνία με δορυφόρο, ξεκινάει το monitoring. Σε αυτή τη διαδικασία, εφόσον ο χρήστης έχει διανύσει τουλάχιστον 5 μέτρα και έχουν περάσει τουλάχιστον 5 δευτερόλεπτα από τη λήψη του τελευταίου σημείου, δίνεται το καινούριο σημείο στο χρήστη η καινούρια δηλαδή τοποθεσία. Οι ακριβείς συντεταγμένες του σημείου εμφανίζονται στο κάτω μέρος της οθόνης προκειμένου να ενημερώνεται ο χρήστης, ενώ ταυτόχρονα δημιουργείται και ένα στίγμα πάνω στο χάρτη της εφαρμογής. Το στίγμα αυτό ενώνεται με τα υπόλοιπα

στίγματα που έχουν δημιουργηθεί από τη λειτουργία του monitoring και έτσι αποτυπώνεται πάνω στο χάρτη ολόκληρη η διαδρομή που διανύει ο χρήστης (trajectory). Μόλις ο χρήστης πατήσει 'start monitoring', και το GPS είναι ενεργοποιημένο, η αντίστοιχη επιλογή στο μενού του Navigation Drawer γίνεται 'stop monitoring'.
Εφόσον η διαδικασία του Monitoring έχει ξεκινήσει, εάν ο χρήστης επιλέξει 'stop monitoring' τότε ο χάρτης 'καθαρίζει', η λειτουργία του monitoring σταματάει, και τα σημεία που δόθηκαν στο χρήστη από αυτή τη λειτουργία αποστέλλονται στο server , με διαδικασία η οποία περιγράφεται στην ενότητα 3.3.2 , ή εάν δεν υπάρχει σύνδεση στο διαδίκτυο αποθηκεύονται στην τοπική βάση της συσκευής.



Εικόνα 25: Εμφάνιση καινούριου σημείου από τη λειτουργία monitoring



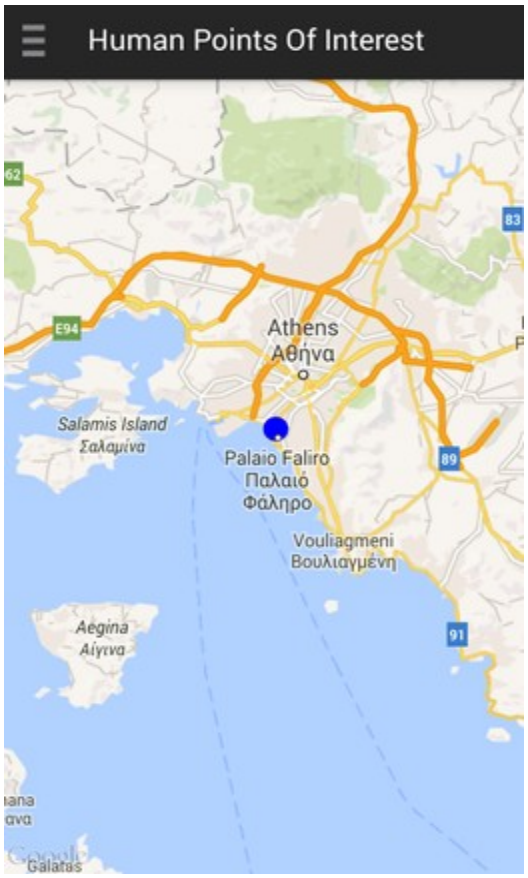
Εικόνα 26: Δημιουργία trajectory στο χάρτη της εφαρμογής

Stay Points

Τα stay points, όπως θα δούμε και στην ανάλυση του server, προκύπτουν από τα σημεία που στέλνει ο χρήστης της εφαρμογής στο server. Κλικάροντας αυτή την επιλογή, γίνεται έλεγχος σχετικά με το εάν η Android συσκευή είναι συνδεδεμένη σε κάποιο δίκτυο. Εάν είναι , τότε 'ζητάει' από το server τα stay points για τα οποία ο χρήστης δεν είναι ενημερωμένος. Εάν δεν είναι συνδεδεμένη, τότε απλά εμφανίζει τα stay points που είναι αποθηκευμένα στην τοπική βάση στο χάρτη. Η ακριβής διαδικασία περιγράφεται στην ενότητα 3.3.3. Τα stay points αναπαρίστανται στο χάρτη με έναν μπλε γεμισμένο κύκλο.

Stay Regions

Επιλέγοντας 'Stay Regions' από το Navigation Drawer, στο χάρτη εμφανίζονται τα stay regions, ή αλλιώς Human Points of Interest. Τα τελευταία εξάγονται βάσει ενός αλγορίθμου ομαδοποίησης (clustering) των stay points , και αφορούν όχι μόνο τον τρέχοντα χρήστη, αλλά όλους τους χρήστες της εφαρμογής.
Και σε αυτή την περίπτωση θα γίνει εκτενέστερη αναφορά του αλγορίθμου στην ανάλυση του server. Ομοίως με τα stay points, εάν δεν υπάρχει σύνδεση με το διαδίκτυο, τότε εμφανίζονται στο χάρτη τα stay regions για τα οποία ο χρήστης είναι ήδη ενημερωμένος και συνεπώς υπάρχουν στην τοπική βάση της συσκευής. Τα stay regions όπως δείχνει και η εικόνα 9, αναπαρίστανται με την ένωση 4 σημείων, σχηματίζοντας ένα ορθογώνιο παραλληλόγραμμο.



Εικόνα 27: Αναπαράσταση stay point

Εικόνα 28: Αναπαράσταση stay region

GPS Settings

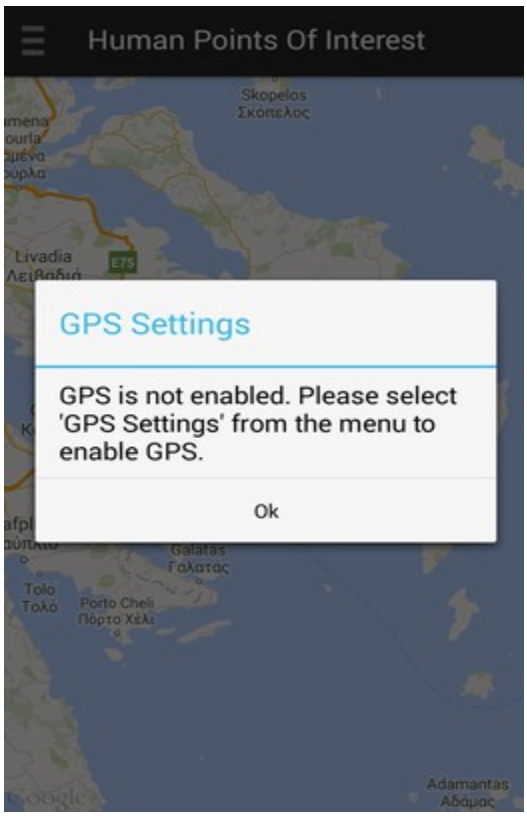
Επειδή, όπως αναφέρθηκε προηγουμένως, για τη λειτουργία του monitoring το GPS είναι απαραίτητο, προς διευκόλυνση του χρήστη προστέθηκε η επιλογή 'GPS Settings' στο μενού της εφαρμογής. Πατώντας αυτή την επιλογή, ο χρήστης ανακατευθύνεται αμέσως στο σημείο του συστήματος της συσκευής Android όπου μπορεί να ενεργοποιήσει τη λειτουργία GPS.

Swap to Hybrid View/Swap to Normal View

Με αυτή την επιλογή, δίνεται η δυνατότητα στο χρήστη να αλλάξει την όψη του χάρτη της εφαρμογής. Ο χάρτης, με την εκκίνηση της εφαρμογής, εμφανίζεται σε κανονική μορφή, ενώ εάν ο χρήστης επιλέξει 'Swap to Hybrid View' τότε η μορφή του χάρτη μετατρέπεται σε υβριδική, ενώ η αντίστοιχη επιλογή στο μενού γίνεται 'Swap to Normal View'. Αντίστοιχα, όποτε θέλει ο χρήστης διαλέγοντας 'Swap to Normal View' μπορεί να επιστρέψει στην κανονική όψη του χάρτη.
Να σημειωθεί ότι όταν γίνεται αλλαγή της όψης του χάρτη, όλες οι πληροφορίες που εμφανίζονται στο χάρτη, όπως πχ στίγματα από Monitoring ή stay points, παραμένουν ως έχουν.

