



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΤΟΥ ΦΟΙΤΗΤΗ

ΜΩΥΣΙΑΔΗ ΑΝΑΣΤΑΣΙΟΥ

με θέμα:

Εφαρμογή συλλογής γεωχωρικών δεδομένων απόδοσης
ασύρματων δικτύων με χρήση υπηρεσίας παρασκηνίου για
έξυπνες συσκευές

-

Crowdsourcing Application for Personalized Location – Based
QoE of Wireless Networks via Background Service on Smart
Devices

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ ΣΤΗΝ

ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗ

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ:

Β. Δαλάκας, Α. Τσαδήμας, Μ. Νικολαΐδου

Περιεχόμενα

1	ΕΙΣΑΓΩΓΗ	7
2	ΕΠΙΣΚΟΠΗΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑΣ	9
2.1	Παραδείγματα location – based crowdsourcing εφαρμογών.....	9
2.2	Τεχνικές Drive Testing από location – based crowdsourcing εφαρμογές.....	13
3	ΠΡΟΒΛΗΜΑ - ΤΟΠΟΘΕΤΗΣΗ.....	21
4	ΤΟ ANDROID	22
4.1	Γενικά	22
4.2	Αρχιτεκτονική	23
4.3	Εκδόσεις.....	25
4.4	Android Components (Συστατικά).....	27
4.4.1	Activities.....	27
4.4.2	Broadcast Receivers	31
4.4.3	Services.....	31
4.4.4	Intents	33
4.5	Alarm Manager	35
4.6	Location API.....	36
5	ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ	37
6	ΥΛΟΠΟΙΗΣΗ	41
6.1	Περιγραφή Εφαρμογής.....	41
6.1.1	Προϋπάρχουσα Υλοποίηση	41
6.1.2	Επέκταση	42
6.2	Εργαλεία Ανάπτυξης Λογισμικού.....	48
6.2.1	Επιλογή IDE (Integrated Development Environment).....	48
6.2.2	Eclipse with ADT plugin.....	48
6.2.3	NetBeans 8.0.2	52

6.3	Η υπηρεσία GCM.....	54
6.3.1	Περιγραφή	54
6.3.2	GCM Αρχιτεκτονική	56
6.3.3	Υλοποίηση της υπηρεσίας GCM στην Εφαρμογή.....	57
6.4	REST Web Services	62
6.4.1	Περιγραφή	62
6.4.2	Υλοποίηση REST web services στην Εφαρμογή.....	65
6.5	Η Βάση δεδομένων MySQL	69
6.5.1	Περιγραφή	69
6.5.2	Οι Πίνακες της Βάσης Δεδομένων MySQL στην Εφαρμογή.....	69
6.6	Βιβλιοθήκες	77
6.6.1	Google Play Services.....	77
6.6.2	Google Maps V2	80
6.6.3	MPAndroid Chart.....	84
6.7	Χρήση GeoServer για Οπτικοποίηση Μετρήσεων	85
6.8	Ελάχιστες Απαιτήσεις - Περιορισμοί.....	85
7	ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΕΦΑΡΜΟΓΗΣ – ΚΩΔΙΚΑΣ	87
7.1	NetBeans REST Project.....	87
7.1.1	SpeedTestServer.....	87
7.1.2	RESTFulHuaLocation	88
7.2	Android Project	91
7.2.1	Έλεγχοι Λειτουργίας	93
7.2.2	Καμπάνιες	94
7.2.3	Υπηρεσία Παρασκηνίου	99
7.2.4	Προτιμήσεις Χρήστη.....	104
7.2.5	Google Maps.....	106
7.2.6	Στατιστικά Μετρήσεων	112

7.3	Παραδείγματα Απεικόνισης Μετρήσεων σε Google Maps.....	116
7.4	Παραδείγματα Απεικόνισης Στατιστικών	119
8	ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΕΠΕΚΤΑΣΕΙΣ.....	123
8.1	Απολογισμός.....	123
8.2	Συμπεράσματα	123
8.3	Επεκτάσεις	124
	Βιβλιογραφία - Αναφορές.....	126
	ΠΑΡΑΡΤΗΜΑΤΑ.....	130
	Παράρτημα Α: Android Project's manifest	130
	Παράρτημα Β: Κώδικας υπηρεσίας παρασκηνίου	134

Περίληψη

Στη σημερινή εποχή οι πληθοποριστικές εφαρμογές βασισμένες σε χωρικά δεδομένα (location – based crowdsourcing apps), στο χώρο του διαδικτύου, συνεχώς αυξάνονται. Εφαρμογές που βασίζουν τη λειτουργία τους σε μαζικά δεδομένα που έχουν προκύψει από τους ίδιους τους χρήστες των εφαρμογών, είναι πολύ χρήσιμες λόγω του τεράστιου πλήθους των στοιχείων που επεξεργάζονται. Η επιτυχία τέτοιων εφαρμογών εξαρτάται άμεσα από τη συμμετοχή των χρηστών σε τέτοιου είδους διαδικασίες.

Τα τελευταία χρόνια, με την έξαρση της διάδοσης των έξυπνων κινητών τηλεφώνων, που διαθέτουν τις κατάλληλες τεχνολογίες, έχουν αρχίσει να αναπτύσσονται location – based crowdsourcing εφαρμογές σε τέτοιου είδους συσκευές. Η επιτυχία τους είναι πολύ μεγαλύτερη λόγω της μαζικής συμμετοχής των χρηστών αλλά και της 24ωρης διαθεσιμότητάς τους από τη στιγμή που οι χρήστες έχουν σχεδόν πάντοτε μαζί τα κινητά τους τηλέφωνα.

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της συγκεκριμένης εργασίας είναι μία location-based crowdsourcing εφαρμογή υλοποιημένη σε Android έξυπνες συσκευές, που έχει ως σκοπό τη μέτρηση της απόδοσης ασύρματων δικτύων, στα οποία είναι συνδεδεμένη. Οι μετρήσεις γίνονται από τους ίδιους τους χρήστες, δίνοντας τη δυνατότητα για συγκέντρωση δεδομένων σε πραγματικό χρόνο από διαφορετικές γεωγραφικές περιοχές, με ταυτόχρονη δυνατότητα αναπαράστασής τους σε χάρτες της Google και αποτύπωσης στατιστικών διαγραμμάτων. Αποτελεί συνέχεια και επέκταση της εφαρμογής που αναπτύχθηκε από τον Κορωνιώτη Νίκο [1] στα πλαίσια της πτυχιακής του εργασίας.

Με την υλοποίηση αυτής της εργασίας δίνεται η δυνατότητα στους χρήστες να ενημερώνονται για την ποιότητα του δικτύου όλων των παρόχων κινητής τηλεφωνίας, όπως και για την ποιότητα ασύρματων δικτύων WiFi, εξάγοντας πολύ χρήσιμα συμπεράσματα. Επίσης, θα μπορούσε να θεωρηθεί ως εργαλείο για τις εταιρείες κινητής τηλεφωνίας, ώστε να διερευνήσουν τις ατέλειες των υφιστάμενων δικτύων τους, προχωρώντας σε εργασίες βελτιστοποίησης, σε περιοχές που από τις μετρήσεις φαίνεται να υστερούν.

Abstract

Nowadays location – based crowdsourcing applications via Internet are constantly on the rise. Applications that base their operation on public data coming from users themselves are very useful because of the huge number of processing data. Success of such applications directly depends on user participation in such processes.

In recent years, the rise of smart mobile phones, which possess the necessary technologies, has led to the development of location – based crowdsourcing applications. Their success is much greater due to massive participation of users and their 24-hour availability, since users are almost always carrying with them their mobile phones.

The application developed in the context of the current thesis is a location-based crowdsourcing application implemented on Android mobile devices, designed to measure the performance of wireless networks to which these devices are connected. Measurements are made by the users themselves and this fact enables the gathering of a huge amount of data throughout the country in a short period of time. The data, once collected, are presented both in Google Maps as well as in statistical diagrams. This work is a continuation and extension of the application developed by Koroniotis Nikolaos [1].

With the implementation of this thesis, users are given the possibility to get information for the network quality of all mobile phone providers, as well as for the quality of WiFi networks, extracting very useful conclusions. In addition, this application could serve as a tool for mobile network operators, in order for them to explore the imperfections of their existing networks and perform the necessary improvements in areas where seem to fall behind.

1 ΕΙΣΑΓΩΓΗ

Η εξέλιξη των κινητών τηλεφώνων στη σημερινή εποχή είναι ραγδαία. Πλέον τα κινητά τηλέφωνα ενσωματώνουν λειτουργικό σύστημα και λογισμικό παρόμοια με το λογισμικό των προσωπικών υπολογιστών PC. Έτσι, τα κινητά τηλέφωνα αποκαλούνται πλέον ως «έξυπνες κινητές συσκευές» (smartphones [2]), αφού η χρήση τους δεν περιορίζεται μόνο στην πληροφόρηση του χρήστη για την ώρα και ημερομηνία, αλλά ενσωματώνουν πληθώρα λειτουργιών που κάνουν τη ζωή του χρήστη ευκολότερη. Αυτό έχει σαν συνέπεια τη μαζική διάδοση των smartphones στο καταναλωτικό κοινό.

Ένα μεγάλο πλεονέκτημα των έξυπνων κινητών τηλεφώνων είναι ότι έχουν τη δυνατότητα να ενσωματώνουν διαφόρων ειδών τεχνολογίες, όπως τη δυνατότητα σύνδεσής τους στο διαδίκτυο μέσω τεχνολογιών που βασίζονται στο πρότυπο IEEE 802.11 ή Wi-Fi όπως κοινώς αποκαλείται, αλλά και στο σύστημα GSM (Global System for Mobile Communications) 2ης (2G), UMTS (Universal Mobile Telecommunications System) 3ης (3G) ή LTE (telecommunication) 4ης γενιάς (4G). Επίσης, η τεχνολογία GPS (Global Positioning System) βοηθά στην ανίχνευση της θέσης του χρήστη σε πραγματικό χρόνο. Αυτές οι τεχνολογίες-υπηρεσίες δίνουν τη δυνατότητα για ανάπτυξη εφαρμογών σε smartphones που έχουν πρόσβαση στο διαδίκτυο και στηρίζονται σε γεωχωρικά δεδομένα. Οι εφαρμογές που χρησιμοποιούν το διαδίκτυο και τα γεωχωρικά δεδομένα που προέρχονται από χρήστες, για να δημιουργήσουν διαδικασίες προς επίλυση με τη βοήθεια των μαζών, ονομάζονται location – based crowdsourcing [3] εφαρμογές. Σύμφωνα με την ιστοσελίδα wikipedia [4], πληθοπορισμός (crowdsourcing) είναι η διαδικασία απόκτησης υπηρεσιών, ιδεών ή γενικά οποιουδήποτε είδους περιεχομένου, «μία μορφή συλλογικής διαδικτυακής δραστηριότητας στην οποία ένα άτομο, ένα ίδρυμα, ένας μη κερδοσκοπικός οργανισμός ή μία εταιρεία προτείνει σε μία ομάδα ατόμων με ποικίλες γνώσεις, ετερογένεια και αριθμό, μέσω μίας ανοικτής πρόσκλησης, να αναλάβουν εθελοντικά μια εργασία. Η ανάληψη της εργασίας, η οποία ποικίλλει σε πολυπλοκότητα, περιλαμβάνει πάντοτε αμοιβαίο όφελος και για τις δύο πλευρές». Η λέξη crowdsourcing στα αγγλικά προέρχεται από την ένωση των λέξεων crowd (πλήθος) και outsourcing (εξωτερική ανάθεση εργασιών). Οι εφαρμογές crowdsourcing είναι αρκετά διαδεδομένες στο χώρο του διαδικτύου και πλέον έχουν αρχίσει να επεκτείνονται μαζικά και στο χώρο των έξυπνων κινητών τηλεφώνων.

Μέσα από τη μαζική χρήση τέτοιων εφαρμογών γεννιέται η ανάγκη για διερεύνηση της ποιότητας του δικτύου κινητής τηλεφωνίας αλλά και Wi-Fi, ώστε να απορρέουν ασφαλή συμπεράσματα για την ποιοτική αλλά και ποσοτική υποστήριξη τέτοιων εφαρμογών. Η

διερεύνηση αυτή γίνεται από τις ίδιες τις εταιρείες κινητής τηλεφωνίας με τη λήψη ανάλογων μετρήσεων που αφορούν το δίκτυο κινητής τηλεφωνίας. Οι δυνατότητες που μας παρέχουν όμως τα σημερινά κινητά τηλέφωνα μας οδηγούν στην ανάπτυξη location – based crowdsourcing εφαρμογών που έχουν ως σκοπό τη λήψη τέτοιου είδους μετρήσεων από τους ίδιους τους χρήστες κινητών τηλεφώνων. Σε αυτό βοηθά σημαντικά και το γεγονός ότι το λειτουργικό σύστημα των κινητών τηλεφώνων πλέον είναι, στην πλειονότητά του, συγκεκριμένο (android ή iOS) με αποτέλεσμα η υλοποίηση τέτοιων εφαρμογών σε 2 πλατφόρμες να αφορά σχεδόν το σύνολο των έξυπνων κινητών τηλεφώνων.

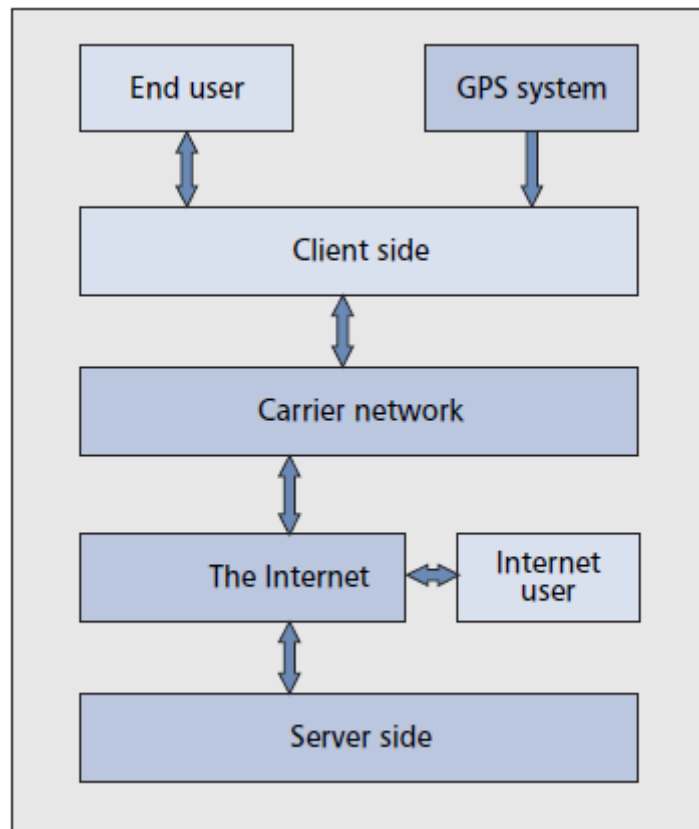
Στη συνέχεια της εργασίας, θα γίνει μία ιστορική αναδρομή για την εξέλιξη των location – based crowdsourcing εφαρμογών στο διαδίκτυο και παρουσίαση κάποιων χαρακτηριστικών τους. Θα αναλυθεί η Αρχιτεκτονική του συστήματος που υλοποιήσαμε και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξή του. Θα γίνει αναφορά στο λειτουργικό σύστημα Android και στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος, όπως REST web services, GCM (Google Cloud Messaging) υπηρεσία, βάση δεδομένων MySQL, όπως και για τις βιβλιοθήκες που ενσωματώθηκαν. Στη συνέχεια, θα γίνει μία σύντομη περιγραφή της εφαρμογής και των πιο βασικών λειτουργιών της, παραθέτοντας και σχετικά τμήματα κώδικα. Τέλος, θα παρουσιαστούν τα συμπεράσματα της μελέτης μας και οι τυχόν προεκτάσεις που μπορούν να υλοποιηθούν στο μέλλον.

2 ΕΠΙΣΚΟΠΗΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑΣ

2.1 Παραδείγματα location – based crowdsourcing εφαρμογών

Η ιδέα για ανάπτυξη εφαρμογών που θα βασίζονται σε γεωχωρικά δεδομένα (location – based apps) είναι γνωστή εδώ και μία δεκαετία τουλάχιστον. Το 2006 οι **Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Pérez, and Nevine Labib Georggi** [5] είχαν επισημάνει την ανάγκη για την ύπαρξη τέτοιων εφαρμογών. Είχαν αναφέρει ως παράδειγμα μία υπάρχουσα εφαρμογή εκείνης της εποχής, το προηγμένο σύστημα πληροφόρησης ταξιδιώτη (ATIS), η οποία πληροφορούσε τους εγγεγραμμένους χρήστες μέσω γραπτών μηνυμάτων για προορισμούς του ενδιαφέροντος του χρήστη που αφορούσαν αξιοθέατα, την κίνηση στους δρόμους κλπ. Αυτές οι πληροφορίες όμως θα ήταν πιο χρήσιμες εάν με κάποιο τρόπο μπορούσε το σύστημα να ανιχνεύσει τη θέση του ταξιδιώτη ώστε να τον πληροφορεί για προορισμούς σχετικούς με την υπάρχουσα θέση του. Να αναπτυχθεί δηλαδή ένα είδος location – based εφαρμογής. Άλλο ένα παράδειγμα αφορούσε μία εφαρμογή Amber Alert όπου ο χρήστης θα μπορούσε να στείλει εικόνα ή και video ενός ατόμου που μοιάζει με το προς αναζήτηση άτομο.

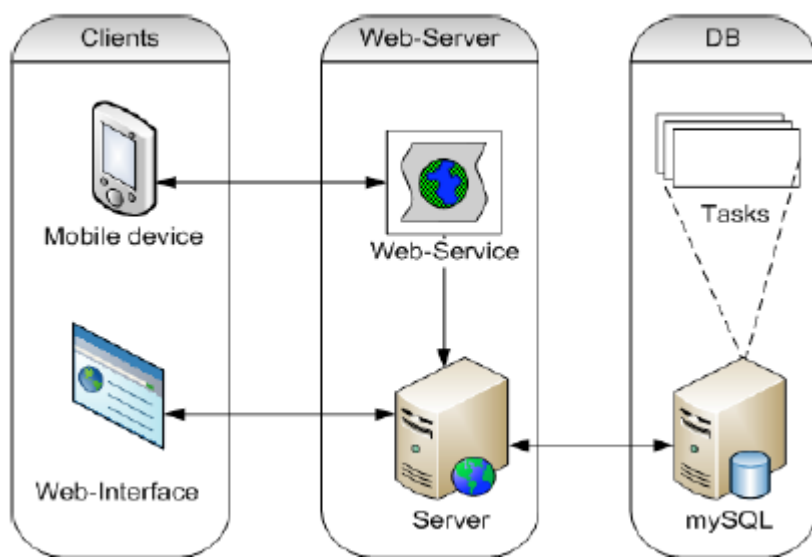
Η αρχιτεκτονική τέτοιων εφαρμογών περιγράφεται στο άρθρο [5], όπως φαίνεται στην Εικόνα 2.1. Σε γενικές γραμμές η ιδέα μιας τέτοιας αρχιτεκτονικής, σε υψηλό επίπεδο, μπορεί να εφαρμοστεί και σήμερα, αν προσθέσουμε σαν δικτυακή δυνατότητα την τεχνολογία Wi-Fi. Η πλευρά του πελάτη (client side) μπορεί να περιλαμβάνει κινητά τηλέφωνα, PDAs (Personal Digital Assistant) ενώ η πλευρά του διακομιστή (server side) μπορεί να υλοποιεί κάποιον web ή application server και να τηρεί τα δεδομένα σε κάποια βάση δεδομένων.



Εικόνα 2.1: Αρχιτεκτονική συστήματος location – based εφαρμογών σε υψηλό επίπεδο

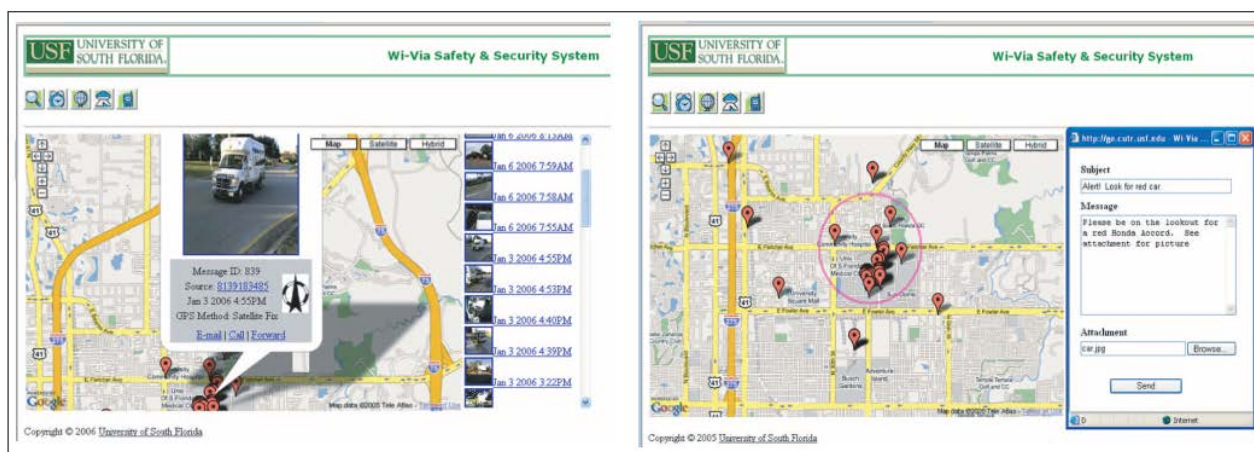
Μία αντίστοιχη αρχιτεκτονική αναφέρεται σε μεταγενέστερο άρθρο, το 2010, από τους **Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, Zahid Nawaz** [6], όπως φαίνεται και στην Εικόνα 2.2. Το συγκεκριμένο άρθρο περιγράφει μία αρχιτεκτονική για location – based crowdsourcing εφαρμογές. Σε σχέση με το προηγούμενο άρθρο που είναι χρονικά παλαιότερο, εισάγει την έννοια του crowdsourcing σε συσκευές κινητής τηλεφωνίας. Στην πλευρά του διακομιστή (server side) έχει συμπεριληφθεί η έννοια του web service ως διαδικτυακή υπηρεσία που εξυπηρετεί τους πελάτες (clients). Ως clients γίνεται αναφορά και σε ιστοσελίδες, από τη στιγμή που υπάρχουν πολλές crowdsourcing εφαρμογές στο διαδίκτυο. Για παράδειγμα, το Amazon’s Mechanical Turk [7], μία διαδικτυακή αγορά (marketplace) για έργα που απαιτούν την ανθρώπινη νοημοσύνη και στα οποία ο χρήστης μπορεί να ορίσει τις διαδικασίες που ζητά να ολοκληρωθούν με κάποιο χρηματικό αντίτιμο. Όπως επίσης και το iStockPhoto [8], μία διαδικτυακή εταιρεία που προσφέρει συλλογές φωτογραφιών και video αγορασμένες από φωτογράφους, τις οποίες μπορούν να αγοράσουν οι χρήστες μέσω πίστωσης (credit) αλλά και πολλές άλλες crowdsourcing εφαρμογές. Αντίστοιχες υλοποιήσεις μπορούν να δημιουργηθούν και για smartphones. Για παράδειγμα, ο **Nathan Eagle** δημιούργησε την εφαρμογή *txteagle* [9],

με την οποία δίνεται η δυνατότητα σε κατόχους έξυπνων κινητών, χωρών της Αφρικής όπως Κένυα και Ρουάντα, να συμμετέχουν σε απλές διαδικασίες μετάφρασης κειμένων της τοπικής τους γλώσσας έναντι χρηματικής αμοιβής. Οι διαδικασίες αυτές βοηθούν και εταιρείες του Δυτικού κόσμου που χρειάζονται αυτές τις μεταφράσεις, αλλά και τους ίδιους τους χρήστες της εφαρμογής, αφού πληρώνονται.



Εικόνα 2.2: Αρχιτεκτονική location – based crowdsourcing συστήματος

Στο άρθρο [5], οι συγγραφείς προτείνουν την υλοποίηση μίας εφαρμογής, η οποία μπορεί σήμερα να χαρακτηριστεί ως πληθοποριστική (crowdsourcing). Βασιζόμενοι στην αρχιτεκτονική της Εικόνας 1 σε υψηλό επίπεδο, προτείνουν το σύστημα Wi-Via, το οποίο θα μπορούσε να χρησιμοποιηθεί από τους πολίτες στέλνοντας στο διακομιστή της εφαρμογής μηνύματα κειμένου, εικόνες ή video σε πραγματικό χρόνο που αφορούν κάποια γεγονότα. Αυτή η πληροφορία θα ήταν προσβάσιμη από το κοινό μέσω διαδικτυακής διεπαφής που χρησιμοποιεί ηλεκτρονικούς χάρτες της μορφής της Εικόνας 2.3. Αυτές οι πληροφορίες θα μπορούσαν να είναι πολύ χρήσιμες από το νόμο, εάν αφορούν κάποιο ύποπτο άτομο, από υπηρεσίες επειγόντων περιστατικών Amber Alert, αν πχ αφορούν κάποιο εξαφανισμένο άτομο ή από την υπηρεσία πρώτων βοηθειών, αν αφορούν κάποιο ατύχημα και πολλά άλλα.



■ Figure 6. Two screen shots of the Wi-Via Web interface.

Εικόνα 2.3: Διαδικτυακή διεπαφή συστήματος Wi-Via

Οι crowdsourcing εφαρμογές, που αναφέρονται στο άρθρο [6] χωρίζουν τους χρήστες σε δύο μεγάλες κατηγορίες, τους *seekers* που είναι οι χρήστες που ορίζουν έργα προς διεκπεραίωση με ή χωρίς αμοιβή και τους *solvers* που είναι οι χρήστες που αναλαμβάνουν να διεκπεραιώσουν τα έργα. Μία τέτοια εφαρμογή για να είναι επιτυχημένη χρειάζεται να δίνει κίνητρο (συνήθως οικονομικό) στους χρήστες που την χρησιμοποιούν και το πλήθος των συμμετοχών να είναι μεγάλο, διότι οι εργασίες δεν ορίζονται από το σύστημα αλλά από τους ίδιους τους χρήστες. Οι συγγραφείς προτείνουν μία πλατφόρμα η οποία θα υποστηρίζει και συσκευές κινητής τηλεφωνίας (mobile platforms) και οι εργασίες θα έχουν χωροχρονικούς περιορισμούς, εισάγοντας την έννοια των location – based crowdsourcing εφαρμογών σε smartphones. Οπότε, η ανάθεση εργασιών θα γίνεται με βάση τη γεωγραφική θέση του χρήστη, δηλαδή θα πρέπει ο *solver* να βρίσκεται σε κοντινή περιοχή με τον *seeker*.

2.2 Τεχνικές Drive Testing από location – based crowdsourcing εφαρμογές

Στη σημερινή εποχή πλέον, όλες οι έξυπνες κινητές συσκευές διαθέτουν αισθητήρες wi-fi, GPS και έχουν πρόσβαση στο διαδίκτυο, με αποτέλεσμα οι location-based crowdsourcing εφαρμογές να είναι αρκετά διαδεδομένες. Σύμφωνα με το άρθρο [3], οι crowdsourcing εφαρμογές σε περιβάλλον κινητών συσκευών παρουσιάζουν σήμερα σημαντικά πλεονεκτήματα, όπως α) η μεταφερσιμότητα, β) η πολύ συχνή χρήση τους και το πλήθος των χρηστών που είναι σχεδόν καθολικό, γ) η χρήση του GPS για προσδιορισμό της θέσης του χρήστη σε πραγματικό χρόνο, καθώς και δ) η ευκολία εγκατάστασης του απαραίτητου λογισμικού. Όπως αναφέραμε και στην Εισαγωγή, η μέτρηση της ποιότητας αλλά και η παρακολούθηση του δικτύου κινητής τηλεφωνίας αποτελεί ένα μέτρο για την ικανοποίηση του χρήστη κατά την χρησιμοποίηση τέτοιων εφαρμογών. Οι **Fedor Chernogorov, Jani Puttonen** [10] αναφέρονται στην τάση για ελαχιστοποίηση τέτοιου είδους μετρήσεων (Minimization of Drive Test - MDT) από τους παρόχους τηλεπικοινωνιών, η οποία αποτυπώνεται και στο αντίστοιχο πρότυπο τηλεπικοινωνιών του οργανισμού 3GPP [11] (3rd Generation Partnership Project). Αυτό συμβαίνει διότι το κόστος για την υλοποίηση και τη συντήρηση του εξοπλισμού που απαιτείται είναι μεγάλο για τους παρόχους κινητής τηλεφωνίας. Οι μετρήσεις αυτές πλέον, μπορούν να γίνονται από τους ίδιους τους χρήστες μέσω των έξυπνων κινητών τηλεφώνων, με το οικονομικό όφελος για τους παρόχους να είναι μεγάλο.

Σύμφωνα με τα παραπάνω, δημιουργούνται οι προϋποθέσεις για την ανάπτυξη location – based crowdsourcing εφαρμογών σε έξυπνες κινητές συσκευές που θα έχουν ως σκοπό τη μέτρηση της απόδοσης δικτύου κινητής τηλεφωνίας όσον αφορά την πρόσβαση στο διαδίκτυο, αλλά και wi-fi δικτύου. Στο άρθρο [10] προτείνεται ένα πλαίσιο εξόρυξης δεδομένων, το οποίο χρησιμοποιεί τις μετρικές απόδοσης δικτύου (KPIs¹) για να εξάγει συμπεράσματα για την ποιότητα του δικτύου σε σχέση με τις υπηρεσίες που υποστηρίζει. Αν για παράδειγμα μας ενδιαφέρει η υπηρεσία VoIP, τότε η όποια καθυστέρηση είναι ενοχλητική στο χρήστη. Αντίθετα εάν μας ενδιαφέρει η μεταφορά αρχείων, μία μικρή καθυστέρηση στο δίκτυο δεν είναι ενοχλητική. Οπότε ο αλγόριθμος κατηγοριοποίησης που χρησιμοποιείται, εστιάζει στη συνιστώσα QoS (ποιότητα υπηρεσίας), η οποία επιτρέπει να γίνεται διάκριση μεταξύ ικανοποιημένων και δυσαρεστημένων χρηστών στο LTE² [12] δίκτυο κινητής τηλεφωνίας. Σκοπός του αλγορίθμου είναι η κατηγοριοποίηση των χρηστών, αξιοποιώντας συγκεκριμένες μετρικές απόδοσης δικτύου (ανάλογα με την υπηρεσία που επιθυμούμε να εξετάσουμε). Ένα δείγμα τέτοιων μετρικών

¹ Key Performance Indicators

² [http://en.wikipedia.org/wiki/LTE_\(telecommunication\)](http://en.wikipedia.org/wiki/LTE_(telecommunication))

φαίνονται στην Εικόνα 2.4. Η πιο σημαντική μετρική, σύμφωνα με τους συγγραφείς, θεωρείται το throughput, δηλ. το ποσοστό των επιτυχών μηνυμάτων που λαμβάνονται μέσα από το κανάλι επικοινωνίας του δικτύου³. Επίσης, δίνεται έμφαση στη μείωση των μετρικών απόδοσης δικτύου χωρίς να επηρεάζεται η ακρίβεια των αποτελεσμάτων.

CONTENTS OF PERIODIC QoS MDT LOG

Variable	Unit
Group 1. Data for User Satisfaction Classification	
Wideband CQI	dB
Serving RSRP	dBm
Serving RSRQ	dB
RB load	percentage [%]
Allocation size	RBs (Resource Blocks)
Number of allocated bits	bits
Number of allocation attempts	[-]
Number of handovers	[-]
Number of connection failures	[-]
Packet discard rate	percentage [%]
Packet flush rate	percentage [%]
Packet drop rate	percentage [%]
Group 2. Data for Classification Performance Evaluation	
User throughput	kbps
Target CBR level	kbps
Group 3. Additional data	
UE ID	[-]
UE x-coordinate	meters
UE y-coordinate	meters

Εικόνα 2.4: Δείγμα μετρικών απόδοσης δικτύου

Τελικά, ο συνδυασμός των μετρικών με τις λιγότερες παραμέτρους και τα πιο ακριβή αποτελέσματα είναι ο ενδεδειγμένος. Στην Εικόνα 2.5 αποτυπώνεται το πλαίσιο πάνω στο οποίο εφαρμόζεται ο αλγόριθμος κατηγοριοποίησης K-πλησιέστερων γειτόνων⁴ [13] για τη συνιστώσα QoS του MDT. Πρόκειται για μία πολύ ενδιαφέρουσα προσέγγιση για τον χαρακτηρισμό της ποιότητας του δικτύου και μπορεί να χρησιμοποιηθεί από location – based crowdsourcing

³ <http://en.wikipedia.org/wiki/Throughput>

⁴ http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

εφαρμογές έξυπνων κινητών συσκευών, που έχουν ως σκοπό την πραγματοποίηση μετρήσεων από τους χρήστες για τη σύνδεση στο διαδίκτυο.

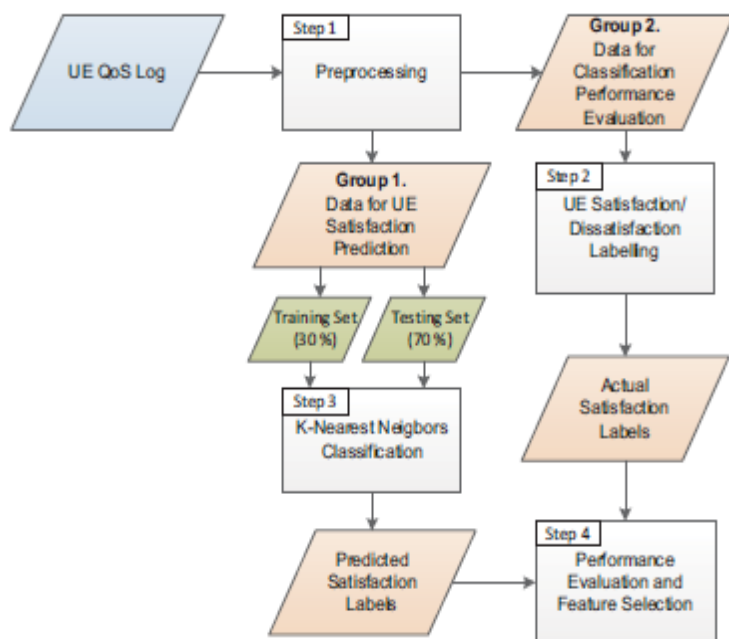


Fig. 1. Classification framework for QoS MDT

Εικόνα 2.5: Πλαίσιο εφαρμογής αλγόριθμου κατηγοριοποίησης για QoS MDT

Στο άρθρο [3], οι συγγραφείς αναλύουν τα πλεονεκτήματα και τα μειονεκτήματα μίας πλατφόρμας crowdsourcing υλοποιημένη σε έξυπνες συσκευές κινητής τηλεφωνίας που έχει ως σκοπό την ανίχνευση και τη μέτρηση δικτύων κινητής τηλεφωνίας και wi-fi. Τα πλεονεκτήματα είναι τα εξής:

- Γίνεται ανίχνευση της τοπολογίας του δικτύου και της απόδοσής του, ώστε να πραγματοποιηθούν οι απαιτούμενες εργασίες βελτιστοποίησής του, από τις εταιρείες κινητής τηλεφωνίας.
- Οι τεχνικές crowdsourcing (τα πλήθη των χρηστών πραγματοποιούν τις μετρήσεις ταυτόχρονα σε κάθε γεωγραφική θέση) έχουν τη δυνατότητα να διαιρούν το έργο σε επιμέρους μικρές διεργασίες που εκτελούνται παράλληλα, με αποτέλεσμα τη πραγματοποίηση του έργου σε πολύ σύντομα χρονικά διαστήματα.
- Δεν απαιτείται για τις μετρήσεις στο δίκτυο κάποιος εξοπλισμός, αφού εκτελούνται στο υλικό και λογισμικό των χρηστών.

Από την άλλη πλευρά, ως μειονέκτημα θεωρείται το κίνητρο που πρέπει να δοθεί στους χρήστες για να χρησιμοποιήσουν την πλατφόρμα, ενώ ως περιορισμός θεωρείται οι πεπερασμένοι πόροι μίας κινητής συσκευής, αφού διαθέτει μπαταρία και το λογισμικό που υλοποιείται θα πρέπει να το λάβει υπόψη του για να μην την εξαντλεί. Επίσης, επειδή οι χρήστες πρέπει να είναι πολλοί, το σύστημα πρέπει να είναι προετοιμασμένο για να εξυπηρετεί μεγάλο αριθμό μετρήσεων, ενώ θα πρέπει να είναι ικανό να αναγνωρίζει κακόβουλες συμπεριφορές ή λανθασμένες μετρήσεις (outliers).

Το 2014 οι **Adriano Faggiani, Enrico Gregori, Luciano Lenzini, Valerio Luconi, and Alessio Vecchio** [3] δημιούργησαν μία τέτοια πλατφόρμα και την ονόμασαν Portolan (Εικόνα 2.6). Σκοπός της εφαρμογής είναι να εκμεταλευτεί τη δυναμική του crowdsourcing και να δημιουργήσει έναν λεπτομερή χάρτη απεικόνισης της κάλυψης σήματος κινητής τηλεφωνίας και Internet, χρησιμοποιώντας τους χάρτες της Google⁵ [14] και τις μετρήσεις των χρηστών.

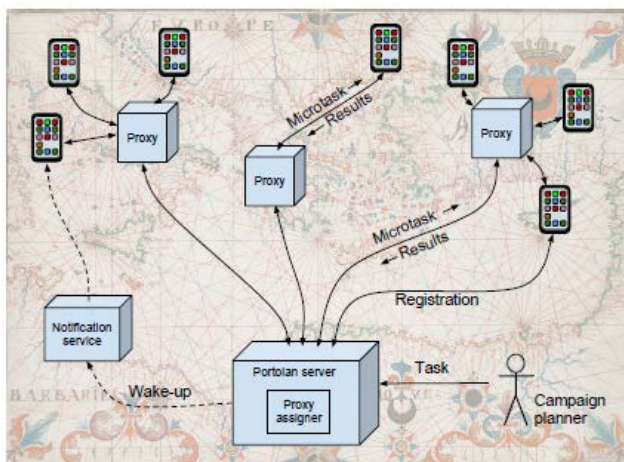


Fig. 1: The Portolan system

Εικόνα 2.6: Αρχιτεκτονική του συστήματος Portolan

Στην Εικόνα 2.6 απεικονίζεται η αρχιτεκτονική της πλατφόρμας Portolan. Αποτελείται κυρίως από τρία μέρη:

Portolan Server

Χρησιμοποιείται για το συντονισμό των χρηστών και την αποθήκευση των μετρήσεων του δικτύου από τους χρήστες. Εδώ συνδέεται ο διαχειριστής του συστήματος (Campaign planner), ο οποίος ορίζει τις εργασίες που θα πρέπει να εκτελέσουν οι χρήστες, δηλ. δημιουργεί καμπάνιες

⁵ <https://www.google.gr/maps>

με συγκεκριμένα χαρακτηριστικά μετρήσεων και χωροχρονικούς περιορισμούς. Οι χρήστες ειδοποιούνται αυτόματα για την καταχώρηση νέας καμπάνιας, μέσω της υπηρεσίας Google Cloud Messaging [15] (GCM) που υποστηρίζεται.

Proxies

Είναι περιφερειακές μονάδες που ελέγχουν τοπικά τις κινητές συσκευές μίας περιοχής με βάση τη γεωγραφική τους θέση. Οι καμπάνιες χωρίζονται σε μικρές δραστηριότητες που αποστέλλονται στους proxies για να ανατεθούν στη συνέχεια στις συσκευές.

Mobile Devices

Είναι οι έξυπνες κινητές συσκευές των χρηστών της εφαρμογής που κάνουν τις μετρήσεις. Η ανάθεση των συσκευών στον αρμόδιο proxy γίνεται από την κεντρική μονάδα Proxy Assigner του portolan server. Οι συσκευές με το που εγκαθιστούν την εφαρμογή, γίνονται register στην GCM υπηρεσία της Google για να ενημερώνονται για τις νέες καμπάνιες που στέλνει ο Portolan server.

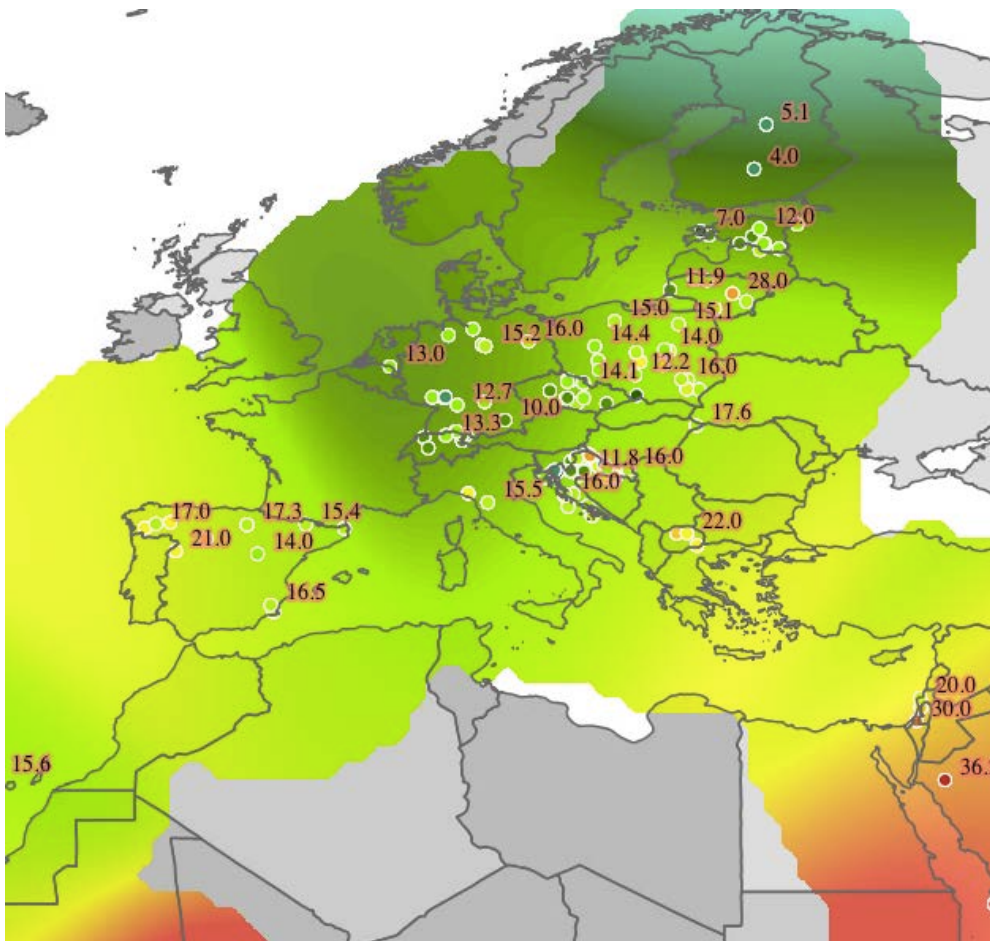
Κάποιες παρατηρήσεις που έχουμε να κάνουμε, όσον αφορά την παραπάνω εφαρμογή, είναι ότι οι μετρήσεις που γίνονται από τους χρήστες μπορεί να είναι: α) ενεργές, απαιτούν κόστος επικοινωνίας – bandwidth (πχ. traceroute, throughput), β) παθητικές, με μηδαμινό κόστος επικοινωνίας, όπως πχ η ανίχνευση του σήματος για το χαρακτηρισμό του τύπου του δικτύου (πχ. Wi-Fi, GPRS, GSM, LTE). Επίσης, η λογική αυτής της crowdsourcing εφαρμογής (Portolan) βλέπουμε ότι διαφέρει από αυτές που περιγράφονται στο άρθρο [6], στο γεγονός ότι οι εργασίες ορίζονται από το διαχειριστή του συστήματος κι όχι από τους ίδιους τους χρήστες (*seekers*).