Logout

Επιλέγοντας 'Logout' ο χρήστης αποσυνδέεται από την εφαρμογή, και για να υπάρξει οποιαδήποτε δράση με την εφαρμογή θα πρέπει να ξανασυνδεθεί, είτε με τα στοιχεία πρόσβασης του, είτε με τον google λογαριασμό του. Μόλις γίνει η επιλογή 'Logout' , ο χρήστης ανακατευθύνεται στην αρχική οθόνη της εφαρμογής (MainActivity).



Εικόνα 29: Προτροπή χρήστη να ενεργοποιήσει το GPS για το 'Start Monitoring'

Εικόνα 30: Υβριδική όψη χάρτη

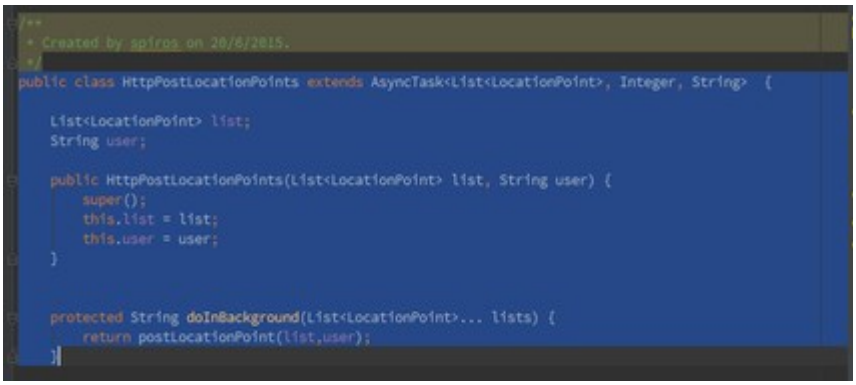
3.3.2 Web service

Στην ενότητα 2 μιλήσαμε γενικά για το τί είναι το web service, αλλά και πού μας χρησιμεύει. Σε αυτό το κεφάλαιο θα δούμε πιο συγκεκριμένα πώς λειτουργεί το web service από την πλευρά του client. Θα δούμε δηλαδή πώς ο client συνδέεται με το server προκειμένου να

στείλει ένα http request, αλλά και πώς στέλνει ή ζητάει δεδομένα από αυτόν ανάλογα με τη μέθοδο που χρησιμοποιεί, get ή post.
Αρχικά, να αναφέρουμε ότι η σύνδεση με το server σε android εφαρμογή δεν μπορεί σε καμία περίπτωση να γίνει στο κύριο thread της εφαρμογής, αλλά πρέπει να γίνεται σε ξεχωριστή κλάση.
Έτσι, όταν ο χρήστης για παράδειγμα επιλέξει από το μενού 'Stop Monitoring', και υπάρχει σύνδεση με το διαδίκτυο, εκτελείται (μεταξύ άλλων) η ακόλουθη γραμμή κώδικα:

```
try {
    String result = new HttpPostLocationPoints(locationPoints, user).execute().get();
```

Τα ορίσματα που παίρνει η παραπάνω μέθοδος είναι το σύνολο των location points που προέκυψαν από τη λειτουργία του Monitoring, καθώς και το username του χρήστη που είναι συνδεδεμένος.
Ας δούμε όμως τη μορφή της κλάσης HttpPostLocationPoints:



Βλέπουμε ότι η κλάση HttpPostLocationPoints κάνει extend την κλάση Asynctask. Να σημειωθεί ακόμα ότι έχει δημιουργηθεί και ένας constructor αυτής της κλάσης, καθώς τα ορίσματα που περάσαμε στην HttpLocationPoints είναι παραπάνω από ένα και διαφορετικού τύπου. Το πρώτο όρισμα είναι μία λίστα από Location Points και το δεύτερο είναι ένα String.
Μόλις δημιουργήσουμε αυτή την κλάση η μέθοδος που εκτελείται είναι η doInBackground, στην οποία όπως διακρίνουμε από την παραπάνω εικόνα, καλείται η μέθοδος postLocationPoint.
Σε αυτή τη μέθοδο δημιουργούμε έναν HttpClient και με μέθοδο HttpPost συνδεόμαστε στην διεύθυνση του server. Στο server πέρα από το username του χρήστη, το οποίο δίνεται στο μονοπάτι στο οποίο γίνεται η σύνδεση, στέλνουμε ένα πίνακα JSON (JSONArray) ο οποίος αποτελείται από JSONObjects, που το καθένα από αυτά αντιπροσωπεύει ένα Location Point. Το JSON είναι μία μορφή η οποία χρησιμοποιείται για να μεταδώσει αντικείμενα δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικό-τιμής. Θα γίνει εκτενέστερη αναφορά για το JSON, όπως και για την διεύθυνση στην οποία εξυπηρετεί ο server στο κεφάλαιο 3.4 .
Στην συνεχεια παρατίθεται ένα screenshot το οποίο δείχνει πώς δημιουργείται το JSONArray, που στέλνει ο πελάτης στον εξυπηρέτη.

```
        JSONArray array = new JSONArray();

        for (int j = 0; j < locationPointList.size(); j++) {

            jsonObject = new JSONObject();
            jsonObject.put("lat", locationPointList.get(j).getLatitude());
            jsonObject.put("long", locationPointList.get(j).getLongitude());
            jsonObject.put("time", locationPointList.get(j).getTime());

            array.put(jsonObject);

        }

        Log.d("JSON: ", array.toString());
```

Εκτός από τις κλασσικές μεθόδους που παρει το API του android, χρησιμοποιήθηκε και μία εξωτερική βιβλιοθήκη για τη σύνδεση με το server, η com.loopj. Η βιβλιοθήκη αυτή είναι ευρέως διαδεδομένη ενώ αξίζει να αναφερθεί ότι χρησιμοποιείται από πολλές γνωστές εφαρμογές, όπως για παράδειγμα το Instagram. Η σύνδεση με το server γίνεται ασύγχρονα, κάτι ποθ καθιστά πιο ευέλικτη την εφαρμογή.
Με αυτή τη βιβλιοθήκη δημιουργείται ένα αντικείμενο πελάτη (AsyncHttpClient), ενώ υπάρχει και ένας handler ο οποίος διαχειρίζεται την απάντηση που στέλνει πίσω ο server στον πελάτη. Σε περίπτωση που δεν υπάρχει πρόβλημα στην επικοινωνία μεταξύ server και πελάτη, ή αλλιώς η απάντηση στο αίτημα του πελάτη είναι 200, εκτελείται η μέθοδος OnSuccess, ενώ εάν υπάρξει κάποιο πρόβλημα, δηλαδή η απάντηση είναι για παράδειγμα 404 εκτελείται η μέθοδος onFailure.

```
httpClient.get("http://192.168.2.4:8080/RestWebService/rest/stayPoints/" + user,
        new AsyncHttpResponseHandler() {

    @Override
    public void onStart() {
        // called before request is started
    }

    public void onSuccess(String response) {
        Log.d("The response is", response);
        stayPoints = response;
        Log.d("Stay points are ", stayPoints);
    }


    @Override
    public void onFailure(int statusCode, Header[] headers, byte[] responseBody, Throwable erro
        Log.d("Connection status: ", "Failed to connect!");
        // mssg to check internet connection...?
    }

});
```