Fig. 2: Signal coverage map (Urban map ©Google. Google and the Google logo are registered trademarks of Google Inc., used with permission).

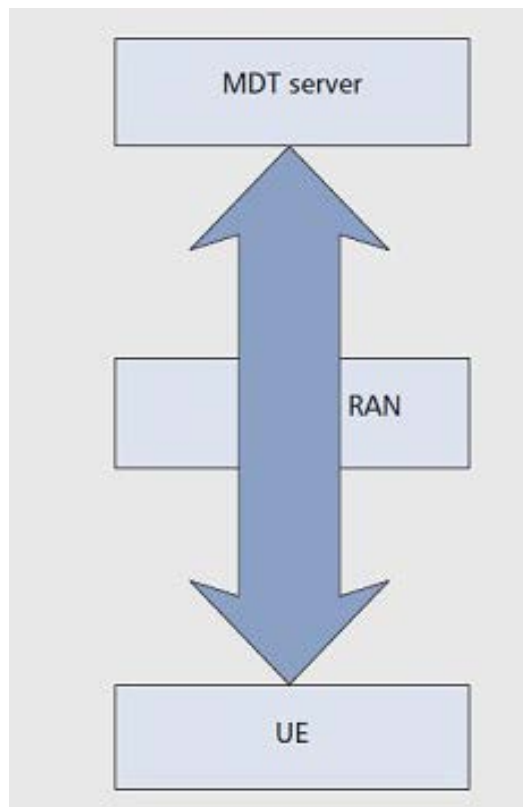
Εικόνα 2.7: Χάρτης μετρήσεων σήματος κάλυψης δικτύου (Πίζα Ιταλίας)

Στην Εικόνα 2.7 αποτυπώνεται ο χάρτης μετρήσεων του σήματος κάλυψης δικτύου στην περιοχή της Πίζας της Ιταλίας, όπου έγιναν οι περισσότερες μετρήσεις. Σύμφωνα με τους συγγραφείς, άποψη που μας βρίσκει και εμάς σύμφωνους, αυτός ο χάρτης είναι πολύ χρήσιμος για τους παρόχους του δικτύου αλλά και για τους ίδιους τους χρήστες. Οι πάροχοι λαμβάνουν πληροφορίες για τις ατέλειες του δικτύου ώστε να προχωρήσουν στη βελτιστοποίησή του, ενώ οι χρήστες πληροφορούνται για την ποιότητα του δικτύου σε περιοχές που τους ενδιαφέρει ώστε να επιλέξουν τον κατάλληλο πάροχο. Οι μετρήσεις στο χάρτη απεικονίζονται ως κουκίδες χρωματιστές, με το κάθε χρώμα να δηλώνει κι ένα εύρος τιμών, χωρίς να έχει γίνει καμία περαιτέρω επεξεργασία. Τα αποτελέσματα των μετρήσεων θα μπορούσαν να έχουν υποστεί κάποιου είδους επεξεργασία, όπως πχ σύμφωνα με τον αλγόριθμο Barnes Surface [16], που χρησιμοποιείται συνήθως από μετεωρολογικούς σταθμούς, όπως φαίνεται στην Εικόνα 2.8. Ο αλγόριθμος αυτός χρησιμοποιώντας έναν μετασχηματισμό – τεχνική σύντηξης δεδομένων, υπολογίζει τις τιμές των μετρήσεων στην παρεμβλλόμενη επιφάνεια μεταξύ των πραγματικών μετρήσεων, ώστε η κάλυψη της επιφάνειας να είναι συνεχής κι όχι διακεκομμένη.



Εικόνα 2.8: Απεικόνιση μετρήσεων με εφαρμογή του αλγορίθμου Barnes Surface

Το 2014 επίσης, ο φοιτητής Κορωνιώτης Νικόλαος του Τμήματος Πληροφορικής και Τηλεματικής του Χαροκόπειου Πανεπιστημίου, στα πλαίσια εκπόνησης της πτυχιακής του εργασίας, δημιούργησε μία εφαρμογή σε περιβάλλον Android με σκοπό τη μέτρηση της απόδοσης ασύρματων δικτύων [1]. Στα πλαίσια της τάσης που υπάρχει για ελαχιστοποίηση των Drive Test, όπως αναφέραμε και παραπάνω, δημιουργήθηκε η συγκεκριμένη location – based crowdsourcing εφαρμογή, όπου πλέον ο κάθε χρήστης, μέσω της κινητής συσκευής του, θα μπορεί να πραγματοποιεί μέτρηση 4 μετρικών απόδοσης ασύρματου δικτύου (downlink, uplink, ping to personal Server, ping to Google Server). Όπως αναφέρεται στο [1], η τεχνική των μετρήσεων που ακολουθείται είναι τύπου U-Plane MDT (Εικόνα 2.9). Στη U-plane αρχιτεκτονική η μέτρηση εκτελείται ανάμεσα στον MDT Server και στην συσκευή του χρήστη (UE) με μία άμεση σύνδεση μεταξύ τους, χωρίς την επίγνωση πως εκτελείται δοκιμή του δικτύου από τον πάροχο του δικτύου (RAN). Τα αποτελέσματα μετά την μέτρηση αποστέλλονται στον MDT Server. Η εφαρμογή τέλος, απεικονίζει γεωγραφικά τις μετρήσεις αυτές σε χάρτες της Google.



Εικόνα 2.9: U-Plane Αρχιτεκτονική

3 ΠΡΟΒΛΗΜΑ - ΤΟΠΟΘΕΤΗΣΗ

Στόχος της μεταπτυχιακής εργασίας είναι να δημιουργηθεί μία εφαρμογή για έξυπνες κινητές συσκευές, που θα πραγματοποιεί μετρήσεις που αφορούν την ασύρματη σύνδεση στο διαδίκτυο και θα παρουσιάζει τα αποτελέσματα σε ηλεκτρονικούς χάρτες. Σκοπός είναι η γεωγραφική χαρτογράφηση της πρόσβασης στο διαδίκτυο από συσκευές κινητής τηλεφωνίας και ταμπλέτες και η αξιολόγηση της εμπειρίας του χρήστη. Επίσης, παρουσιάζονται στατιστικά στοιχεία σε γραφήματα φιλικά προς το χρήστη που αφορούν τη σύντηξη των δεδομένων από τις μετρήσεις μας, όπως πχ. μέση τιμή μέτρησης ανά μετρική σε μια συγκεκριμένη περιοχή, για συγκεκριμένο τύπο δικτύου, αλλά και γραφήματα μετρήσεων ανά πάροχο, με σκοπό τη σύγκριση της ποιότητας των επιμέρους δικτύων.

Οι μετρήσεις γίνονται από τις συσκευές των χρηστών της εφαρμογής, σύμφωνα με τις αρχές του MDT, που περιγράφονται στο άρθρο [10]. Η υλοποίηση της εφαρμογής μας βασίζεται στις αρχές της εφαρμογής Portolan, που περιγράφεται στο άρθρο [3], χρησιμοποιώντας όμως την αρχιτεκτονική που περιγράφεται στο άρθρο [6] (Εικόνα 2.2). Υπάρχει η δυνατότητα οι μετρήσεις να γίνονται κατά παραγγελία (on demand) ή στο παρασκήνιο (as a service). Η συχνότητα των μετρήσεων και ο τύπος του δικτύου που λαμβάνονται οι μετρήσεις, καθορίζονται με βάση τις προτιμήσεις του χρήστη. Οι μετρήσεις στο παρασκήνιο γίνονται σε συγκεκριμένες περιοχές με συγκεκριμένα χωροχρονικά κριτήρια (καμπάνιες). Τα αποτελέσματα των μετρήσεων υπόκεινται σε επεξεργασία, σύμφωνα με τον αλγόριθμο σύντηξης δεδομένων Barnes Surface που αναφέραμε παραπάνω (Εικόνα 2.8), ώστε ο χάρτης των μετρήσεων να είναι όσο το δυνατόν περισσότερο φιλικός προς το χρήστη.

Σκοπός μας είναι να εκμεταλλευτούμε την προϋπάρχουσα γνώση στο χώρο των location – based crowdsourcing εφαρμογών, ώστε να δημιουργήσουμε μία εφαρμογή για έξυπνες συσκευές, που συνδυάζει αυτές τις γνώσεις, για την εξαγωγή ενός αποτελέσματος όσο το δυνατόν πιο ολοκληρωμένου. Γι' αυτό το λόγο, η συγκεκριμένη μεταπτυχιακή εργασία χρησιμοποιεί την πρότερη γνώση που έχει εξαχθεί από την πτυχιακή του Κορωνιώτη Νικόλαου [1] και προχωρά ένα βήμα παραπάνω, ενσωματώνοντας νέες λειτουργίες και βελτιστοποιώντας τις ήδη υπάρχουσες.

4 TO ANDROID

4.1 Γενικά

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google [17] και αργότερα από την Open Handset Alliance [18]. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα (smartphones) και οι ταμπλέτες (tablet), με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV⁶), αυτοκίνητα (Android Auto⁷) και ρολόγια χειρός (Android Wear⁸). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ (π.χ. το HP Slate 21) και σε άλλες ηλεκτρονικές συσκευές.

Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί⁹.

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 καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License¹⁰, μιας ελεύθερης άδειας λογισμικού. Το λογότυπο για το λειτουργικό σύστημα Android (Εικόνα 4.1) είναι ένα ρομπότ σε χρώμα πράσινου μήλου και σχεδιάστηκε από τη γραφίστρια Irina Blok. (πηγή [19])

⁶ http://en.wikipedia.org/wiki/Android_TV

⁷ http://en.wikipedia.org/wiki/Android_Auto

⁸ http://en.wikipedia.org/wiki/Android_Auto

⁹ Πηγή: <https://www.abiresearch.com/press/4q-2014-smartphone-os-results-android-smartphone-s/>

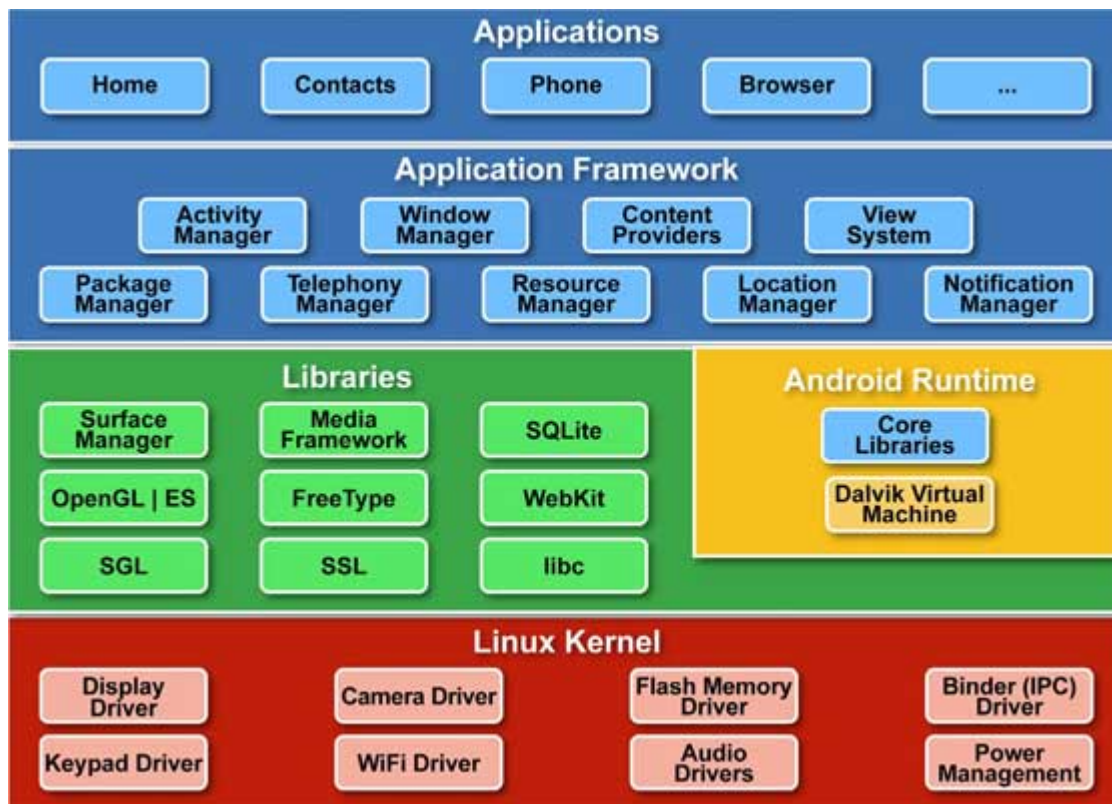
¹⁰ http://en.wikipedia.org/wiki/Apache_License



Εικόνα 4.1: Λογότυπο Android

4.2 Αρχιτεκτονική

Το λειτουργικό σύστημα Android, όπως φαίνεται και στην Εικόνα 4.2, χωρίζεται σε πέντε τομείς (sections) και τέσσερα επίπεδα (layers). (πηγή [20])



Εικόνα 4.2 : Αρχιτεκτονική Android

Στο χαμηλότερο επίπεδο βρίσκεται ο πυρήνας του Linux. Αυτό παρέχει τις βασικές λειτουργίες του συστήματος όπως τη διαχείριση διεργασιών, μνήμης, αλλά και συσκευής, όπως κάμερα, πληκτρολόγιο, οθόνη. Επίσης, ο πυρήνας είναι υπεύθυνος για τη δικτύωση, ενώ παρέχει κι ένα σύνολο από οδηγούς (drivers) για την επικοινωνία με το περιφερειακό υλικό (hardware).

Libraries

Πάνω από τον πυρήνα υπάρχει ένα σύνολο από βιβλιοθήκες, όπως η βιβλιοθήκη WebKit για περιήγηση στο διαδίκτυο (web browsing), η libc, η SQLite βάση δεδομένων για αποθήκευση και διαμοιρασμό δεδομένων, βιβλιοθήκες για αναπαραγωγή και εγγραφή ήχου και εικόνας, βιβλιοθήκες SSL για προστασία στο διαδίκτυο κ.α. (Εικόνα 4.3)



Εικόνα 4.3: Οι βιβλιοθήκες του Android¹¹

Android Runtime

Βρίσκεται στο ίδιο επίπεδο με τις βιβλιοθήκες. Εδώ παρέχεται ένα κύριο συστατικό (component) του Android, η εικονική μηχανή (Virtual machine) Dalvik , που αποτελεί παραλλαγή της Java εικονικής μηχανής, βελτιστοποιημένη για το Android λειτουργικό σύστημα. Η Dalvik VM

¹¹ Πηγή: <http://www.tbray.org/ongoing/When/201x/2010/11/14/-big/Libraries.png.html>

χρησιμοποιεί βασικά χαρακτηριστικά του Linux, όπως διαχείριση μνήμης και multi-threading, τα οποία είναι σημαντικά στοιχεία στην Java. Επίσης, ενεργοποιεί κάθε Android εφαρμογή να εκτελείται σε δική της διεργασία ως στιγμιότυπο της εικονικής μηχανής Dalvik. Τέλος το Android runtime παρέχει ένα σύνολο από βασικές βιβλιοθήκες ώστε οι προγραμματιστές να χρησιμοποιούν ως γλώσσα προγραμματισμού για Android εφαρμογές την Java.

Application Framework

Το συγκεκριμένο επίπεδο παρέχει υπηρεσίες υψηλού επιπέδου σε εφαρμογές, όπως πχ ο Activity Manager που διαχειρίζεται το κύκλο ζωής των εφαρμογών στο λειτουργικό σύστημα. Οι προγραμματιστές εφαρμογών Android έχουν τη δυνατότητα να χρησιμοποιούν αυτές τις υπηρεσίες στις εφαρμογές τους.

Applications

Το επίπεδο εφαρμογών είναι το υψηλότερο επίπεδο στην Αρχιτεκτονική Android. Οι εφαρμογές που είναι προεγκατεστημένες στο λειτουργικό, όπως πχ. επαφές, τηλέφωνο κ.α., όπως και εφαρμογές που εγκαθιστούμε από το Google Play ή από οποιαδήποτε άλλη πηγή, ανήκουν σε αυτό το επίπεδο.

4.3 Εκδόσεις

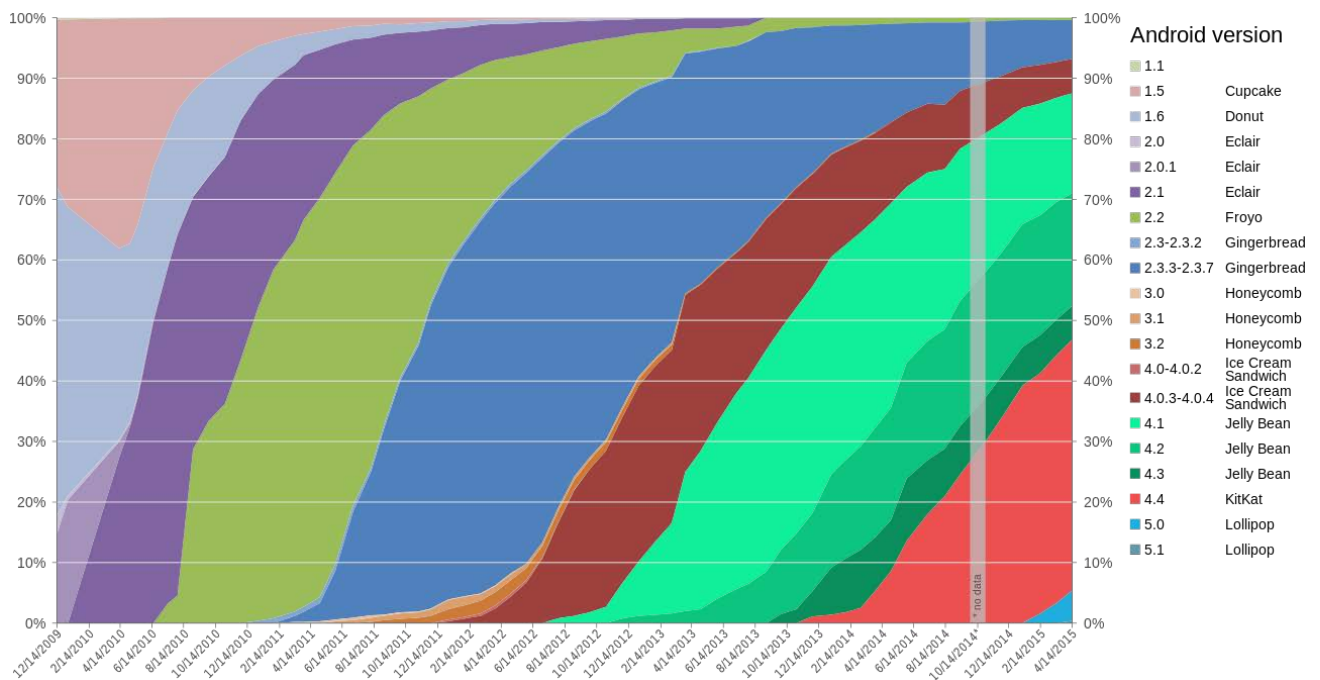
Η πρώτη έκδοση του λειτουργικού Android κυκλοφόρησε σε δοκιμαστική μορφή τον Νοέμβριο του 2007. Η πρώτη επίσημη εμπορική έκδοση (Android 1.0) κυκλοφόρησε τον Σεπτέμβριο του 2008. (Πηγή [21])

Η πιο πρόσφατη Android έκδοση είναι η 5.0 με την κωδική ονομασία "Lollipop", η οποία κυκλοφόρησε στις 3 Νοεμβρίου του 2014. Από τον Απρίλιο του 2009 οι εκδόσεις Android κυκλοφορούν με κωδικές ονομασίες με αλφαβητική σειρά (από έκδοση 1.5 με ονομασία "Cupcake"), όπως φαίνεται και στον Πίνακα 4.1. (Πηγή [21])

Πίνακας 4.1 (Εκδόσεις Android)

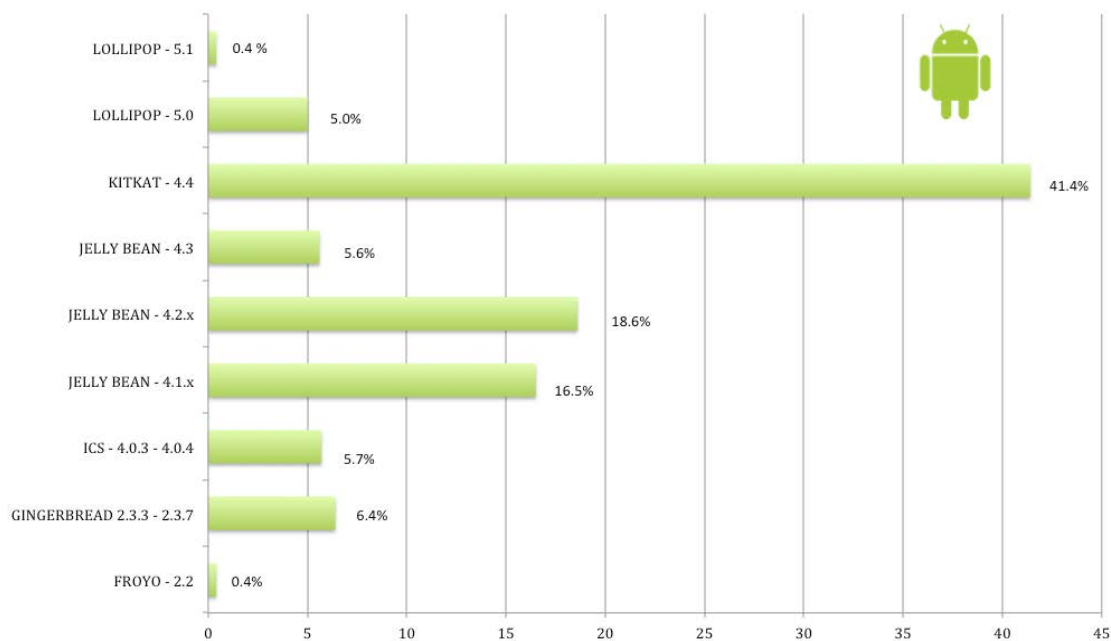
“Alpha” (1.0)	Éclair (2.0 – 2.1)	Ice Cream Sandwich (4.0 – 4.0.4)
“Beta” (1.1)	Froyo (2.2 – 2.2.3)	Jelly Bean (4.1 – 4.3.1)
Cupcake (1.5)	Gingerbread (2.3 – 2.3.7)	KitKat (4.4 – 4.4.4, 4.4W–4.4W.2)
Donut (1.6)	Honeycomb (3.0 – 3.2.6)	Lollipop (5.0 – 5.1.1)

Στην Εικόνα 4.4, βλέπουμε την ιστορική εξέλιξη των εκδόσεων Android στην αγορά. Όπως βλέπουμε και είναι απόλυτα φυσικό, η διανομή παλαιότερων εκδόσεων συνεχώς μειώνεται, ενώ φαίνεται ότι τον Απρίλιο του 2015 η πιο διαδεδομένη έκδοση στην αγορά είναι η KitKat.



Εικόνα 4.4: Ιστορικό εκδόσεων λειτουργικού Android

Η Εικόνα 4.5 μας δείχνει τα ποσοστά χρήσης των Android εκδόσεων τον Απρίλιο του 2015, όπου φαίνεται ότι η πιο δημοφιλής έκδοση είναι η KitKat 4.4 με ποσοστό 41.4%.



Εικόνα 4.5: Χρήση εκδόσεων Android (Απρίλιος 2015)¹²

4.4 Android Components (Συστατικά)

Τα Android Components (συστατικά) είναι Java κλάσεις που χρησιμοποιούνται για να εκτελέσουν συγκεκριμένες λειτουργίες μέσα σε μία Android εφαρμογή. Στη συνέχεια, θα αναλυθούν τα συστατικά αυτά που θα χρησιμοποιήσω για την υλοποίηση της εφαρμογής – πελάτη.

4.4.1 Activities

Το Activity είναι ένα συστατικό (component) μίας Android εφαρμογής. Συγκεκριμένα, αποτελούν τις οθόνες της εφαρμογής με τις οποίες ο χρήστης μπορεί να αλληλεπιδρά με την συσκευή με σκοπό να κάνει κάτι, όπως να πάρει τηλέφωνο, να βγάλει φωτογραφίες, να στείλει email, να δει έναν χάρτη. Κάθε activity δημιουργεί ένα παράθυρο στην εφαρμογή, όπου σχεδιάζεται η διεπαφή του χρήστη. (πηγή [22])

¹² Πηγή: <http://www.plusqa.com/2015/04/android-stats-april-2015/>

Μια εφαρμογή αποτελείται συνήθως από πολλαπλά Activities που είναι χαλαρά συνδεδεμένα μεταξύ τους. Συνήθως, ένα Activity σε μια εφαρμογή ορίζεται ως η «κύρια» δραστηριότητα, η οποία παρουσιάζεται στο χρήστη κατά την εκκίνηση της εφαρμογής. Ο χρήστης της εφαρμογής μπορεί να περιηγηθεί από ένα Activity σε κάποιο άλλο, προκειμένου να εκτελέσει διάφορες ενέργειες, αφού τα Activities επικοινωνούν μεταξύ τους και ανταλλάσσουν δεδομένα. Κάθε φορά που ένα νέο Activity ξεκινά, σταματά το προηγούμενο και το σύστημα το διατηρεί σε μια στοίβα. Η στοίβα λειτουργεί με τη λογική lifo (last in first out), έτσι ώστε όταν ο χρήστης πατήσει το κουμπί “back” της συσκευής το τρέχον Activity να καταστραφεί και το τελευταίο να ανακτηθεί.

Activity Lifecycle (Κύκλος Ζωής Δραστηριότητας)

Τα Activities, όπως όλα τα components του Android, είναι Java κλάσεις και έχουν συγκεκριμένο κύκλο ζωής, που εξαρτάται από τις ενέργειες του χρήστη [23]. Δηλαδή, το στιγμιότυπο του Activity αλλάζει καταστάσεις κατά τη διάρκεια της ζωής του, εκτελώντας συγκεκριμένες μεθόδους - events (συμβάντα), όπως φαίνεται και στην Εικόνα 4.6, με αποτέλεσμα να μπορούμε να ελέγξουμε τη συμπεριφορά του υλοποιώντας κώδικα στις αντίστοιχες μεθόδους [24].

➤ *onCreate(Bundle)*

Καλείται όταν το Activity δημιουργείται για πρώτη φορά. Το Bundle είναι αντικείμενο παραμέτρων που περιέχει όλη την προηγούμενη κατάσταση του Activity. Πάντοτε ακολουθεί η εκτέλεση της μεθόδου onStart().

➤ *onRestart()*

Καλείται όταν το activity έχει σταματήσει και πριν ξεκινήσει ξανά. Πάντοτε ακολουθεί η εκτέλεση της μεθόδου onStart().

➤ *onStart()*

Καλείται όταν το Activity γίνεται ορατό στο χρήστη. Ακολουθεί η εκτέλεση της μεθόδου onResume() εάν η δραστηριότητα έρχεται στο προσκήνιο ή η εκτέλεση της μεθόδου onStop() εάν η δραστηριότητα σταματά να βρίσκεται στο προσκήνιο.

➤ *onResume()*

Εκτελείται όταν το Activity αρχίζει να αλληλεπιδρά με το χρήστη. Σε αυτή τη φάση το Activity βρίσκεται στην κορυφή της στοίβας δραστηριοτήτων. Πάντοτε ακολουθεί η εκτέλεση της μεθόδου onPause().

➤ *onPause()*

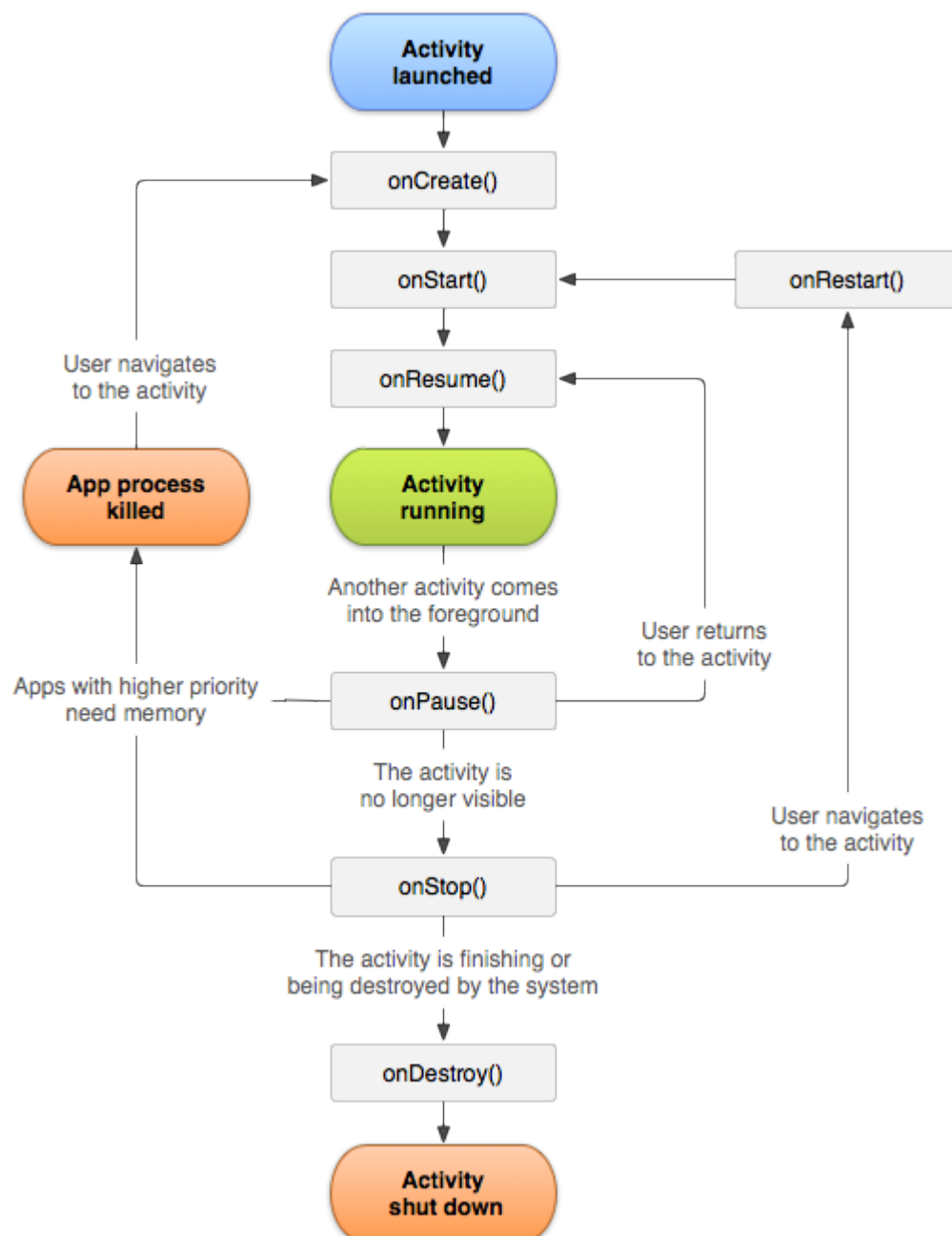
Καλείται όταν το σύστημα ετοιμάζεται να επαναφέρει στο προσκήνιο ένα προηγούμενο Activity. Χρησιμοποιείται για αποθήκευση όλων των αλλαγών σε καθολικά δεδομένα. Ακολουθεί η εκτέλεση της μεθόδου *onResume()* εάν η δραστηριότητα γυρίζει ξανά στο προσκήνιο ή η εκτέλεση της μεθόδου *onStop()* εάν η δραστηριότητα σταματά να βρίσκεται στο προσκήνιο.

➤ *onStop()*

Καλείται όταν το Activity δεν είναι ορατό πλέον στο χρήστη, επειδή κάποιο άλλο έχει έρθει στο προσκήνιο. Ακολουθεί η εκτέλεση της μεθόδου *onRestart()* εάν η δραστηριότητα έρχεται στο προσκήνιο για να αλληλεπιδράσει με το χρήστη ή η εκτέλεση της μεθόδου *onDestroy()* εάν η δραστηριότητα καταστρέφεται.

➤ *onDestroy()*

Καλείται πριν το Activity καταστραφεί από ενέργεια του χρήστη ή και του συστήματος, σε περιπτώσεις που χρειάζεται εξοικονόμηση πόρων.



Εικόνα 4.6: Κύκλος ζωής Android Activity

4.4.2 Broadcast Receivers

Ένας Broadcast Receiver είναι ένα συστατικό (component) του Android, το οποίο εκτελείται σε συγκεκριμένα συμβάντα (events) που προκαλούνται είτε από την εφαρμογή είτε από το σύστημα. Για παράδειγμα, μπορεί να δημιουργηθεί ένας Broadcast Receiver για το συμβάν συστήματος ACTION_BOOT_COMPLETED, ο οποίος ενεργοποιείται κάθε φορά που η συσκευή ολοκληρώνει τη διαδικασία εκκίνησής της.

Για να λειτουργήσει ο receiver, θα πρέπει να έχουμε δηλώσει τον receiver αλλά και το αντίστοιχο event στο αρχείο – μανιφέστο (AndroidManifest.xml) της εφαρμογής. Επίσης, υπάρχει η δυνατότητα να γίνει εγγραφή (register) του receiver δυναμικά, μέσω κώδικα, χρησιμοποιώντας τη μέθοδο `Context.registerReceiver()`. (πηγή [25], [26])

4.4.3 Services

Ένα Service είναι ένα συστατικό (component) του Android, το οποίο εκτελεί μακροχρόνιες εργασίες στο παρασκήνιο και δεν απαιτεί διεπαφή χρήστη (user interface). Τα Services λειτουργούν στο παρασκήνιο ακόμα κι αν η λειτουργία της εφαρμογής σταματήσει. (Πηγή [27])

Υπάρχουν τριών ειδών service:

Service

Αποτελεί τη βασική υλοποίηση του Service component και είναι αυτό που θα χρησιμοποιηθεί και στην Android εφαρμογή – πελάτη για την υλοποίηση της υπηρεσίας παρασκηνίου. Η λειτουργία του ξεκινά με την εκτέλεση της εντολής `Context.startService()` από ένα συστατικό όπως πχ ένα Activity και για να σταματήσει πρέπει να εκτελεστεί η εντολή `stopSelf()` ή `stopService()`. Το μειονέκτημά του είναι ότι λειτουργεί στο κύριο νήμα εκτέλεσης της εφαρμογής (main thread), οπότε εάν θέλουμε να εκτελέσει απαιτητικές σε υπολογιστικούς πόρους εργασίες, πρέπει να το εκτελέσουμε σε ξεχωριστό νήμα (thread). Αυτό το επιτυγχάνουμε με τη χρησιμοποίηση της κλάσης `AsyncTask` [28]. (πηγή [27])

Ο κύκλος ζωής ενός Service υλοποιείται με τις παρακάτω μεθόδους – συμβάντα (events):

➤ *onCreate()*

Είναι η πρώτη μέθοδος που καλείται στο κύκλο ζωής του Service, όταν αυτό δημιουργείται για πρώτη φορά. Οπότε εδώ μπορούμε να υλοποιήσουμε εντολές και συναρτήσεις που θέλουμε να εκτελεστούν την πρώτη φορά που δημιουργείται το service.

➤ *onStartCommand()*

Το σύστημα καλεί αυτή τη μέθοδο όταν εκτελείται η εντολή `startService()`. Η κλήση αυτή της μεθόδου μας δηλώνει ότι το Service λειτουργεί στο παρασκήνιο, οπότε εδώ είναι το μέρος όπου μπορούμε να γράψουμε κώδικα για να ορίσουμε τη λειτουργία του. Για να σταματήσει το service πρέπει να καλέσουμε την εντολή `stopSelf()` ή `stopService()`.

➤ *onDestroy()*

Το σύστημα καλεί αυτή τη μέθοδο όταν το service σταματά να λειτουργεί. Εδώ πρέπει να υλοποιήσουμε κώδικα που να ελευθερώνει δεσμευμένους πόρους από την υλοποίησή μας κατά τη λειτουργία του service.

Intent Service

Το `IntentService` αποτελεί υποκλάση της κλάσης `Service` και ξεκινά και αυτό τη λειτουργία του με την εκτέλεση της εντολής `Context.startService()`. Χρησιμοποιείται για εκτέλεση συγκεκριμένης εργασίας που λειτουργεί στο παρασκήνιο και στη συνέχεια σταματά, δηλαδή δε χρειάζεται η εκτέλεση της εντολής `stopSelf()`.

Ο κύκλος ζωής του, επιπρόσθετα από το βασικό service, περιέχει την υλοποίηση της μεθόδου `onHandleIntent()` που εκτελείται σε ξεχωριστό νήμα (worker thread), με αποτέλεσμα να μην χρειάζεται να υλοποιήσουμε κώδικα γι' αυτό το σκοπό (πχ `AsyncTask`). Όλος ο κώδικας που ορίζει τη λειτουργία του `IntentService` υλοποιείται σε αυτή τη μέθοδο. (πηγή [27])

Bound Service

Το bound service ξεκινά με την εκτέλεση της εντολής `bindService()`. Παρέχει μία διεπαφή τύπου πελάτη – εξυπηρέτη, που επιτρέπει σε συστατικά της εφαρμογής να είναι συνδεδεμένα με το service και να αλληλεπιδρούν μαζί του, στέλνοντας αιτήματα (requests) και λαμβάνοντας τα αποτελέσματα, χρησιμοποιώντας τεχνική ενδοεπικοινωνίας διαδικασιών (IPC). Το bound service λειτουργεί όσο το συστατικό της εφαρμογής που είναι συνδεδεμένο μαζί του λειτουργεί. Μόλις πάψει η λειτουργία του, σταματά να λειτουργεί και το service. (πηγή [27])

Με τα bound service δε θα ασχοληθώ καθόλου στην υλοποίηση της εφαρμογής, διότι χρειαζόμαστε ένα service που να λειτουργεί συνεχώς στο παρασκήνιο, ακόμα και μετά το κλείσιμο της εφαρμογής.

4.4.4 Intents

Το Intent είναι ένα Java αντικείμενο ανταλλαγής μηνυμάτων μεταξύ Android συστατικών (Activities, Broadcast Receivers, Services) με σκοπό την πραγματοποίηση μίας ενέργειας (action) από το συστατικό που δέχεται την κλήση. Υπάρχουν τρεις θεμελιώδεις περιπτώσεις χρήσης: (πηγή [29])

➤ *Εναρξη νέου Activity*

Μπορούμε να δημιουργήσουμε ένα νέο στιγμιότυπο ενός Activity μέσω της μεθόδου `startActivity()`, περνώντας ως παράμετρο ένα Intent. Το Intent περιγράφει ποιο Activity θα ξεκινήσει και περιέχει όλα τα απαραίτητα δεδομένα γι' αυτό.

➤ *Εναρξη νέου Service*

Μπορούμε να ξεκινήσουμε ένα Service μέσω της μεθόδου `startService()`, περνώντας ως παράμετρο ένα Intent. Το Intent περιγράφει το Service που θα ξεκινήσει και περιέχει όλα τα απαραίτητα δεδομένα γι' αυτό.

➤ *Παράδοση ενός Broadcast*

Τα Broadcast είναι μηνύματα που λαμβάνουν οι εφαρμογές και συγκεκριμένα οι receivers, που αναλύσαμε προηγουμένως. Τα Broadcast μηνύματα τα στέλνουμε με τις μεθόδους `sendBroadcast()`, `sendOrderedBroadcast()`, `sendStickyBroadcast()` με παράμετρο το Intent που περιγράφει την ενέργεια.

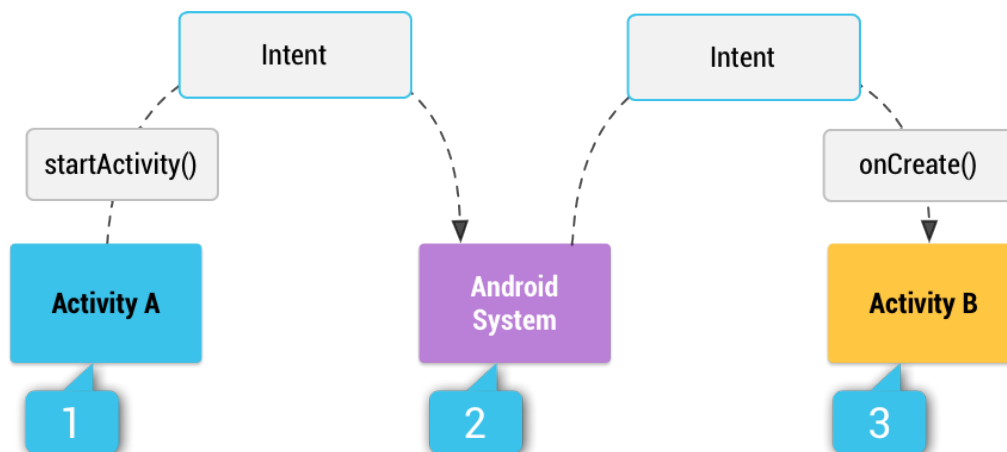
Τα Intents χωρίζονται σε δύο κατηγορίες: (πηγή [29])

➤ *Explicit Intents*

Καθορίζουν το συστατικό (Android component) που θα ξεκινήσει δηλώνοντας το όνομά του. Κυρίως χρησιμοποιούνται όταν θέλουμε να ξεκινήσουμε μία δραστηριότητα (Activity) στην ίδια εφαρμογή, αφού γνωρίζουμε το όνομα της κλάσης.

➤ *Implicit Intents*

Δεν καθορίζουν το όνομα του συστατικού που θα ξεκινήσει, αλλά δηλώνουν την ενέργεια που θέλουν να πραγματοποιηθεί. Το σύστημα του Android είναι αυτό που αποφασίζει ποιο συστατικό θα ξεκινήσει, γνωρίζοντας, μέσω των Intent φίλτρων, τις ενέργειες που υποστηρίζει (Εικόνα 4.7). Τα Implicit Intents χρησιμοποιούνται κυρίως όταν δεν ξέρουμε το όνομα του συστατικού που θέλουμε να πραγματοποιήσει την ενέργεια, ή όταν θέλουμε η συγκεκριμένη ενέργεια να πραγματοποιηθεί από οποιοδήποτε συστατικό οποιασδήποτε εφαρμογής υποστηρίζει την ενέργεια.



Εικόνα 4.7: Έναρξη νέου Activity μέσω Implicit Intent

4.5 Alarm Manager

Ο Alarm Manager διαχειρίζεται τα Alarms, τα οποία δίνουν τη δυνατότητα σε εφαρμογές Android να προχωρούν σε λειτουργίες που έχουν σχέση με το χρόνο, οι οποίες μπορεί να πραγματοποιούνται και εκτός λειτουργίας της εφαρμογής. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε τον Alarm Manager για να ξεκινήσουμε μία διαδικασία η οποία να επαναλαμβάνεται κάθε μέρα. (πηγή [30])

Τα χαρακτηριστικά τους είναι τα εξής:

- Θέτουν σε λειτουργία Intents σε καθορισμένη χρονική στιγμή ή περιοδικά
- Χρησιμοποιούνται σε συνδυασμό με Broadcast receivers για να ξεκινήσουν services και να πραγματοποιήσουν άλλες λειτουργίες.
- Δυνατότητα λειτουργίας εκτός εφαρμογής, οπότε μπορεί να χρησιμοποιηθεί για να ανιχνεύσει ενέργειες και συμβάντα, ακόμα και όταν η συσκευή βρίσκεται σε κατάσταση «ύπνου» (sleep mode).
- Προσφέρουν εξοικονόμηση πόρων, από τη στιγμή που μας δίνουν τη δυνατότητα να προγραμματίζουμε ενέργειες και διαδικασίες, διαφορετικά θα ήμασταν αναγκασμένοι να τρέχουμε συνεχώς στο παρασκήνιο services που να πραγματοποιούν τις εργασίες.

Τα Alarms διακρίνονται στους παρακάτω τύπους:

- *ELAPSED_REAL_TIME*
Εκτελεί την ενέργεια με βάση το χρόνο που έχει παρέλθει από τη στιγμή που η συσκευή τέθηκε σε λειτουργία, αλλά δεν εκτελείται εάν η συσκευή βρίσκεται σε sleep mode.
- *ELAPSED_REAL_TIME_WAKEUP*
Η διαφορά του από την προηγούμενη περίπτωση είναι ότι αν η συσκευή βρίσκεται σε sleep mode, την θέτει σε λειτουργία και εκτελεί την ενέργεια.
- *RTC*
Εκτελεί την ενέργεια σε καθορισμένη χρονική στιγμή, αλλά δεν εκτελείται εάν η συσκευή βρίσκεται σε sleep mode.
- *RTC_WAKEUP*
“Ξυπνά” την συσκευή και εκτελεί την ενέργεια σε καθορισμένη χρονική στιγμή.

Ο Alarm Manager θα χρησιμοποιηθεί στην εφαρμογή – πελάτης για να ξεκινά το service που θα πραγματοποιεί τις μετρήσεις, με βάση τις προτιμήσεις του χρήστη.

4.6 Location API

Ένα από τα μοναδικά χαρακτηριστικά των εφαρμογών σε έξυπνες συσκευές είναι η δυνατότητα επίγνωσης θέσης (location awareness). Ο κύριος λόγος που αυτή η δυνατότητα είναι πολύ χρήσιμη, έγκειται στο γεγονός ότι οι χρήστες τέτοιων συσκευών έχουν μαζί τους συνέχεια αυτές τις συσκευές, με αποτέλεσμα η δυνατότητα επίγνωσης θέσης σε εφαρμογές να προσφέρει ολοκληρωμένα αποτελέσματα. (πηγή [31])

Η δυνατότητα αυτή επιτυγχάνεται σε Android εφαρμογές μέσω τεχνολογιών που μας παρέχουν αυτές οι συσκευές, όπως το GPS, η μέθοδος τριγωνισμού σε δίκτυα κινητής τηλεφωνίας, αλλά και μέσω wi-fi δικτύων. (πηγή [32])

Το λειτουργικό Android μας παρέχει δύο διαφορετικά API για να επιτύχουμε επίγνωση θέσης. Το **android.location** πακέτο, που είναι και το παλαιότερο και το **Google Play services location** API, που προωθείται από την Google και απαιτεί την εγκατάσταση της υπηρεσίας Google Play.

Για τις μετρήσεις της υπηρεσίας παρασκηνίου, στην εφαρμογή που δημιουργήθηκε, χρησιμοποιήθηκε η διεπαφή (interface) **FusedLocationProviderApi**, που παρέχεται στο **Google Play services location** API, λόγω του γεγονότος ότι προτείνεται από την Google, ενώ στην υλοποίηση της OnDemand μέτρησης, που βασίστηκε στην παλαιότερη υλοποίηση του Κορωνιώτη Νικόλαου, χρησιμοποιήθηκε ο **LocationManager** από το πακέτο **android.location**.

5 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

Στην Εικόνα 5.1 αποτυπώνεται η αρχιτεκτονική του συστήματος που περιλαμβάνει την εφαρμογή που υλοποιήσαμε. Η πλατφόρμα αποτελείται από δύο μέρη:

- Εφαρμογή – πελάτης (client app)
- Εξυπηρετητές εφαρμογών (app servers)

Η Εφαρμογή – πελάτης αποτελεί την location – based crowdsourcing εφαρμογή που υλοποίησα για συσκευές κινητής τηλεφωνίας και ταμπλέτες σε λειτουργικό σύστημα Android για λόγους που θα αναλύσω στο επόμενο κεφάλαιο. Η εφαρμογή είναι υπεύθυνη για την πραγματοποίηση των μετρήσεων και την αναπαράστασή τους σε ψηφιακούς γεωγραφικούς χάρτες. Οι μετρικές απόδοσης δικτύου για τις οποίες γίνονται οι μετρήσεις είναι η ταχύτητα λήψης δεδομένων (**downlink**), ταχύτητα αποστολής (**uplink**), χρόνος ping στον εξυπηρετή μας στη διεύθυνση 83.212.113.47 (**ping to personal server**) και χρόνος ping σε εξυπηρετή της google (**ping to google server**).

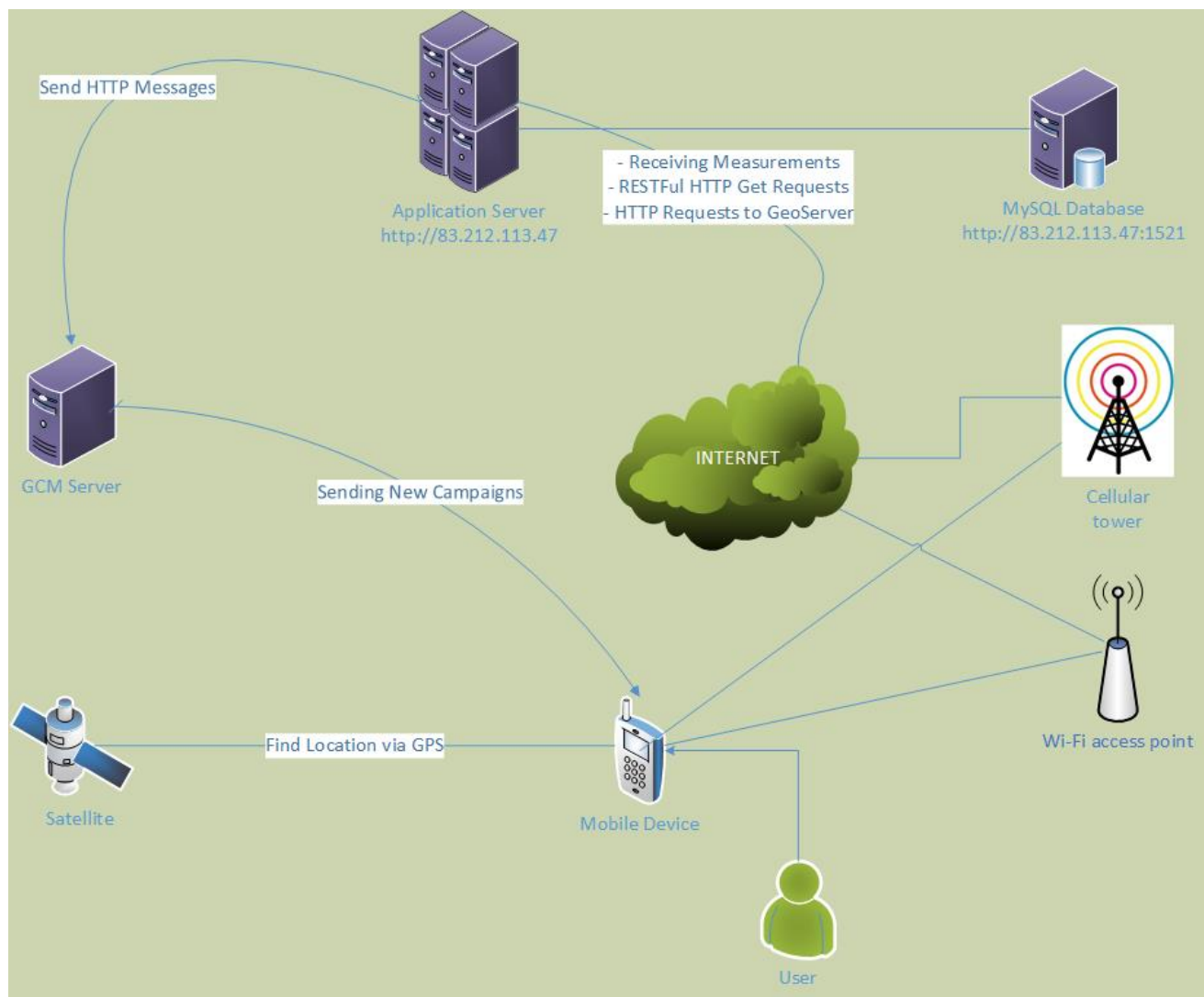
Οι Εξυπηρετητές επικοινωνούν με τις εφαρμογές – πελάτες (client apps) και είναι υπεύθυνοι για τη συλλογή και αποθήκευση των δεδομένων που προέρχονται από τις μετρήσεις των συσκευών. Επίσης, παρέχουν αυτά τα δεδομένα στις κινητές συσκευές για την αποτύπωση των μετρήσεων σε ψηφιακούς χάρτες, αλλά και για την παρουσίαση στατιστικών στοιχείων για τις μετρήσεις. Πιο συγκεκριμένα, οι εξυπηρετητές βρίσκονται σε εικονική μηχανή του «ωκεανού» (okeanos cloud) στη διεύθυνση <http://83.212.113.47>, με λειτουργικό σύστημα Ubuntu 12.04 LTS (Long Term Support). Αναλυτικά περιέχουν:

- Έναν πολυνηματικό εξυπηρετή (multithreaded server), ο οποίος έχει υλοποιηθεί σε Java με τεχνική ServerSocket (ανταλλαγή TCP μηνυμάτων), που χρησιμοποιείται από τις εφαρμογές – πελάτες για την προσομοίωση της μέτρησης των μετρικών απόδοσης ασύρματων δικτύων downlink, uplink και δύο ειδών ping. Η συγκεκριμένη υλοποίηση έχει υιοθετηθεί από την πτυχιακή του Κορωνιώτη Νικόλαου [1], όπου υπάρχουν αναλυτικές οδηγίες.
- Μία βάση δεδομένων MySQL 5.5.41 στην πόρτα 3306, όπου αποθηκεύονται τα δεδομένα των μετρήσεων και οι συσκευές που συμμετέχουν σ'αυτές.
- Ένας Apache web server στην πόρτα 80 με εγκατεστημένη PHP 5.3.10, όπου έχει υλοποιηθεί κώδικας σε PHP, που δέχεται τα δεδομένα των μετρήσεων σε μορφή HTTP POST Request και τα αποθηκεύει στη βάση δεδομένων. Η συγκεκριμένη υλοποίηση έχει

υιοθετηθεί από την πτυχιακή του Κορωνιώτη Νικόλαου [1] με κάποιες μικρές προσθήκες και βελτιώσεις. Επίσης, έχει υλοποιηθεί κώδικας PHP για την αποθήκευση στη βάση δεδομένων των συσκευών Android που εγγράφονται στην υπηρεσία GCM.

- Ένας Apache Tomcat 7.0.26 στην πόρτα 8888, όπου έχει εγκατασταθεί ο Geoserver 2.6.0, ο οποίος χρησιμοποιείται από τις εφαρμογές – πελάτες για την απεικόνιση των μετρήσεων σε γεωγραφικούς χάρτες. Αναλυτικές οδηγίες εγκατάστασης και χρήσης του Geoserver αναφέρονται στην πτυχιακή του Κορωνιώτη Νικόλαου [1].
- Ένας Glassfish server 3.1.2 στην πόρτα 8080, στον οποίο έχω δημιουργήσει ένα NetBeans Java project με RESTFul services, με κύριο σκοπό την παροχή στατιστικών στοιχείων στις εφαρμογές – πελάτες των μετρήσεων των ασύρματων δικτύων.
- GCM application server [33], ο οποίος υλοποιείται σε Glassfish και στέλνει HTTP μηνύματα στις εφαρμογές – πελάτες σε μορφή JSON (JavaScript Object Notation), κάθε φορά που εισάγεται νέα καμπάνια (γεωγραφική περιοχή μετρήσεων) στη βάση. Πρόκειται για υπηρεσία της Google που θα αναλύσουμε στην παράγραφο 6.3.

Ο λόγος που χρησιμοποίησα τις παραπάνω τεχνολογίες για την υλοποίηση των εξυπηρετητών και του λειτουργικού συστήματος, είναι διότι βασίστηκα στην προϋπάρχουσα υλοποίηση του Κορωνιώτη Νικόλαου (εκτός από τα δύο τελευταία bullets). Σκοπός μου, όπως ανέφερα και στην προηγούμενη παράγραφο, ήταν να ασχοληθώ περισσότερο με την επέκταση της εφαρμογής – πελάτη για κινητές συσκευές κι όχι να δημιουργήσω κάτι από την αρχή από τη στιγμή που κάτι τέτοιο είχε ήδη υλοποιηθεί.



Εικόνα 5.1: Αρχιτεκτονική Συστήματος

Όπως φαίνεται και στην Εικόνα 5.1, ο χρήστης χρησιμοποιεί την εφαρμογή – πελάτη μέσω κινητών συσκευών (smartphones ή ταμπλέτες). Η εφαρμογή, μέσω GPS, ανακτά τις συντεταγμένες της γεωγραφικής θέσης του χρήστη. Μπορεί να συνδέεται στο διαδίκτυο μέσω wi-fi ή μέσω του δικτύου κινητής τηλεφωνίας, οπότε μπορεί να επικοινωνεί με τον application server για να πραγματοποιεί τις μετρήσεις των τεσσάρων μετρικών απόδοσης του δικτύου, που αναφέραμε παραπάνω. Μετά την πραγματοποίηση των μετρήσεων, μπορεί να αποστέλλει στον application server τα αποτελέσματα, ώστε να αποθηκεύονται στη βάση δεδομένων MySQL. Η εφαρμογή επίσης, πραγματοποιεί HTTP αιτήματα (requests) στον GeoServer για την γεωγραφική απεικόνιση των μετρήσεων σε χάρτες της google και RESTful HTTP GET αιτήματα στον Glassfish για να ανακτήσει δεδομένα που έχουν αποθηκευτεί στη βάση δεδομένων MySQL. Ο application server, τέλος, κάθε φορά που στη βάση δεδομένων εισάγεται

νέα καμπάνια (θα αναλυθεί στο επόμενο κεφάλαιο), στέλνει στον GCM server, HTTP μηνύματα, που περιέχουν σε μορφή JSON όλες τις ενεργές καμπάνιες (η συγκεκριμένη διαδικασία έχει υλοποιηθεί στη μεταπτυχιακή εργασία του Gerti Nurka [34]). Ο GCM server, στη συνέχεια, στέλνει σε όλες τις κινητές συσκευές που έχουν εγγραφεί στη συγκεκριμένη υπηρεσία (register) τις νέες καμπάνιες, οι οποίες λαμβάνονται στην εφαρμογή – πελάτης της συσκευής από τον GCM client [35], που έχει υλοποιηθεί από εμάς και θα αναλυθεί στο επόμενο κεφάλαιο.

6 ΥΛΟΠΟΙΗΣΗ

6.1 Περιγραφή Εφαρμογής

6.1.1 Προϋπάρχουσα Υλοποίηση

Όπως αναφέραμε και παραπάνω, η εφαρμογή που δημιουργήθηκε χρησιμοποιεί την ίδια μεθοδολογία μετρήσεων με την εφαρμογή της πτυχιακής εργασίας του Κορωνιώτη Νικόλαου. Πιο συγκεκριμένα, χρησιμοποιήθηκε για τις μετρήσεις ο κώδικας της [1], με κάποιες τεχνικές βελτιώσεις. Υιοθετήθηκε στην εφαρμογή – πελάτη η socket client υλοποίηση σε Java (SpeedTester.java, Ping.java) και στη μεριά του διακομιστή η socket server υλοποίηση (Server.class, SubServerThread.class, TestingServer.class) για την προσομοίωση των μετρικών απόδοσης δικτύου και αποστολής ping. Η εγκατάσταση και παραμετροποίηση του GeoServer στον Tomcat, όπως και η δημιουργία των styles απεικόνισης των μετρήσεων (downlink, uplink, ping1, ping2) έγιναν σύμφωνα με τις οδηγίες της εργασίας [1]. Επιπρόσθετα, για την απεικόνιση των μετρήσεων με τεχνικές σύντηξης δεδομένων, σύμφωνα με τον αλγόριθμο Barnes Surface, δημιουργήθηκαν από τον Gerti Nurka (εργασία [34]) νέα styles για κάθε μετρική (DownLinkBarnesSurface, UplinkBarnesSurface, PingOurServerBarnesSurface, PingGoogleBarnesSurface), αφού εγκαταστάθηκε στον GeoServer πρώτα η βιβλιοθήκη WPS¹³. Οι μετρήσεις που πραγματοποιούνται αποθηκεύονται σε υπάρχοντα πίνακα της [1] της βάσης δεδομένων MySQL, στον οποίο έχουμε προσθέσει επιπλέον πεδία που αφορούν το χρήστη, την ακρίβεια μέτρησης, το όνομα (SSID (*service set identifier*) για wi-fi και πάροχο για κινητή τηλεφωνία) και την ταχύτητα του δικτύου.

Η εργασία [1], πραγματοποιεί μετρήσεις επί τόπου κατά παραγγελία του χρήστη. Επειδή αυτή η λειτουργία υπήρχε διαθέσιμη, θεώρησα ότι θα μπορούσε να ενσωματωθεί στη νέα εφαρμογή ως μία επιπλέον δυνατότητα (OnDemandActivity.java).

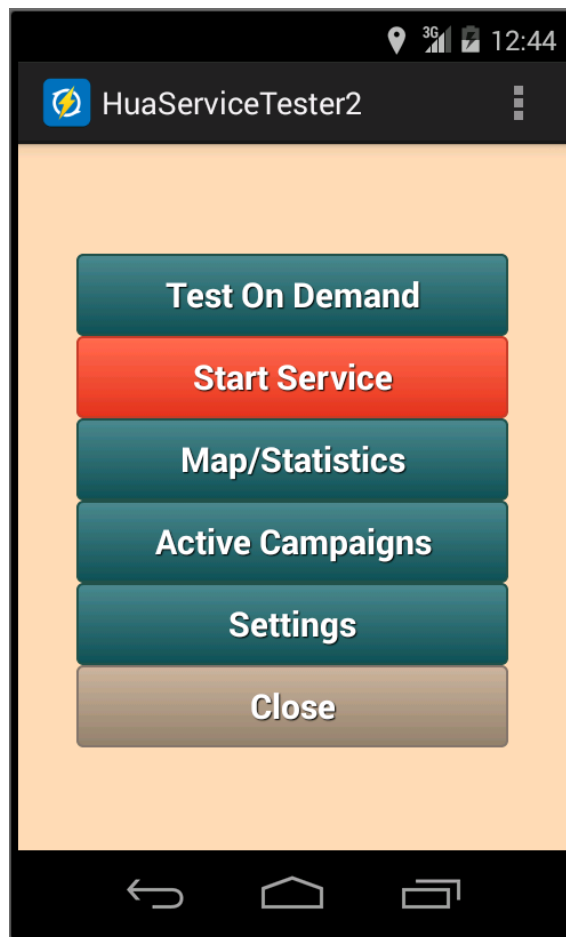
¹³ <http://docs.geoserver.org/stable/en/user/extensions/wps/install.html>

6.1.2 Επέκταση

Η εργασία βασίστηκε στην ανάπτυξη της εφαρμογής – πελάτη, η οποία υλοποιήθηκε σε λειτουργικό σύστημα Android. Ο λόγος που επιλέχθηκε το συγκεκριμένο λειτουργικό σύστημα είναι διότι, σύμφωνα με τη μελέτη που έγινε στην παράγραφο 4.1, το Android είναι το πιο διαδεδομένο λειτουργικό για έξυπνες συσκευές και ταμπλέτες [19].

Η εφαρμογή λειτουργεί σε συσκευές που έχουν εγκαταστημένο Android API επιπέδου 11 και πάνω (έκδοση 3.0 και πάνω) με στόχευση τις συσκευές που «τρέχουν» Android API επιπέδου 19 (έκδοση KitKat 4.4.2). Η απόφαση αυτή βασίστηκε στη μελέτη χρήσης που έγινε στην παράγραφο 4.3 όταν ξεκίνησα την υλοποίηση της εφαρμογής (Δεκέμβριος 2014) βασιζόμενος στο διάγραμμα της Εικόνας 4.3, όπου αναλύεται ιστορικά η εξέλιξη των εκδόσεων Android. Σήμερα (Απρίλιος του 2015) φαίνεται ότι η πρόβλεψή μου ήταν επιτυχής, αφού η πιο διαδεδομένη έκδοση Android είναι η KitKat 4.4 με ποσοστό 41,4% (Εικόνα 4.4). Συνολικά, τον Απρίλιο του 2015, μόλις ένα ποσοστό 6,8% χρησιμοποιεί εκδόσεις προγενέστερες της Honeycomb 3.0 (Android API 11), ποσοστό το οποίο τείνει να ελαχιστοποιηθεί. Επειδή λοιπόν, ήθελα να χρησιμοποιήσω κάποιες καινούργιες βιβλιοθήκες που δεν υποστηρίζονται από παλαιότερες εκδόσεις Android, αποφάσισα να βάλω αυτόν τον περιορισμό.

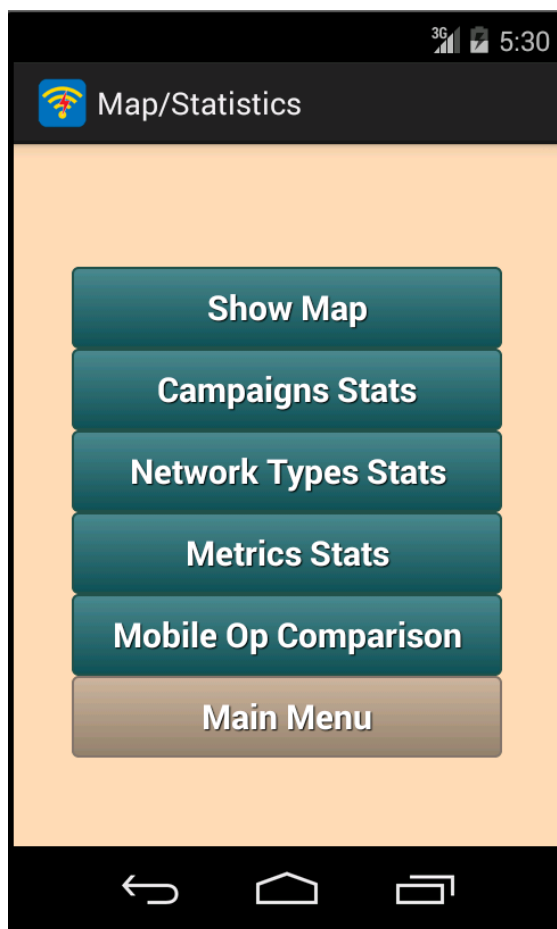
Η αρχική οθόνη της εφαρμογής, όπως φαίνεται στην Εικόνα 6.1, περιέχει το κύριο μενού επιλογών:



Εικόνα 6.1: Αρχική οθόνη εφαρμογής

- *Test On Demand*: Είναι η λειτουργία επιτόπιας μέτρησης που ενσωματώθηκε από την εργασία [1], όπως αναφέραμε παραπάνω.
- *Start Service*: Η βασική λειτουργία της εφαρμογής. Πατώντας ο χρήστης αυτό το κουμπί ξεκινά την υπηρεσία παρασκηνίου, η οποία δουλεύει συνεχώς όσο η συσκευή είναι σε λειτουργία, ακόμα κι αν κλείσουμε την εφαρμογή. Η υπηρεσία παρασκηνίου ανιχνεύει την κατάσταση δικτύου, τις επιλογές του χρήστη (settings) και προχωρά σε περιοδικές μετρήσεις της απόδοσης του δικτύου. Προϋπόθεση για να πραγματοποιηθούν οι μετρήσεις είναι να ικανοποιούνται τα κριτήρια μίας τουλάχιστον ενεργής καμπάνιας και οι επιλογές του χρήστη.
- *Map/Statistics*: Επιλογή που μας οδηγεί σε μία νέα οθόνη με υπομενού, όπως φαίνεται στην Εικόνα 6.2. Οι επιλογές εδώ αφορούν την γραφική απεικόνιση των μετρήσεων σε χάρτες της Google, με διάφορα φίλτρα (θα αναλυθούν σε ειδικό κεφάλαιο), και τις οθόνες στατιστικών στοιχείων όσων αφορά τις καμπάνιες και τις μετρήσεις. Τα

δεδομένα ανακτώνται μέσω RESTful διεπαφής που έχουμε δημιουργήσει στον διακομιστή εφαρμογών (application server) Glassfish.



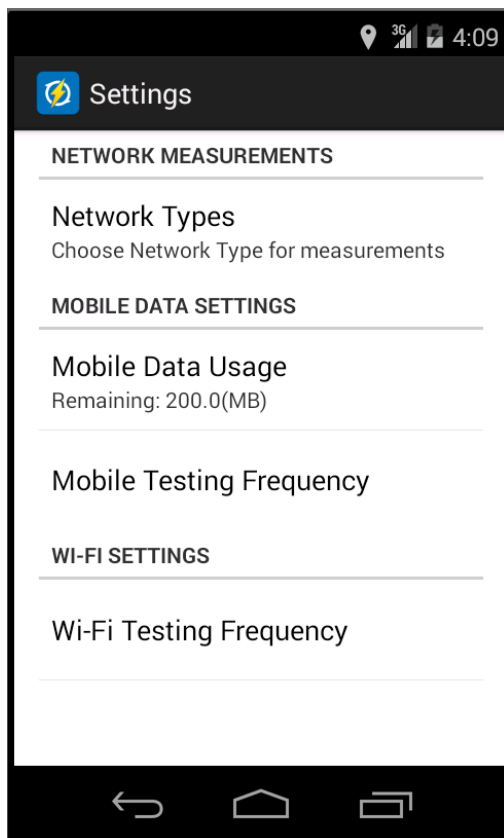
Εικόνα 6.2: Οθόνη υπομενού εφαρμογής

- *Active Campaigns*: Οι ενεργές καμπάνιες είναι γεωγραφικές περιοχές με συγκεκριμένα χαρακτηριστικά. Τις ενεργές καμπάνιες τις ορίζουν οι χρήστες της διαδικτυακής εφαρμογής που έχει δημιουργηθεί από τον Gerti Nurka, στα πλαίσια της μεταπτυχιακής του εργασίας [34]. Η εφαρμογή μας ενημερώνεται με το αρχείο των ενεργών καμπανιών μέσω της υπηρεσίας GCM, που θα αναλύσουμε σε ειδική παράγραφο. Ο ρόλος των καμπανιών είναι να θέτουν χωροχρονικά όρια εντός των οποίων θα γίνονται οι μετρήσεις από την υπηρεσία παρασκήνιου (TestService) ικανοποιώντας ταυτόχρονα συγκεκριμένα χαρακτηριστικά, όπως α) το είδος της σύνδεσης (wi-fi ή mobile), β) το όριο ταχύτητας της συσκευής και γ) το όριο ακρίβειας της θέσης της συσκευής (Εικόνα 6.3).

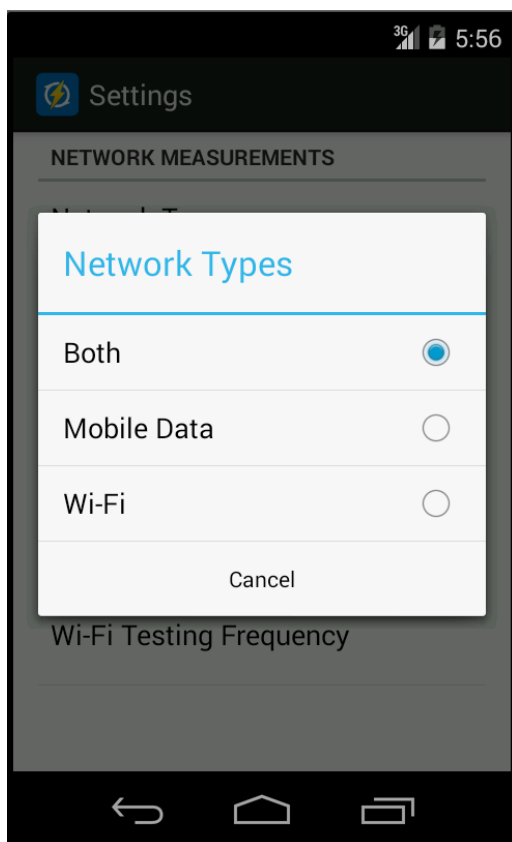


Εικόνα 6.3: Ενεργές Καμπάνιες εφαρμογής

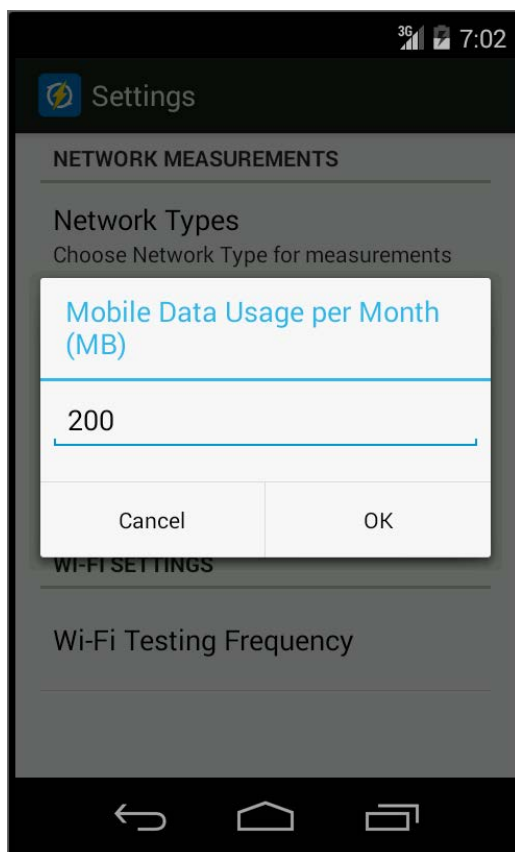
- *Settings*: Οθόνη ρυθμίσεων εφαρμογής (Εικόνα 6.4). Καθορίζονται οι προτιμήσεις του χρήστη σχετικά με την πραγματοποίηση των μετρήσεων από την υπηρεσία παρασκηνίου. Οι διαθέσιμες επιλογές είναι:
 - *Network Types*: Επιλέγεται ο τύπος δικτύου που θέλουμε να πραγματοποιήσουμε μετρήσεις. (Επιλογές: Both, Mobile Data, Wi-Fi), (Εικόνα 6.5)
 - *Mobile Data Usage*: Σε περιπτώσεις μέτρησης σε δίκτυο κινητής τηλεφωνίας, ορίζεται μηνιαίο όριο χρήσης δεδομένων σε MB για να αποφεύγονται ανεπιθύμητες χρεώσεις. (Εικόνα 6.6)
 - *Mobile Testing Frequency*: Καθορίζεται η συχνότητα μέτρησης σε περιπτώσεις μέτρησης σε δίκτυο κινητής τηλεφωνίας. (Επιλογές: 5-10-30 minutes, 1-2-6-12 hours, 1 day)
 - *Wi-Fi Testing Frequency*: Καθορίζεται η συχνότητα μέτρησης σε περιπτώσεις μέτρησης σε δίκτυο wi-fi. (Επιλογές: 5-10-30 minutes, 1-2-6-12 hours, 1 day)
- *Close*: Επιλογή εξόδου από την εφαρμογή.



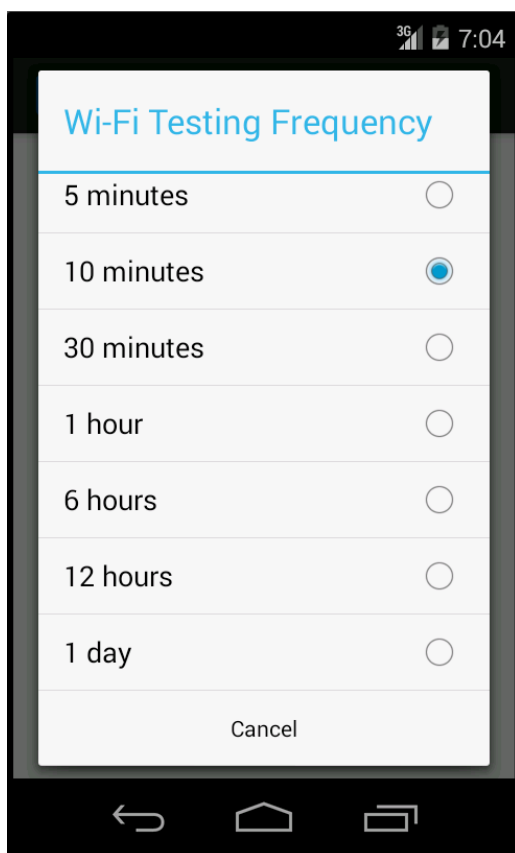
Εικόνα 6.4: Οθόνη Προτιμήσεων χρήστη (Settings)



Εικόνα 6.5: Παράθυρο επιλογής τύπου δικτύου για μετρήσεις



Εικόνα 6.6: όριο χρήσης δεδομένων για mobile data



Εικόνα 6.7: Παράθυρο επιλογής συχνότητας μετρήσεων σε wifi

6.2 Εργαλεία Ανάπτυξης Λογισμικού

6.2.1 Επιλογή IDE (Integrated Development Environment)

Για την επιλογή ανάπτυξης λογισμικού για την εφαρμογή – πελάτης που έχει υλοποιηθεί σε συσκευές με λειτουργικό σύστημα Android, είχα δύο επιλογές. α) Το Eclipse IDE με εγκατεστημένο το ADT (Android Development Toolkit) plugin [36] για ανάπτυξη εφαρμογών σε περιβάλλον Android και β) το Android Studio της JetBrains [37].

Σήμερα, το επίσημα υποστηριζόμενο IDE από την Google είναι το Android Studio. Οπότε εάν ξεκινούσα σήμερα την ανάπτυξη της εφαρμογής, θα επέλεγα ως εργαλείο το συγκεκριμένο. Όμως το Νοέμβριο του 2014 που ξεκίνησα την μεταπτυχιακή μου εργασία, το συγκεκριμένο εργαλείο βρισκόταν σε beta έκδοση. Οπότε, επειδή εκείνη την περίοδο το πιο σταθερό περιβάλλον για ανάπτυξη λογισμικού σε Android ήταν το Eclipse with ADT plugin και επιπλέον επειδή είχα μεγαλύτερη εμπειρία στο συγκεκριμένο εργαλείο επέλεξα το συγκεκριμένο IDE.

Για την ανάπτυξη της υπηρεσίας REST στον διακομιστή εφαρμογών (application server), υπήρχαν αρκετές επιλογές. Καταρχήν, έπρεπε να επιλέξω τη γλώσσα προγραμματισμού που θα χρησιμοποιήσω. Επειδή η πιο διαδεδομένη λύση είναι η JAVA, αλλά και επειδή χρησιμοποιώ ως application server τον Glassfish, επέλεξα τελικά την JAVA. Όσον αφορά το εργαλείο IDE, για λόγους καθαρά πρότερης εμπειρίας, αλλά και επειδή θεωρώ ότι το συγκεκριμένο εργαλείο διακρίνεται για την απλότητά του στη κατασκευή των υπηρεσιών REST, επέλεξα το NetBeans IDE της Oracle.

6.2.2 Eclipse with ADT plugin

Για να εγκαταστήσουμε το εργαλείο πρέπει πρώτα να έχουμε εγκαταστήσει το JDK7+¹⁴ (Java Development Toolkit 7 και πάνω). Αφού μεταφερθούμε στη σελίδα εγκατάστασης (Εικόνα 6.8) επιλέγουμε την έκδοση της Java που θέλουμε να εγκαταστήσουμε για το λειτουργικό μας σύστημα.

¹⁴ <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Java SE 7u79/80

Auto-update Notice & End of Public Updates for Oracle JDK 7

Coincident with the [January 2015 CPU](#) release users with the auto-update feature enabled will be migrated from Oracle JRE 7 to Oracle JRE 8. Also, please note this release will be the last Oracle JDK 7 publicly available update. For more information, and details on how to receive longer term support for Oracle JDK 7, please see the [Oracle Java SE Support Roadmap](#).

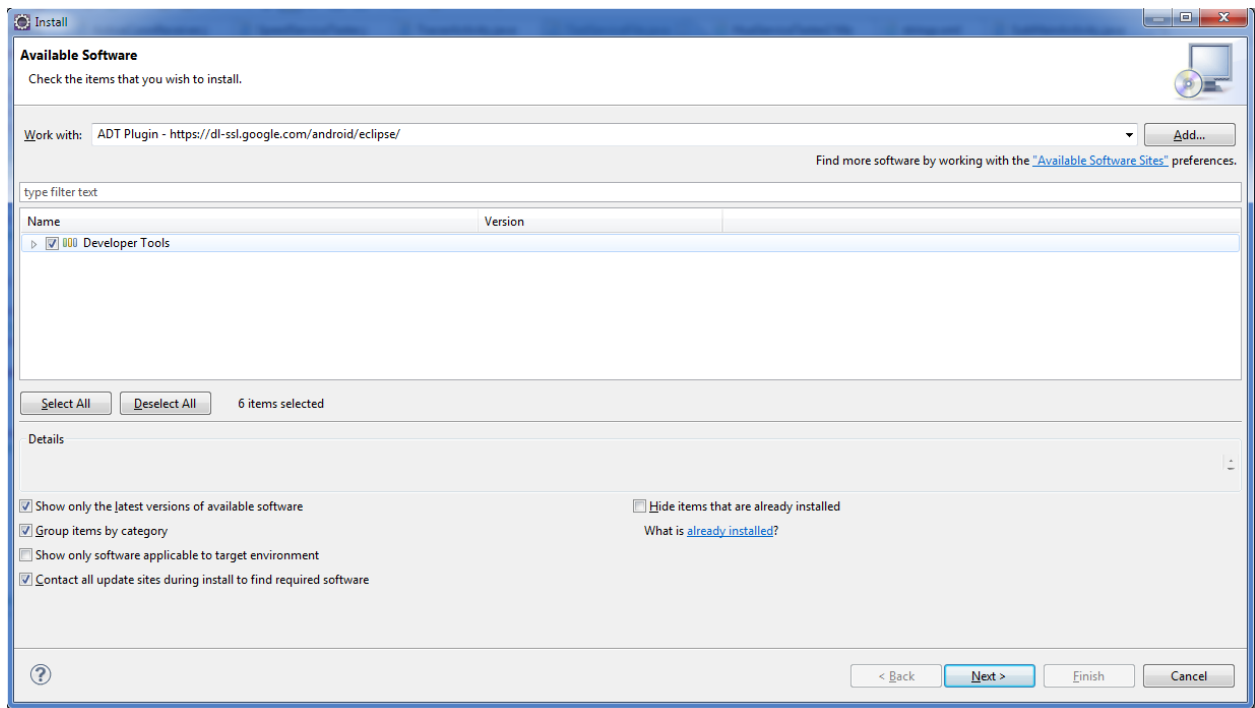
These releases includes important security fixes. Oracle strongly recommends that all Java SE 7 users upgrade to one of these releases.
[Learn more](#) ➔

- Installation Instructions - Release Notes - Oracle License - Java SE Products - Third Party Licenses - Certified System Configurations - Readme Files - JDK Readme - JRE Readme	JDK DOWNLOAD Server JRE DOWNLOAD JRE DOWNLOAD
Early Access Releases Early access versions of future releases of the JDK and the JRE are available for testing. These early access releases include future update and future major releases. These releases are licensed only for testing, not for use in production.	DOWNLOAD
JDK 8 Demos and Samples Demos and samples of common tasks and new functionality available on JDK 8. JavaFX 8 demos and samples are included in the JDK 8 Demos and Samples packages. The source code provided with demos and samples for the JDK is meant to illustrate the usage of a given feature or technique and has been deliberately simplified.	DOWNLOAD
JDK 7 and JavaFX Demos and Samples Demos and samples of common tasks and new functionality available on JDK 7. The source code provided with demos and samples for the JDK is meant to illustrate the usage of a given feature or technique and has been deliberately simplified.	DOWNLOAD

Εικόνα 6.8: Σελίδα μεταφόρτωσης JDK 7

Στη συνέχεια, αφού εγκαταστήσουμε το Eclipse¹⁵ (επιλογή *Eclipse IDE for Java Developers*) το εκτελούμε και μεταφερόμαστε στην επιλογή Help->Install New Software, όπου εισάγουμε την επιλογή ADT Plugin. Πατώντας Next, εγκαθίσταται το plugin. (Εικόνα 6.9)

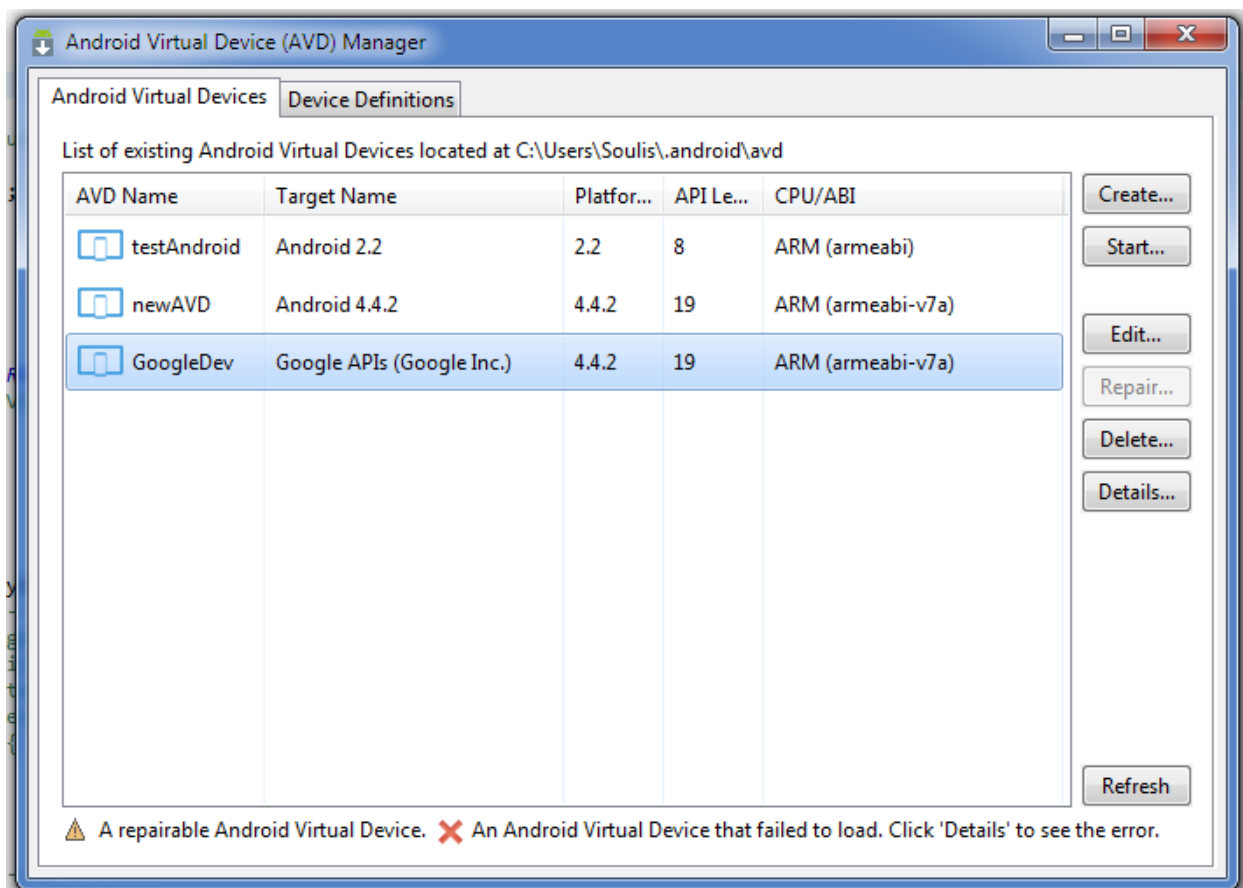
¹⁵ <http://eclipse.org/downloads/>



Εικόνα 6.9: Εγκατάσταση Eclipse ADT plugin

Το συγκεκριμένο εργαλείο ανάπτυξης περιλαμβάνει τον AVD (Android Virtual Devices) Manager, ο οποίος χρησιμοποιεί τον Android Εξομοιωτή (Emulator) για να προσομοιώσει εικονικές κινητές συσκευές ή ταμπλέτες, όπως φαίνεται και στην Εικόνα 6.10. Αυτή η λειτουργία μας βοηθά πολύ κατά την ανάπτυξη της εφαρμογής, αφού δεν είμαστε αναγκασμένοι να εγκαθιστούμε την εφαρμογή σε πραγματικές συσκευές ώστε να ελέγχουμε τη συμπεριφορά της, αλλά μπορούμε να εγκαθιστούμε την εφαρμογή σε εικονική συσκευή με σκοπό τον εντοπισμό σφαλμάτων (debugging).