3.3.3 Βάση Δεδομένων

Για την ανάπτυξη της εφαρμογής κρίθηκε αναγκαία η ύπαρξη μίας βάσης δεδομένων στη μεριά του πελάτη. Σε περίπτωση που η σύνδεση με το server δεν είναι δυνατή, είτε λόγω κάποιου προβλήματος στο server είτε επειδή ο πελάτης δεν είναι συνδεδεμένος στο διαδίκτυο, πρέπει ο πελάτης να έχει τη δυνατότητα να επεξεργάζεται τα δεδομένα για τα οποία είναι ήδη ενημερωμένος. Έτσι, κάθε φορά που ο server στέλνει δεδομένα στο πελάτη, είτε stay points είτε stay regions, αυτά αποθηκεύονται στην τοπική βάση του πελάτη. Με αυτό τον τρόπο, την επόμενη φορά που ο πελάτης δε θα μπορεί να ζητήσει δεδομένα από το server, θα έχει τη δυνατότητα να 'τραβήξει' τα δεδομένα τα οποία έχουν ήδη σταλεί από τον server από την τοπική βάση.
Η βάση που έχει δημιουργηθεί αποτελείται από 3 tables, ένα για τα location points , ένα για τα stay points και ένα για τα stay regions. Το table των location points είναι απαραίτητο σε περίπτωση που ο client επιλέξει 'stop monitoring' και δεν είναι συνδεδεμένος στο διαδίκτυο, τα location points που έχουν προκύψει από τη διαδικασία του monitoring πρέπει να αποθηκευτούνε κάπου αφού δε θα σταλούν εκείνη τη στιγμή στο server. Έτσι,

αποθηκεύονται στην τοπική βάση, έτσι ώστε μόλις ο πελάτης αποκτήσει πρόσβαση στο διαδίκτυο, να σταλούν τα location points της βάσης στο server.
Αυτή η ενέργεια πραγματοποιείται με τη χρήση ενός Broadcast Receiver, έτσι όταν ο πελάτης αποκτήσει πρόσβαση στο διαδίκτυο αποστέλλονται στο server όλα τα location points που υπάρχουν στη βάση και στη συνέχεια διαγράφονται από αυτήν.

Παραπάνω φαίνεται ο τρόπος με τον οποίο δημιουργούνται προγραμματιστικά τα tables της βάσης. Τα LPTABLE, SPTABLE, SRTABLE αφορούν τα location points, stay points και stay regions αντίστοιχα.Στη συνέχεια αναφέρονται επιγραμματικά οι μέθοδοι που έχουν υλοποιηθεί και αφορούν τη βάση δεδομένων:

- public void addLocationPoint (LocationPoint locationPoint)
- public List<LocationPoint> getLocationPoints()
- public void deleteAllLocationPoints()
- public void addStayPoint (StayPoint stayPoint)
- public void addStayRegion (StayRegion stayRegion)
- public List<StayPoint> getStayPoints()
- public List<StayRegion> getStayRegions()

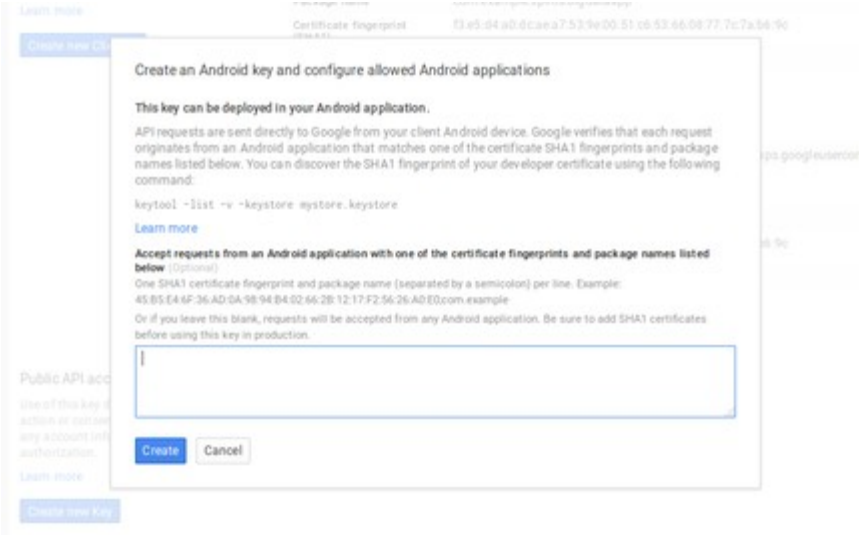
3.3.4. Θέματα Προγραμματισμού

i.Google Maps

Για την εμφάνιση του χάρτη της google στην εφαρμογή, έπρεπε να ακολουθηθεί μία συγκεκριμένη διαδικασία. Αρχικά, να αναφέρουμε ότι για να εμφανίζεται ο χάρτης πρέπει στη συσκευή android να είναι εγκατεστημένο το Google Play Services. Χωρίς αυτό η εμφάνιση του χάρτη δεν είναι εφικτή.
Ο προγραμματιστής που θέλει να κάνει χρήση του google maps πρέπει να προμηθευτεί ένα κλειδί, το Google Maps API key. Αυτό είναι δωρεάν και μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές, ενώ διατίθεται στο Google APIs Console. Για να το προμηθευτεί κάποιος χρειάζεται να συμπληρώσει δύο ζητούμενα,το SHA-1 και το package της εφαρμογής που περιέχει το χάρτη.
Το SHA-1 είναι ένα 'certificate fingerprint' (στα ελληνικά μεταφράζεται πιστοποιητικό δακτυλικών αποτυπωμάτων).Σε Linux λειτουργικό σύστημα το SHA-1 προκύπτει με την ακόλουθη εντολή:

keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android

Στην παρακάτω εικόνα φαίνεται το σημείο κατά το οποίο γίνεται η εισαγωγή του SHA-1 και του ονόματος του πακέτου στη κονσόλα της google.



Εικόνα 31: Εισαγωγή SHA-1 και package

Τέλος, τα ακόλουθα permissions όπως και το κλειδί πρέπει να προστεθούν στο AndroidManifest.xml αρχείο με τον ακόλουθο τρόπο:

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

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

ii.OAuth 2.0

Όπως αναφέρθηκε στην ενότητα 3.3.1 ο χρήστης μπορεί να συνδεθεί με την εφαρμογή χωρίς να εγγραφεί πρώτα, αλλά με το google λογαριασμό του. Για να γίνει αυτό, αρχικά χρειάζεται να αρχικοποιήσουμε ένα αντικείμενο GoogleApiClient στην μέθοδο onCreate του Activity μας. Επίσης πρέπει να γίνουν override οι μέθοδοι onStart και onStop, και σε αυτές να συνδέσουμε και να αποσυνδέσουμε αντίστοιχα το αντικείμενο GoogleApiClient με τις μεθόδους connect και disconnect.
Αφού αρχικοποιήσουμε λοιπόν το αντικείμενο GoogleApiClient,όπως φαίνεται στο παρακάτω screenshot, στη συνέχεια μπορούμε να εισάγουμε στην εφαρμογή μας το κουμπί για σύνδεση με google λογαριασμό (google sign-in button).

```
    mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(Plus.API)
            .addScope(new Scope("profile"))
            .build();
}
```

Για να εμφανίζεται το google sign-in button στο γραφικό της εφαρμογής μας, αρκεί να προστεθεί το ακόλουθο κομμάτι κώδικα στο layout του activity:

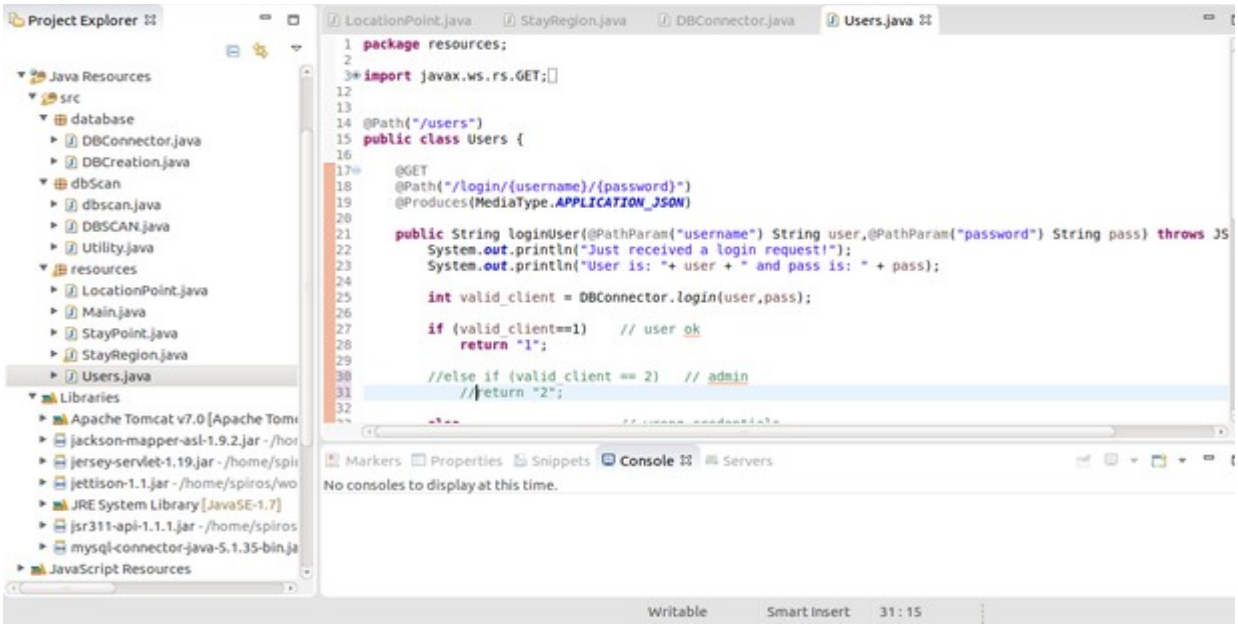
```
<com.google.android.gms.common.SignInButton
    android:id="@+id/sign_in_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Στη συνέχεια, εάν ο χρήστης συνδεθεί επιτυχώς, εκτελείται η μέθοδος onConnected ενώ σε διαφορετική περίπτωση η μέθοδος onConnectionFailed. Περισσότερες πληροφορίες για την εισαγωγή του google sign-in button στον παρακάτω σύνδεσμο: https://developers.google.com/identity/sign-in/android/sign-in

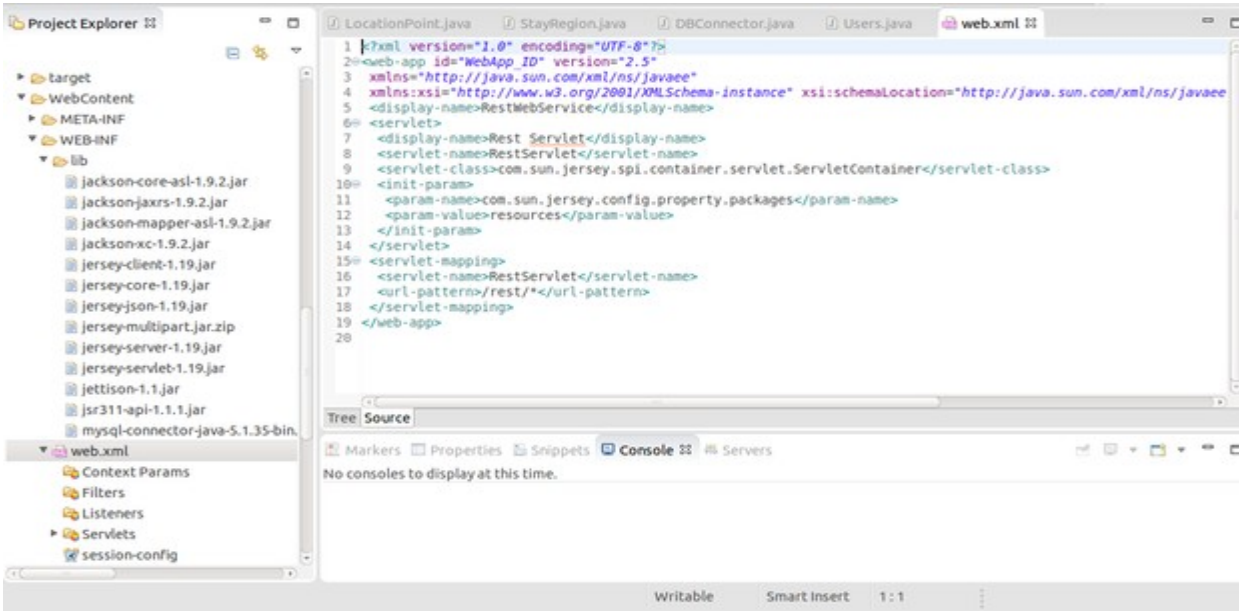
3.4 Server Side

3.4.1 Web Service

Αρχικά παρατίθενται κάποια screenshots, τα οποία δείχνουν τη δομή του project, πώς είναι χωρισμένα τα packages, τις κλάσεις του, καθώς και μερικές από τις βιβλιοθήκες που έχουν χρησιμοποιηθεί για την υλοποίηση του εξυπηρέτη. Ο server που χρησιμοποιήθηκε, όπως αναφέρεται και στην ενότητα 3.2 είναι ο Tomcat 7.0.



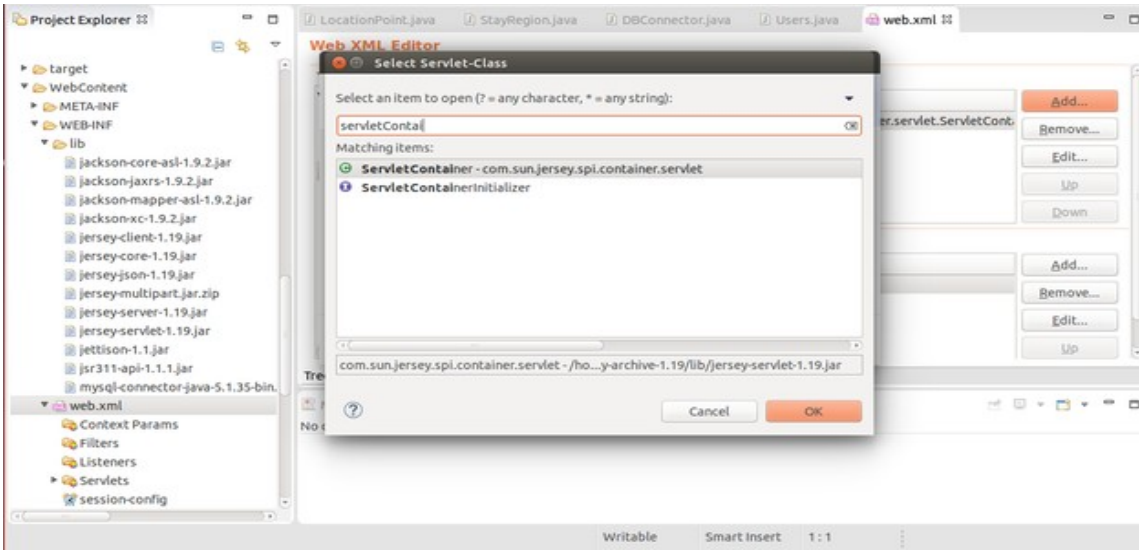
Εικόνα 32: Δομή packages και κλάσεων του server



Εικόνα 33: Libraries που χρησιμοποιήθηκαν και web.xml αρχείο

i.Jersey Framework

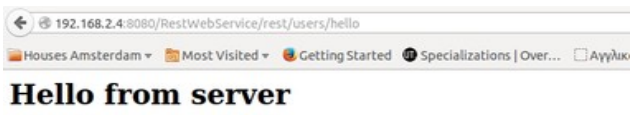
Όπως διακρίνουμε και από τα περιεχόμενα του lib φακέλου στο δεύτερο screenshot, για την υλοποίηση του Restful Web Service έχει γίνει χρήση της βιβλιοθήκης jersey. Το jersey είναι ένα open source framework που χρησιμοποιείται για τη μεταφορά δεδομένων σε μοντέλο πελάτη-εξυπηρέτη και παρέχει ένα JAX-RS API. Περισσότερα για το framework jersey αλλά και για το JAX-RS API αντίστοιχα μπορεί να βρει κάποιος εδώ: https://jersey.java.net/ και https://jax-rs-spec.java.net/ .



Εικόνα 34: Δημιουργία ενός Servlet Container που παρέχεται από το framework Jersey

Αρχικά, στο Web Dynamic Project της εφαρμογής,πρέπει να δημιουργηθεί ενα ServletContainer, όπως φαίνεται στην εικόνα 34, και στη συνέχεια πρέπει να προσδιοριστεί το package στο οποίο θα περιλαμβάνονται οι πόροι του server, τους οποίους θα μοιράζεται με τους πελάτες. Μετά ορίζουμε ένα mapping για το servlet που δημιουργήσαμε, έτσι ώστε να καθοριστεί το μονοπάτι (path) στο οποίο θα εξυπηρετεί ο server. Ολόκληρο το path του server ορίζεται από την ip του ακολουθούμενη από ένα port number, το όνομα του project που δημιουργήσαμε,το url που προσδιορίσαμε στο servlet mapping και ότι άλλο path προσδιορίζουμε μέσα στις κλάσεις.

Εικόνα 35 : Ενδεικτική κλάση server



Εικόνα 36: Αίτημα σύνδεσης με server

Στις εικόνες 35 και 36 φαίνεται ένα παράδειγμα σύνδεσης πελάτη στον εξυπηρέτη. Αρχικά να διευκρινήσουμε ότι για τις ανάγκες της εφαρμογής η σύνδεση του πελάτη με τον εξυπηρέτη γίνονται τοπικά. Αυτό σημαίνει ότι θα πρέπει να είναι συνδεδεμένοι στο ίδιο δίκτυο για να είναι εφικτή η επικοινωνία.Στο server έχει δοθεί χειροκίνητα μία συγκεκριμένη ip, η 192.168.2.4, έτσι ώστε αυτή να είναι σταθερή. Το όνομα του project του server είναι RestWebService, ενώ το path που προσδιορίστηκε στο servlet mapping είναι /rest/*.
Τέλος, το path που χρησιμοποιείται για την κλάση Users είναι '/users', ενώ η μέθοδος sayHello της οποίας η απάντηση είναι σε html κώδικα, και αυτό προγραμματιστικά γίνεται με το annotation @Produces, εκτελείται στο μονοπάτι '/hello'.
Έτσι, όπως φαίνεται από την εικόνα 36, όταν ένα μηχάνημα που είναι συνδεδεμένο στο ίδιο δικτυο με τον server, δώσει σε έναν browser το ακόλουθο url:
'http://192.168.2.4:8080/RestWebService/rest/users/hello' , τότε θα λάβει ως απάντηση τον html κώδικα που παράγει η μέθοδος sayHello. Να αναφέρουμε ότι πρέπει να προσδιορίζεται για κάθε μέδοδο ο τύπος της αίτησης, post ή get συνήθως, καθώς και το τί παράγει, html ή json για παράδειγμα.

ii.JSON

Στο project του server, όλα τα responses που στέλνονται στον πελάτη, είναι μορφής JSON.Όπως αναφέραμε και στην ενότητα 3.3.2, το JSON είναι μία μορφή η οποία χρησιμοποιείται για να μεταδώσει αντικείμενα δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικό-τιμής.

Εικόνα 37: Μέθοδος που διαβάζει το JSON από τον client

Η παραπάνω μέθοδος εκτελείται όταν στείλει ο πελάτης τα location points από τη λειτουργία monitoring στο server. Η μέθοδος readJson δέχεται σαν όρισμα ένα String το οποίο εστάλη από τον πελάτη και είναι όλα τα location points σε JSON format. Στη συνέχεια ο server αποκωδικοποιεί τα δεδομένα παίρνοντας ένα ένα τα JSONObjects από το JSONArray. Κάθε JSONObject αντιπροσωπεύει και ένα location point. Έτσι, ο server διαβάζει για κάθε location point τις συντεταγμένες του (lat και long αντίστοιχα) καθώς και την ώρα που βρέθηκε το location point(time).
Με τον ίδιο τρόπο στέλνει ο server στον πελάτη JSON Objects, όταν ο πελάτης ζητήσει stay points ή stay regions.

Εικόνα 38: Μέθοδος που δημιουργεί το JSONArray των stay points
που θα σταλούν στον πελάτη