Η εικονική συσκευή που χρησιμοποίησα για την εγκατάσταση της εφαρμογής είχε σαν πλατφόρμα στόχευσης (Target Name) την Google APIs διότι ήταν η μόνη που είχε από την αρχή εγκατεστημένα τα Google Play Services [38], τα οποία χρησιμοποιώ στην εφαρμογή. Η χρήση των συγκεκριμένων υπηρεσιών αναλύεται σε ειδική παράγραφο.

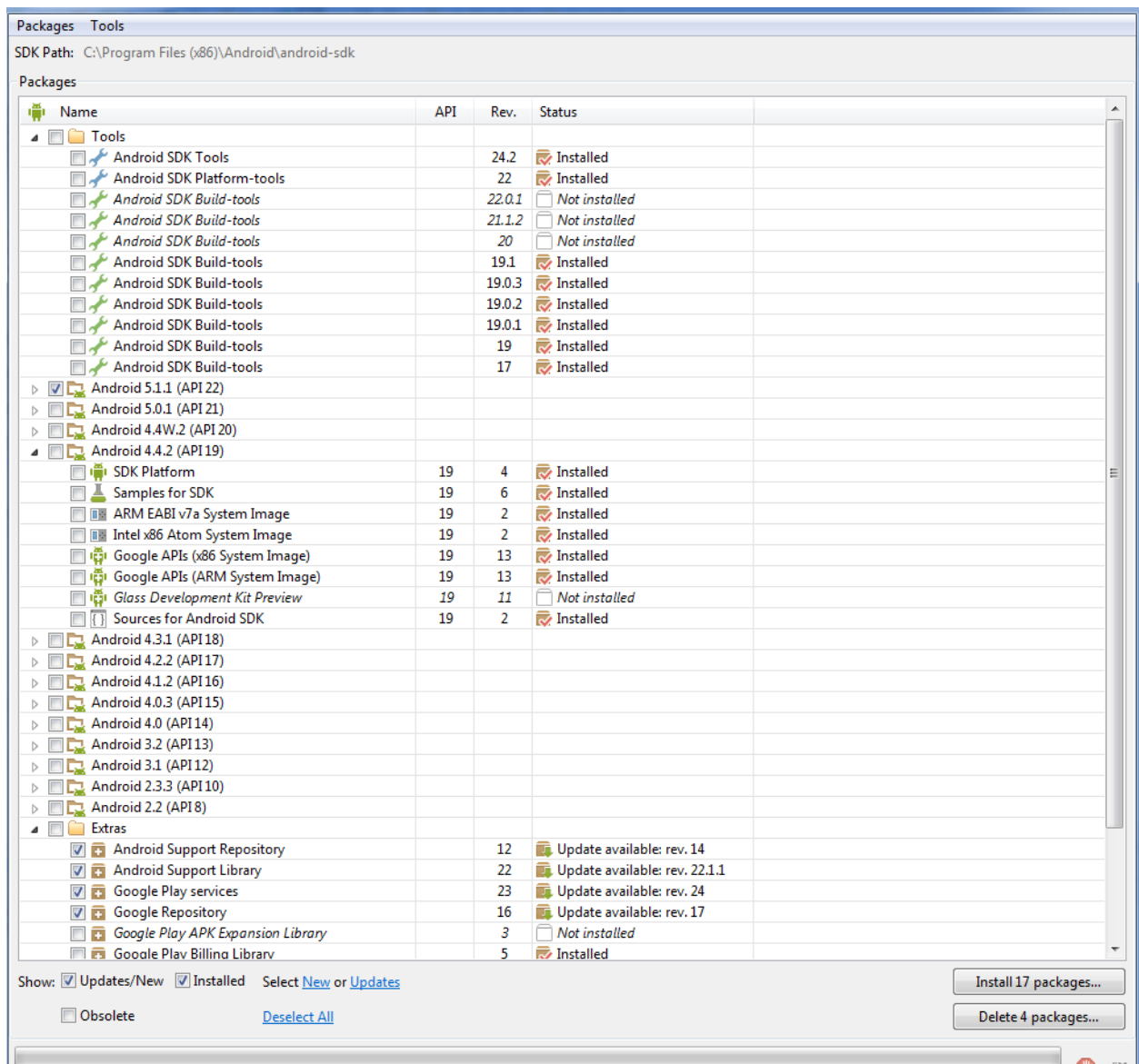


Εικόνα 6.10: AVD (Android Virtual Devices) Manager

Το Android SDK (Standard Development Kit) [39] είναι ένα σύνολο από βιβλιοθήκες και εργαλεία που χρησιμοποιούνται από τους προγραμματιστές για ανάπτυξη εφαρμογών σε περιβάλλον Android.

Ανάλογα με την εφαρμογή που θέλουμε να δημιουργήσουμε, ο SDK Manager μας δίνει τη δυνατότητα να εγκαταστήσουμε πακέτα εκδόσεων Android, εικονικές συσκευές, εκδόσεις εργαλείων οικοδόμησης κώδικα εφαρμογής (SDK Tools, SDK Platforms), αλλά και επιπλέον βιβλιοθήκες όπως Google Play Services.

Για την ανάπτυξη της δικής μου εφαρμογής, όπως φαίνεται και στην Εικόνα 6.11, επέλεξα να εγκαταστήσω την τελευταία έκδοση του Android SDK Tools (24.2) και του Android SDK Platform-tools (22). Η έκδοση Android που χρειάζεται οπωσδήποτε εγκατάσταση είναι η 4.4.2, γιατί αποτελεί την έκδοση – στόχο της εφαρμογής μου. Επίσης, χρειάζεται οπωσδήποτε να εγκατασταθεί ως επιπλέον (extra) βιβλιοθήκη η Google Play Services.



Εικόνα 6.11: Android SDK Manager

6.2.3 NetBeans 8.0.2

Όπως αναφέραμε, για τη δημιουργία των REST υπηρεσιών θα χρησιμοποιήσουμε ως εργαλείο ανάπτυξης το NetBeans IDE 8.0.2. Αρχικά, θα πρέπει να το μεταφορτώσουμε και να το εγκαταστήσουμε στον υπολογιστή μας. Η Εικόνα 6.12, μας δείχνει τη σελίδα μεταφόρτωσης του IDE [40]. Επιλέγουμε να εγκαταστήσουμε την έκδοση JavaEE, η οποία περιέχει όλες τις βιβλιοθήκες που χρειαζόμαστε για την ανάπτυξη της υπηρεσίας. Η συγκεκριμένη έκδοση περιέχει ενσωματωμένο τον application server Glassfish, δυνατότητα πολύ χρήσιμη ώστε πριν

εγκαταστήσουμε την υπηρεσία στον παραγωγικό διακομιστή εφαρμογών, να μπορούμε να κάνουμε deploy την υπηρεσία τοπικά στον υπολογιστή μας για δοκιμαστική χρήση και εντοπισμό σφαλμάτων.

NetBeans IDE 8.0.2 Download 8.0.1 | 8.0.2 | Development | Archive

Email address (optional): IDE Language: English Platform: Windows

Subscribe to newsletters: ☒ Monthly ☐ Weekly ☒ NetBeans can contact me at this address

Note: Greyed out technologies are not supported for this platform.

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	C/C++	HTML5 & PHP	All
NetBeans Platform SDK	•	•			•
Java SE	•	•			•
Java FX	•	•			•
Java EE		•			•
Java ME					•
HTML5		•		•	•
Java Card™ 3 Connected					•
C/C++			•		•
Groovy					•
PHP				•	•
Bundled servers					
GlassFish Server Open Source Edition 4.1		•			•
Apache Tomcat 8.0.15		•			•

[Download](#) Free, 90 MB
 [Download](#) Free, 186 MB
 [Download](#) Free, 63 MB
 [Download](#) Free, 63 MB
 [Download](#) Free, 205 MB

* You can add or remove packs later using the IDE's Plugin Manager (Tools | Plugins).

Java 7 and later versions are required for installing and running the PHP and C/C++ NetBeans Bundles. You can download the [latest Java at java.com](#).

JDK 7 and later versions are required for installing and running the Java SE, Java EE and All NetBeans Bundles. You can download [standalone JDK](#) or download the latest [JDK with NetBeans IDE Java SE bundle](#).

You can start developing applications based on the NetBeans Platform using the NetBeans IDE for Java SE. Learn more about the [NetBeans Platform](#). NetBeans source code and binary builds without bundled runtimes are also available in [zip file format](#). See also [instructions on how to build the IDE from sources](#) or [installation instructions](#).

Important Legal Information:

NetBeans Community Distributions are available under a Dual License consisting of the [Common Development and Distribution License \(CDDL\) v1.0](#) and [GNU General Public License \(GPL\) v2](#). Such distributions include additional components under separate licenses identified in the [License file](#). See the [Third Party License file](#) for external components included in NetBeans and their associated licenses.

Εικόνα 6.12: σελίδα μεταφόρτωσης NetBeans IDE 8.0.2

6.3 Η υπηρεσία GCM

6.3.1 Περιγραφή

Η υπηρεσία GCM (Google Cloud Messaging) για εφαρμογές Android (Εικόνα 6.13), είναι μία υπηρεσία της Google που μας επιτρέπει να στέλνουμε δεδομένα από τον διακομιστή μας σε έξυπνες συσκευές Android (downstream messages) ή από τις συσκευές προς το διακομιστή (upstream messages). Η υπηρεσία GCM χειρίζεται το πλήθος, την αναμονή και την αποστολή των μηνυμάτων σε Android εφαρμογές που εκτελούνται σε έξυπνες συσκευές – στόχους, χωρίς να απαιτείται καμία χρέωση. Πρόκειται δηλαδή για μία δωρεάν υπηρεσία της Google. (πηγή [15])



Εικόνα 6.13: Λογότυπο Υπηρεσίας GCM

GCM Components

Τα GCM components είναι τα συστατικά της υπηρεσίας GCM που διαδραματίζουν πρωτεύοντα ρόλο στην υπηρεσία (πηγή [41]). Είναι τα εξής:

- *GCM Connection Servers*

Οι διακομιστές που παρέχει η Google για την αποστολή και λήψη μηνυμάτων μεταξύ του δικού μας διακομιστή (3rd-party app server) και της GCM εφαρμογής – πελάτη (GCM client app).

- *Client App*

Η GCM εφαρμογή – πελάτη που υλοποιείται στις, εγγεγραμμένες στην GCM υπηρεσία, έξυπνες συσκευές, η οποία επικοινωνεί με τους GCM connection servers.

➤ *3rd-party App Server*

Πρόκειται για διακομιστή εφαρμογών (application server), στον οποίο υλοποιείται η GCM εφαρμογή – διακομιστή (GCM server) από εμάς και έχει ως σκοπό την αποστολή μηνυμάτων προς τις εφαρμογές – πελάτη (client apps) μέσω των GCM connection servers (Εικόνα 6.14).

GCM Credentials

Το σύνολο των IDs και tokens, που χρησιμοποιούνται από την υπηρεσία GCM, για να βεβαιωθούμε ότι όλα τα τμήματα της υπηρεσίας είναι αυθεντικά και τα μηνύματα στέλνονται στους σωστούς παραλήπτες (πηγή [41]). Είναι τα εξής:

➤ *Sender ID*

Το Sender ID παράγεται κατά τη διαδικασία δημιουργίας project στη κονσόλα προγραμματιστών της Google (*Google Developers console*¹⁶). Γενικά, για όλες τις υπηρεσίες της Google θα πρέπει να έχουμε δημιουργήσει λογαριασμό στη Google και τουλάχιστον ένα project. Το **project number** είναι στην ουσία το Sender ID. Η διαδικασία εγγραφής και δημιουργίας project και υπηρεσιών στη Google θα αναλυθεί στη συνέχεια (πηγή [42]). Το Sender ID χρησιμοποιείται κατά τη διαδικασία εγγραφής μίας συσκευής στην υπηρεσία GCM, για να ταυτοποιήσει τον 3rd party app server, ο οποίος στέλνει τα μηνύματα στις εγγεγραμμένες συσκευές.

➤ *Sender Auth Token*

Είναι το API κλειδί (key) το οποίο σώζεται στον 3rd party app server. Χρησιμοποιείται από τον app server για να αποκτή πρόσβαση στις υπηρεσίες της Google.

➤ *Application ID*

Αφορά την GCM εφαρμογή – πελάτη (client app), η οποία έχει εγγραφεί στην υπηρεσία για να λαμβάνει μηνύματα. Έτσι διασφαλίζεται ότι τα μηνύματα θα παραδοθούν στη σωστή Android συσκευή.

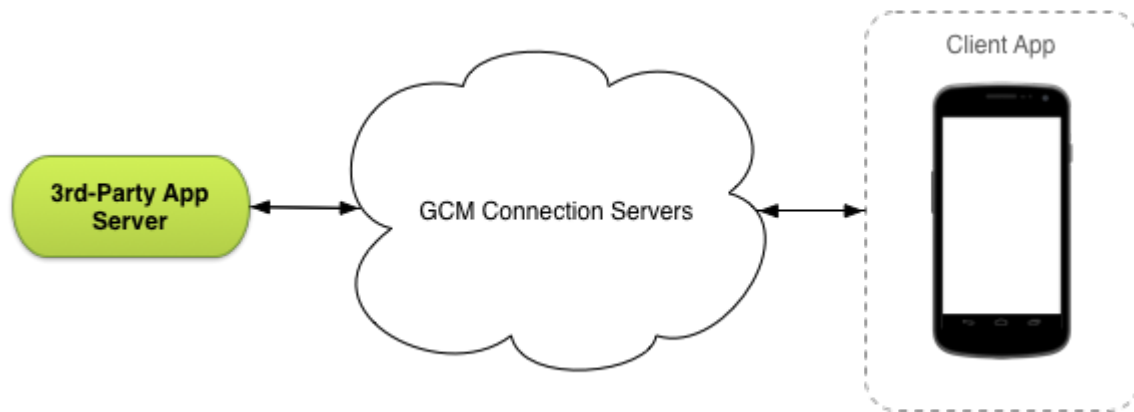
➤ *Registration ID*

Είναι το ID που λαμβάνουν οι συσκευές από τον GCM server κατά τη διαδικασία εγγραφής τους στην υπηρεσία. Το Registration ID πρέπει να παραμένει κρυφό.

¹⁶ <https://code.google.com/apis/console/>

6.3.2 GCM Αρχιτεκτονική

Η υλοποίηση της υπηρεσίας GCM περιλαμβάνει τους GCM connection servers που παρέχονται από την Google, τον 3rd party App server που επικοινωνεί με τους connection servers και στον οποίο υλοποιούμε τον GCM app server [33], και την GCM εφαρμογή – πελάτη (client app) που υλοποιείται σε Android συσκευές [35]. (πηγή [41])



Εικόνα 6.14: Αρχιτεκτονική Υπηρεσίας GCM (πηγή [41])

Στην Εικόνα 6.14 φαίνεται η αρχιτεκτονική της υπηρεσίας και το πώς τα συστατικά (components) της υπηρεσίας αλληλεπιδρούν μεταξύ τους.

- Οι GCM connection servers λαμβάνουν μηνύματα από τον 3rd party app server και στέλνουν μηνύματα στις client app. Οι GCM connection servers μπορεί να είναι τύπου HTTP¹⁷, οπότε στέλνουν μόνο μηνύματα στις client app ή τύπου XMPP¹⁸, οπότε μπορούν να δέχονται επιπλέον μηνύματα από τις client app.
- Ο 3rd party server υλοποιείται από εμάς να δουλεύει σε συνεργασία με τους connection servers της Google. Ο app server στέλνει μηνύματα σε έναν GCM connection server κι αυτός τα αποθηκεύει σε σειρά (επειδή μπορεί οι client app να βρίσκονται εκτός λειτουργίας) και τα στέλνει στις client apps, που είναι εγκατεστημένες σε εγγεγραμμένες Android συσκευές.
- Οι client app είναι οι GCM εφαρμογές – πελάτη που είναι εγκατεστημένες σε Android έξυπνες συσκευές. Για να λαμβάνουν GCM μηνύματα πρέπει να έχουν εγγραφεί στην

¹⁷ <https://developer.android.com/google/gcm/http.html>

¹⁸ <https://developer.android.com/google/gcm/ccs.html>

υπηρεσία (θα αναλυθεί στην παράγραφο 6.3.3) και να έχουν λάβει το *Registration ID*. Εάν χρησιμοποιούν XMPP connection server μπορούν και να στέλνουν μηνύματα στον διακομιστή της Google (upstream messages).

6.3.3 Υλοποίηση της υπηρεσίας GCM στην Εφαρμογή

Η υπηρεσία GCM θα χρησιμοποιηθεί στην εφαρμογή – πελάτη για έξυπνες συσκευές Android που δημιουργήσαμε. Η υλοποίηση του GCM client έγινε από εμένα, με βάση το άρθρο [35] της Google. Η υλοποίηση του GCM server (3rd app server) έγινε στον application server Glassfish (κεφάλαιο 5) από τον Gerti Nurka [34], με βάση το άρθρο [33] της Google. Η υλοποίηση είναι για GCM connection servers τύπου HTTP, αφού θέλουμε ο server να στέλνει μόνο HTTP μηνύματα προς τις συσκευές κι όχι να λαμβάνει μηνύματα από τις συσκευές.

Η υλοποίηση της υπηρεσίας GCM προϋποθέτει την εγκατάσταση της βιβλιοθήκης Google Play Services (παράγραφος 6.6.1) και την εγγραφή μας στις υπηρεσίες της Google. Επίσης, οι Android συσκευές πρέπει να έχουν εγκατεστημένη την έκδοση 2.3 και πάνω.

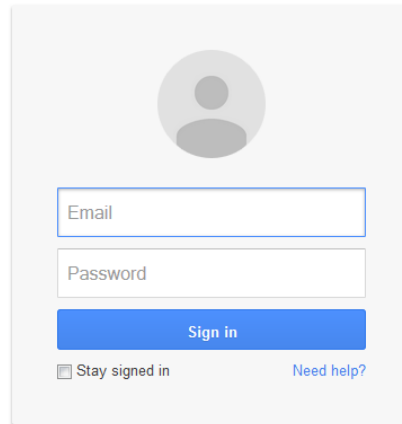
- Όπως φαίνεται στην Εικόνα 6.15 αρχικά πρέπει να δημιουργήσουμε, εάν δεν έχουμε, έναν λογαριασμό στην Google (στη διεύθυνση <http://google.com> επιλέγουμε πάνω δεξιά το κουμπί *Sign In*).
- Ανοίγουμε την κονσόλα προγραμματιστών της Google (*Google Developers console*) και δημιουργούμε ένα Google API project (Εικόνα 6.16). Παράγεται ένα *project number* το οποίο το χρησιμοποιούμε ως **Sender ID** στην υπηρεσία GCM και συγκεκριμένα στην υλοποίηση της client app.
- Από τα διαθέσιμα Google APIs επιλέγουμε το *Cloud Messaging for Android* και το ενεργοποιούμε (επιλογή *Enable API* στην Εικόνα 6.17).
- Στο αριστερό μενού επιλέγουμε *APIs & auth -> Credentials* και στο παράθυρο που εμφανίζεται επιλέγουμε *Create a new key -> Server key* (Εικόνα 6.18). Το κλειδί που παράγεται είναι το **API key** που χρησιμοποιείται στον 3rd party app server, όπως περιγράψαμε στην προηγούμενη παράγραφο. (πηγή [42])

Πλέον είμαστε έτοιμοι να υλοποιήσουμε την υπηρεσία GCM.



One account. All of Google.

Sign in with your Google Account

A sign-in form with a grey user icon at the top. Below it are two input fields: 'Email' and 'Password'. A blue 'Sign in' button is positioned below the password field. At the bottom left, there is a checkbox labeled 'Stay signed in'. At the bottom right, there is a blue link labeled 'Need help?'.

[Create an account](#)

One Google Account for everything Google



Εικόνα 6.15: Δημιουργία Google λογαριασμού

Start using the Google APIs console
to manage your API usage

A blue 3D cube with green curved arrows looping around it, representing the Google APIs console.

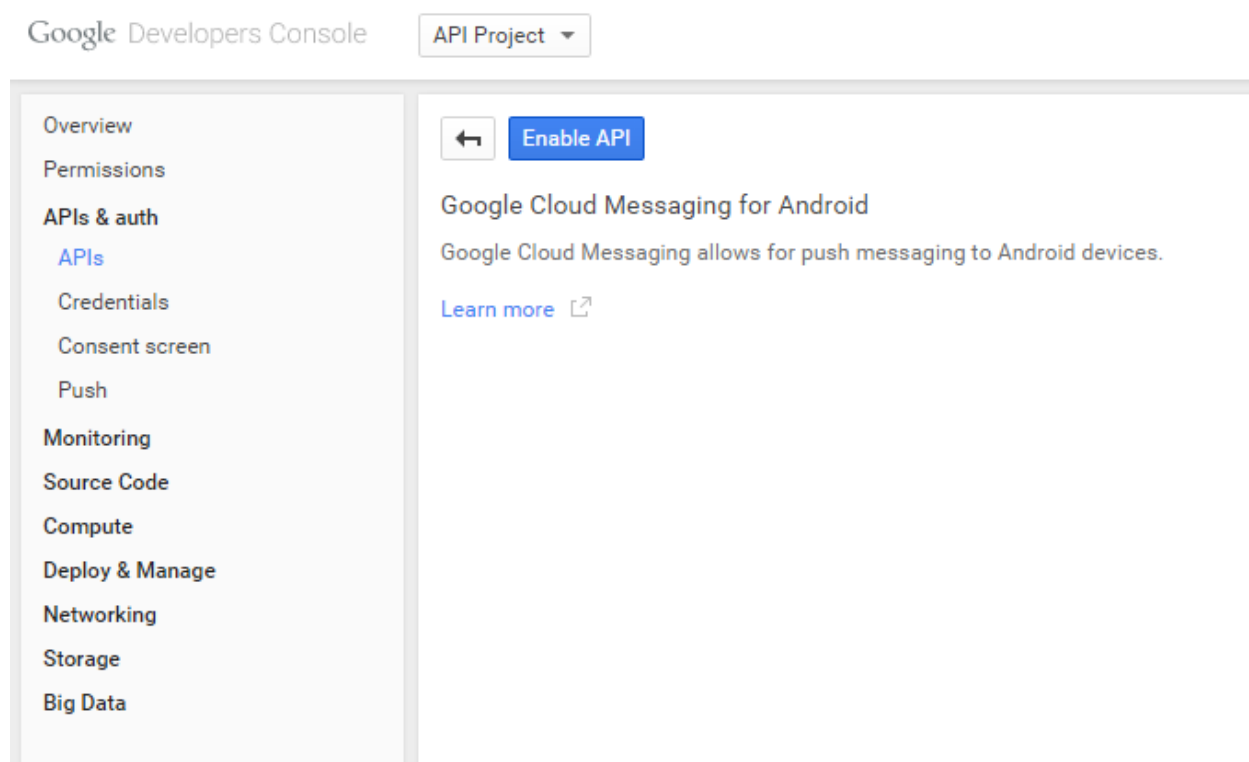
Creating an **APIs project** will let you:

- **Use** Google APIs **beyond anonymous limits**.
- **Monitor** API usage and **control** API access.
- **Share** API management with a team.

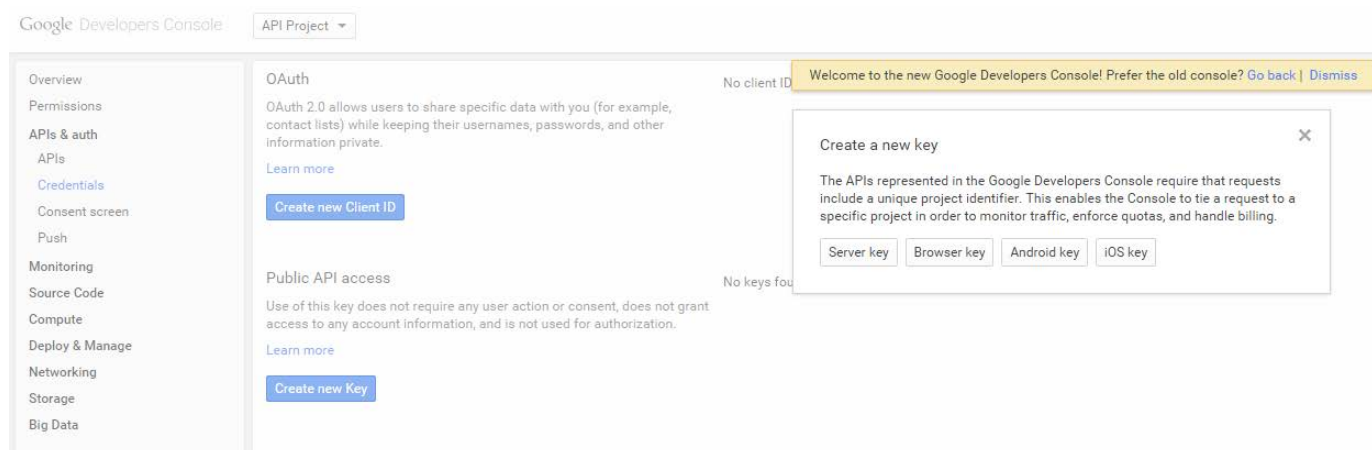
Create project...

[Code Home](#) - [Privacy Policy](#)

Εικόνα 6.16: Σελίδα δημιουργίας Google API project



Εικόνα 6.17: Ενεργοποίηση υπηρεσίας GCM



Εικόνα 6.18: Δημιουργία API κλειδιού (key)

Για να λειτουργήσει η υπηρεσία θα πρέπει να προσθέσουμε στο αρχείο `AndroidManifest.xml` της εφαρμογής κάποια δικαιώματα. Αυτά είναι:

- `com.google.android.c2dm.permission.RECEIVE`
Δικαίωμα για εγγραφή και λήψη μηνυμάτων από την υπηρεσία GCM
- `android.permission.INTERNET`
Δικαίωμα πρόσβασης στο διαδίκτυο, ώστε κατά την εγγραφή της συσκευής να στέλνεται το *Registration_ID* στον 3rd party app server
- `android.permission.WAKE_LOCK`
Δικαίωμα για να μένει η συσκευή ενεργή κατά τη διαδικασία λήψης μηνύματος
- `gr.soulis.huaservicetester2.permission.C2D_MESSAGE`
Δημιουργεί δικαίωμα λήψης μηνυμάτων μόνο από τη συγκεκριμένη συσκευή.
- `com.google.android.c2dm.intent.RECEIVE`
Εγγραφή receiver, με δικαίωμα `com.google.android.c2dm.permission.SEND`, ώστε όταν η συσκευή είναι ενεργοποιημένη να δέχεται μηνύματα μόνο από τους GCM connection servers.

```
<!-- GCM requires a Google account. (ref:
https://code.google.com/p/gcm/source/checkout) -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
<!-- Keeps the processor from sleeping when a message is received. -->
    <uses-permission android:name="android.permission.WAKE_LOCK" />
<!--
Creates a custom permission so only this app can receive its messages.
NOTE: the permission *must* be called PACKAGE.permission.C2D_MESSAGE,
      where PACKAGE is the application's package name.

-->
    <permission
        android:name="gr.soulis.huaservicetester2.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />

    <uses-permission
android:name="gr.soulis.huaservicetester2.permission.C2D_MESSAGE" />

    <!-- This app has permission to register and receive data message. -->
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />

<!-- Permission for accessing the Internet -->
    <uses-permission android:name="android.permission.INTERNET" />

<receiver
    android:name=".GcmBroadcastReceiver"
    android:enabled="true"
    android:exported="true"
    android:permission="com.google.android.c2dm.permission.SEND" >
```

```

<intent-filter>

    <!-- Receives the actual messages. -->
    <action
android:name="com.google.android.c2dm.intent.RECEIVE" />

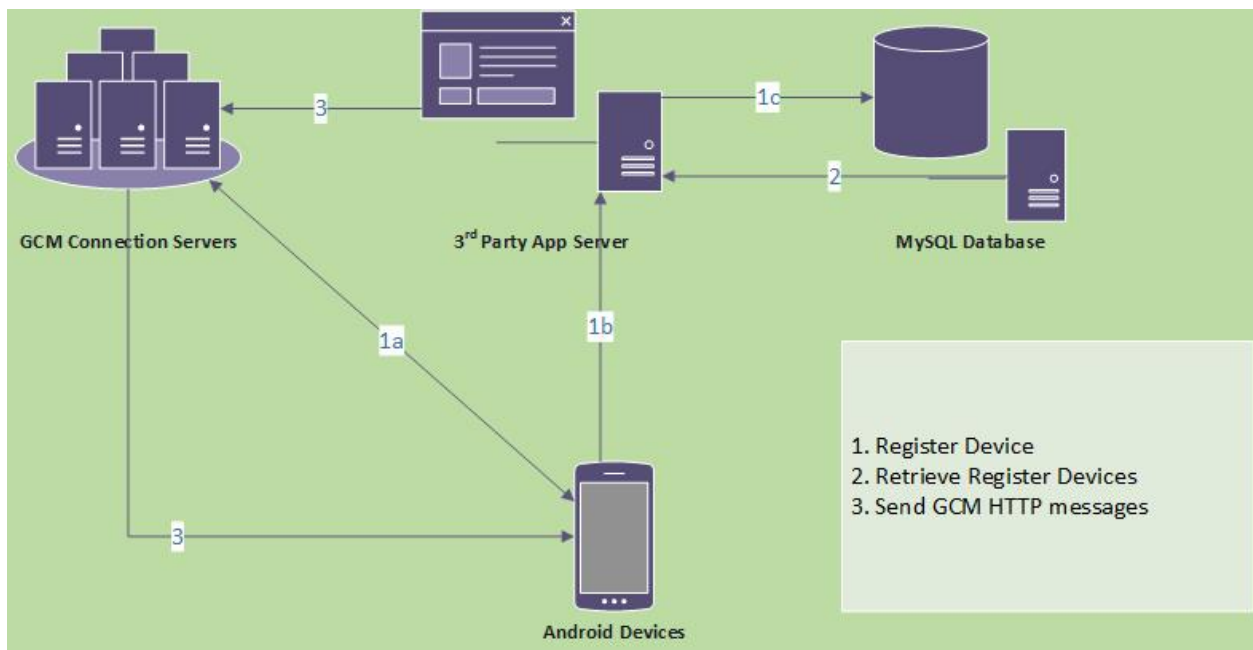
        <category android:name="gr.soulis.huaservicetester2" />
    </intent-filter>
</receiver>

```

Ο GCM client που υλοποιήθηκε στην Android εφαρμογή, χρησιμοποιείται για τη λήψη μηνυμάτων από τον GCM server σχετικά με το νέο αρχείο καμπανιών. Οι καμπάνιες, όπως έχω αναφέρει, είναι περιοχές με χωροχρονικά κριτήρια μέσα στις οποίες πραγματοποιούνται οι μετρήσεις απόδοσης του ασύρματου δικτύου.

Οι συσκευές έχουν αποθηκευμένες, σε αρχείο σε μορφή JSON, τις ενεργές καμπάνιες στις οποίες γίνονται οι μετρήσεις. Όποτε εισάγεται μία νέα καμπάνια στη βάση δεδομένων, ο app server στέλνει HTTP μηνύματα στις Android συσκευές, μέσω των GCM connection servers, και οι client app που «τρέχουν» στις συσκευές λαμβάνουν αυτά τα μηνύματα, σε μορφή JSON, ώστε να ενημερώνονται άμεσα για τις αλλαγές, εμφανίζοντας ένα Notification μήνυμα. Στη συνέχεια, αποθηκεύουν το JSON μήνυμα τοπικά σε αρχείο αντικαθιστώντας το παλιό, ώστε οι μετρήσεις να γίνονται με βάση τα νέα κριτήρια που προκύπτουν από το αρχείο καμπανιών.

Η διαδικασία αποστολής GCM μηνυμάτων στην Android εφαρμογή αναπαριστάται στην Εικόνα 6.19. Αρχικά, όταν εγκαθίσταται η εφαρμογή στη συσκευή, γίνεται εγγραφή (register) της συσκευής στην υπηρεσία GCM, μέσω μηνυμάτων με τους GCM connection servers. Στη συνέχεια, ο GCM client λαμβάνει το *Registration_Id* της συσκευής, το αποθηκεύει στη συσκευή και το στέλνει στον application server για να το αποθηκεύσει στη βάση δεδομένων MySQL. Όταν ο 3rd party App server πρέπει να στείλει τις νέες καμπάνιες στις Android συσκευές, συνδέεται με τη βάση δεδομένων για να ανακτήσει τα *Registration_Ids* των εγγεγραμμένων συσκευών και στέλνει στους GCM connection servers τα μηνύματα με τις καμπάνιες. Οι GCM connection servers, στη συνέχεια, αναλαμβάνουν να αποστείλουν τα HTTP μηνύματα στις εγγεγραμμένες συσκευές Android.



Εικόνα 6.19: Αναπαράσταση αποστολής μηνυμάτων σε GCM αρχιτεκτονική

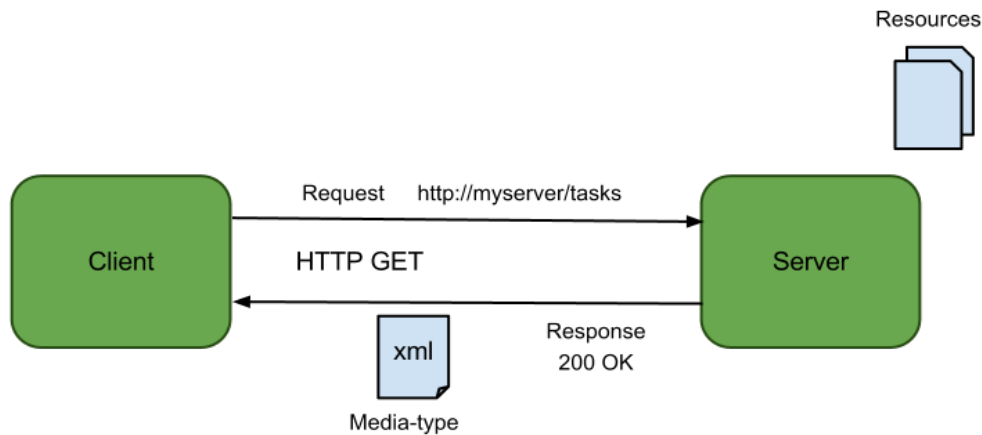
6.4 REST Web Services

6.4.1 Περιγραφή

Το REST (**RE**presentational **S**tate **T**ransfer) είναι ένας τύπος αρχιτεκτονικής λογισμικού που χρησιμοποιείται για την υλοποίηση υπηρεσιών διαδικτύου (web services) [43]. Έχει γίνει ευρέως γνωστό στο χώρο του διαδικτύου ως ένας εναλλακτικός και απλούστερος τρόπος υλοποίησης web service σε σχέση με τον τύπο SOAP (Simple Object Access Protocol) και WSDL (Web Service Definition Language).

Τα RESTful συστήματα, που συμμορφώνονται με τους περιορισμούς της αρχιτεκτονικής REST, για να επικοινωνήσουν μεταξύ τους, χρησιμοποιούν στην πλειονότητά τους το πρωτόκολλο HTTP και τις μεθόδους του POST, GET, PUT και DELETE, τα οποία χρησιμοποιούνται από τους φυλλομετρητές (web browsers) για ανάκτηση ιστοσελίδων και αποστολή δεδομένων σε απομακρυσμένους διακομιστές (Εικόνα 6.20).

Η αρχιτεκτονική REST αναπτύχθηκε από την W3C¹⁹ Technical Architecture Group (TAG) παράλληλα με την HTTP 1.1 και βασίστηκε στον υπάρχον σχεδιασμό του HTTP 1.0.



Εικόνα 6.20: Αρχιτεκτονική REST Web Service²⁰

Τα RESTful συστήματα πρέπει να υπακούουν σε έξι περιορισμούς [44]:

➤ *Uniform Interface*

Καθορίζει τη διεπαφή μεταξύ πελάτη (client) και εξυπηρετή (server). Απλοποιεί και διαχωρίζει τμήματα της αρχιτεκτονικής, με αποτέλεσμα να επιτρέπει κάθε τμήμα της να εξελίσσεται ανεξάρτητα. Οι τέσσερις αρχές στις οποίες υπακούει είναι:

– *Resource – Based*

Επιμέρους πόροι αναγνωρίζονται από αιτήματα (requests) χρησιμοποιώντας τα URIs ως αναγνωριστικά πόρων. Οι πόροι διαχωρίζονται εννοιολογικά από τις παραστάσεις που επιστρέφονται στον πελάτη. Αν για παράδειγμα ο πόρος αναφέρεται σε βάση δεδομένων, δεν επιστρέφεται η βάση δεδομένων αλλά οι εγγραφές σε μορφή XML (Extensible Markup Language) ή JSON.

– *Manipulation of Resources Through Representations*

Όταν ένας πελάτης κατέχει μία αναπαράσταση ενός πόρου, μπορεί και έχει το δικαίωμα να διαγράψει ή να μεταβάλλει τον πόρο στον διακομιστή.

– *Self-descriptive Messages*

¹⁹ World Wide Web Consortium (http://en.wikipedia.org/wiki/World_Wide_Web_Consortium)

²⁰ www.prideparrot.com

Κάθε μήνυμα συμπεριλαμβάνει την απαραίτητη πληροφορία που περιγράφει τον τρόπο επεξεργασίας του. Για παράδειγμα, ποιος parser πρέπει να χρησιμοποιηθεί για MIME type μηνύματα.

– *Hypermedia as the Engine of Application State (HATEOAS)*

Οι πελάτες μπορούν να πραγματοποιήσουν μετάβαση κατάστασης στα services μέσω περιεχομένου (body content), query string παραμέτρων, κεφαλίδας αίτησης (request header) και URI (περιλαμβάνει το όνομα του πόρου). Τα services μέσω περιεχομένου (body content), κωδικών απάντησης (response codes) και κεφαλίδων απάντησης (response header).

➤ *Stateless*

Κατά την επικοινωνία client – server καμία πληροφορία δε διατηρείται στον server για το περιεχόμενο του client μεταξύ των αιτημάτων (requests). Δεν υπάρχει η έννοια της συνόδου (session), όπου η κατάσταση διατηρείται μεταξύ πολλαπλών HTTP αιτημάτων (requests).

➤ *Cacheable*

Οι πελάτες πρέπει να είναι ικανοί να αποθηκεύουν προσωρινά (cacheable) τις απαντήσεις των υπηρεσιών (services) και οι υπηρεσίες να δηλώνουν εάν η πληροφορία είναι cacheable ώστε να αποτρέπουν τους πελάτες να εμφανίζουν άκυρη πληροφορία. Η σωστή διαχείριση αυτής της δυνατότητας βελτιώνει σημαντικά την απόδοση της υπηρεσίας.

➤ *Client – Server*

Η παραδοχή *uniform interface* διαχωρίζει τους πελάτες (clients) από τους εξυπηρέτες (servers). Αυτό σημαίνει ότι για παράδειγμα οι πελάτες δεν χρειάζεται να έχουν επίγνωση της αποθήκευσης δεδομένων στον εξυπηρέτη και οι εξυπηρέτες επίγνωση για τη διεπαφή χρήστη στους πελάτες. Ο κώδικας του πελάτη και του εξυπηρέτη είναι ανεξάρτητη με αποτέλεσμα να είναι μεταφέρσιμος (portability).

➤ *Layered system*

Ένας πελάτης δε μπορεί να δημοσιοποιήσει σε ποιον τελικό διακομιστή συνδέεται. Ενδιάμεσοι διακομιστές μπορούν να βελτιώσουν την επεκτασιμότητα του συστήματος. Επίπεδα πρόσβασης (Layers) μπορούν να επιβάλουν πολιτική ασφαλείας.