Εικόνα 39: Μέθοδος που δημιουργεί το JSONArray των stay regions
που θα σταλούν στον πελάτη

Όπως διακρίνουμε από τις παραπάνω εικόνες, αφού ο server κάνει το κατάλληλο query στη βάση δεδομένων, στέλνει στον πελάτη για κάθε stay point τις συντεταγμένες του, οι οποίες όπως έχουμε αναφέρει αποτελούν το κεντροειδές των location points, ενώ για κάθε stay region στέλνει τις μέγιστες και ελάχιστες συντεταγμένες του (minLat,maxLat, minLong, maxLong) προκειμένου να έχει τη δυνατότητα ο client να το αναπαραστήσει στο χάρτη της εφαρμογής.

3.4.2 Λειτουργικότητα Server

i.Login/Register user

Πριν εμφανιστεί το MapsActivity στη μεριά του client, αυτός υπάρχει περίπτωση να στείλει αίτημα στο server σε 2 περιπτώσεις. Είτε για να συνδεθεί στην εφαρμογή με username και password και δεν έχει επιλέξει απομνημόνευση των στοιχείων του, είτε για να εγγραφεί στην εφαρμογή. Η κλάση η οποία εξυπηρετεί αυτά τα αιτήματα είναι η κλάση Users, η οποία φαίνεται στην εικόνα 35.
Στο μονοπάτι στο οποίο στέλνει αίτημα ο χρήστης προστίθεται το '/Users' και μετά από

αυτό το '/login' ή '/register' ανάλογα με το αν ο χρήστης θέλει να συνδεθεί να εγγραφεί στην εφαρμογή. Στο τέλος του μονοπατιού προστίθενται τα username και password που έχει διαλέξει ο χρήστης και αυτά περνιούνται και σαν ορίσματα στην αντίστοιχη μέθοδο.

Εικόνα 40: Μέθοδος που εκτελείται μετά από αίτημα του χρήστη για εγγραφή

Διακρίνουμε παραπάνω ότι μόλις ο server λάβει ένα αίτημα για εγγραφή στην εφαρμογή, τότε εκτελείται η μέθοδος registerUser. Σε αυτήν ο server συνδέεται με τη βάση, και απαντάει με '1' ή '0' στον πελάτη, ανάλογα με το εάν η εγγραφή πραγματοποιήθηκε επιτυχώς ή όχι.

ii.Location Points

Μόλις ο client επιλέξει από το μενού 'Stop Monitoring' τότε όλα τα location points που έχουν προκύψει αποστέλλονται στο server. Σε περίπτωση που δεν είναι συνδεδεμένος στο διαδίκτυο, μόλις συνδεθεί στέλνει όλα τα location points που είναι αποθηκευμένα στην τοπική βάση στο server, όπως είδαμε σε προηγούμενη ενότητα.
Ο server από τη μεριά του, δέχεται αιτήματα για τα location points στο μονοπάτι 'http://192.168.2.4:8080/RestWebService/rest/locationPoints ακολουθούμενο από το username του χρήστη που έστειλε το αίτημα. Στη συνέχεια, αφού αποκωδικοποιήσει το JSON που έλαβε, με τη χρήση ενός αλγορίθμου που θα αναλυθεί σε επόμενη ενότητα, ελέγχει εάν προκύπτουν stay points από τα Location points που του έστειλε ο πελάτης.

Εικόνα 41: Μέθοδος που εκτελείται μόλις ο server
λάβει από έναν χρήστη location points

Η μέθοδος findStayPoints που διακρίνουμε στην εικόνα 41, επιστρέφει μία λίστα με όλα τα

stay points που προέκυψαν, όπως αναφέραμε παραπάνω. Εάν αυτή η λίστα δεν είναι κενή, δηλαδή προέκυψε τουλάχιστον ένα stay point, τότε ανανεώνονται και τα stay regions, τα οποία αφορούν όλους τους χρήστες.

Εικόνα 42: Συνέχεια μεθόδου εικόνας 41

Να αναφέρουμε σε αυτό το σημείο, ότι για οποιαδήποτε ενέργεια σχετική με τα stay points και stay regions, ενημερώνεται κατάλληλα και η βάση, ενώ και ο αλγόριθμος εξαγωγής των stay regions θα αναλυθεί σε επόμενη ενότητα.

iii.Stay Points

Το μονοπάτι στο οποίο στέλνει αίτημα ο χρήστης για να λάβει τα stay points είναι το 'http://192.168.2.4:8080/RestWebService/rest/stayPoints ακολουθούμενο από το username του. Ο εξυπηρέτης από την πλευρά του, ανοίγει μία σύνδεση με τη βάση δεδομένων και επιστρέφει στον πελάτη σε JSON format μόνο τα stay points για τα οποία ο τελευταίος δεν είναι ενημερωμένος.

Εικόνα 43: Μέθοδος sendStayPoints

Στην παραπάνω εικόνα βλέπουμε ότι στη μέθοδο sendStayPoints καλείται η μέθοδος getStayPoints της κλάσης DBConnector. Η μέθοδος getStayPoints είναι static και για αυτό το λόγο καλείται χωρίς δημιουργία αντικειμένου DBConnector. Περισσότερα για τη σχεδίαση της βάσης δεδομένων και για τα ερωτήματα που γίνονται σε αυτήν θα αναλυθούν στην επόμενη ενότητα.

iv. Stay Regions

Το μονοπάτι στο οποίο στέλνει αίτημα ο χρήστης για να λάβει τα stay regions είναι το 'http://192.168.2.4:8080/RestWebService/rest/stayRegions/ , ακολουθούμενο από {username} , το όνομα του χρήστη δηλαδή. Ο server . Όπως και με τα stay points, ανοίγει μία σύνδεση με τη βάση, απλά αυτή τη φορά τα stay regions δε σχετίζονται με έναν χρήστη αλλά με όλους τους χρήστες της εφαρμογής. Στην ενότητα 3.4.3 θα αναλύσουμε τη βάση δεδομένων που χρησιμοποιείται στην πλευρά του εξυπηρέτη, και στην 3.4.4 τους αλγόριθμους εξαγωγής των stay points και stay regions.

3.4.3 Βάση Δεδομένων

Όπως αναφέραμε και στην ενότητα 3.2 η βάση δεδομένων που χρησιμοποιήθηκε είναι η mysql. Μόλις εκτελείται ο server, καλείται η μέθοδος, createPOIDB, η οποία δημιουργεί τη βάση δεδομένων του server.