➤ *Code On Demand (προαιρετικό)*

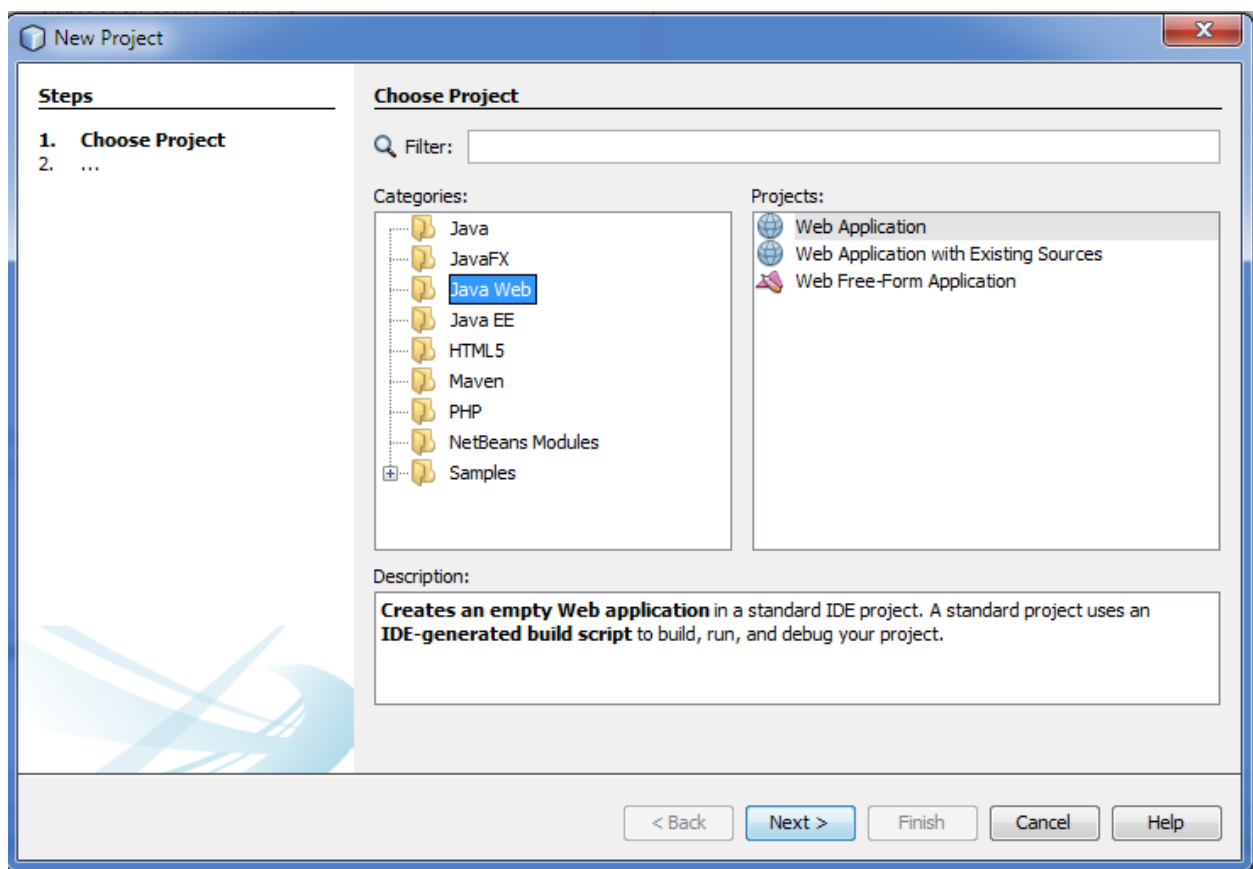
Οι Εξυπηρέτες είναι ικανοί προσωρινά να επεκτείνουν ή να διαμορφώσουν τη λειτουργικότητα του πελάτη παρέχοντας προγραμματιστική λογική η οποία μπορεί να

εκτελεστεί, όπως πχ Java applets ή μία γλώσσα προγραμματισμού που εκτελείται σε clients όπως Javascript.

6.4.2 Υλοποίηση REST web services στην Εφαρμογή

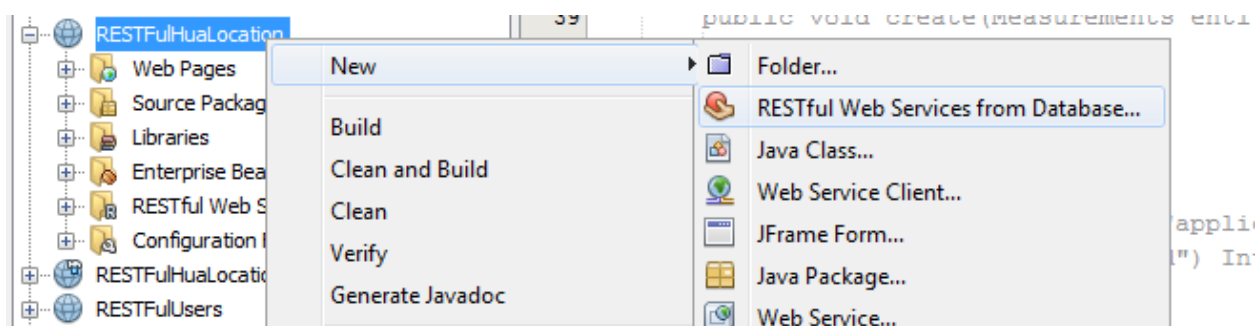
Τα REST web services που δημιουργήθηκαν για την Android εφαρμογή έχουν ως σκοπό την παροχή πρόσβασης στα δεδομένα της MySQL βάσης δεδομένων. Οι λειτουργίες της εφαρμογής που χρησιμοποιούν τις RESTful υπηρεσίες είναι σχετικές με την ανάκτηση στοιχείων σχετικά με τις μετρήσεις δικτύου στις οθόνες παρουσίασης στατιστικών και στοιχείων σχετικά με τις ενεργές καμπάνιες της εφαρμογής.

Για τη δημιουργία της RESTful εφαρμογής χρησιμοποιήθηκε ως εργαλείο υλοποίησης το NetBeans 8.0.2 και εγκαταστάθηκε σε διακομιστή Glassfish. Αρχικά δημιουργήθηκε ένα NetBeans web project.



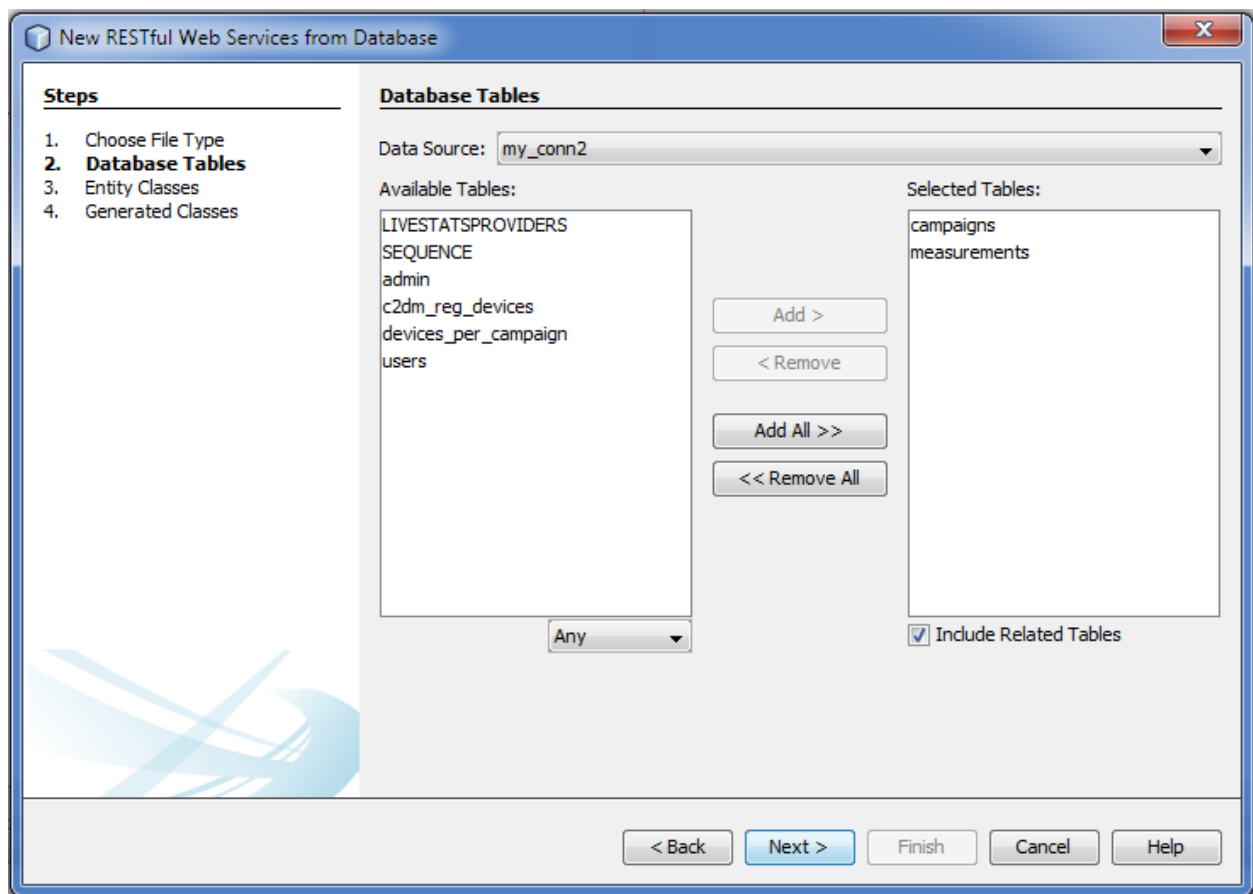
Εικόνα 6.21: Δημιουργία NetBeans web application

Αφού ανοίξουμε το εργαλείο NetBeans, από το μενού επιλέγουμε *File->New->Project..* και εμφανίζεται το παράθυρο της Εικόνας 6.21. Επιλέγουμε *Java Web -> Web Application* και στη συνέχεια διαλέγουμε ένα όνομα για την εφαρμογή μας (*RESTFulHuaLocation*). Το Project δημιουργείται και τώρα πλέον πρέπει να προσθέσουμε στην εφαρμογή τα RESTful services. Όπως φαίνεται και στην Εικόνα 6.22, κάνοντας δεξί κλικ πάνω στο όνομα του project εμφανίζεται ένα πλαίσιο επιλογών, στο οποίο επιλέγουμε *New -> RESTful Web Services from Database*. Το NetBeans μας δίνει τη δυνατότητα να δηλώσουμε εάν επιθυμούμε το service να συνδεθεί με κάποια βάση δεδομένων, ώστε να μας διευκολύνει και να μην είμαστε υποχρεωμένοι να αναπτύξουμε κώδικα για αυτή τη σύνδεση.



Εικόνα 6.22: Δημιουργία RESTful σε NetBeans Web application

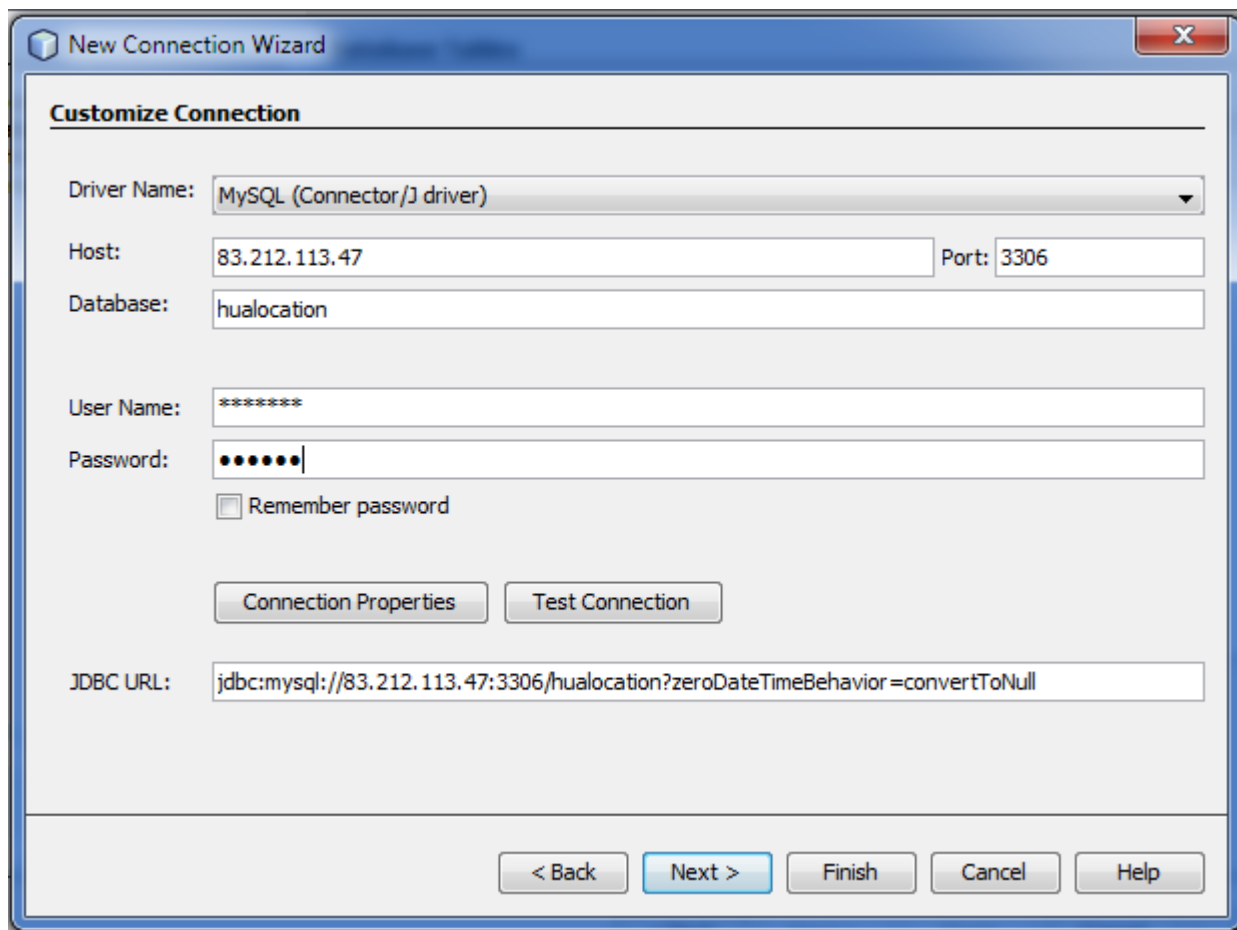
Στη συνέχεια εμφανίζεται νέο παράθυρο, όπως φαίνεται στην Εικόνα 6.23, όπου επιλέγοντας το data source που συνδέεται με τη βάση, εμφανίζονται οι πίνακες της βάσης δεδομένων.



Εικόνα 6.23: Σύνδεση RESTful service με βάση δεδομένων

Στο Data Source, όπως φαίνεται και στην Εικόνα 6.24, πρέπει να πληκτρολογήσουμε τον τύπο της βάσης δεδομένων και του οδηγού της (driver), τα username και password σύνδεσης στη βάση και το URL του οδηγού JDBC που είναι εγκατεστημένος στο διακομιστή Glassfish. (*jdbc:mysql://83.212.113.47:3306/huaLocation?zeroDateTimeBehavior=convertToNull*).

Ο οδηγός που εγκατέστησα στον Glassfish είναι ο **mysql-connector-java-5.1.24-bin.jar**, μέσα στο φάκελο lib/ της διεύθυνσης εγκατάστασής του.



Εικόνα 6.24: Οδηγός σύνδεσης με τη βάση δεδομένων MySQL

Αφού συνδεθούμε επιτυχώς στη βάση δεδομένων, επιλέγουμε από τους διαθέσιμους πίνακες τους *campaigns* και *measurements*, αφού θέλουμε το service να ανακτήσει δεδομένα μόνο από αυτούς τους πίνακες. Το NetBeans δημιουργεί entity κλάσεις για αυτούς τους πίνακες και κάποια βασικά ερωτήματα (queries) ώστε να μην χρειαστεί να γράψουμε κώδικα για τις βασικές λειτουργίες. Πατάμε *Finish* και το Project έχει δημιουργηθεί.

6.5 Η Βάση δεδομένων MySQL

6.5.1 Περιγραφή

Η MySQL είναι από τις πιο διαδεδομένες βάσεις δεδομένων στο χώρο του διαδικτύου. Μέχρι τον Ιούλιο του 2013 ήταν η πιο διαδεδομένη, αλλά λόγω της ραγδαίας αύξησης των Android συσκευών που χρησιμοποιούν τοπικά ως βάση δεδομένων την SQLite, έπεσε στη δεύτερη θέση. Χρησιμοποιείται ευρύτατα σε συστήματα ως σχεσιακή βάση δεδομένων (RDBMS²¹).

Το project της MySQL είναι ανοικτού κώδικα (open source), υπό τους όρους της GNU General Public License²² καθώς και κάτω από μια ποικιλία ιδιόκτητων συμφωνιών. Η MySQL ανήκε και προωθούνταν από μία σουηδική εταιρία την MySQL AB [45], εξαγοράστηκε όμως από την Oracle Corporation. Σήμερα, λογισμικά ανοικτού κώδικα στο διαδίκτυο χρησιμοποιούν στην πλειονότητά τους την MySQL ως σχεσιακή βάση δεδομένων. (πηγή [46])

Η MySQL είναι απόλυτα συμβατή και φιλική προς συστήματα που χρησιμοποιούν ως γλώσσα προγραμματισμού την PHP και ως web server τον Apache. Γι' αυτό το σκοπό έχει δημιουργηθεί το ευρέως γνωστό λογισμικό πακέτο LAMP (Linux, Apache, MySQL, PHP/Perl/Python). Επίσης, για τη διαχείριση της MySQL βάσης δεδομένων, σε γραφικό περιβάλλον, έχει αναπτυχθεί το επίσης ευρέως διαδεδομένο λογισμικό εργαλείο phpMyAdmin [47], με τη βοήθεια του οποίου μπορούμε να έχουμε μία γραφική απεικόνιση όλων των πινάκων της βάσης δεδομένων, να αλλάξουμε τα δικαιώματα, να δημιουργήσουμε χρήστες, γενικά να έχουμε πλήρη πρόσβαση στη βάση δεδομένων μας.

6.5.2 Οι Πίνακες της Βάσης Δεδομένων MySQL στην Εφαρμογή

Η βάση δεδομένων MySQL εγκαταστάθηκε στο εικονικό μηχάνημα στον «oceanos» που διαθέτουμε στη διεύθυνση 83.212.113.47. Επειδή χρειάζεται και η εγκατάσταση της PHP και Apache web server, διάλεξα να χρησιμοποιήσω το πακέτο LAMP [48], δίνοντας τις εντολές:

```
$ sudo apt-get update
$ sudo apt-get install lamp-server^
```

²¹ Relational Database Management System

²² http://en.wikipedia.org/wiki/GNU_General_Public_License

Το πακέτο αυτό περιέχει και το εργαλείο phpMyAdmin, που χρησιμοποίησα για τη δημιουργία και διαχείριση της βάσης δεδομένων και των χρηστών της. Είναι εγκατεστημένο στη διεύθυνση <http://83.212.113.47/phpmyadmin> και χρειάζεται πιστοποίηση (username και password του διαχειριστή της βάσης δεδομένων) για να αποκτήσουμε πρόσβαση στη βάση δεδομένων (Εικόνα 6.25).



Εικόνα 6.25: Σελίδα εισαγωγής στο phpMyAdmin

Αρχικά δημιουργούμε μία νέα βάση δεδομένων. Επιλέγουμε ως character set (κωδικοσειρά για χαρακτήρες) **utf8** και collation **utf8_general_ci**, ώστε η βάση δεδομένων μας να είναι πολυγλωσσική. Στη συνέχεια, εισάγουμε το όνομά της ως *huaLocation*, όπως φαίνεται και στην Εικόνα 6.26 και η βάση δεδομένων μας δημιουργείται.

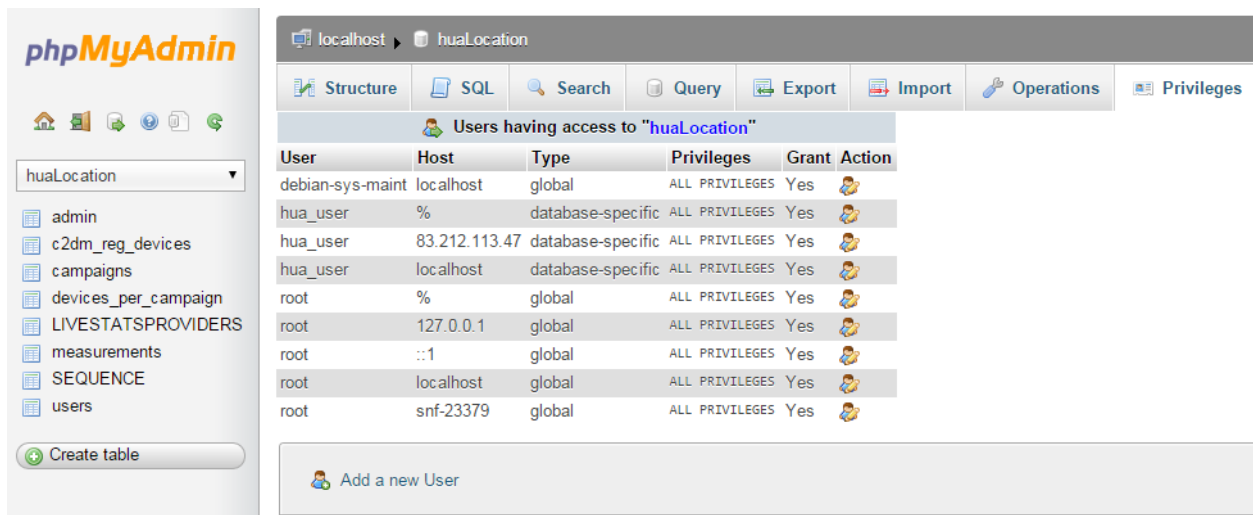


Εικόνα 6.26: Δημιουργία νέας βάσης δεδομένων MySQL

Δημιουργία Χρήστη Βάσης Δεδομένων

Για λόγους ασφαλείας, θα πρέπει να δημιουργηθεί χρήστης για τη βάση δεδομένων *huaLocation*, ο οποίος θα έχει πλήρη δικαιώματα μόνο στα στοιχεία της συγκεκριμένης βάσης. Είναι λάθος να χρησιμοποιούμε το διαχειριστή της MySQL για να συνδεόμαστε στη βάση δεδομένων, διότι ο χρήστης αυτός έχει πλήρη δικαιώματα όχι μόνο στη βάση *huaLocation* αλλά σε όλη τη MySQL, δηλαδή σε πίνακες και παραμέτρους του συστήματος και αυτό βέβαια δημιουργεί τεράστιο κενό ασφαλείας.

Στο οριζόντιο μενού του phpMyAdmin υπάρχει η επιλογή *Privileges*, όπου μπορούμε να δούμε τους υπάρχοντες χρήστες της βάσης και να δημιουργήσουμε καινούργιους (Εικόνα 6.27). Επιλέγουμε *Add New User* και στη νέα οθόνη που εμφανίζεται δηλώνουμε το όνομα και το password του χρήστη, ενώ στα δικαιώματα του χρήστη στη βάση *huaLocation* επιλέγουμε *Global Privileges (Check All)*, ώστε ο χρήστης να έχει πλήρη δικαιώματα στη βάση δεδομένων *huaLocation*.



Εικόνα 6.27: Διαχείριση χρηστών βάσης δεδομένων στο phpMyAdmin

Δημιουργία Πινάκων της Βάσης Δεδομένων

Οι πίνακες που δημιουργήθηκαν για την Android εφαρμογή είναι οι εξής (Εικόνα 6.28):

1. *campaigns*

Πίνακας αποθήκευσης των χαρακτηριστικών κάθε καμπάνιας, που εισάγεται στη βάση για τον καθορισμό των γεωγραφικών περιοχών και κριτηρίων για την πραγματοποίηση των μετρήσεων.

Πεδία:

- *camp_id (int(11))*: PRIMARY_KEY
Σειριακός μοναδικός αριθμός που χαρακτηρίζει την καμπάνια
- *create_date (timestamp)*: Default: CURRENT_TIMESTAMP
Η χρονική στιγμή καταχώρησης της καμπάνιας. Η τιμή καταχωρείται αυτόματα.
- *end_date (date)*:
Η ημ/νία λήξης ισχύος της καμπάνιας. Μετά την παρέλευση της ημ/νίας, η καμπάνια θεωρείται ανενεργή.
- *latitude (double)*:
Το γεωγραφικό πλάτος στο οποίο βρίσκεται το επίκεντρο της καμπάνιας.
- *longitude (double)*:
Το γεωγραφικό πλάτος στο οποίο βρίσκεται το επίκεντρο της καμπάνιας.
- *network_type (varchar(10))*:
Ο τύπος του δικτύου που επιτρέπονται οι μετρήσεις (both, mobile, wifi).

- *accuracy (smallint(5))*: σε m
Η επιτρεπτή ακρίβεια της μέτρησης (ανώτατο όριο).
- *radius (int(10))*: σε m
Η επιτρεπτή ακτίνα μέτρησης από το επίκεντρο της καμπάνιας (ανώτατο όριο).
- *velocity (tinyint(3))*: σε m/s
Το άνω όριο ταχύτητας της συσκευής για αποδοχή της μέτρησης
- *name (varchar(70))*:
Το όνομα της καμπάνιας
- *active (tinyint(1))*: index
Κατάσταση καμπάνιας. Τιμές: 0 (ανενεργή), 1 (ενεργή). Η καμπάνια λαμβάνεται υπόψη για μέτρηση μόνο όταν βρίσκεται σε ενεργή κατάσταση.

2. *c2dm_reg_devices*

Πίνακας καταχώρησης συσκευών που έχουν εγγραφεί στην υπηρεσία GCM.

Πεδία:

- *c2dm_id (int11)*: PRIMARY_KEY
Μοναδικός σειριακός αριθμός εγγραφής.
- *c2dm_regid (varchar(250))*: index (UNIQUE TYPE)
Το *Registration_ID* που λαμβάνει η συσκευή κατά τη διαδικασία εγγραφής της στην υπηρεσία GCM, από τους GCM connection servers. Υπάρχει περιορισμός μοναδικότητας του αριθμού στον πίνακα.
- *c2dm_active (tinyint(1))*: index
Κατάσταση εγγραφής. Τιμές: 0 (ανενεργή), 1 (ενεργή). Ανενεργή σημαίνει ότι η εφαρμογή έχει απεγκατασταθεί από την συσκευή.
- *c2dm_date (timestamp)*: Default: CURRENT_TIMESTAMP
Η χρονική στιγμή καταχώρησης της συσκευής στην υπηρεσία GCM. Η τιμή καταχωρείται αυτόματα.
- *c2dm_androidid (varchar(128))*: index
Μοναδικό αναγνωριστικό της συσκευής. Είναι ο κωδικός της συσκευής που χρησιμοποιείται και στον πίνακα των μετρήσεων.

3. *devices_per_campaign*

Πίνακας που δηλώνει σε ποιες κινητές συσκευές στάλθηκε η κάθε καμπάνια (μέσω μηνυμάτων GCM). Η σημασία αυτού του πίνακα έχει να κάνει με την ιστορικότητα αποστολής μηνυμάτων.

Πεδία:

- *id (int(11))*: PRIMARY_KEY
Μοναδικός σειριακός αριθμός εγγραφής.
- *dc_campid (int(11))*: FOREIGN_KEY (campaigns.camp_id)
Ο κωδικός αριθμός που χαρακτηρίζει την κάθε καμπάνια
- *dc_devid (int(11))*: FOREIGN_KEY (c2dm_reg_devices. c2dm_id)
Ο κωδικός αριθμός που χαρακτηρίζει την κάθε συσκευή που έχει εγγραφεί στην υπηρεσία GCM
- *sent_date (timestamp)*: Default: CURRENT_TIMESTAMP
Η χρονική στιγμή αποστολής μηνύματος που αφορά την καμπάνια *dc_campid* στην συσκευή *dc_devid*. Η τιμή καταχωρείται αυτόματα.

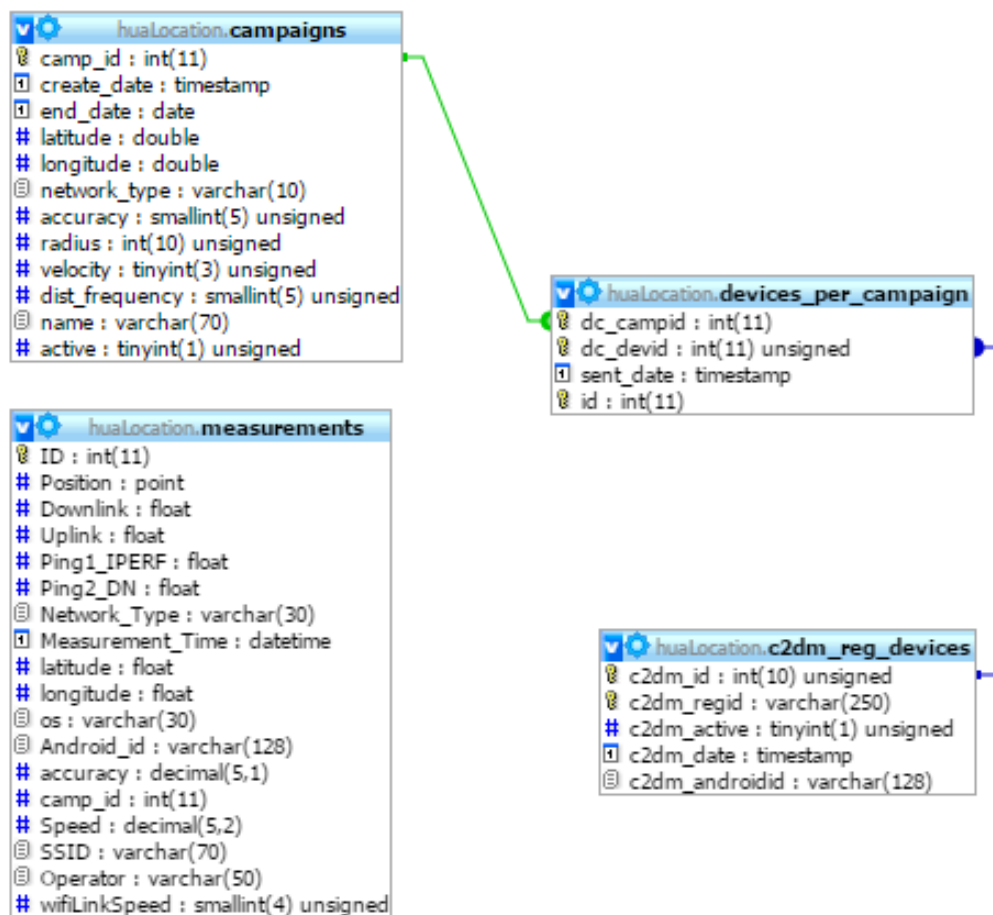
4. *measurements*

Είναι ο πίνακας στον οποίο αποθηκεύονται τα αποτελέσματα μετρήσεων των μετρικών των ασύρματων δικτύων. Κάποια πεδία προϋπήρχαν στην αρχική υλοποίηση του Κορωνιώτη Νικόλαου.

Πεδία:

- *ID(int11)*: PRIMARY_KEY
Μοναδικός σειριακός αριθμός μέτρησης
- *Position (point)*: index (SPATIAL TYPE)
Πεδίο τύπου point που περιέχει τη γεωγραφική θέση της μέτρησης. Το ευρετήριο τύπου SPATIAL χρειάζεται για τη λειτουργία του Geoserver.
- *Downlink (float)*:
Η τιμή της μετρικής ταχύτητας λήψης δεδομένων που προκύπτει από τα αποτελέσματα της μέτρησης.
- *Uplink (float)*:
Η τιμή της μετρικής ταχύτητας αποστολής δεδομένων που προκύπτει από τα αποτελέσματα της μέτρησης.
- *Ping1_IPERF (float)*:
Η τιμή της μετρικής χρόνου ping στον προσωπικό διακομιστή 83.212.113.47, που προκύπτει από τα αποτελέσματα της μέτρησης.
- *Ping2_DN (float)*:
Η τιμή της μετρικής χρόνου ping σε διακομιστή της Google, που προκύπτει από τα αποτελέσματα της μέτρησης.

- *Network_Type (varchar(30))*: index
Ο τύπος δικτύου (wifi ή mobile) που πραγματοποιήθηκε η μέτρηση.
- *Measurement_Time (datetime)*:
Ο χρόνος καταγραφής της μέτρησης.
- *latitude (float)*:
Το γεωγραφικό πλάτος του τόπου όπου έγινε η μέτρηση.
- *longitude (float)*:
Το γεωγραφικό μήκος του τόπου όπου έγινε η μέτρηση.
- *os (varchar(30))*:
Ο τύπος (Android για τη συγκεκριμένη υλοποίηση) και η έκδοση του λειτουργικού συστήματος της συσκευής.
- *Android_id (varchar(128))*: index
Ο μοναδικός κωδικός που αποτελεί το αναγνωριστικό της συσκευής. Μπορεί να είναι το IMEI της συσκευής ή το android.os.Build.SERIAL
- *accuracy (decimal(5,1))*:
Η τιμή της ακρίβειας της μέτρησης από την Android συσκευή σε m.
- *camp_id (int(11))*: index
Ο κωδικός αριθμός που αποτελεί αναγνωριστικό της καμπάνιας.
- *Speed (decimal(5,2))*:
Η ταχύτητα της συσκευής τη στιγμή που πραγματοποιούσε τη μέτρηση σε m/s.
- *SSID (varchar(70))*:
Το όνομα του wifi δικτύου στο οποίο έγινε η μέτρηση (αν πρόκειται για μέτρηση σε wifi δίκτυο).
- *Operator (varchar(50))*:
Ο πάροχος του δικτύου κινητής τηλεφωνίας, στο οποίο έγινε η μέτρηση (αν πρόκειται για μέτρηση σε mobile δίκτυο).
- *wifiLinkSpeed (smallint(4))*:
Η ταχύτητα συγχρονισμού με το WiFi access point

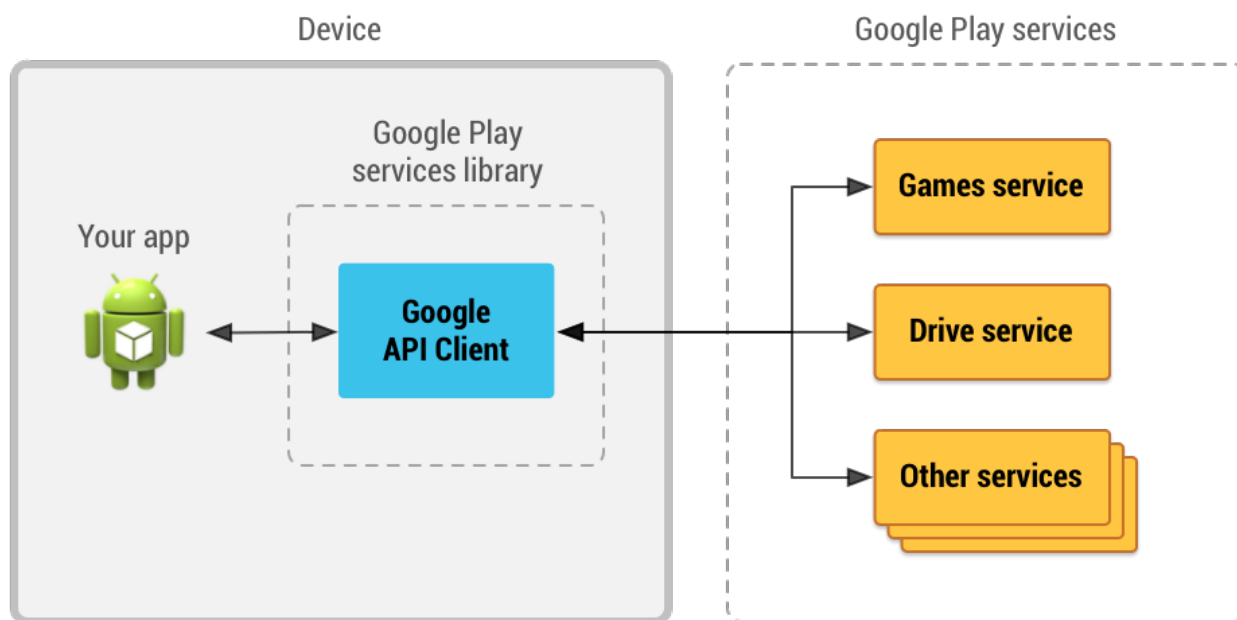


Εικόνα 6.28: Οι πίνακες της βάσης δεδομένων *huaLocation*

6.6 Βιβλιοθήκες

6.6.1 Google Play Services

Η Google Play Services είναι μία βιβλιοθήκη, η οποία προσφέρει πλειάδα χαρακτηριστικών και υπηρεσιών της Google σε Android εφαρμογές, όπως Google Maps, FusedLocationApi, Google+ και πολλές άλλες. Επίσης, αποτελεί προαπαιτούμενο εάν θέλουμε να δημοσιοποιήσουμε την εφαρμογή μας σε μορφή APK στα Google Play Stores, ενώ για τις εφαρμογές σε πλατφόρμα Android 2.3 και άνω, υποστηρίζεται η αυτόματη ενημέρωση [38]. Η πρόσβαση στις υπηρεσίες που προσφέρει το Google Play επιτυγχάνεται με την υλοποίηση της κλάσης `GoogleApiClient` (Εικόνα 6.29). [49]



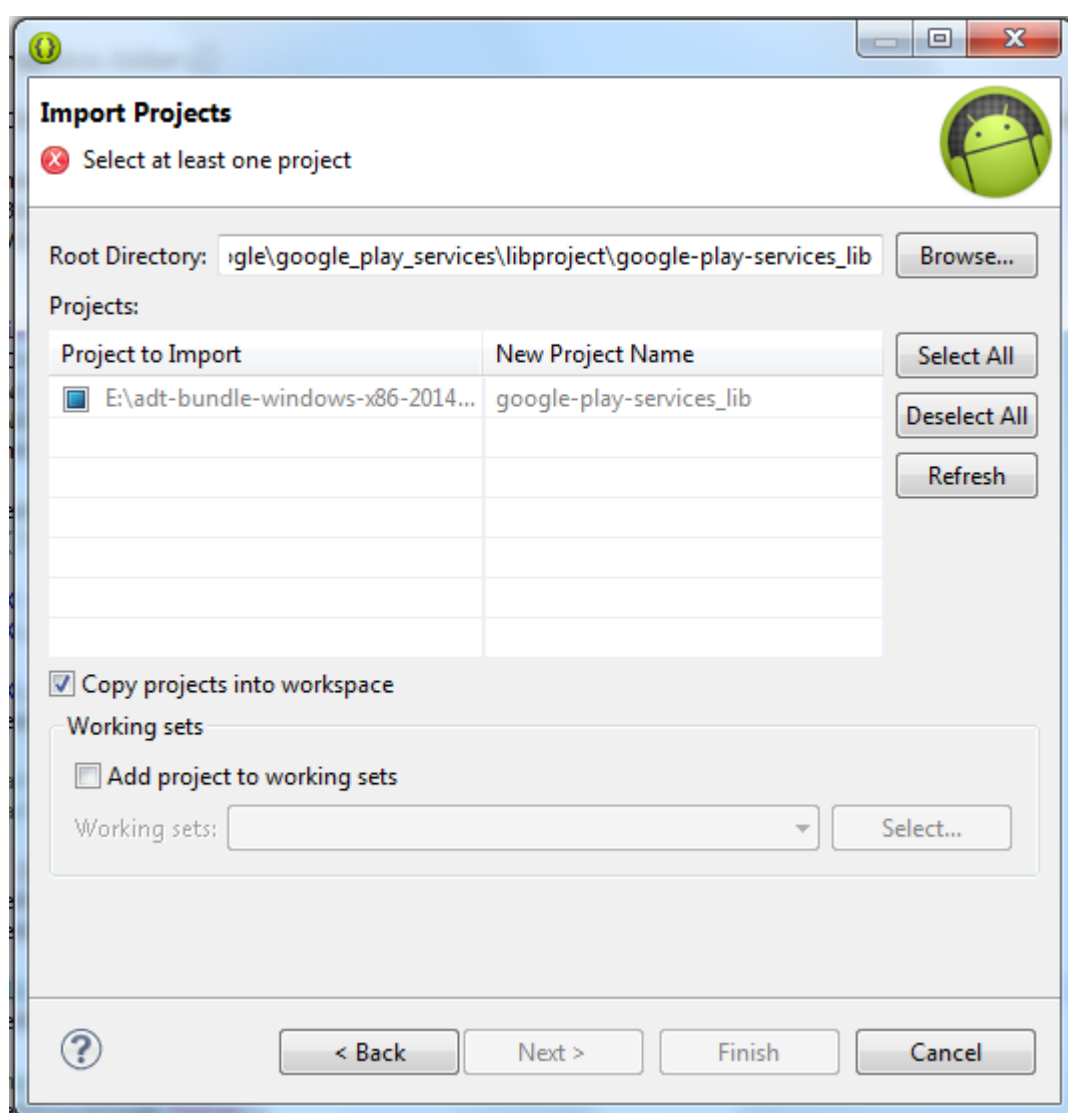
Εικόνα 6.29: Πρόσβαση στις υπηρεσίες Google Play μέσω Google API Client

Για να μπορέσουμε να χρησιμοποιήσουμε τα Google Play Services APIs της βιβλιοθήκης, πρέπει πρώτα να την εγκαταστήσουμε στο Android project της εφαρμογής μας.

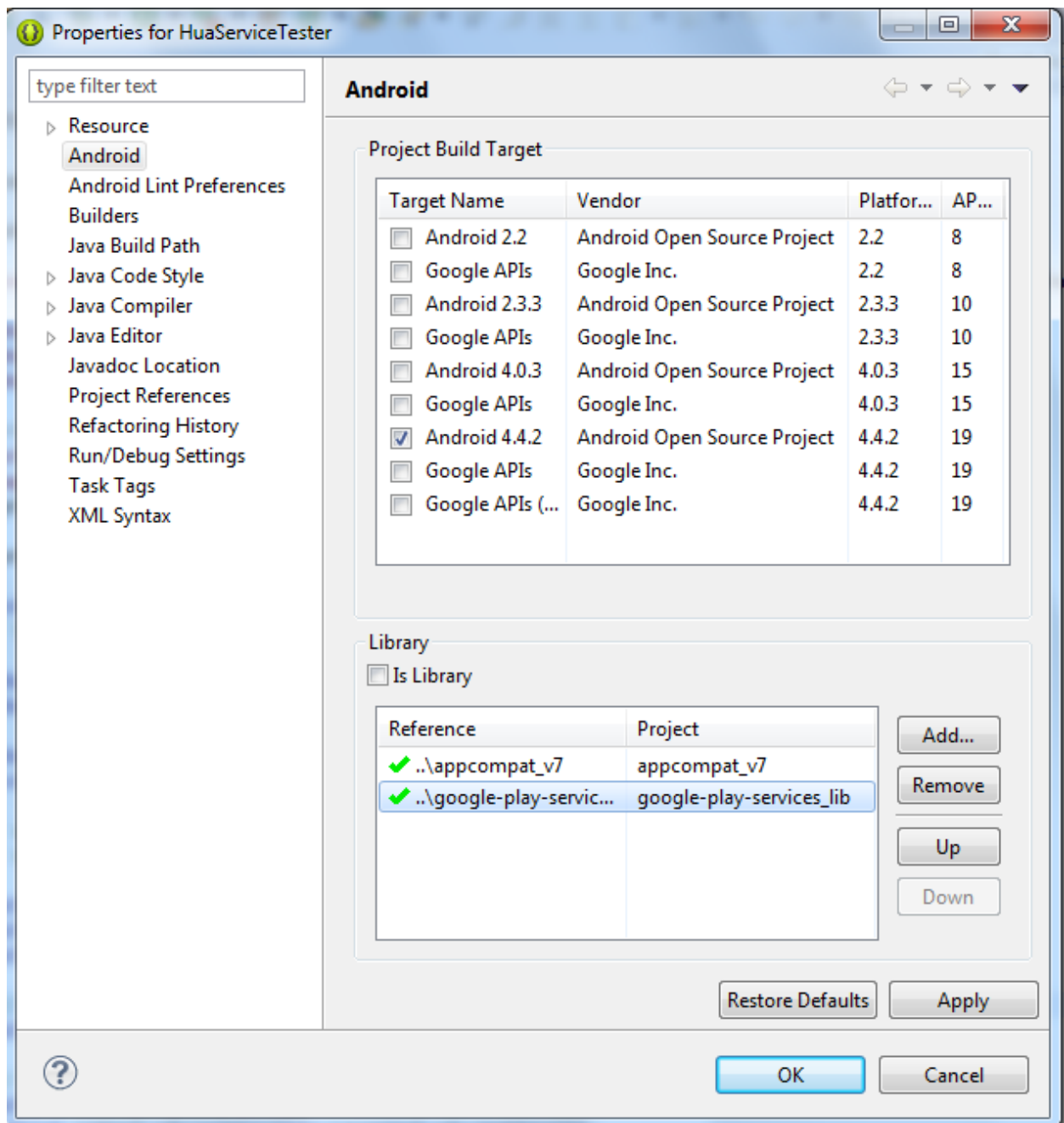
Ανοίγουμε τον SDK Manager και στον φάκελο *extras*, τσεκάρουμε την επιλογή *Google Play services* και πατάμε το κουμπί *Install* (Εικόνα 6.11). Με αυτή την ενέργεια γίνεται μεταφόρτωση της βιβλιοθήκης στο φάκελο `<android-sdk-φάκελος>/extras/google/google_play_services`. Στη συνέχεια, ανοίγουμε το Eclipse IDE με το ADT plugin εγκατεστημένο, ώστε να εισάγουμε τη βιβλιοθήκη στο workspace που δουλεύουμε (Εικόνα 6.30). Στο μενού επιλέγουμε *File->Import-*

>Android->Existing Android Code Into Workspace και στη συνέχεια πατάμε Next. Μέσα στο φάκελο `google_play_services` επιλέγουμε τη βιβλιοθήκη `libproject/google_play_services_lib`, τσεκάρουμε το project που εμφανίζεται και πατάμε Finish.