```
63⊖   public static void createPOIDB() {
64        DBCreation sql = DBCreation.getInstance();
65        Connection conn;
66        PreparedStatement st;
67        conn = sql.connectWithMysql();
68
69        try {
70            st = conn.prepareStatement("CREATE DATABASE IF NOT EXISTS POIDB");
71            st.executeUpdate();
72        }
73        catch (SQLException e) {
74            System.out.println("Error while creating POIDB");
75            return;
76        }
77        finally {
78            try {
79                conn.close();
80            }
81            catch (SQLException e) {
82                e.printStackTrace();
83            }
```

Εικόνα 44: Μέθοδος createPOIDB

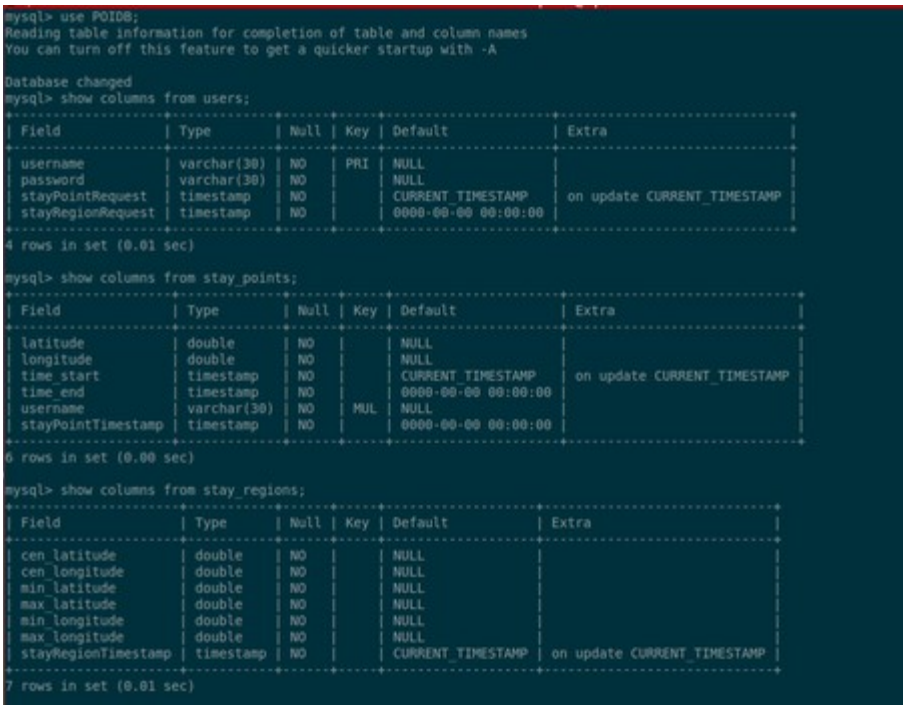
Η μέθοδος createPOIDB ανοίγει μία σύνδεση με τη mysql, και στη συνέχεια δημιουργεί τη βάση POIDB. Τέλος, δημιουργεί τα απαραίτητα tables στη βάση που δημιουργήθηκε . Η βάση περιέχει 3 tables, Το table users, το table stay points και το table stay regions.

1. Users table: Περιέχει το όνομα χρήστη, τον κωδικό πρόσβασης, καθώς και 2 timestamp, τα οποία αποτελούν τη χρονική στιγμή που ο χρήστης έστειλε αίτημα για stay points και stay regions. Πρωτεύοον κλειδί αυτού του table είναι το username.

2. Stay_points table: Περιέχει το κεντροειδές του stay point(latitude και longitude), 2 timestamp που αντιπροσωπεύουν τις χρονικές στιγμές που ο χρήστης έφτασε και έφυγε από το σημείο, το username του χρήστη στον οποίο ανήκει το stay point, αλλά και το timestamp που αντιπροσωπεύει τη χρονική στιγμή που εισήλθε αυτή η εγγραφή στη βάση δεδομένων. Το username του χρήστη όπως φαίνεται και στην εικόνα 45, είναι δευτερεύον κλειδί και κάνει reference στο table users.

3. Stay_regions table: Περιέχει το κεντροειδές της περιοχής (cen_latitude και cen_longitude), τις ελάχιστες και μέγιστες συντεταγμένες(min_latitude, max_latitude, min_longitude και max_longitude) καθώς και το timestamp που εισήχθη το stay region στη βάση. Εδώ δεν υπάρχει στήλη user, καθώς όπως έχουμε ήδη αναφέρει τα stay regions προκύπτουν από τα stay points όλων των χρηστών.

Παρακάτω παρατίθεται ένα screenshot όπου φαίνεται η δομή των tables της POIDB.



Εικόνα 45: Tables της βάσης δεδομένων POIDB

Οι μέθοδοι που έχουν υλοποιηθεί για επικοινωνία με τη βάση POIDB, περιλαμβάνονται στο αρχείο DBConnector και είναι οι εξής:

- ***public static boolean register(String username, String password)***
  Η μέθοδος αυτή εισάγει έναν νέο χρήστη στο table users.

- ***public static boolean login(String username,String password)***

Η μέθοδος αυτή ελέγχει εάν τα στοιχεία που έδωσε ένας χρήστης είναι σωστά.

- ***public static void insertStayPoints(List<StayPoint> list, String user)***
  Η μέθοδος αυτή εισάγει τα stay points που προέκυψαν για το χρήστη user.

- ***public static void insertStayRegion(StayRegion sr)***

Εισάγει ένα stay region στο αντίστοιχο table.

- ***public static String getStayPoints(String username)***

Επιστρέφει τα stay points που αφορούν το χρήστη username

- ***public static Vector<Stay Point> getAllStayPoints()***

Επιστρέφει όλα τα stay points που υπάρχουν στο αντίστοιχο table.

- ***public static String getStayRegions(String username)***

Επιστρέφει όλα τα stay regions για τα οποία ο χρήστης username δεν είναι ενημερωμένος.

- ***public static String deleteStayRegions()***

Διαγράφει από το table stay_regions όλες τις εγγραφές.

3.4.4 Θέματα Προγραμματισμού

i. Εξαγωγή Stay Points

Σε αυτό το σημείο θα αναλύσουμε τον αλγόριθμο που χρησιμοποιείται για την εξαγωγή των stay points βάσει των location points που έλαβε από το χρήστη. Η μέθοδος findStayPoints, ένα κομμάτι της οποίας παρατίθεται παρακάτω σε ψευδοκώδικα, παίρνει ως όρισμα μία λίστα από location points.