Στο workspace, έχει δημιουργηθεί το project της βιβλιοθήκης Google Play services και πλέον μπορούμε να το χρησιμοποιήσουμε ως βιβλιοθήκη σε οποιαδήποτε Android εφαρμογή. Πατώντας δεξί κλικ πάνω στο όνομα του project επιλέγουμε *Properties->Android* (Εικόνα 6.31). Στο πλαίσιο *Library* πατάμε το κουμπί Add και επιλέγουμε το `google-play-services_lib`. Η βιβλιοθήκη Google Play services έχει πλέον ενσωματωθεί στην εφαρμογή μας και μπορούμε να χρησιμοποιήσουμε το αντίστοιχο API.



Εικόνα 6.30: Εισαγωγή βιβλιοθήκης Google Play services στο workspace του Eclipse IDE



Εικόνα 6.31: Ενσωμάτωση βιβλιοθήκης google-play-services σε Android project

Τέλος, η βιβλιοθήκη και η έκδοσή της πρέπει να δηλωθεί στο αρχείο `AndroidManifest.xml` της εφαρμογής.

```
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

6.6.2 Google Maps V2

Για την αναπαράσταση των μετρήσεων της εφαρμογής σε ηλεκτρονικούς γεωγραφικούς χάρτες χρησιμοποιήθηκε η βιβλιοθήκη Google Maps V2 για Android εφαρμογές, η οποία διανέμεται δωρεάν από την Google. (πηγή [50])

Για να μπορέσουμε να χρησιμοποιήσουμε τη βιβλιοθήκη θα πρέπει πρώτα να έχουμε εγκαταστήσει την Google Play services, σύμφωνα με τη διαδικασία που παρουσιάστηκε στην προηγούμενη παράγραφο. Επίσης, για να χρησιμοποιήσουμε το API της βιβλιοθήκης πρέπει να προσθέσουμε στην εφαρμογή ένα Maps API κλειδί (key) από την κονσόλα προγραμματιστών της Google (*Google Developers console*). Στο κλειδί πρέπει να δηλώσουμε την εφαρμογή που θα έχει πρόσβαση στο Google Maps API. Το αναγνωριστικό της εφαρμογής είναι το ψηφιακό πιστοποιητικό γνωστό ως **SHA-1 fingerprint**, από τον αλγόριθμο κρυπτογράφησης SHA-1, μαζί με το package της εφαρμογής μας. Το πιστοποιητικό SHA-1 fingerprint είναι μοναδικό και μπορούμε να το δούμε, αν βρισκόμαστε σε *debug mode*, στο Eclipse Android SDK επιλέγοντας *Window->Preferences->Android->Build* (Εικόνα 6.32)

Η διαδικασία προσθήκης του Maps API κλειδιού στην εφαρμογή μας είναι παρόμοια με αυτή που περιγράφηκε στην παράγραφο 6.3.3 με την εγγραφή της υπηρεσίας GCM. Πρέπει να έχουμε δημιουργήσει στην Google λογαριασμό και ένα τουλάχιστον API Project. Συνδεόμαστε μέσω της προγραμματιστικής κονσόλας στο API Project και ενεργοποιούμε το **Google Maps Android API**. Στη συνέχεια, στο αριστερό μενού επιλέγουμε *Credentials* και πατάμε το κουμπί *Create new Key*. Στο παράθυρο που ανοίγει επιλέγουμε να δημιουργήσουμε νέο *Android key* και εμφανίζεται το παράθυρο της Εικόνας 6.33. Στο πλαίσιο κειμένου εισάγουμε το ψηφιακό πιστοποιητικό SHA-1 fingerprint, τον χαρακτήρα “:” και το όνομα του package της εφαρμογής μας, δηλαδή:

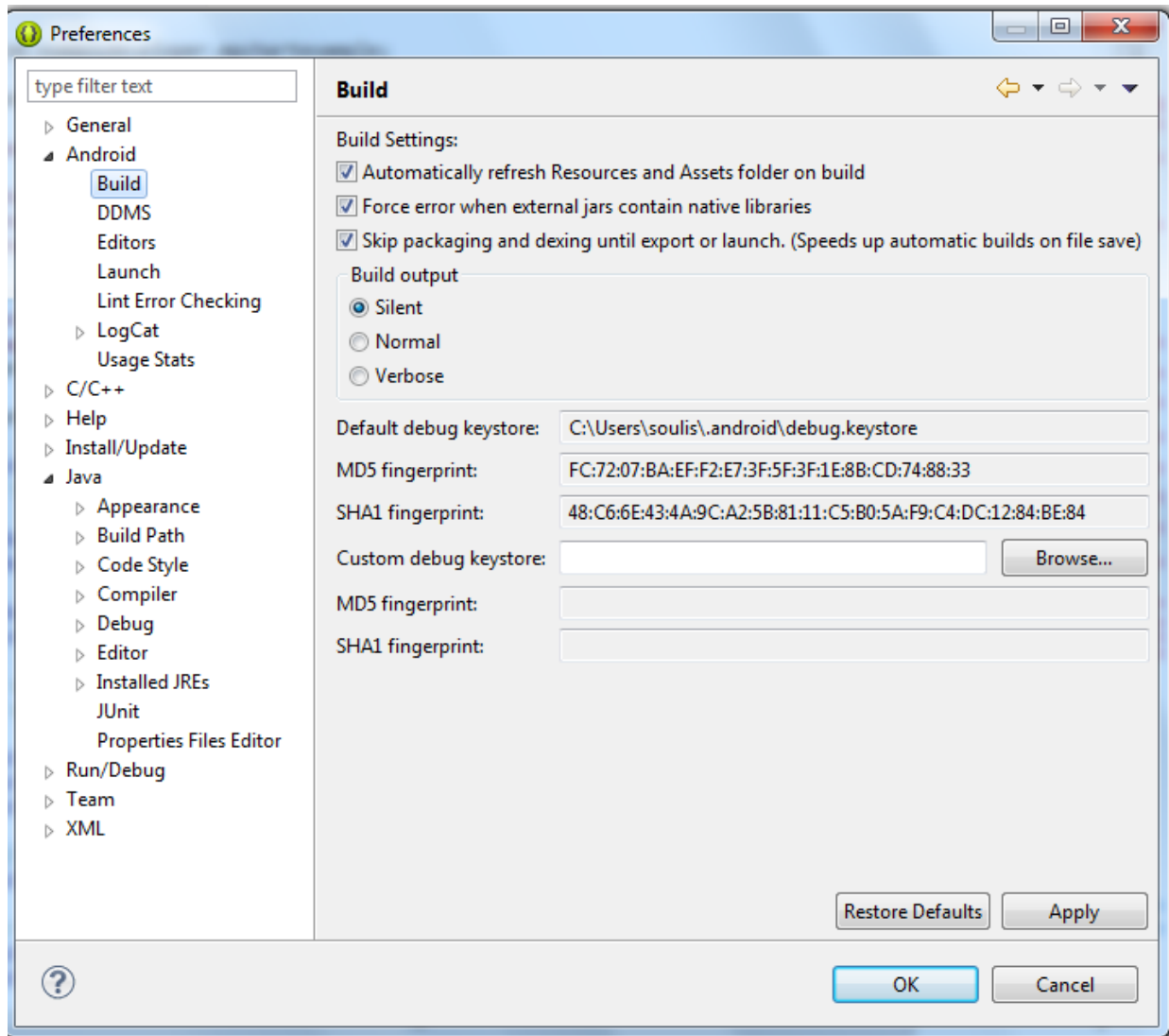
```
48:C6:6E:43:4A:9C:A2:5B:81:11:C5:B0:5A:F9:C4:DC:12:84:BE:84;gr.soulis.huaserv  
icetester2
```

Με την καταχώρηση του αναγνωριστικού το API κλειδί λαμβάνει μία τιμή, πχ

```
AIzaSyAYDlBS0MiT4Nb01Ezo09kf8v04ZCNKk2k
```

Αυτή την τιμή πρέπει να δηλώσουμε στο αρχείο *AndroidManifest.xml* της εφαρμογής μας.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyAYDLBSMiT4Nb01Ezo09kf8v04ZCNKk2k" />
```



Εικόνα 6.32: Το ψηφιακό πιστοποιητικό SHA-1 fingerprint

Create an Android key and configure allowed Android applications

This key can be deployed in your Android application.

API requests are sent directly to Google from your client Android device. Google verifies that each request originates from an Android application that matches one of the certificate SHA1 fingerprints and package names listed below. You can discover the SHA1 fingerprint of your developer certificate using the following command:

```
keytool -list -v -keystore mystore.keystore
```

[Learn more](#)

Accept requests from an Android application with one of the certificate fingerprints and package names listed below (Optional)

One SHA1 certificate fingerprint and package name (separated by a semicolon) per line. Example:

```
45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example
```

Or if you leave this blank, requests will be accepted from any Android application. Be sure to add SHA1 certificates before using this key in production.

```
48:C6:6E:43:4A:9C:A2:5B:81:11:C5:B0:5A:F9:C4:DC:12:84:BE:84;gr.soulis.huaservicetester2
```

Create

Cancel

Εικόνα 6.33: Δημιουργία Google API κλειδιού για πρόσβαση στο Google Maps API

Επιπλέον, για να χρησιμοποιήσουμε το Google Maps API, χρειάζεται να αποκτήσουμε και κάποια δικαιώματα, τα οποία δηλώνονται στο αρχείο `AndroidManifest.xml` της εφαρμογής μας.

➤ *`android.permission.INTERNET`*

Χρειάζεται διαδικτυακή πρόσβαση για την επικοινωνία με τους διακομιστές της Google.

➤ *`android.permission.ACCESS_NETWORK_STATE`*

Επιτρέπει τον έλεγχο της κατάστασης δικτύου, ώστε να γνωρίζει η εφαρμογή πότε είναι συνδεδεμένη διαδικτυακά.

➤ *`android.permission.WRITE_EXTERNAL_STORAGE`*

Επιτρέπει στην εφαρμογή να αποθηκεύει δεδομένα, όπως εικόνες από τους χάρτες σε εξωτερική συσκευή αποθήκευσης δεδομένων.

Τα παρακάτω δικαιώματα μπορούν να παραλειφθούν (δεν είναι υποχρεωτικά), αλλά προτείνεται να δηλωθούν:

➤ *android.permission.ACCESS_COARSE_LOCATION*

Επιτρέπει στο API να υπολογίζει τη γεωγραφική θέση της συσκευής μέσω δικτύου WiFi ή δικτύου κινητής τηλεφωνίας.

➤ *android.permission.ACCESS_FINE_LOCATION*

Επιτρέπει στο API να υπολογίζει τη γεωγραφική θέση της συσκευής μέσω GPS (Global Positioning System) με μεγάλη ακρίβεια.

```
<!-- Use permission for Google Maps (ref: https://developers.google.com/maps/documentation/android/start#getting\_the\_google\_maps\_android\_api\_v2) -->
<uses-permission
android:name="gr.soulis.huaservicetester2.permission.MAPS_RECEIVE" />
<!-- Permission for accessing the Internet -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- Permission for writing to external storage -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!-- Use permission for Google Maps -->
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!-- Permissions for user's location -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Με την ολοκλήρωση των παραπάνω διαδικασιών, η εφαρμογή μας αποκτά πρόσβαση στο API της βιβλιοθήκης Google Maps.

6.6.3 MPAndroid Chart

Η συγκεκριμένη βιβλιοθήκη έχει δημιουργηθεί από τον Philipp Jahoda και χρησιμοποιείται στην Android εφαρμογή για την αναπαράσταση στατιστικών στοιχείων σχετικά με τις μετρήσεις των ασύρματων δικτύων και τις καμπάνιες στις οποίες γίνονται οι μετρήσεις. Είναι βιβλιοθήκη ανοικτού κώδικα δημοσιοποιημένη υπό τους όρους της άδειας Apache License v2 [51].

Ο λόγος που επέλεξα τη συγκεκριμένη βιβλιοθήκη είναι ότι διαθέτει ποικιλία γραφικών διαγραμμάτων και χαρακτηριστικών με πολλά παραδείγματα υλοποίησης που μας δίνουν τη δυνατότητα να αναπτύξουμε και να αναπαραστήσουμε πολλούς συνδυασμούς μετρικών και μεταβλητών. (πηγή [52])

Τα βασικά χαρακτηριστικά της βιβλιοθήκης είναι τα εξής:

- Συμβατή με λειτουργικό Android 2.2 (API level 8) και άνω
- 7 διαφορετικοί τύποι διαγραμμάτων (line, bar, pie, scatter, bubble, candlestick, radar)
- Συνδυασμοί διαγραμμάτων
- Κλιμάκωση και στους δύο άξονες
- Δυνατότητα διπλού άξονα Y
- Δυνατότητα συρσίματος/διαλογής (dragging/panning)
- Αλλαγή χρώματος με την επιλογή του στοιχείου
- Δυνατότητα επιλογής τιμής για highlight
- Δυνατότητα αποθήκευσης γραφικού διαγράμματος σε εξωτερική μνήμη
- Δυνατότητα αναπαράστασης δεδομένων από αρχείο τύπου txt (text)
- Έτοιμα templates χρωμάτων για τα διαγράμματα
- Δυνατότητα κίνησης (animation)
- Δυνατότητα ολικής αναπροσαρμογής του κώδικα (fully customizable)

Για την εγκατάσταση της βιβλιοθήκης στην Android εφαρμογή πρέπει να μεταφορτώσουμε την τελευταία έκδοση του αρχείου *jar* της βιβλιοθήκης²³ και στη συνέχεια να το αντιγράψουμε μέσα στο φάκελο *lib* του Android project της εφαρμογής μας.

²³ <https://github.com/PhilJay/MPAndroidChart/releases>

6.7 Χρήση GeoServer για Οπτικοποίηση Μετρήσεων

Ο GeoServer είναι ένας εξυπηρετητής ανοικτού κώδικα γραμμένος σε Java και είναι συμβατός με τις κυριότερες τεχνολογίες παροχής γεωχωρικών δεδομένων, όπως αυτές έχουν οριστεί από τον οργανισμό Open Geospatial Consortium (www.opengeospatial.org) [53]. Χρησιμοποιεί open standards για να επιτύχει τη λειτουργικότητά του και είναι εύκολο να ενσωματωθεί στις δημοφιλέστερες υπηρεσίες παροχής χαρτών (όπως Google Maps, OpenStreetMaps και Yahoo Maps). Στην περίπτωση μας, χρησιμοποιείται σε συνδυασμό με το OpenLayers για την αναπαράσταση των γεωχωρικών δεδομένων (δεδομένων τύπου SPATIAL από τη βάση δεδομένων MySQL) σε ηλεκτρονικούς χάρτες της Google. Για να ανακτήσει τα γεωχωρικά δεδομένα, ο Geoserver έχει συνδεθεί με τη βάση δεδομένων MySQL με όνομα *hualocation*, που έχουμε δημιουργήσει, και συγκεκριμένα με τον πίνακα *measurements*.

Ο GeoServer εγκαθίσταται ως plugin μέσα σε web container. Στην περίπτωση μας, έχει εγκατασταθεί η έκδοση 2.6.0 σε Tomcat 7, στη διεύθυνση <http://83.212.113.47:8888/geoserver>. Η διαδικασία που ακολουθήθηκε για την εγκατάστασή του περιγράφεται αναλυτικά στην εργασία του Κορωνιώτη Νικόλαου (εργασία [1]), στην οποία βασιστήκαμε κι εμείς. Επιπρόσθετα, έγινε εγκατάσταση της βιβλιοθήκης WPS από τον Gerti Nurka (εργασία [34]) και δημιουργία νέων styles, για την απεικόνιση των μετρήσεων με τεχνικές σύντηξης δεδομένων, σύμφωνα με τον αλγόριθμο Barnes Surface.

6.8 Ελάχιστες Απαιτήσεις - Περιορισμοί

Για να λειτουργήσει η εφαρμογή σε συσκευές με λειτουργικό Android, χρειάζεται να ικανοποιηθούν κάποιες απαιτήσεις. Όπως έχει αναφερθεί, ο κώδικας που χρησιμοποιήθηκε είναι συμβατός με συσκευές Android API 11 έκδοσης 3.0 (Honeycomb) και άνω. Αυτό οφείλεται στο γεγονός ότι κάποιες βιβλιοθήκες (πχ GCM) και χαρακτηριστικά (MapFregment, PreferenceFregment, ActionBar, Google Play Services API, Switch View) δε λειτουργούν σωστά ή και καθόλου στις παλαιότερες εκδόσεις.

Επίσης, για να λειτουργήσει η εφαρμογή χρειάζεται να έχουν εγκατασταθεί στη συσκευή τα Google Play Services και να είναι ενεργοποιημένο το Google location services, ώστε να μπορεί

να λειτουργήσει το Location API της Google. Επίσης, πρέπει να είναι ενεργοποιημένη η δυνατότητα εντοπισμού δικτύου WiFi ή mobile ώστε να μπορεί η συσκευή να συνδεθεί στο διαδίκτυο. Το διαδίκτυο είναι απαραίτητο για την πραγματοποίηση των μετρήσεων αλλά και την παρουσίαση των στατιστικών, με άλλα λόγια για την λειτουργία της εφαρμογής. Τέλος, είναι καλό να είναι ενεργοποιημένο και το GPS για τον εντοπισμό της γεωγραφικής μας θέσης με μεγάλη ακρίβεια, κάτι που απαιτείται από τις μετρήσεις. Αυτό βέβαια ίσως κοστίζει σε χρόνο, εάν βρισκόμαστε σε κλειστό χώρο, όπου η σύνδεση με δορυφόρους είναι δύσκολη και ο εντοπισμός της θέσης με ακρίβεια μπορεί να αργήσει. Οι μετρήσεις μπορούν να γίνουν και χωρίς GPS και ο εντοπισμός θέσης να γίνει με τη βοήθεια των Google location services, μέσω WiFi ή mobile δικτύου, αλλά σε αυτή την περίπτωση ο εντοπισμός της γεωγραφικής θέσης είναι λιγότερο ακριβής.

Κατά την εγκατάσταση της εφαρμογής χρειάζεται να αποδεχτούμε κάποιους όρους και απαιτήσεις που ζητούν την άδειά μας.

➤ *Your location*

Άδεια για δυνατότητα εντοπισμού της γεωγραφικής μας θέσης

➤ *Network communication*

Άδεια για πλήρη πρόσβαση στο διαδίκτυο και για λήψη δεδομένων, όπως και για προβολή κατάστασης δικτύου.

➤ *Storage*

Άδεια για μεταβολή/διαγραφή δεδομένων σε κάρτα SD

➤ *Phone calls*

Άδεια για πρόσβαση στην κατάσταση του τηλεφώνου και το IMEI της συσκευής

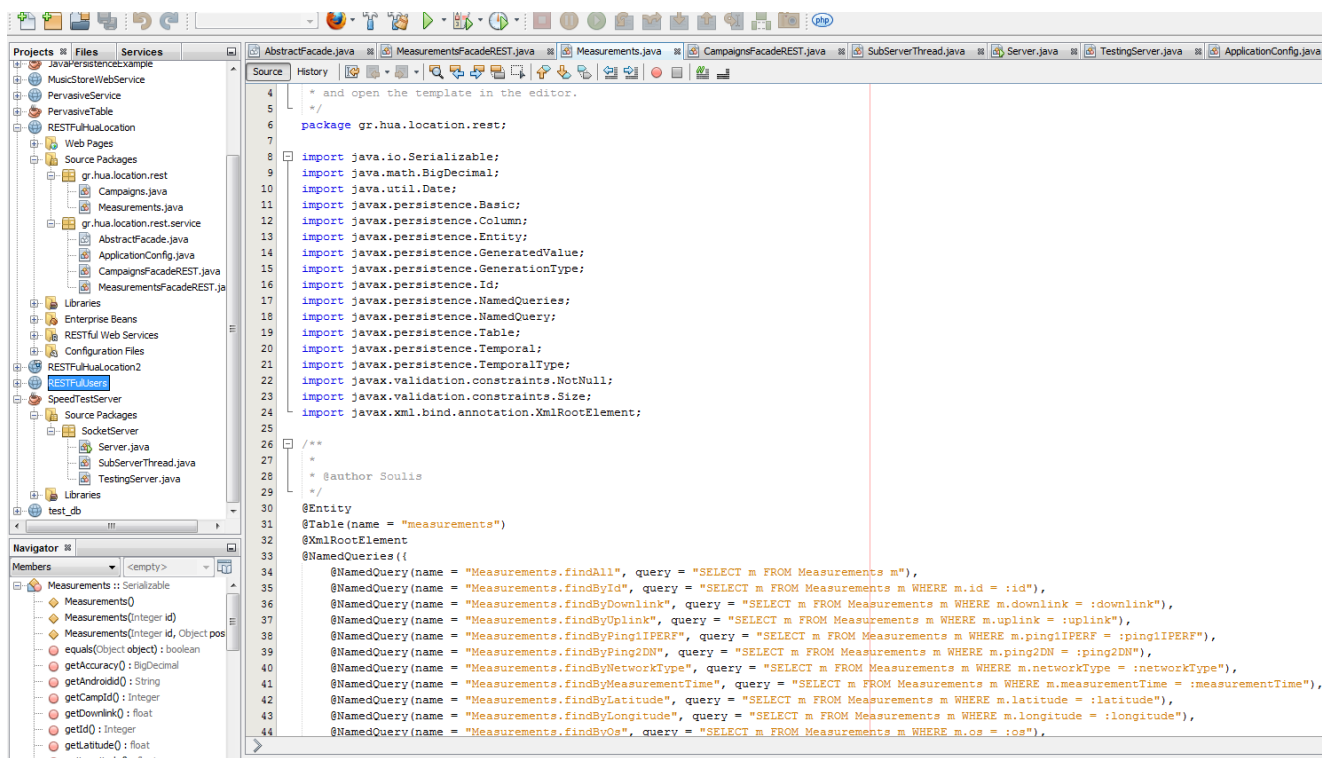
➤ *System tools*

Άδεια για μεταβολή ρυθμίσεων του συστήματος

7 ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΕΦΑΡΜΟΓΗΣ – ΚΩΔΙΚΑΣ

7.1 NetBeans REST Project

Όπως έχω αναφέρει, το κομμάτι της εφαρμογής που αφορά τον application server έχει υλοποιηθεί με τη βοήθεια του εργαλείου NetBeans 8.0.2 (Εικόνα 7.1). Έχουν δημιουργηθεί δύο projects το **SpeedTestServer** και το **RESTFulHuaLocation**.



Εικόνα 7.1: Κώδικας από τα 2 projects της εφαρμογής μας σε NetBeans

7.1.1 SpeedTestServer

Είναι η υλοποίηση ενός SocketServer (Κορωνιώτης πηγή [1]). Το αρχείο *SpeedTestServer.jar* που παράγεται κατά την εκτέλεση του προγράμματος στο NetBeans, εγκαθίσταται στον εξυπηρετητή μας στο φάκελο */usr/share/server/*. Ο εξυπηρετητής εκτελείται σε νέα σύνοδο με

τη βοήθεια του προγράμματος **screen** που εγκαθιστούμε στον application server και «ακούει» (listen) στην πόρτα 9222 συνεχώς. Η ακολουθία των εντολών είναι οι εξής:

```
$ screen
```

```
$ cd /usr/share/server
```

```
$ java -jar ./SpeedTestServer.jar
```

Ο εξυπηρετητής χρησιμοποιείται από την Android εφαρμογή κατά τη διαδικασία πραγματοποίησης των μετρήσεων των τεσσάρων μετρικών απόδοσης ασύρματου δικτύου.

7.1.2 RESTfulHuaLocation

Είναι η υλοποίηση της RESTful υπηρεσίας στον application server που έχει ως σκοπό την αποστολή δεδομένων σε μορφή JSON, από τη βάση δεδομένων MySQL *huaLocation* στην Android εφαρμογή.

Κλάση Campaigns.java

Entity κλάση που δημιουργείται για σύνδεση με τον πίνακα *campaigns* της βάσης δεδομένων *huaLocation*. Έχουν δημιουργηθεί αναζητήσεις στον πίνακα (NamedQueries), ώστε να ανακτώνται τα αντίστοιχα δεδομένα με βάση τα HTTP GET αιτήματα από την Android εφαρμογή.

Κλάση Measurements.java

Entity κλάση που δημιουργείται για σύνδεση με τον πίνακα *measurements* της βάσης δεδομένων *huaLocation*. Έχουν δημιουργηθεί αναζητήσεις στον πίνακα (NamedQueries), ώστε να ανακτώνται τα αντίστοιχα δεδομένα με βάση τα HTTP GET αιτήματα από την Android εφαρμογή.

Κλάση AbstractFacade.java

Υπερκλάση στην οποία δηλώνονται οι μέθοδοι, βάσει των οποίων γίνονται οι αναζητήσεις (NamedQueries ή NativeQueries) στη βάση δεδομένων. Τα NamedQueries ανακτώνται από την αντίστοιχη entity κλάση, ενώ με τα NativeQueries δηλώνουμε την αναζήτηση δυναμικά.

Παράδειγμα Κώδικα:

```
package gr.hua.location.rest.service;

import java.util.List;
import javax.persistence.EntityManager;
/**
 *
 * @author Soulis
 */
public abstract class AbstractFacade<T> {
    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }

    protected abstract EntityManager getEntityManager();
    /// SOULIS REST WSs -----
    // // device's amount of mobile measurements in current month
    public int countMetrics(String androidID) {
        javax.persistence.Query q = getEntityManager()
            .createNativeQuery("SELECT COUNT(*) FROM measurements WHERE
Android_id='"+androidID+"' AND Network_Type<>'wifi' AND
month(Measurement_Time)=month(CURRENT_TIMESTAMP) AND
year(Measurement_Time)=year(CURRENT_TIMESTAMP)");
        return ((Long) q.getSingleResult()).intValue();
    }

    // retrieve active campaigns
    public List<T> getActive(short active) {
        return
getEntityManager().createNamedQuery("Campaigns.findByActive").setParameter("active",
active).getResultList();
    }
}
```

Κλάση CampaignsFacadeREST.java

Υποκλάση της *AbstractFacade* μέσω της οποίας γίνεται η αντιστοίχιση του HTTP GET αιτήματος σε Java μέθοδο. Τα αιτήματα αυτά αφορούν ερωτήματα στον πίνακα *campaigns*. Επίσης, δηλώνεται η μορφή της απάντησης στο αίτημα (plain text/xml/json).

Παράδειγμα Κώδικα:

```
/**
 *
 * @author Soulis
 */
@Stateless
@Path("gr.hua.location.rest.campaigns")
public class CampaignsFacadeREST extends AbstractFacade<Campaigns> {
    @PersistenceContext(unitName = "RESTFulHuaLocationPU")
    private EntityManager em;

    public CampaignsFacadeREST() {
        super(Campaigns.class);
    }
    @GET
    @Path("getActives/{active}")
    @Produces("application/json")
    public List<Campaigns> getActive(@PathParam("active") short active) {
        return super.getActive(active);
    }

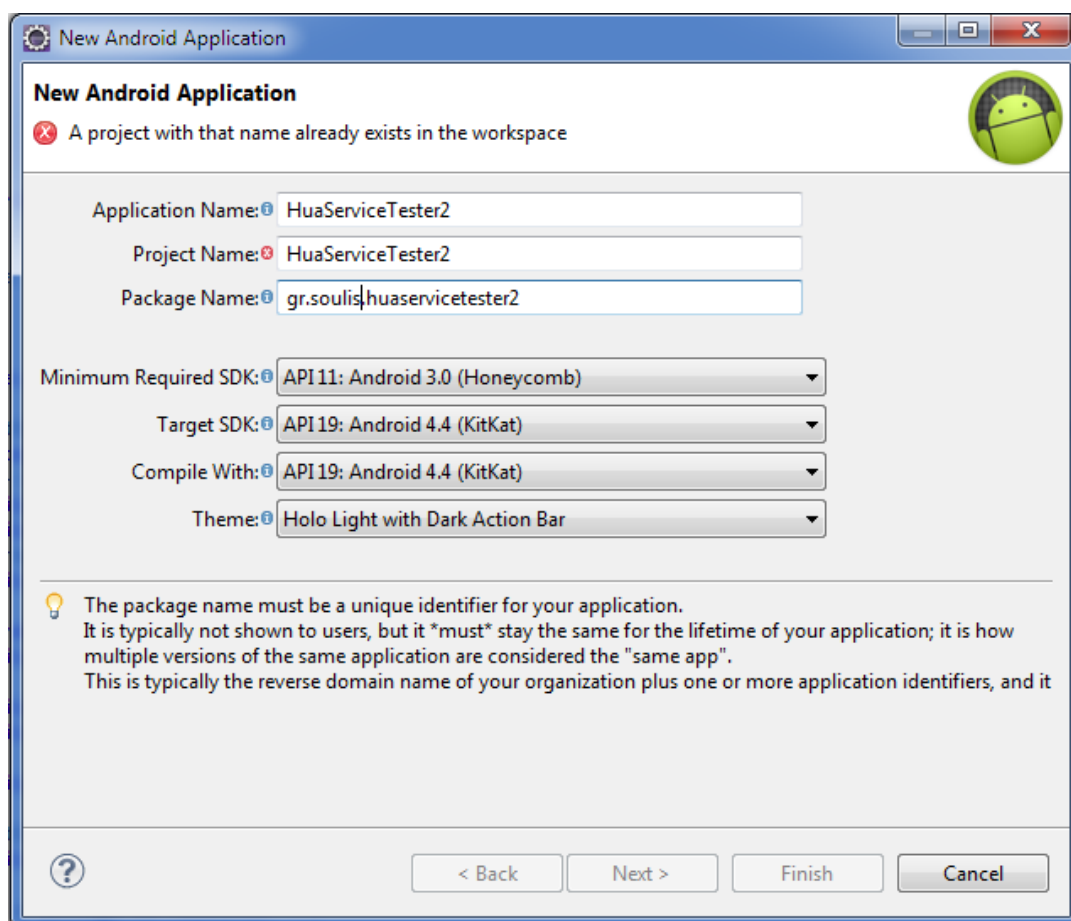
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
}
```

Κλάση MeasurementsFacadeREST.java

Υποκλάση της *AbstractFacade* μέσω της οποίας γίνεται η αντιστοίχιση του HTTP GET αιτήματος σε Java μέθοδο. Τα αιτήματα αυτά αφορούν ερωτήματα στον πίνακα *measurements*. Επίσης, δηλώνεται η μορφή της απάντησης στο αίτημα (plain text/xml/json). Ο κώδικας είναι παρόμοιος λογικής με της κλάσης *CampaignsFacadeREST*.

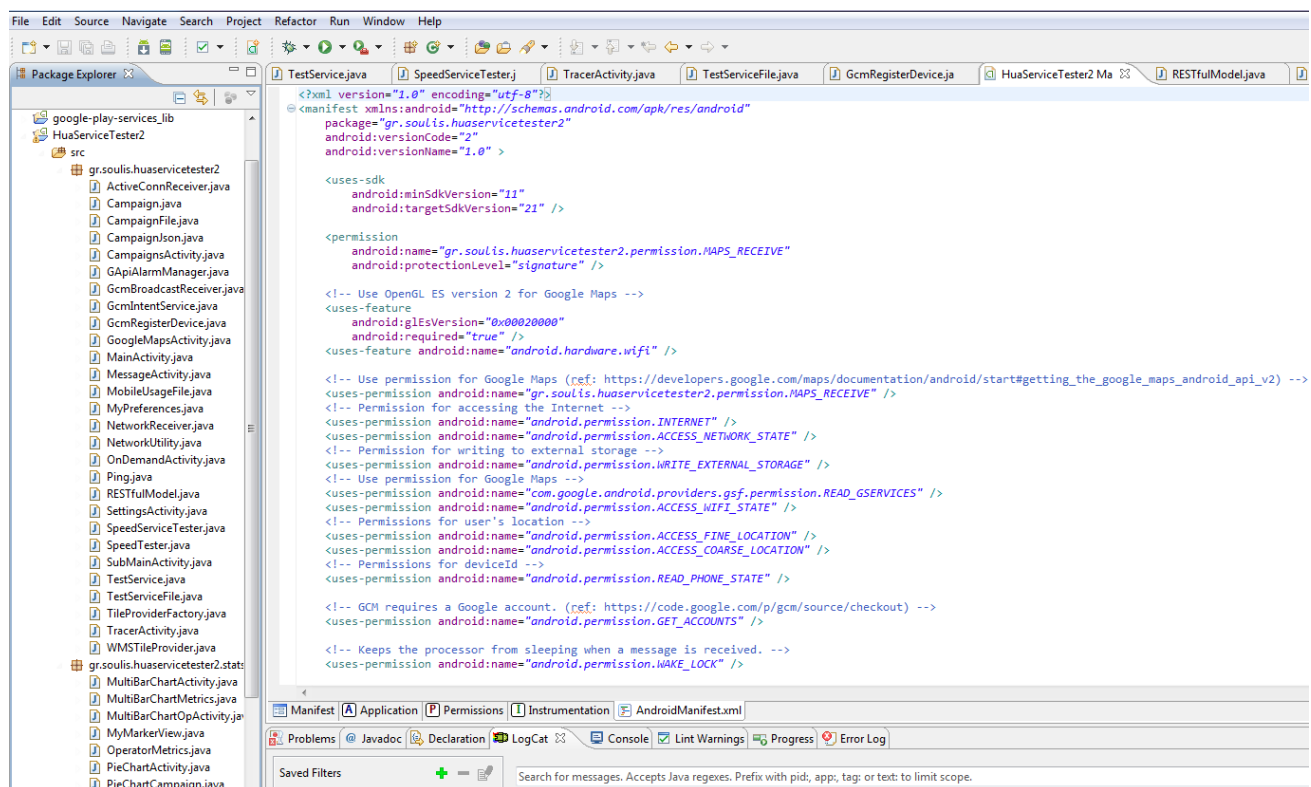
7.2 Android Project

Όπως έχω αναφέρει, η εφαρμογή – πελάτης έχει υλοποιηθεί με τη βοήθεια του εργαλείου Eclipse IDE με το ADT plugin. Για τη δημιουργία του project ανοίγουμε το Eclipse και επιλέγουμε στο μενού *File->New->Android Application project*. Στο παράθυρο που εμφανίζεται (Εικόνα 7.2), εισάγουμε το όνομα της εφαρμογής (**HuaServiceTester2**), το όνομα του package μέσα στο οποίο θα αναπτυχθεί η εφαρμογή (**gr.soulis.huaservicetester2**) και ορίζουμε την μικρότερη Android έκδοση συμβατότητας της εφαρμογής, την έκδοση – στόχος και την έκδοση βάσει της οποίας θα γίνει η μεταγλώττιση του κώδικα.



Εικόνα 7.2: Δημιουργία του project της εφαρμογής μας σε Eclipse

Επίσης, όπως φαίνεται και στην Εικόνα 6.31, στο project της εφαρμογής μας, πρέπει να ενσωματώσουμε τις βιβλιοθήκες *google-play-services_lib* (παρ. 6.6.1) και *appcompat_v7* (για συμβατότητα με παλαιότερες εκδόσεις) και να επιλέξουμε την έκδοση – στόχο (v. 4.4.2).



Εικόνα 7.3: Κώδικας από το project της εφαρμογής μας σε Eclipse

Το περιβάλλον ανάπτυξης του λογισμικού της εφαρμογής φαίνεται στην Εικόνα 7.3. Το αρχείο με πρωτεύοντα ρόλο στην υλοποίηση είναι το *AndroidManifest.xml*, στο οποίο δηλώνονται όλα τα αρχεία και οι άδειες που απαιτούνται για να λειτουργήσουν οι βιβλιοθήκες και τα συστατικά του Android. Ο κώδικας του αρχείου παρουσιάζεται στο Παράρτημα Α. Επίσης, τα styles που χρησιμοποιούνται στην εφαρμογή για τη διαμόρφωση του γραφικού περιβάλλοντος, δηλαδή των οθονών της εφαρμογής, έχουν βασιστεί στην ιστοσελίδα <http://www.mindfreakerstuff.com> [54].

Η εφαρμογή ξεκινά τη λειτουργία της με το άνοιγμα του *activity launcher* που δηλώνεται να είναι το αρχείο *MainActivity.java*, το οποίο παρουσιάζεται στη συνέχεια.

7.2.1 Έλεγχοι Λειτουργίας

Η εφαρμογή για να λειτουργήσει χρειάζεται να έχει πρόσβαση στο διαδίκτυο. Επίσης, πρέπει να είναι εγκατεστημένα τα Google Play services. Μία από τις πιο σημαντικές λειτουργίες της εφαρμογής που πραγματοποιούνται συνεχώς από τη στιγμή που η εφαρμογή εγκαθίσταται στη συσκευή είναι ο έλεγχος δικτύου. Εάν δηλαδή η συσκευή είναι συνδεδεμένη σε δίκτυο WiFi ή mobile ώστε να έχουμε πρόσβαση στο διαδίκτυο.

TracerActivity.java

Είναι activity (χωρίς δική του οθόνη) που λειτουργεί ως υπερκλάση άλλων activities για να κληρονομήσουν τα χαρακτηριστικά του. Εδώ δηλώνονται όλες οι σταθερές του προγράμματος, όπως πχ *SERVER_IP* (η IP του application server), *SERVER_PORT*, *ANDROID_ID* κλπ, φορτώνονται οι προτιμήσεις του χρήστη, όπως πχ σε τί είδους δίκτυο (Wi-Fi, mobile ή και τα δύο) επιθυμεί να γίνονται οι μετρήσεις, η συχνότητα μέτρησης κλπ. Επίσης, ελέγχεται εάν η υπηρεσία Google Play είναι διαθέσιμη και εάν δεν είναι εμφανίζεται ανάλογο μήνυμα. Εδώ γίνεται δυναμικά και η δήλωση (register) του receiver *NetworkReceiver*, ο οποίος ελέγχει εάν υπάρχει πρόσβαση στο διαδίκτυο και σε περίπτωση που δεν υπάρχει εμφανίζει ανάλογο μήνυμα (μέσω του *MessageActivity*).

MainActivity.java

Είναι το Activity με το οποίο ξεκινά η εφαρμογή (Activity Launcher). Με το άνοιγμα της εφαρμογής, εμφανίζεται το αρχικό μενού επιλογών, όπως φαίνεται και στην Εικόνα 6.1. Το συγκεκριμένο activity αποτελεί υποκλάση της *TracerActivity.java*, η οποία πραγματοποιεί όλους του ελέγχους για την ομαλή λειτουργία της εφαρμογής (πχ έλεγχος δικτύου) και περιέχει τις δηλώσεις σταθερών της εφαρμογής. Επιπρόσθετα, η *MainActivity* ελέγχει εάν έχει γίνει εγγραφή στην υπηρεσία GCM, ανακτώντας το *Register_Id* και εάν έχει φορτωθεί το αρχείο με τις καμπάνιες. Επίσης, ελέγχει εάν η υπηρεσία παρασκηνίου για τις μετρήσεις είναι σε λειτουργία, ώστε να εμφανίσει στο αντίστοιχο κουμπί το σωστό λεκτικό (Start/Stop Service).

NetworkReceiver.java

Είναι ο receiver που ελέγχει συνεχώς κατά τη λειτουργία της εφαρμογής την κατάσταση δικτύου. Η δήλωσή του γίνεται δυναμικά στην υπερκλάση *TracerActivity.java* για τις παρακάτω ενέργειες:

- `android.net.conn.CONNECTIVITY_CHANGE`
- `android.location.PROVIDERS_CHANGED`
- `android.net.wifi.WIFI_STATE_CHANGED`

Σε περίπτωση έλλειψης σύνδεσης με το διαδίκτυο (WiFi ή mobile) καλεί μέσω *Intent* το *MessageActivity* για εμφάνιση ανάλογου μηνύματος στο χρήστη.

7.2.2 Καμπάνιες

Οι καμπάνιες είναι γεωγραφικές περιοχές, οι οποίες ορίζονται από τους χρήστες μίας διαδικτυακής πλατφόρμας που έχει αναπτυχθεί από τον Gerti Nurka, στα πλαίσια της μεταπτυχιακής του εργασίας [34]. Οι περιοχές αυτές έχουν συγκεκριμένα χαρακτηριστικά, τα οποία πρέπει να ικανοποιούνται ώστε να πραγματοποιηθούν μετρήσεις. Τα χαρακτηριστικά αυτά σώζονται στον πίνακα *campaigns* της βάσης δεδομένων *huaLocation* και έχουν αναλυθεί στην παράγραφο 6.5.2. Ο ρόλος των καμπανιών είναι να θέτουν χωροχρονικά όρια εντός των οποίων θα γίνονται οι μετρήσεις από την υπηρεσία παρασκηνίου.

Η Android εφαρμογή ενημερώνεται με τις ενεργές καμπάνιες μέσω της υπηρεσίας GCM, η οποία στέλνει HTTP μηνύματα σε μορφή JSON, σε όλες τις εγγεγραμμένες συσκευές στην υπηρεσία GCM. Ένα δείγμα ενός τέτοιου μηνύματος είναι ως εξής:

JSON Campaigns

```
{"campaigns": [  
  {  
    "id": "1",  
    "create_date": "2014-11-15 11:34:22",  
    "end_date": "2015-11-25 ",  
    "latitude": 38.0141,  
    "longtitude": 23.6335,  
    "network_type": "wifi",
```