```
i=1
lsp
while i < N
    j = i + 1
    while j <= N
        if(TimeDiff(pj,pj-1)>Tmax)
            if(TimeDiff(pi,pj-1)>Tmin)
                lsp.append(EstimateStayPoint(pi ... pj-1))
            i=j
            break
        else if(Distance(pi,pj)>Dmax)
            if(TimeDiff(pi,pj-1)>Tmin)
                lsp.append(EstimateStayPoint(pi ... pj-1))
                i=j
                break
            i++
            break
        else if(j==N)
            if(TimeDiff(pi,pj)>Tmin)
                lsp.append(EstimateStayPoint(pi ... pj))
            i=j
           break
        j++

StayPoint estimateStayPoint(PointsList L)
    lat,long = EstimateCendroid(L)
    Tstart = L.getFirst().T
    Tend = L.getLast().T
    StayPoint sp = new StayPoint(lat,long,Tstart,Tend)
    return StayPoint
```

Εικόνα 46: Μέθοδος findStayPoints σε ψευδογλώσσα

Αρχικά, για τον αλγόριθμο αυτόν χρειάζονται 3 thresholds, ή αλλιώς κατώφλια, τα Tmax, Tmin και Dmax. Το Tmin είναι ο ελάχιστος χρόνος που πρέπει να έχει παρέλθει ανάμεσα σε 2 location points που βρέθηκαν, προκειμένου αυτά να σχηματίσουν ένα stay point. Το Tmax δηλώνει το μέγιστο χρόνο που μπορεί να έχει περάσει ανάμεσα σε δύο location points που βρέθηκαν. Εάν η χρονική διαφορά ανάμεσα σε 2 location points είναι μεγαλύτερη από Tmax, είναι πολύ πιθανό είτε να υπήρξε κάποια δυσλειτουργία με το GPS, είτε τα location points αυτά να έχουν προκύψει από διαφορετική χρήση monitoring. Τέλος, το Dmax είναι η μέγιστη απόσταση που μπορεί να απέχουν 2 location points προκειμένου να σχηματίσουν ένα stay point. Για τον υπολογισμό του κεντροειδούς του stay point, απλά προστίθενται όλες οι συντεταγμένες των location points, και στη συνέχεια διαιρούνται με το πλήθος τους, έτσι ώστε να προκύψει η μέση τετμημένη και τεταγμένη αντίστοιχα.
Τέλος, παρατίθεται μία εικόνα στην οποία φαίνεται η μέθοδος που καλείται για τον υπολογισμό της απόστασης μεταξύ 2 σημείων. Σημειώνεται ότι η ακτίνα της γης υπολογίζεται σε 6371000 μέτρα.

```
public static float calculateDistance(LocationPoint lp1,LocationPoint lp2) {

    float lat1 = (float) lp1.getLatitude();
    float long1 = (float) lp1.getLongitude();
    float lat2 = (float) lp2.getLatitude();
    float long2 = (float) lp2.getLongitude();

    double earthRadius = 6371000; //meters
    double dLat = Math.toRadians(lat2-lat1);
    double dLng = Math.toRadians(long2-long1);
    double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
               Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) *
               Math.sin(dLng/2) * Math.sin(dLng/2);
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    float dist = (float) (earthRadius * c);

    //System.out.println("Distance is "+ dist);

    return dist;
    }
```

Εικόνα 47: Μέθοδος υπολογισμού απόστασης 2 σημείων

ii. Εξαγωγή Stay Regions – DBSCAN algorithm

Μία από τις δυσκολίες που αντιμετωπίστηκαν στην υλοποίηση του server, ήταμ η μέθοδος από την οποία θα προέκυπταν τα stay regions, ή αλλιώς ο τρόπος με τον οποίο θα ομαδοποιούνταν τα stay points. Τελικά προτιμήθηκε ο αλγόριθμος DBScan, λόγω της πολύ καλής απόδοσης που έχει όσον αφορά το χρόνο εκτέλεσης. Ο αλγόριθμος DBScan είναι ένας αλγόριθμος ομαδοποίησης (clustering algorithm) ο οποίος βασίζεται στην πυκνότητα των δεδομένων.

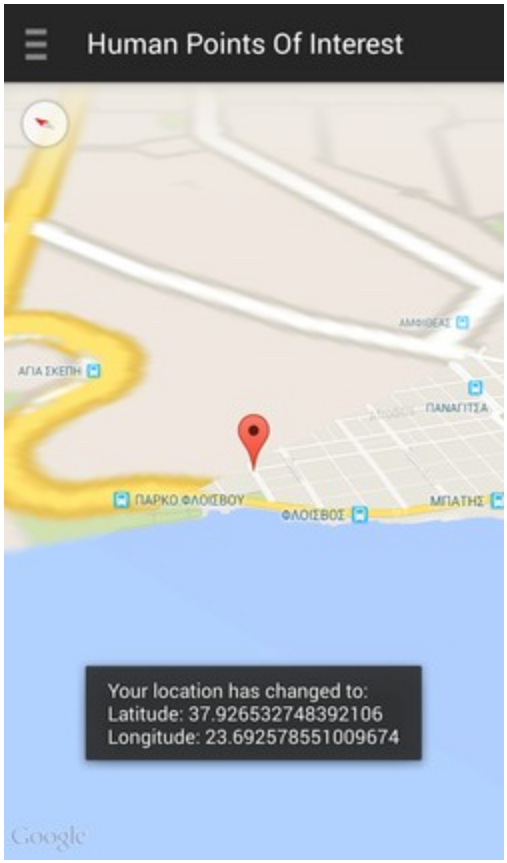
```
public static Vector<List<StayPoint>> applyDbscan(Vector<StayPoint> list)
{
    resultList.clear();
    Utility.VisitList.clear();

    pointList = list;
    int index2 =0;

    while (pointList.size()>index2){
        StayPoint p =pointList.get(index2);
         if(!Utility.isVisited(p)){

            Utility.Visited(p);