```

        "accuracy":150,
        "radius":100,
        "velocity":10
    },
    {
        "id":"2",
        "create_date":"2014-11-15 11:34:22",
        "end_date":"2015-11-26 ",
        "latitude":31.0141,
        "longtitude": 27.6335,
        "network_type":"mobile",
        "accuracy":150,
        "radius":1500,
        "velocity":10
    }
] }

```

Η Android εφαρμογή αποθηκεύει τις ενεργές καμπάνιες που λαμβάνει από την υπηρεσία GCM σε αρχείο (*CampaignFile.txt*). Αυτό συμβαίνει όποτε εισάγονται νέες καμπάνιες στη διαδικτυακή πλατφόρμα που έχει αναπτυχθεί από τον Gerti Nurka. Όμως, μετά την εγκατάσταση της εφαρμογής, την πρώτη φορά που θα εκτελεστεί δεν θα έχει ενημερωθεί από την υπηρεσία GCM με τις ενεργές καμπάνιες. Ο έλεγχος αυτός γίνεται στην αρχική οθόνη της εφαρμογής, στο activity *MainActivity*. Εάν το αρχείο *CampaignFile.txt* δεν υπάρχει, γίνεται ανάκτηση των ενεργών καμπανιών μέσω RESTful service και συγκεκριμένα της μεθόδου *getCampaigns()*, που αντικαθιστά την υπηρεσία GCM, μόνο για αυτή την περίπτωση. Επίσης, υπάρχει η δυνατότητα ανάκτησης των καμπανιών από την οθόνη 6.3, πατώντας το κουμπί **Retrieve Active Campaigns**.

Κώδικας:

```

try {

    String msgCampFile = CampaignFile.readCampaignFile(getApplicationContext());

} catch (IOException e1) {
    // IF Campaign file not found, create Campaign File from RESTful Service
    RESTfulModel restModel = new RESTfulModel(getApplicationContext());
    restModel.getCampaigns((short) 1); // get active campaigns and write to file -
--
}

```

Campaign.java

Κλάση που αναπαριστά το αντικείμενο campaign με όλα τα χαρακτηριστικά του με getters και setters.

CampaignFile.java

Κλάση που χρησιμοποιείται για εγγραφή των καμπανιών σε αρχείο και ανάγνωσή τους από αυτό.

CampaignsActivity.java

Οθόνη παρουσίασης των ενεργών καμπανιών, όπως φαίνεται και στην Εικόνα 6.3, με δυνατότητα ανάκτησης των νέων ενεργών καμπανιών, μέσω RESTful service.

GcmRegisterDevice.java

Καλείται στην αρχική οθόνη της εφαρμογής (*MainActivity*) για να ανακτήσει το *registration_id* της συσκευής στην GCM υπηρεσία. Εάν το id δεν υπάρχει (δεν έχει αποθηκευτεί στο *SharedPreferences* της συσκευής) θεωρεί ότι η συσκευή δεν είναι εγγεγραμμένη στην υπηρεσία και στέλνει μήνυμα εγγραφής στην GCM υπηρεσία, μέσω του *GoogleCloudMessaging* API. Ταυτόχρονα, με τη μέθοδο *AsyncTask*, στέλνει το *registration_id* μέσω ενός HTTP POST στον application server και συγκεκριμένα στο PHP script **c2dm_reg.php**, το οποίο αναλαμβάνει να αποθηκεύσει το *registration_id* στον πίνακα *c2dm_reg_devices*.

Κώδικας στην MainActivity.java:

```
/// REGISTER DEVICE TO GCM -----
GcmRegisterDevice regDevice = new GcmRegisterDevice(getBaseContext());

try{
    regid = regDevice.getRegistrationId();
}
catch(RuntimeException ex){
    Toast.makeText(getApplicationContext(), ex.getMessage(),
    Toast.LENGTH_SHORT).show();
}
```

```

        if (regid.isEmpty()) {
            regDevice.registerInBackground();
            Toast.makeText(getApplicationContext(), "The Device is REGISTERED to GCM!",
            Toast.LENGTH_SHORT).show();
        }
    }

```

GcmBroadcastReceiver.java

Είναι ο receiver ο οποίος ανιχνεύει τα GCM μηνύματα που στέλνονται από τους GCM connection servers και εκκινεί το Intent Service *GcmIntentService.java* για να παραλάβει το μήνυμα. Για να λειτουργήσει ο receiver πρέπει να δηλώσουμε στο *AndroidManifest.xml* την άδεια χρήσης, τις ενέργειες και κατηγορίες ανίχνευσης.

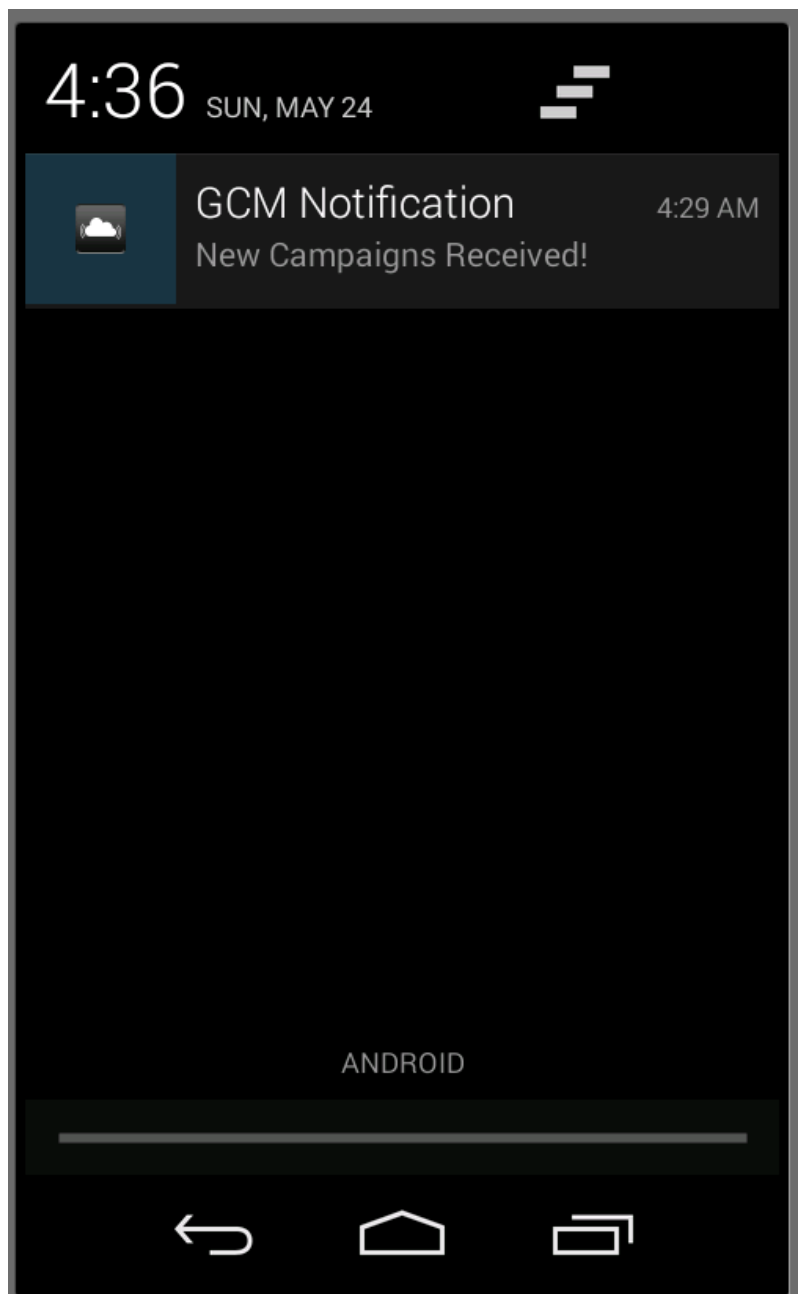
```

<receiver
    android:name=".GcmBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND"
    android:enabled="true" android:exported="true">
    <intent-filter>
        <!-- Receives the actual messages. -->
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="gr.soulis.huaservicetester2" />
    </intent-filter>
</receiver>

```

GcmIntentService.java

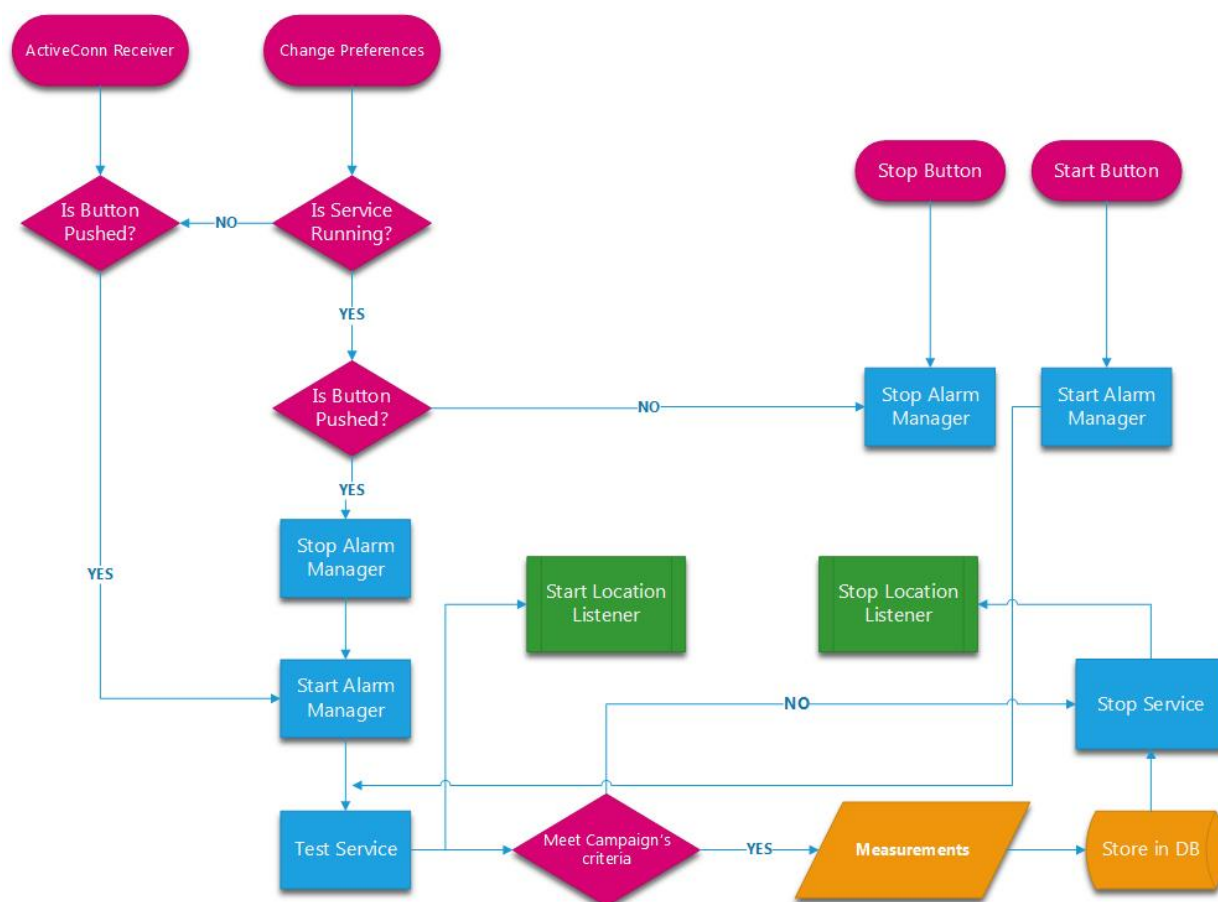
Το service καλείται από τον receiver *GcmBroadcastReceiver.java* με σκοπό την αποθήκευση του GCM μηνύματος που περιέχει τις ενεργές καμπάνιες. Χρησιμοποιεί το *GoogleCloudMessaging* API ώστε να διαβάσει το GCM μήνυμα, να το αποθηκεύσει σε αρχείο και να στείλει ένα Notification μήνυμα στην οθόνη της συσκευής (Εικόνα 7.4), που να πληροφορεί το χρήστη για τη λήψη ενεργών καμπανιών.



Εικόνα 7.4: Notification μήνυμα λήψης ενεργών καμπανιών από την υπηρεσία GCM

7.2.3 Υπηρεσία Παρασκηνίου

Στην αρχική οθόνη της εφαρμογής, στην Εικόνα 6.1, πατώντας το κουμπί **Start Service**, ξεκινά να εκτελείται η Υπηρεσία Παρασκηνίου. Η υπηρεσία αυτή πραγματοποιεί μετρήσεις του ασύρματου δικτύου στο οποίο είναι συνδεδεμένη η συσκευή, στο παρασκήνιο, με βάση τις προτιμήσεις του χρήστη (οθόνη *Settings*, Εικόνα 6.4) και το αρχείο των ενεργών καμπανιών, **ακόμα και όταν η εφαρμογή είναι κλειστή**. Για παράδειγμα, εάν είμαστε συνδεδεμένοι σε WiFi δίκτυο, γίνεται έλεγχος στις προτιμήσεις του χρήστη, εάν έχει δηλαδή επιλέξει να πραγματοποιούνται μετρήσεις στο συγκεκριμένο τύπο δικτύου (*Network Type: wifi ή both*) και κάθε πότε να πραγματοποιεί τη μέτρηση (κάθε 5, 10, 30 λεπτά). Στη συνέχεια, ελέγχεται το αρχείο καμπανιών, εάν η γεωγραφική μας θέση βρίσκεται εντός των ορίων μίας καμπάνιας. Εάν ναι, ελέγχονται τα περαιτέρω χαρακτηριστικά της καμπάνιας (τύπος δικτύου, ταχύτητα συσκευής, ακρίβεια μέτρησης) και αν ικανοποιούνται, η υπηρεσία προχωρά στη μέτρηση των τεσσάρων μετρικών (downlink, uplink, ping Google server, ping Personal server).



Εικόνα 7.5: Διάγραμμα Ροής λειτουργίας της υπηρεσίας παρασκηνίου για τις μετρήσεις δικτύου

Στην Εικόνα 7.5 δίνεται το διάγραμμα ροής λειτουργίας της υπηρεσίας παρασκηνίου. Αρχικά, η υλοποίηση της υπηρεσίας είχε γίνει με διαφορετικό τρόπο. Με το πάτημα του κουμπιού καλούταν μία κλάση η οποία υλοποιούσε τον `LocationListener`, ο οποίος, με βάση τη συχνότητα μέτρησης από τις προτιμήσεις του χρήστη, καλούσε με τη σειρά του το `IntentService` που έκανε τους παραπάνω ελέγχους, πραγματοποιούσε τις μετρήσεις και στη συνέχεια καταστρεφόταν. Το πρόβλημα σε αυτή την υλοποίηση ήταν ότι η μέθοδος `requestUpdates()` του `LocationListener` δεν εκτελείται σε σταθερά χρονικά διαστήματα, με αποτέλεσμα την έλλειψη αξιοπιστίας όσον αφορά τη συχνότητα των μετρήσεων. Αυτό συνέβαινε εάν η συσκευή έμενε για πολύ ώρα αδρανής, οπότε έπεφτε σε κατάσταση «ύπνου» (sleep mode). Η αδρανοποίηση της συσκευής μπορεί να αντιμετωπιστεί με τον `AlarmManager`, ο οποίος λειτουργεί σαν ξυπνητήρι στη συσκευή. Οπότε στην τελική υλοποίηση της υπηρεσίας παρασκηνίου, όπως φαίνεται και στην Εικόνα 7.5, με το πάτημα του κουμπιού υλοποιείται ο `AlarmManager`, ο οποίος καλεί το `Service` μέσα στο οποίο γίνεται η υλοποίηση του `LocationListener`, για τον προσδιορισμό της γεωγραφικής θέσης και στη συνέχεια, αφού γίνουν οι απαραίτητοι έλεγχοι, πραγματοποιείται η μέτρηση των μετρικών απόδοσης δικτύου. Τέλος, τα αποτελέσματα των μετρήσεων αποθηκεύονται στη βάση δεδομένων και το `Service` καταστρέφεται.

TestServiceFile.java

Η κλάση χρησιμοποιείται για να αποθηκεύουμε σε αρχείο την ενέργεια του χρήστη, όσον αφορά την υπηρεσία παρασκηνίου, αν δηλαδή πάτησε το κουμπί για να ξεκινήσει η υπηρεσία ή για να σταματήσει (μέθοδος `isMyServiceBtnRunning()`). Επίσης, έχει υλοποιηθεί και η μέθοδος `isMyServiceRunning()` η οποία ελέγχει εάν η υπηρεσία παρασκηνίου λειτουργεί ή όχι.

ActiveConnReceiver.java

Receiver με καθολική ισχύ, λειτουργεί δηλαδή ακόμα κι όταν η εφαρμογή είναι κλειστή. Ελέγχει τη λειτουργία της υπηρεσίας παρασκηνίου κάθε φορά που μεταβάλλεται η κατάσταση του δικτύου. Δηλώνεται στο *AndroidManifest.xml*:

```
<receiver
    android:name=".ActiveConnReceiver"
    android:enabled="true" >
    <intent-filter>
```

```

        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        <action android:name="android.net.wifi.WIFI_STATE_CHANGED" />
    </intent-filter>
</receiver>

```

Ο Receiver, όπως φαίνεται και στην Εικόνα 7.5, ελέγχει σε περίπτωση που έχει πατηθεί το κουμπί για ενεργοποίηση της υπηρεσίας αν όντως η υπηρεσία εκτελείται στο παρασκήνιο. Αν για οποιοδήποτε λόγο η υπηρεσία δε λειτουργεί ενώ ο χρήστης έχει πατήσει το κουμπί για έναρξη της υπηρεσίας, δίνει εντολή στον AlarmManager να ξεκινήσει την υπηρεσία.

Κώδικας:

```

/// This Receiver is running globally
public class ActiveConnReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        String activeConn = "";

        TestServiceFile tsf = new TestServiceFile(context.getApplicationContext());
        activeConn = tsf.activeConnection();

        /// THE SERVICE IS NOT RUNNING. WE HAVE TO CHECK IF THE USER HAS PUSHED THE BUTTON TO
        START
        if(!tsf.isMyServiceRunning()){
            try {
                if(tsf.isMyServiceBtnRunning()){
                    if(!(activeConn.isEmpty())){

                        GApiAlarmManager.startAlarmManager(context.getApplicationContext());

                    }
                }
            } catch (IOException e) {
                Toast.makeText(context.getApplicationContext(),
                    e.getStackTrace().toString(), Toast.LENGTH_LONG).show();
            }
        }
    }
}

```

GApiAlarmManager.java

Είναι η κλάση που εκτελείται όταν ο χρήστης πατήσει το κουμπί *Start/Stop Service*. Η κλάση ελέγχει την κατάσταση δικτύου και εάν οι προτιμήσεις του χρήστη συμπίπτουν, καλεί τον *AlarmManager*, ο οποίος με τη σειρά του ξεκινά το *Service* για την πραγματοποίηση των μετρήσεων (*Start Service*) ή διακόπτει τη λειτουργία του *AlarmManager* (*Stop Service*). Οι μέθοδοι και τα αντικείμενα της κλάσης είναι *static*, διότι η διαχείριση του *Service* πρέπει να είναι ενιαία και καθολική.

CampaignJson.java

Κλάση η οποία ανοίγει και αναλύει (*parse*) το αρχείο των καμπανιών, που είναι σε μορφή *JSON*. Η μέθοδος *findCampaign()* χρησιμοποιεί τη μέθοδο *matchCampaignCriteria()* για να βρει την καμπάνια που πληροί τις προϋποθέσεις για την πραγματοποίηση της μέτρησης και επιστρέφει τα στοιχεία της. Χρησιμοποιείται από την υπηρεσία παρασκηνίου (*TestService*).

Κώδικας διαδικασίας εύρεσης κατάλληλης καμπάνιας για πραγματοποίηση μετρήσεων

```
public boolean matchCampaignCriteria(Location location, String end_dateStr, double
camp_lat, double camp_lng, int camp_acc, int radius, int velocity) throws
ParseException{

    Date cur_date = new Date();
    // check if the campaign has expired
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd",
java.util.Locale.ITALIAN); // Locale.Greek doesn't exists
    Timestamp tStampEndDate;
    Date end_date;

    if(end_dateStr.contains("-")){ /// is Date
        try{
            end_date = dateFormat.parse(end_dateStr);
        }catch(ParseException pe){
            pe.printStackTrace();
            throw new ParseException(pe.getMessage(), pe.getErrorOffset());
        }
    }
    else{ // is TIMESTAMP
        tStampEndDate = new Timestamp(Long.parseLong(end_dateStr));
        end_date = new Date(tStampEndDate.getTime());
    }

    if(end_date.compareTo(cur_date) < 0)
        return false;

    float accuracy = 0;
    if(location.hasAccuracy())
        accuracy = location.getAccuracy();
}
```

```

        float speed = 0;
        if(location.hasSpeed())
            speed = location.getSpeed();

        // the location of each campaign
        Location camp_loc = new Location("");
        camp_loc.setLatitude(camp_lat);
        camp_loc.setLongitude(camp_lng);

        // check if our location fits with location of campaign
        if(location.distanceTo(camp_loc) > (radius+camp_acc))
            return false;

        // check the accuracy
        if(accuracy > camp_acc)
            return false;

        // check the speed
        if(speed > velocity)
            return false;

        return true;
    }

```

TestService.java

Η κλάση που υλοποιεί την **υπηρεσία παρασκηνίου**. Πρόκειται για Service που καλείται από την κλάση *GApiAlarmManager.java* με το πάτημα του κουμπιού **Start Service** από το χρήστη. Μέσα στην κλάση γίνεται υλοποίηση του *GoogleApiClient* και του *LocationListener* για ανάκτηση της γεωγραφικής θέσης της συσκευής, με μεγάλη ακρίβεια (*PRIORITY_HIGH_ACCURACY*). Αυτός είναι ο λόγος που δε χρησιμοποιείται *IntentService* αλλά *Service*. Το *Service* λειτουργεί στο παρασκήνιο μέχρι την εκτέλεση της εντολής *stopSelf()*. Το *IntentService* σταματά τη λειτουργία του προτού προλάβει ο *LocationListener* να επιστρέψει την γεωγραφική θέση της συσκευής, με αποτέλεσμα να μην πραγματοποιείται η μέτρηση.

Η κλάση, επίσης, ρωτά αν υπάρχει η κατάλληλη καμπάνια για την πραγματοποίηση της μέτρησης και ανακτά τα στοιχεία της (κλάση *CampaignJson.java*) με τη βοήθεια της μεθόδου *findCampaign()*. Στη συνέχεια καλεί την κλάση ***SpeedServiceTester*** για την πραγματοποίηση των μετρήσεων και την αποθήκευση των αποτελεσμάτων στη βάση δεδομένων. Ο κώδικας της κλάσης βρίσκεται στο Παράρτημα Β.

SpeedServiceTester.java

Είναι η κλάση που καλείται από την υπηρεσία παρασκηνίου για να εκτελέσει τις μετρήσεις των τεσσάρων μετρικών και να αποθηκεύσει τα αποτελέσματα στη βάση δεδομένων. Η συγκεκριμένη διαδικασία εκτελείται σε δύο διαφορετικά νήματα. Και τα δύο νήματα δημιουργούνται με τη βοήθεια της κλάσης `AsyncTask`. Το πρώτο νήμα πραγματοποιεί τις μετρήσεις και επιστρέφει ανάλογα μηνύματα στην εφαρμογή. Αν επιστραφεί μήνυμα επιτυχίας των μετρήσεων, δημιουργείται το δεύτερο νήμα, μέσω του οποίου στέλνονται τα αποτελέσματα των μετρήσεων με HTTP POST στον application server και συγκεκριμένα στο PHP script ***index_measures.php***, το οποίο αναλαμβάνει να συνδεθεί με τη βάση δεδομένων MySQL ***huaLocation*** και να αποθηκεύσει τις μετρήσεις στον πίνακα ***measurements***.

Η συγκεκριμένη υλοποίηση βασίστηκε στην υλοποίηση των μετρήσεων από την εργασία [1] του Κορωνιώτη Νικόλαου, προσαρμοσμένη στα δεδομένα της δικής μου υλοποίησης. Επίσης, για λόγους ασφαλείας στην αποθήκευση των μετρήσεων προστέθηκε η απαίτηση το αναγνωριστικό της συσκευής *Android_id*, να είναι αποθηκευμένο στον πίνακα ***c2dm_reg_devices***, ώστε να είμαστε σίγουροι ότι οι μετρήσεις προέρχονται από αξιόπιστη συσκευή Android.

7.2.4 Προτιμήσεις Χρήστη

Οι προτιμήσεις χρήστη αφορούν τις μετρήσεις απόδοσης ασύρματου δικτύου μέσω της υπηρεσίας παρασκηνίου. Δηλώνονται στην οθόνη ρυθμίσεων (settings) της εφαρμογής (Εικόνα 6.4) και έχουν αναλυθεί στην παράγραφο 6.1.2.

SettingsActivity.java

Κλάση που υλοποιεί την οθόνη ρυθμίσεων με τις προτιμήσεις του χρήστη. Είναι Activity που κληρονομεί από την κλάση `PreferenceActivity` [55], που χρησιμοποιείται για αποθήκευση των ρυθμίσεων των Android εφαρμογών στη διεπαφή `SharedPreferences`.

Κατά την αποθήκευση των ρυθμίσεων, ελέγχεται εάν η υπηρεσία παρασκηνίου βρίσκεται σε λειτουργία. Σε αυτή την περίπτωση δίνεται εντολή για επανεκκίνηση της υπηρεσίας, διότι οι αλλαγές στις προτιμήσεις του χρήστη μπορεί να επηρεάσουν τη διαδικασία μέτρησης.

Σχετικός Κώδικας:

```
//// listen for changes on Preferences for restarting the Service if running

PreferenceManager.getDefaultSharedPreferences(getActivity().getApplicationContext()).
registerOnSharedPreferenceChangeListener(new
SharedPreferences.OnSharedPreferenceChangeListener() {

    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                           String key) {
        // TODO Auto-generated method stub

        Toast.makeText(SettingsFragment.this.getActivity().getApplicationContext(),
"ConfigurationChanged!!", Toast.LENGTH_LONG).show();
        Log.d("PREF", "ConfigurationChanged!!");

        TestServiceFile tsf = new
TestServiceFile(getActivity().getApplicationContext());
        try {
            if(tsf.isMyServiceBtnRunning()){
                Intent i = new
Intent(getActivity().getApplicationContext(),gr.soulis.huaservicetester2.TestS
ervice.class);
                getActivity().stopService(i);
                getActivity().startService(i);
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
});
```

MyPreferences.java

Κλάση για ανάκτηση των ρυθμίσεων της εφαρμογής από το αντικείμενο SharedPreferences.

Σχετικός Κώδικας:

```
SharedPreferences mPrefs = context.getSharedPreferences("SharedPreferences",
Context.MODE_PRIVATE);
```

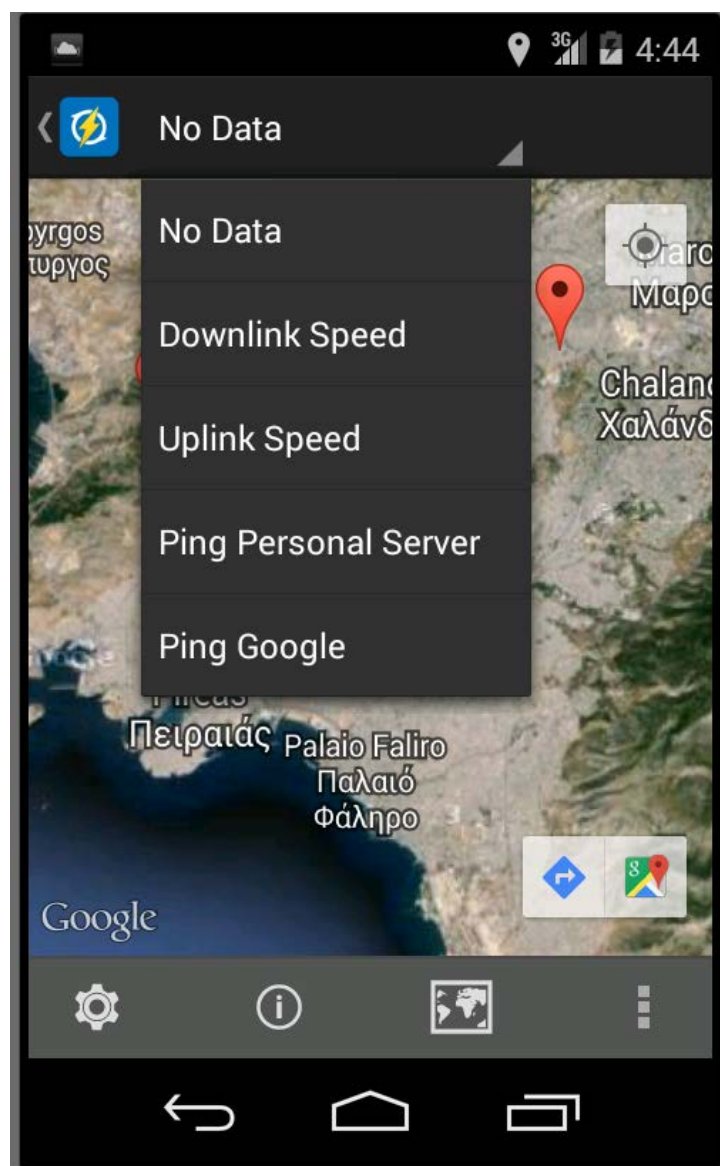
7.2.5 Google Maps

Για την αναπαράσταση των μετρήσεων απόδοσης ασύρματων δικτύων, χρησιμοποιούμε χάρτες της Google. Η εμφάνιση των Google Maps στην εφαρμογή βρίσκεται στο υπομενού, όπως φαίνεται στην Εικόνα 6.2, στο κουμπί *Show Map*.



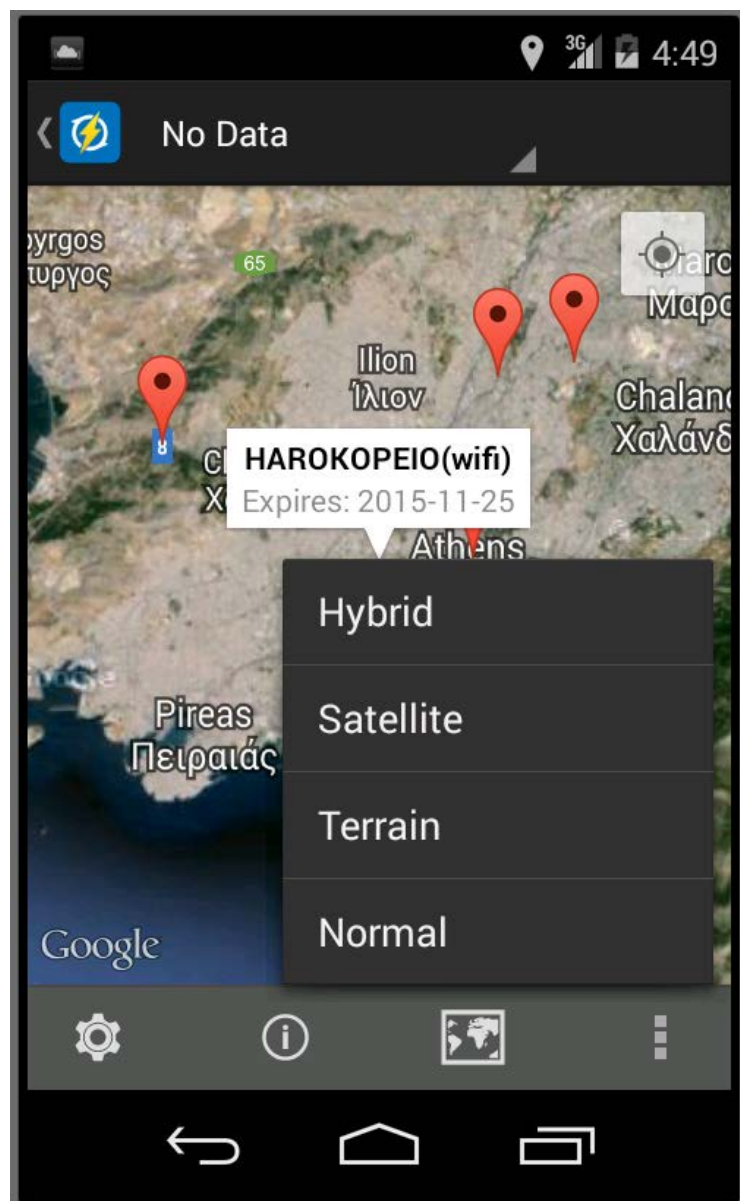
Εικόνα 7.6: Οι ενεργές καμπάνιες σε Google Maps

Στην Εικόνα 7.6 φαίνονται στο χάρτη τα σημεία που αναπαριστούν το επίκεντρο των ενεργών καμπανιών (markers). Κάνοντας «κλικ» πάνω τους με το αριστερό κουμπί του ποντικιού, εμφανίζονται στοιχεία που αφορούν τη συγκεκριμένη καμπάνια, όπως το όνομά της, ο τύπος δικτύου που υποστηρίζει και η ημερομηνία λήξης της.



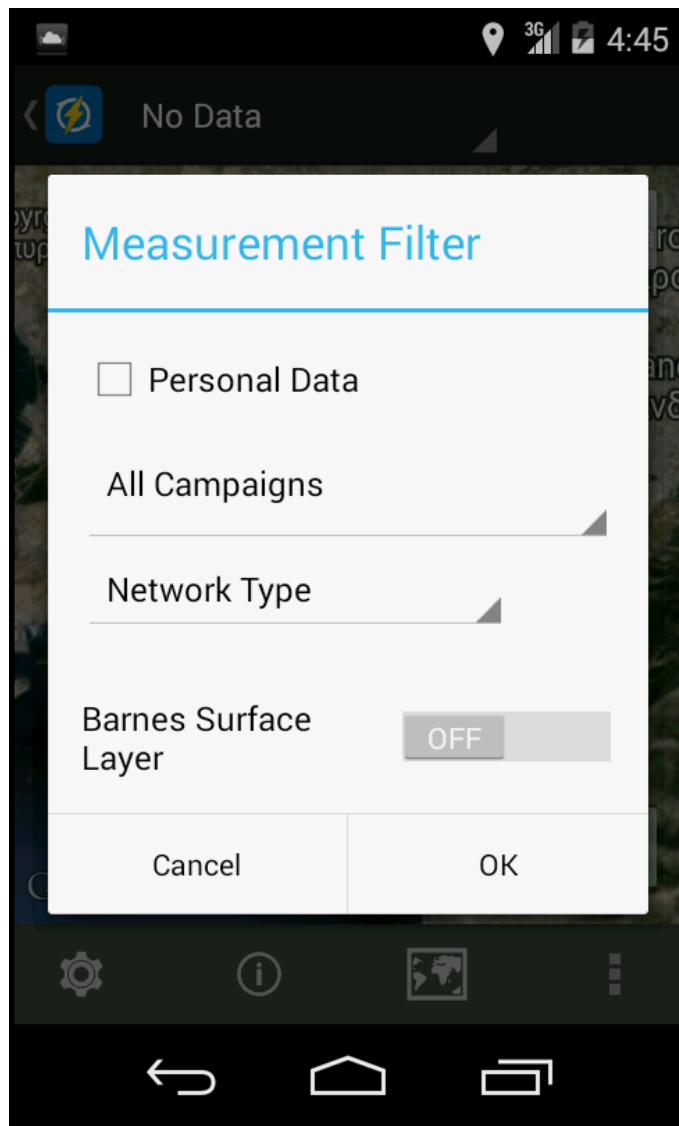
Εικόνα 7.7: Επιλογή μετρικών για αναπαράσταση σε Google Maps

Στην Εικόνα 7.7, εμφανίζονται οι επιλογές μετρικών απόδοσης, για τις οποίες έχουν γίνει οι μετρήσεις, ώστε να αποτυπωθούν στο χάρτη της Google. Ο χάρτης, αρχικά δεν εμφανίζει μετρήσεις διότι η εξ' ορισμού επιλογή είναι *No Data*.



Εικόνα 7.8: Επιλογή τύπου χάρτη σε Google Maps

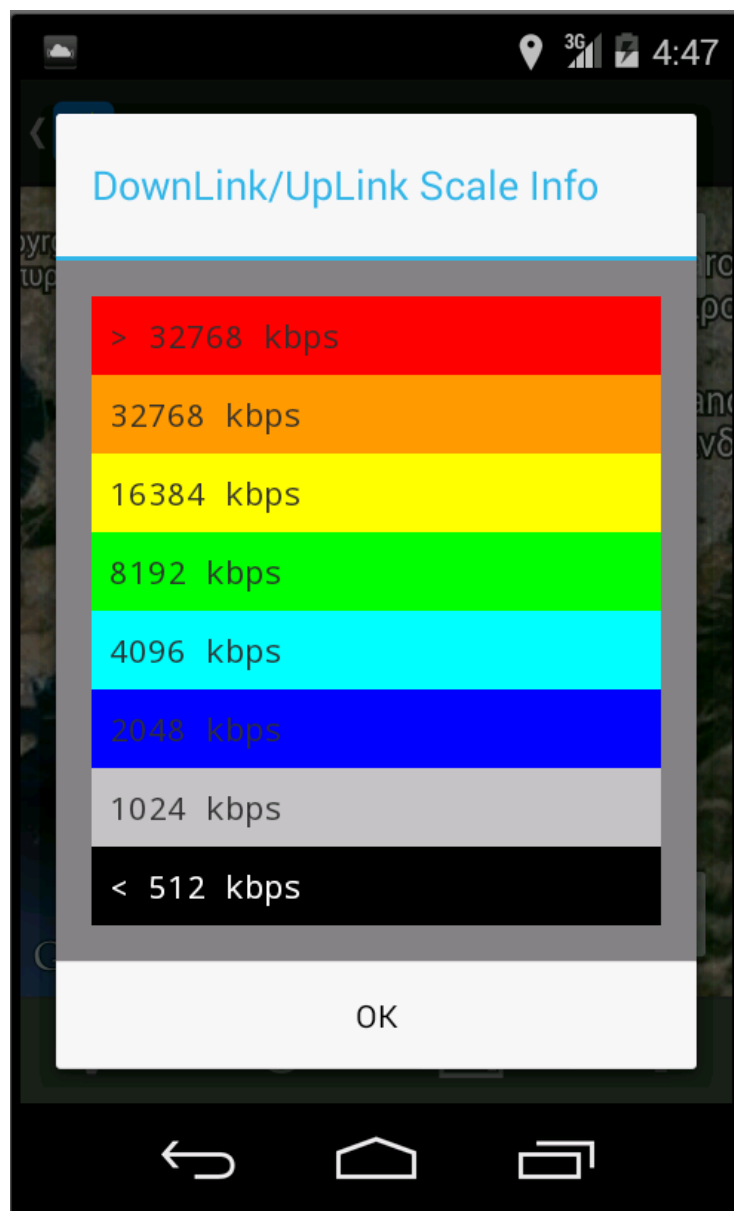
Στην Εικόνα 7.8 εμφανίζονται οι επιλογές για τους τύπους χαρτών που υποστηρίζονται. Είναι οι τύποι που υποστηρίζονται από τους χάρτες της Google (Google Maps).



Εικόνα 7.9: Επιλογή φίλτρων για αναπαράσταση σε Google Maps

Στην Εικόνα 7.9 φαίνεται το παράθυρο διαλόγου με τις επιλογές φίλτρων που έχουμε. Υπάρχει και η δυνατότητα τα φίλτρα να συνδυαστούν μεταξύ τους.

- *Personal Data*
Δυνατότητα για εμφάνιση μετρήσεων που έγιναν μόνο από τη συγκεκριμένη συσκευή
- *Campaigns*
Εμφάνιση μετρήσεων για όλες τις καμπάνιες/συγκεκριμένη καμπάνια/κατά παραγγελία
- *Network Type*
Εμφάνιση μετρήσεων μόνο για συγκεκριμένο τύπο δικτύου (wifi/mobile/All Types)
- *Barnes Surface Layer*
Απεικόνιση μετρήσεων μέσα από τη συγκεκριμένη επεξεργασία σύντηξης δεδομένων.



Εικόνα 7.10: Χρωματική κατηγοριοποίηση μετρήσεων

Στην Εικόνα 7.10 απεικονίζονται τα χρώματα που λαμβάνουν οι μετρήσεις σε αντιστοιχία με το εύρος τιμών στο οποίο ανήκουν. Η συγκεκριμένη χρωματική απεικόνιση αφορά τις μετρικές ταχύτητας λήψης δεδομένων (Downlink) και ταχύτητας αποστολής δεδομένων (Uplink). Αντίστοιχη χρωματική απόδοση υφίσταται και για τις 2 μετρικές μέτρησης ping. Το μόνο που αλλάζει είναι η μονάδα μέτρησης (sec αντί kbps).

GoogleMapsActivity.java

Είναι η κλάση – Activity η οποία υλοποιεί την αναπαράσταση των μετρήσεων σε χάρτες της Google. Η υλοποίηση, όπως έχει αναφερθεί προηγουμένως, έχει πραγματοποιηθεί με το **Google Maps API v2**. Η απεικόνιση του χάρτη γίνεται με τη βοήθεια της κλάσης MapFragment.

Σχετικός Κώδικας:

```
GoogleMap map = null;
MapFragment mapF = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
    map = mapF.getMap();
    map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    map.setMyLocationEnabled(true);           // focus to current location
```

Το αντικείμενο mapF δηλώνεται στο *layout* αρχείο του activity *activity_google_maps.xml*

```
<fragment
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    class="com.google.android.gms.maps.MapFragment"/>
```

Οι επιλογές και τα φίλτρα έχουν σχεδιαστεί σαν στοιχεία του ActionBar. Πιο συγκεκριμένα, η επιλογή μετρικών έχει σχεδιαστεί σε μορφή λίστας (Εικόνα 7.7):

```
// Set up the action bar to show a dropdown list.
    final ActionBar actionBar = getActionBar();
    actionBar.setDisplayShowTitleEnabled(false);
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_LIST);
```

ενώ τα φίλτρα σαν στοιχείο του ActionBar. Επιλέγοντας το αντίστοιχο εικονίδιο (Εικόνα 7.9), ανοίγει ένα παράθυρο διαλόγου τύπου AlertDialog.

Η επιλογή τύπου χάρτη έχει σχεδιαστεί ως μενού (δήλωση στο αρχείο – μενού του activity *menu/google_maps.xml*), με δυνατότητα να εμφανίζεται και ως στοιχείο του ActionBar, εάν υπάρχει διαθέσιμος χώρος (Εικόνα 7.8).