            Neighbours =Utility.getNeighbours(p);


            if (Neighbours.size()>=minpt){


                int ind=0;
                while(Neighbours.size()>ind){

                    StayPoint r = Neighbours.get(ind);
                    if(!Utility.isVisited(r)){
                        Utility.Visited(r);
                    Vector<StayPoint> Neighbours2 = Utility.getNeighbours(r);
                    if (Neighbours2.size() >= minpt){
                        Neighbours=Utility.Merge(Neighbours, Neighbours2);
                        }
                    } ind++;
                }
```

Εικόνα 48: Υλοποίηση αλγορίθμου DBScan

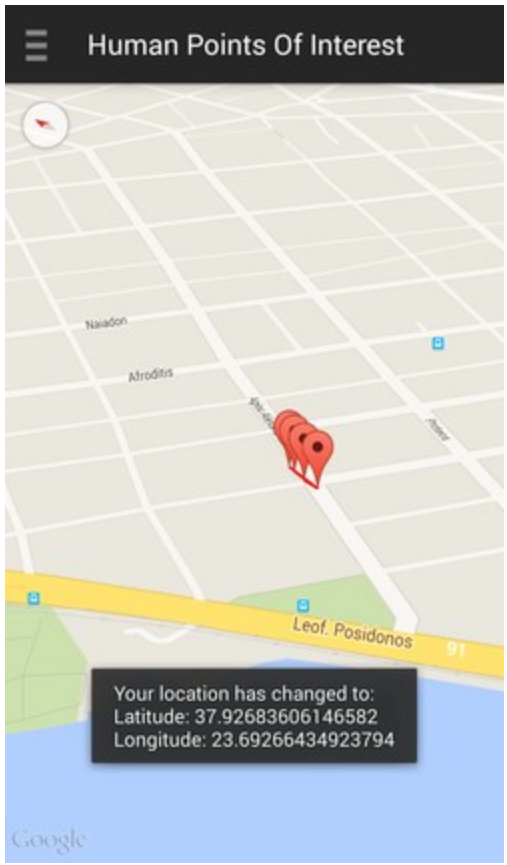
Στην παραπάνω εικόνα βλέπουμε το κύριο κομμάτι κώδικα κατά το οποίο εφαρμόζεται ο αλγόριθμος DBScan. Όσον αφορά τη δομή του project, έχει δημιουργηθεί το package dbScan, το οποίο περιέχει τις κλάσεις Utility και dbscan. Η κλάση dbscan περιέχει την υλοποίηση του αλγορίθμου, ενώ η κλάση Utility περιέχει βοηθητικές μεθόδους για τη dbscan, όπως για παράδειγμα την isVisited(StayPoint c) η οποία ελέγχει εάν ένα σημείο έχει επισκεφτεί κατά την εκτέλεση μίας επανάληψης του dbscan, και την getNeighbours(StayPoint p) η οποία επιστρέφει μία λίστα από stay points που είναι γειτονικά με ένα συγκεκριμένο σημείο.

3.5 Πειραματική εκτέλεση εφαρμογής

Σε αυτή την ενότητα παρουσιάζεται μία ενδεικτική εκτέλεση της εφαρμογής. Ο χρήστης admin έχει εισαχθεί στην εφαρμογή και αφού έχει ενργοποιήσει τη λειτουργία του GPS επιλέγει 'Start Monitoring'. Μερικά από τα location points που προέκυψαν παρατίθενται στις εικόνες 49 και 50. Στη δεύτερη έχει γίνει zoom για να φανεί και το trajectory, η τροχιά δηλαδή, η οποία έχει προκύψει από τη λειτουργία monitoring.



Εικόνα 49: Location Point που βρέθηκε κατά τη λειτουργία Monitoring



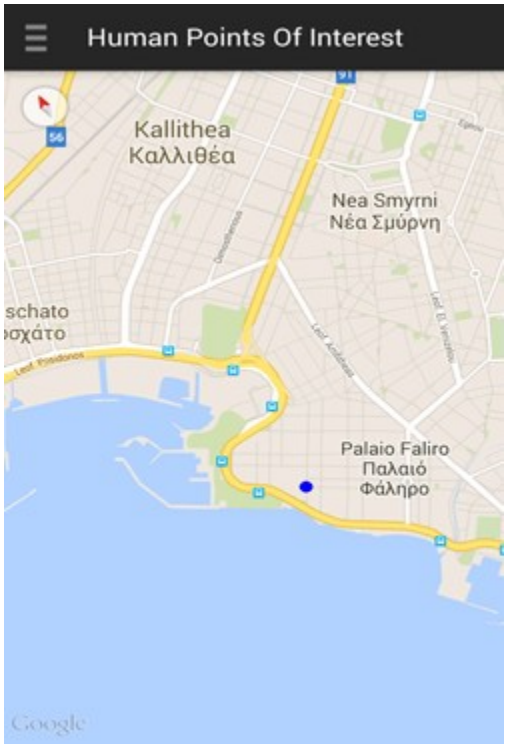
Εικόνα 50: Trajectory που έχει σχηματιστεί από 4 location points

Ο χρήστης στη συνέχεια συνδέεται στο ίδιο δίκτυο με το server , και μόλις αυτός επεξεργάζεται τα location points που έλαβε, προκύπτει ένα stay point και ένα stay region.

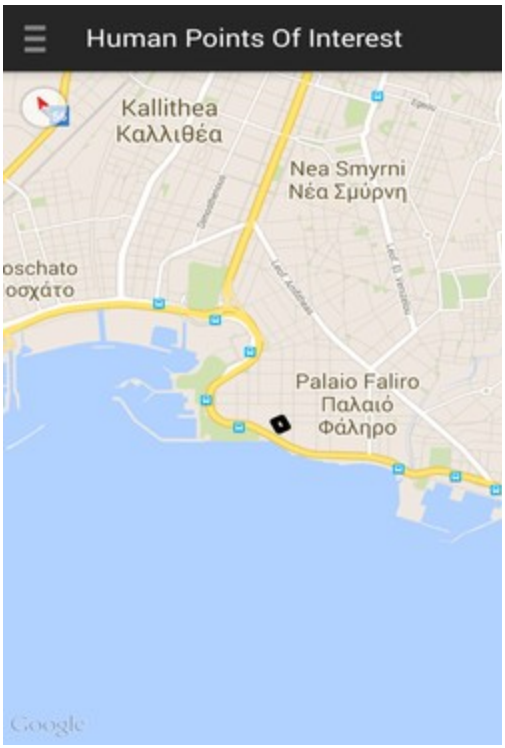
```
Location Point01  is :
 Latitude: 37.926532748392106
 Longitude: 23.692578551009674
Location Point11  is :
 Latitude: 37.92683606146582
 Longitude: 23.69266434923794
Location Point21  is :
 Latitude: 37.92713664962728
 Longitude: 23.6928134857499
Location Point31  is :
 Latitude: 37.713664962728
 Longitude: 23.134857499
STAY POINT IS: 37.9266844049289623.69262145012381
N6
This cluster is [latitude = 37.926532748392106, longitude = 23.692578551009674, , timeStart =
Minimum and maximum latitude is 37.926532748392106 37.92668440492896
Minimum and maximum longitude is 23.692578551009674 23.69262145012381
The cendroid of the cluster is  37.92655802448158 23.692585700862O3
```

Εικόνα 51: Logcat server, μόλις λάβει τα location points

Παραπάνω διακρίνουμε τις συντεταγμένες του stay point αλλά και του stay region που προέκυψε. Στη συνέχεια ο client επιλέγει από το μενού της εφαρμογής 'Stay Points' και 'Stay Regions' αντίστοιχα, και το αποτέλεσμα φαίνεται παρακάτω.



Εικόνα 52: Stay point που προέκυψε



Εικόνα 53: Stay region μετά την εφαμογή του DBScan

ΒΙΒΛΙΟΓΡΑΦΙΑ-ΠΗΓΕΣ

[1] https://el.wikipedia.org/wiki/Android
[2] http://www.linuxinsider.gr/forum/εισαγωγή-στη-δημιουργία-mobile-εφαρμογών-για-android
[3] http://developer.android.com/guide/components/activities.html
[4] http://developer.android.com/guide/components/fragments.html
[5] http://cs.northwestern.edu/~akuzma/classes/CS495-s09/doc/Mining%20Interesting%20Locations%20and%20Travel%20Sequences%20From%20GPS%20Trajectories.pdf
[6] http://www.idiap.ch/~gatica/publications/MontoliuGatica-mum10.pdf
[7] https://jersey.java.net/
[8] https://en.wikipedia.org/wiki/Web_service
[9] https://en.wikipedia.org/wiki/JSON
[10] http://json.org/
[11] https://developer.android.com/training/implementing-navigation/nav-drawer.html
[12] https://developers.google.com/identity/protocols/OAuth2
[13] http://source.android.com/source/build-numbers.html