```
app:showAsAction="ifRoom"
```

Τέλος, με τη βοήθεια της μεθόδου MapData() γίνεται η αναπαράσταση των μετρήσεων στο χάρτη, σε μορφή χρωματιστών κουκίδων ή μέσω της τεχνικής *Banes Surface* (θα παρουσιαστούν στην παράγραφο 7.3).

7.2.6 Στατιστικά Μετρήσεων

Η εφαρμογή, εκτός από την αναπαράσταση των μετρήσεων σε Google Maps, προσφέρει κι ένα σύνολο από οθόνες στατιστικών για τις μετρήσεις απόδοσης ασύρματων δικτύων σε διαγράμματα γραφικών, χρησιμοποιώντας τη βιβλιοθήκη **MPAndroidChart**. Οι οθόνες αυτές βρίσκονται στο υπομενού της εφαρμογής (Εικόνα 6.2).

➤ *Campaigns Stats*

Οθόνη με γράφημα τύπου – πίτας, που αναπαριστά τα ποσοστά μετρήσεων ανά καμπάνια σε κάθε τύπο δικτύου (wifi, mobile, both).

➤ *Network Types Stats*

Οθόνη με γράφημα τύπου – πίτας, που αναπαριστά τα ποσοστά μετρήσεων ανά τύπο δικτύου (wifi, LTE, HSPA, HSPAP κλπ) για όλους τους παρόχους ή για κάθε πάροχο.

➤ *Metrics Stats*

Οθόνη με γράφημα – τύπου μπάρες, που αναπαριστά τη μέση τιμή μετρήσεων ανά μετρική στις καμπάνιες για κάθε τύπο δικτύου.

➤ *Mobile Op Comparison*

Οθόνη με γράφημα – τύπου μπάρες, που αναπαριστά για κάθε καμπάνια τη μέση τιμή μετρήσεων ανά μετρική των παρόχων κινητής τηλεφωνίας VODAFONE GR, WIND GR, GR COSMOTE (συγκριτικά αποτελέσματα).

Η ανάκτηση των δεδομένων που αφορούν τις μετρήσεις γίνονται μέσω HTTP GET αιτημάτων σε RESTful υπηρεσίες που έχουν δημιουργηθεί στον διακομιστή εφαρμογών (παράγραφος 7.1.2).

RESTfulModel.java

Κλάση η οποία υλοποιεί τα HTTP GET αιτήματα στις παρακάτω RESTful υπηρεσίες :

```
protected final static String REST_URL_MEASURES =  
"http://83.212.113.47:8080/RESTFulHuaLocation/webresources/gr.hua.location.rest.measu  
rements";  
protected final static String REST_URL_CAMPAGNS =  
"http://83.212.113.47:8080/RESTFulHuaLocation/webresources/gr.hua.location.rest.campa  
igns";
```

Οι κλήσεις γίνονται σε διαφορετικό νήμα (thread) με υλοποίηση της κλάσης AsyncTask. Τα περισσότερα αιτήματα αφορούν την ανάκτηση των μετρήσεων για χρήση από τις οθόνες στατιστικών. Τα δεδομένα προέρχονται από τους πίνακες *measurements* και *campaigns* της βάσης δεδομένων MySQL, *hwaLocation*. Ανάλογα με την επιπλέον διαμόρφωση του URL κλήσης καλείται και διαφορετική μέθοδο υλοποίησης της RESTful υπηρεσίας.

- `setUserMobileDataUsage(String userID):`
`String[] uri = {REST_URL_MEASURES+"/countMetrics/"+userID};`
Ανακτάται το πλήθος των μετρήσεων του χρήστη `userID`, τον τρέχοντα μήνα σε mobile δίκτυο. Χρησιμοποιείται για τον υπολογισμό χρήσης δεδομένων (mobile data usage).
- `getCampaigns():`
`String[] uri = {REST_URL_CAMPAIGNS+"/getActives/1";}`
Ανάκτηση των ενεργών καμπανιών σε μορφή JSON παρόμοια με τη μορφή που στέλνεται από την υπηρεσία GCM.
- `getMeasurements(String nType):`
`String[] uri = {REST_URL_MEASURES+"/getMeasurements/"+nType};`
Ανακτά το πλήθος μετρήσεων ανά καμπάνια και ανά τύπο δικτύου.
- `getNetworkTypes(String operator):`
`String[] uri = {REST_URL_MEASURES+"/getNetworkTypes/"+operator};`
Ανακτά το πλήθος μετρήσεων ανά τύπο δικτύου για κάθε πάροχο (ή για όλους).
- `getMetricsPerCampaign():`
`String[] uri = {REST_URL_MEASURES+"/getMetricsPerCamp";}`
Ανακτά τη μέση τιμή των μετρήσεων για κάθε μετρική ανά καμπάνια.
- `getMetricsPerCampaign(String operator):`
`String[] uri = {REST_URL_MEASURES+"/getMetricsPerCamp/"+operator};`
Ίδια υπηρεσία με την προηγούμενη για συγκεκριμένο πάροχο κινητής τηλεφωνίας.
- `getOperatorMetrics():`
`String[] uri = {REST_URL_MEASURES+"/getOperatorMetrics";}`
Ανακτά τη μέση τιμή των μετρήσεων για κάθε μετρική ανά πάροχο κινητής τηλεφωνίας.
- `getOperatorMetrics(int campId):`
`String[] uri = {REST_URL_MEASURES+"/getOperatorMetrics/"+String.valueOf(campId)};`
Ίδια υπηρεσία με την προηγούμενη για συγκεκριμένη καμπάνια.

StatsBase.java

Υπερκλάση της βιβλιοθήκης MPAndroidChart.

PieChartActivity.java

Η κλάση – activity που υλοποιεί την οθόνη *Campaigns Stats* (Εικόνα 7.14). Χρησιμοποιεί την κλήση `getMeasurements()` στην αντίστοιχη RESTful υπηρεσία, για ανάκτηση των δεδομένων.

PieChartCampaign.java

Κλάση που δημιουργεί αντικείμενα τύπου *PieChartCampaign* που περιέχουν όλη την πληροφορία που χρειάζεται η οθόνη *Campaigns Stats* για να απεικονίσει το γράφημα. Στέλνονται ως `Parcelable` αντικείμενα – παράμετροι στον `receiver` που κάνει την κλήση στην αντίστοιχη RESTful υπηρεσία.

PieOpNetworkActivity.java

Η κλάση – activity που υλοποιεί την οθόνη *Network Types Stats* (Εικόνα 7.15). Χρησιμοποιεί την κλήση `getNetworkTypes()` στην αντίστοιχη RESTful υπηρεσία, για ανάκτηση των δεδομένων.

PieChartOperator.java

Κλάση που δημιουργεί αντικείμενα τύπου *PieChartOperator* που περιέχουν όλη την πληροφορία που χρειάζεται η οθόνη *Network Types Stats* για να απεικονίσει το γράφημα. Στέλνονται ως `Parcelable` αντικείμενα – παράμετροι στον `receiver` που κάνει την κλήση στην αντίστοιχη RESTful υπηρεσία.

MultiBarChartActivity.java

Η κλάση – activity που υλοποιεί την οθόνη *Metrics Stats* (Εικόνες 7.16, 7.17). Χρησιμοποιεί τις κλήσεις `getMetricsPerCampaign()` στις αντίστοιχες RESTful υπηρεσίες, για ανάκτηση των δεδομένων.

MultiBarChartMetrics.java

Κλάση που δημιουργεί αντικείμενα τύπου *MultiBarChartMetrics* που περιέχουν όλη την πληροφορία που χρειάζεται η οθόνη *Metrics Stats* για να απεικονίσει το γράφημα. Στέλνονται ως *Parcelable* αντικείμενα – παράμετροι στον receiver που κάνει την κλήση στην αντίστοιχη RESTful υπηρεσία.

MultiBarChartOpActivity.java

Η κλάση – activity που υλοποιεί την οθόνη *Mobile Op Comparison* (Εικόνες 7.18, 7.19). Χρησιμοποιεί τις κλήσεις `getOperatorMetrics()` στις αντίστοιχες RESTful υπηρεσίες, για ανάκτηση των δεδομένων.

OperatorMetrics.java

Κλάση που δημιουργεί αντικείμενα τύπου *OperatorMetrics* που περιέχουν όλη την πληροφορία που χρειάζεται η οθόνη *Mobile Op Comparison* για να απεικονίσει το γράφημα. Στέλνονται ως *Parcelable* αντικείμενα – παράμετροι στον receiver που κάνει την κλήση στην αντίστοιχη RESTful υπηρεσία.

7.3 Παραδείγματα Απεικόνισης Μετρήσεων σε Google Maps

Στην αρχική οθόνη της εφαρμογής (Εικόνα 6.1) πατώντας το κουμπί *Start Service* θέτουμε σε λειτουργία την υπηρεσία παρασκηνίου. Εφόσον βρεθεί η κατάλληλη καμπάνια, η υπηρεσία ξεκινά να πραγματοποιεί μετρήσεις, οι οποίες αποθηκεύονται στον πίνακα *measurements*.

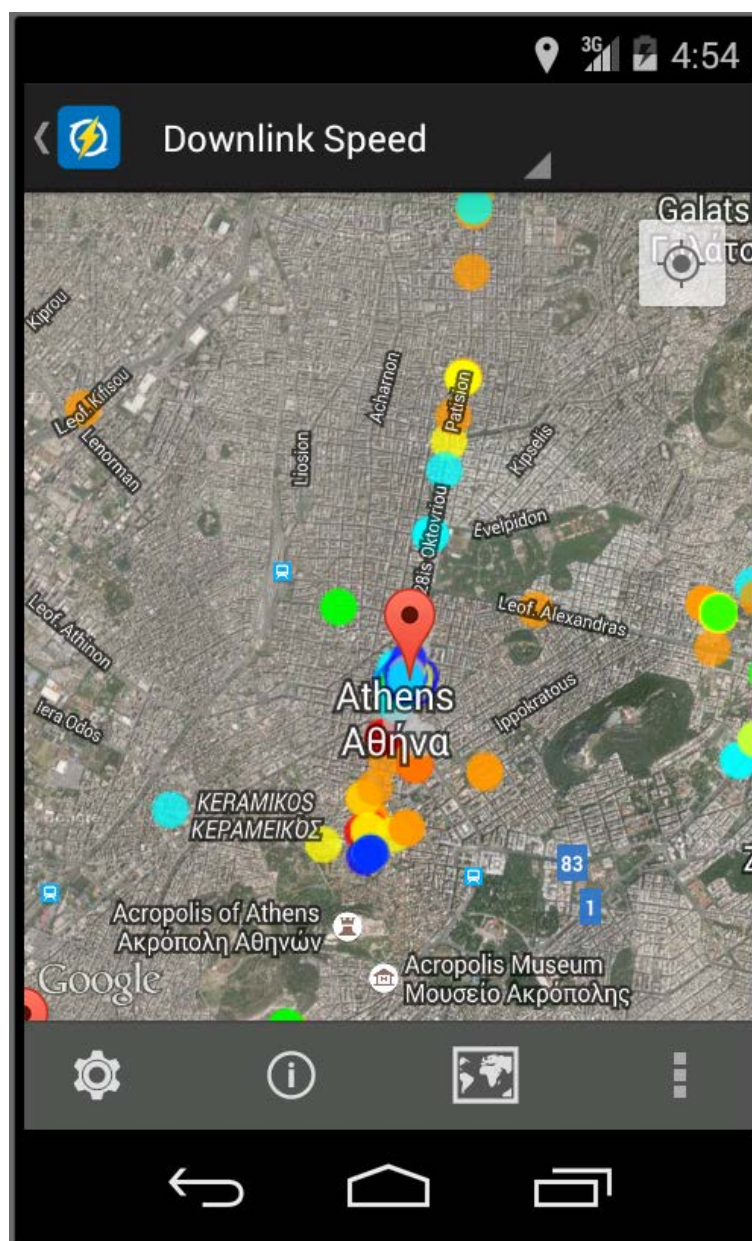
Στην Εικόνα 7.11, βλέπουμε ένα δείγμα από τις μετρήσεις που αποθηκεύονται στον πίνακα *measurements*. Όπως φαίνεται, οι μετρήσεις αυτές γίνονται κάθε πέντε λεπτά, διότι αυτή είναι η προτίμηση του χρήστη για τη συχνότητα των μετρήσεων στο συγκεκριμένο τύπο δικτύου (wifi). Η καμπάνια στα πλαίσια της οποίας γίνονται οι μετρήσεις έχει αναγνωριστικό (*camp_id*) ίσο με 2. Οι μετρήσεις έχουν γίνει από μία συγκεκριμένη συσκευή (φαίνεται από το *Android_id*). Στην Εικόνα 6.1 υπάρχουν και μετρήσεις που έχουν ως *camp_id* = 0. Αυτές οι μετρήσεις δεν έχουν γίνει από την υπηρεσία παρασκηνίου, αλλά κατά παραγγελία (OnDemand) του χρήστη, πατώντας στο μενού, το κουμπί *Test On Demand*.

	Downlink	Uplink	Ping1_IPERF	Ping2_DN	Network_Type	Measurement_Time	latitude	longitude	os	Android_id	accuracy	camp_id	Speed	SSI
'	11785.6	864.709	46.4011	61.592	wifi	2015-05-09 08:47:59	37.9902	23.7538	Android_19	357507051627340	88.5	2	0.00	LUN
'	11885.8	870.354	41.0009	56.6563	wifi	2015-05-09 08:53:00	37.9902	23.7538	Android_19	357507051627340	70.5	2	0.00	LUN
'	11951.6	867.444	42.1498	56.016	wifi	2015-05-09 08:57:59	37.9902	23.7537	Android_19	357507051627340	58.1	2	0.00	LUN
'	12038.4	868.228	40.7968	55.7291	wifi	2015-05-09 09:03:00	37.9902	23.7538	Android_19	357507051627340	61.5	2	0.00	LUN
'	11958.6	863.542	48.2784	55.7596	wifi	2015-05-09 09:08:00	37.9902	23.7538	Android_19	357507051627340	70.5	2	0.00	LUN
'	11959.6	870.354	43.577	56.1021	wifi	2015-05-09 09:13:00	37.9902	23.7538	Android_19	357507051627340	70.5	2	0.00	LUN
'	11900.7	869.802	46.4409	56.5591	wifi	2015-05-09 09:18:00	37.9902	23.7538	Android_19	357507051627340	70.5	2	0.00	LUN
'	12032.3	869.329	54.4937	54.5738	wifi	2015-05-09 09:22:59	37.9902	23.7537	Android_19	357507051627340	55.3	2	0.00	LUN
'	11973.7	866.817	37.2429	57.0875	wifi	2015-05-09 09:28:00	37.9902	23.7538	Android_19	357507051627340	88.5	2	0.00	LUN
'	11733.2	829.732	36.6734	53.0363	wifi	2015-05-09 09:32:59	37.9902	23.7537	Android_19	357507051627340	62.1	2	0.00	LUN
'	11820.7	869.644	44.7728	57.5096	wifi	2015-05-09 09:37:59	37.9902	23.7538	Android_19	357507051627340	63.0	2	0.00	LUN
'	11768	869.959	36.3007	57.9702	wifi	2015-05-09 09:42:59	37.9902	23.7538	Android_19	357507051627340	38.8	2	0.00	LUN
'	11858.1	870.354	45.2026	55.1054	wifi	2015-05-09 09:47:59	37.9902	23.7537	Android_19	357507051627340	61.5	2	0.00	LUN
'	11809	869.723	44.4991	54.141	wifi	2015-05-09 09:52:59	37.9902	23.7537	Android_19	357507051627340	41.8	2	0.00	LUN
'	11797.3	850.461	72.0033	68.6564	wifi	2015-05-09 09:57:59	37.9902	23.7538	Android_19	357507051627340	70.5	2	0.00	LUN
'	11826.6	869.329	50.1723	55.6972	wifi	2015-05-09 10:02:59	37.9902	23.7537	Android_19	357507051627340	59.5	2	0.00	LUN
'	11959.6	864.553	48.3223	55.5456	wifi	2015-05-09 10:07:59	37.9902	23.7537	Android_19	357507051627340	61.5	2	0.00	LUN
'	8188.35	870.432	44.4932	70.8865	wifi	2015-05-09 10:12:59	37.9902	23.7538	Android_19	357507051627340	67.5	2	0.00	LUN
'	5912.33	850.838	95.9098	68.6776	wifi	2015-05-09 16:14:35	38.0144	23.6333	Android_15	868329010984692	27.0	0	0.00	my_
'	2084.94	659.703	125.801	69.9631	wifi	2015-05-09 16:27:08	38.0143	23.6335	Android_15	868329010984692	34.0	0	0.00	my_
'	6159.66	836.747	608.791	79.679	wifi	2015-05-09 16:32:06	38.0142	23.6335	Android_15	868329010984692	1058.0	3	0.00	my_

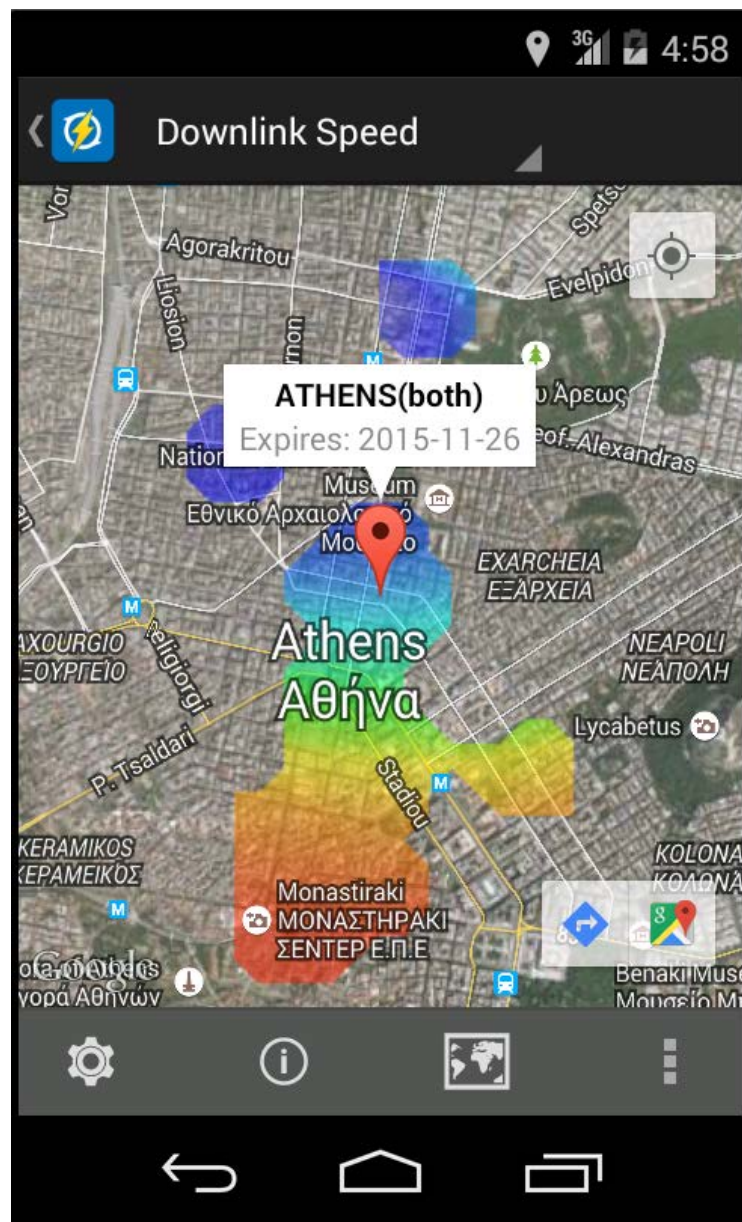
Εικόνα 7.11: Παράδειγμα μετρήσεων στον πίνακα *measurements*

Η εφαρμογή δίνει τη δυνατότητα απεικόνισης αυτών των μετρήσεων σε χάρτες της Google, όπως φαίνεται στην Εικόνα 7.12. Η οθόνη αυτή, όπως έχουμε δει, βρίσκεται στην επιλογή *Maps/Statistics -> Show Map*. Οι κουκίδες αναπαριστούν τη γεωγραφική θέση της μέτρησης και το χρώμα τους την τάξη μεγέθους της μέτρησης, με βάση την Εικόνα 7.10.

Εάν θέλουμε να περιορίσουμε τα αποτελέσματα σε συγκεκριμένη περιοχή ή για συγκεκριμένο τύπο δικτύου ή να εμφανιστούν μόνο οι προσωπικές μετρήσεις, μπορούμε να χρησιμοποιήσουμε τα κατάλληλα φίλτρα, σύμφωνα με την Εικόνα 7.9.



Εικόνα 7.12: Αναπαράσταση μετρήσεων απόδοσης δικτύου σε Google Maps



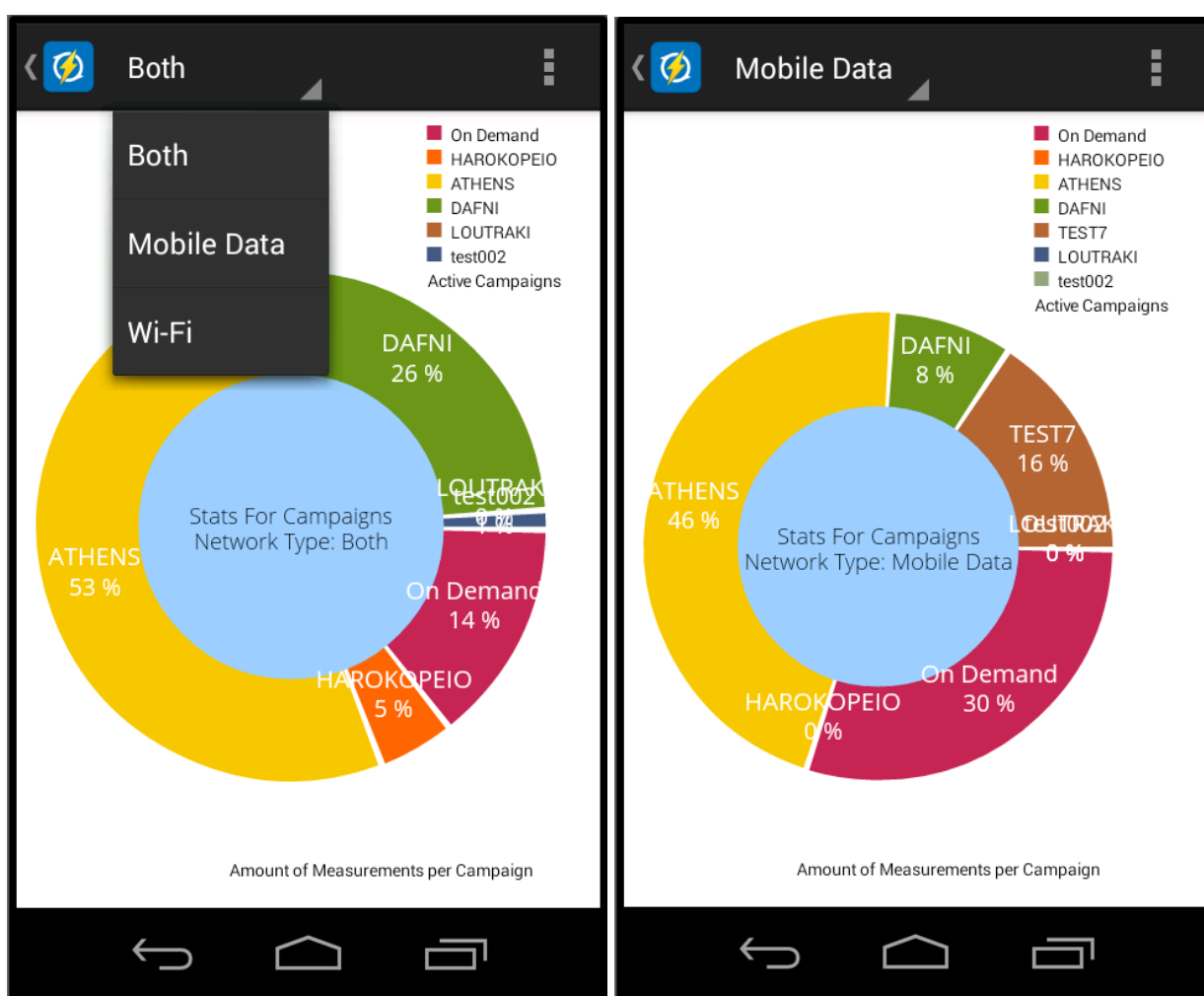
Εικόνα 7.13: Ενεργοποίηση Barnes Surface layer σε Google Maps

Επίσης, στις επιλογές των φίλτρων υπάρχει η δυνατότητα ενεργοποίησης του **Barnes Surface Layer**, όπως φαίνεται στην Εικόνα 7.13, όπου οι μετρήσεις έχουν υποστεί επεξεργασία από αλγόριθμο σύντηξης δεδομένων, ο οποίος έχει την ικανότητα να προβλέπει τιμές μετρήσεων σε θέσεις που δεν έχουν γίνει μετρήσεις, αλλά είναι κοντινές σε άλλες μετρήσεις, με το οπτικό αποτέλεσμα να είναι πολύ πιο φιλικό προς το χρήστη.

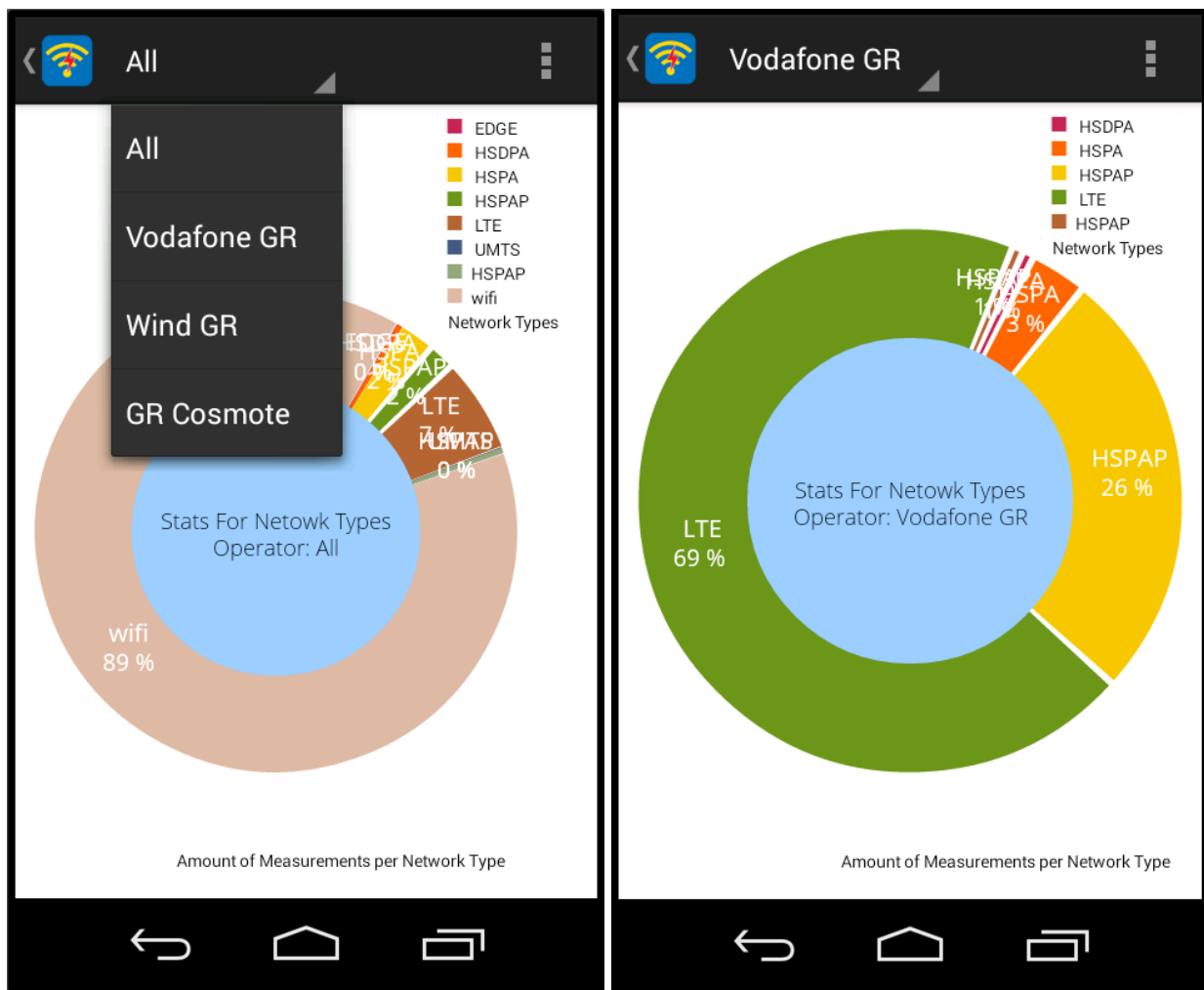
7.4 Παραδείγματα Απεικόνισης Στατιστικών

Με τη βοήθεια της βιβλιοθήκης **MPAndroidChart** έχουν δημιουργηθεί 4 οθόνες παρουσίασης στατιστικών των μετρήσεων απόδοσης δικτύου σε γραφικά περιβάλλοντα.

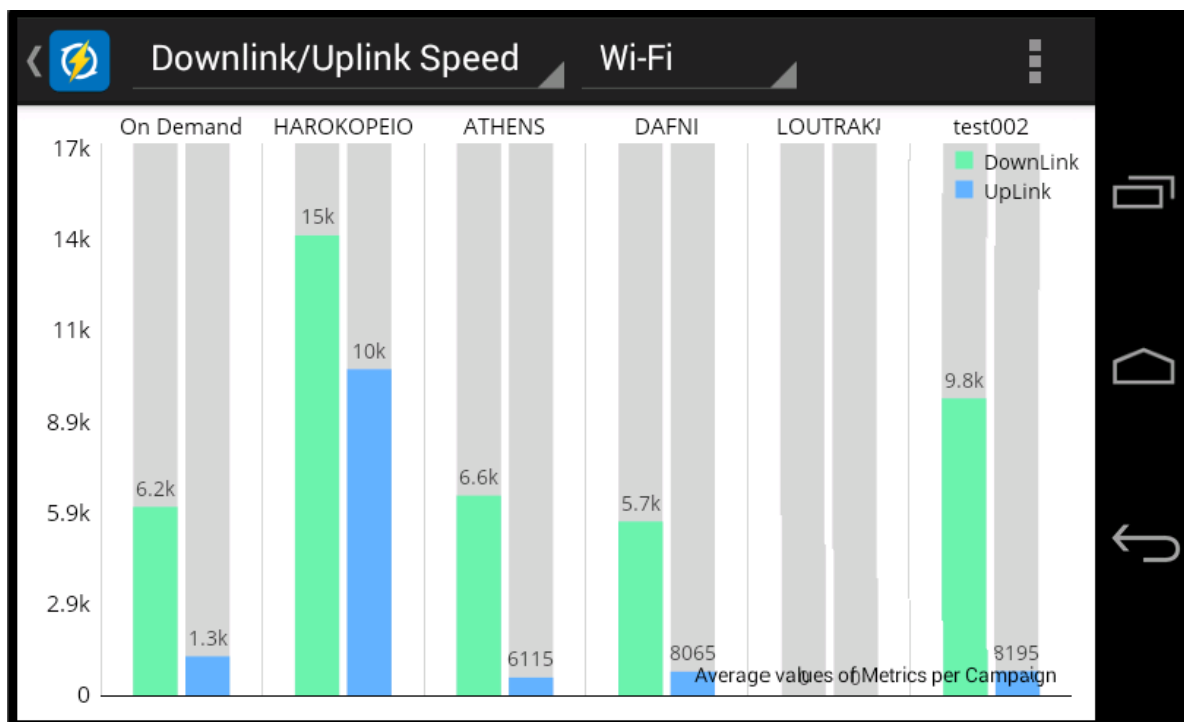
Ακολουθούν παραδείγματα απεικόνισης αυτών των μετρήσεων (Εικόνες 7.14, 7.15, 7.16, 7.17, 7.18) σε διαγράμματα γραφικών με αντίστοιχη περιγραφή των μεταβλητών που αναπαριστούν.



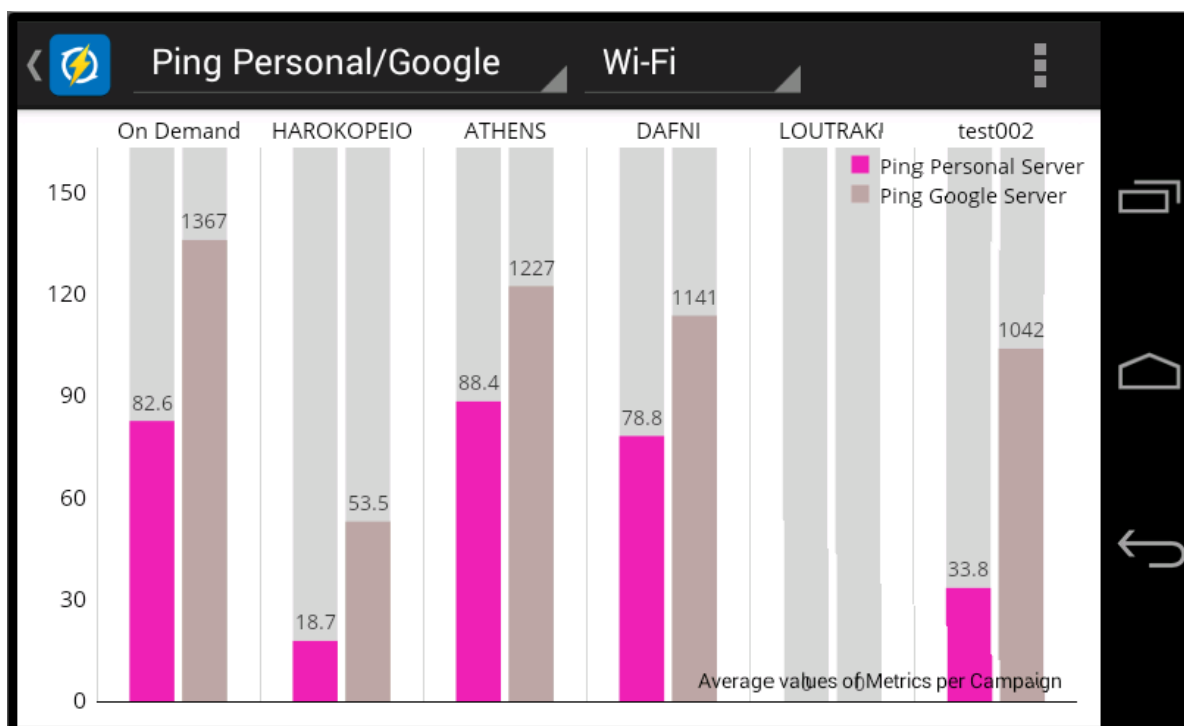
Εικόνα 7.14: Η οθόνη *Campaigns Stats* για διαφορετικούς τύπους δικτύων



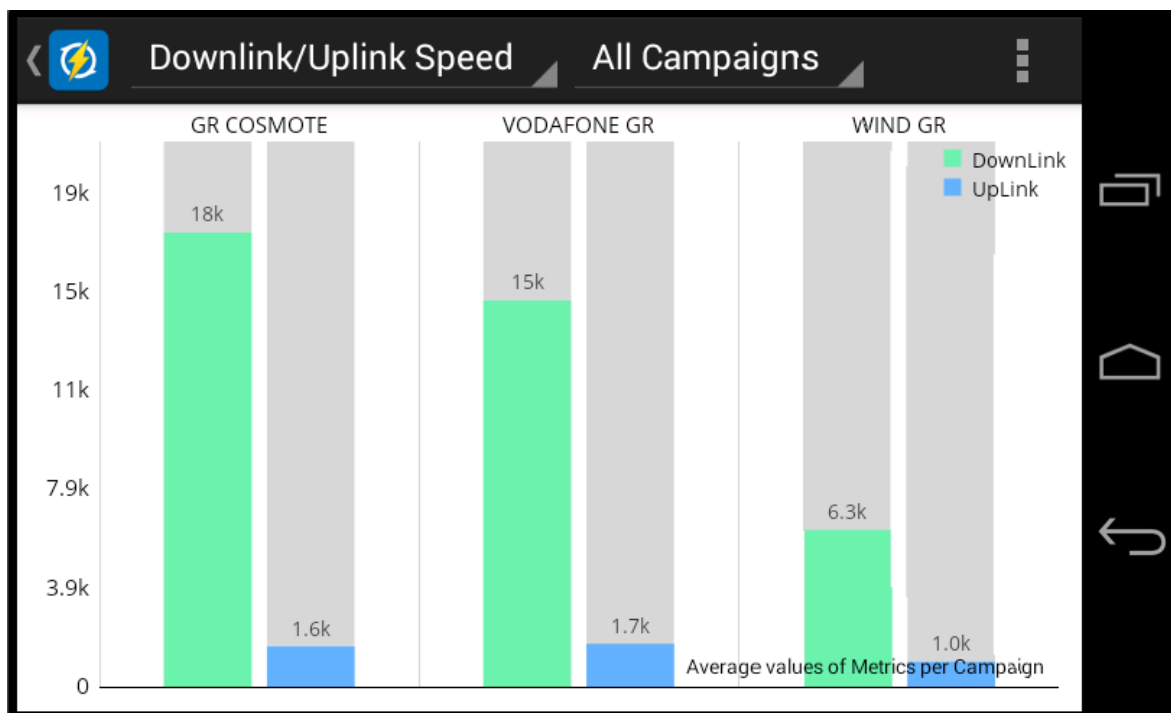
Εικόνα 7.15: Η οθόνη *Network Types Stats* για όλους τους τύπους δικτύου (αριστερά) και για συγκεκριμένο πάροχο δικτύου (δεξιά)



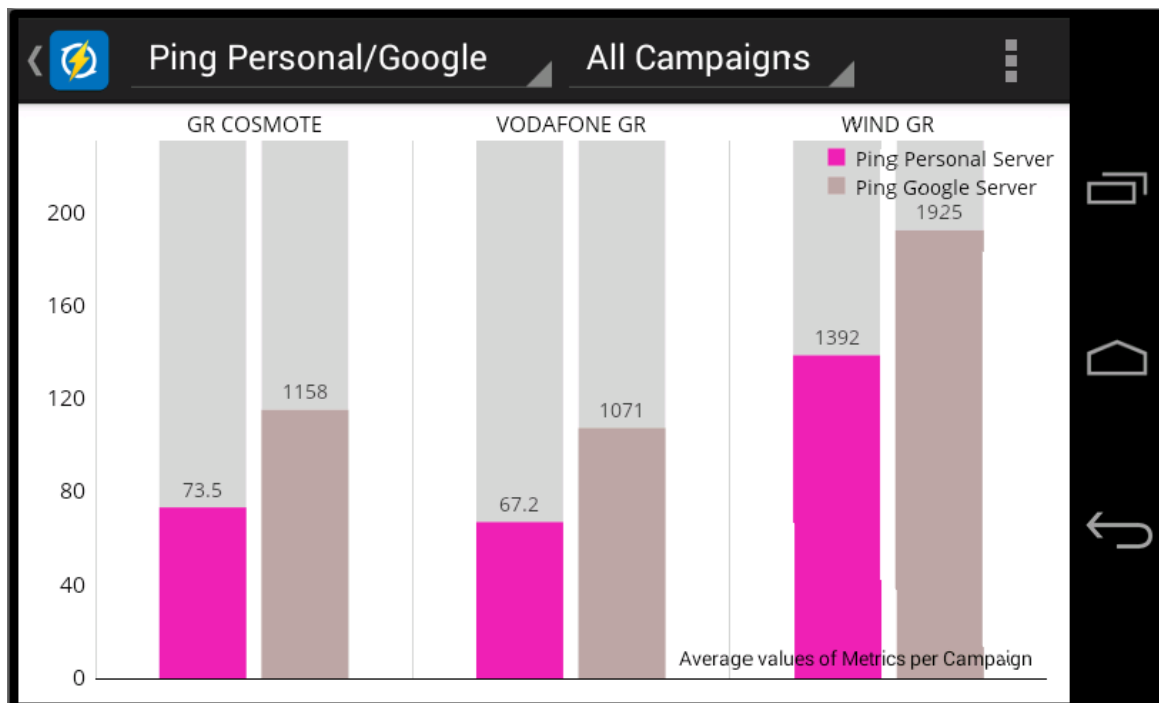
Εικόνα 7.16: Η οθόνη *Metrics Stats* για τις μετρικές *Downlink/Uplink* και δίκτυο *wifi*. Υπάρχει η δυνατότητα επιλογής μετρικών και τύπου δικτύου (*Wi-Fi*, *Vodafone*, *Wind*, *Cosmote*)



Εικόνα 7.17: Η οθόνη *Metrics Stats* για τις μετρικές *Ping Personal/Google* και δίκτυο *wifi*. Υπάρχει η δυνατότητα επιλογής μετρικών και τύπου δικτύου (*Wi-Fi*, *Vodafone*, *Wind*, *Cosmote*)



Εικόνα 7.18: Η οθόνη *Mobile Op Comparison* για τις μετρικές *Downlink/Uplink*. Υπάρχει η δυνατότητα επιλογής μετρικών και καμπάνιας.



Εικόνα 7.19: Η οθόνη *Mobile Op Comparison* για τις μετρικές *Ping Personal/Google*. Υπάρχει η δυνατότητα επιλογής μετρικών και καμπάνιας.

8 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΕΠΕΚΤΑΣΕΙΣ

8.1 Απολογισμός

Μέσα από την ενασχόλησή μου με τις τεχνολογίες που είναι σχετικές με το χώρο των έξυπνων συσκευών, διαπίστωσα ότι η ανάπτυξη εφαρμογών σε αυτό το χώρο παρουσιάζει δυσκολίες σε θέματα συμβατότητας και εναρμόνισης με τις τελευταίες τεχνολογικές τάσεις. Ο λόγος έγκειται στο γεγονός ότι υπάρχουν πολλοί διαφορετικοί κατασκευαστές τέτοιων συσκευών, που θέτουν τους δικούς τους περιορισμούς και προσεγγίζουν διαφορετικά τα διάφορα τεχνολογικά θέματα. Αυτό έχει επίπτωση στην υλοποίηση της εφαρμογής, που πρέπει να είναι συμβατή με όσο το δυνατόν περισσότερα είδη συσκευών. Θέτοντας κάποιους περιορισμούς και κάνοντας κάποιες παραδοχές, κατάφερα να δημιουργήσω τελικά μία εφαρμογή που είναι συμβατή με το 94% των συσκευών με λειτουργικό σύστημα Android (Απρίλιος 2015). Όμως, σε αυτό το χώρο, οι τεχνολογικές εξελίξεις είναι ραγδαίες. Ακόμα κι αυτή την ώρα, κάποιες νέες βιβλιοθήκες δημιουργούνται, που μπορεί να απλοποιούν και να βελτιστοποιούν κάποιες διαδικασίες, ενώ κάποιες άλλες που μπορεί να έχω χρησιμοποιήσει να καταργούνται. Για παράδειγμα, στις 28 Μαΐου 2015 άλλαξε ο τρόπος που γίνεται η εγγραφή των συσκευών στην υπηρεσία GCM (πλέον γίνεται μέσω *token*), η οποία βελτιστοποιεί τη διαδικασία και επιπλέον την καθιστά πιο ασφαλή. Επίσης, άλλαξε το περιβάλλον ανάπτυξης κώδικα, από Eclipse, το οποίο παρουσίαζε σημαντικά προβλήματα, πλέον το επίσημα υποστηριζόμενο περιβάλλον είναι το Android Studio. Βλέπουμε λοιπόν, ότι τέτοιου είδους εφαρμογές θέλουν συνεχώς αναβάθμιση για να μπορούν να θεωρούνται αξιόπιστες και σύγχρονες.

8.2 Συμπεράσματα

Η πρόσβαση στο διαδίκτυο μέσω έξυπνων συσκευών συνεχώς αυξάνεται. Πλέον όλες αυτές οι συσκευές προσφέρουν δυνατότητα σύνδεσης στο διαδίκτυο, με αποτέλεσμα να είναι πολύ ενδιαφέρουσα η διερεύνηση της ποιότητάς του και της εμπειρίας του χρήστη. Η εφαρμογή που δημιουργήθηκε αποτελεί μία ολοκληρωμένη προσπάθεια ανάπτυξης μίας πλατφόρμας πληθοπορισμού, που λειτουργεί σε έξυπνες συσκευές για τη συλλογή χωρικών δεδομένων, με σκοπό την αξιολόγηση των ασύρματων δικτύων. Η δυνατότητα αποτύπωσης των μετρήσεων σε γεωγραφικούς χάρτες και η πληθοποριστική (*crowdsourced*) του ιδιότητα, μπορούν να το καταστήσουν εργαλείο για τους χρήστες αλλά και για τις εταιρείες κινητής τηλεφωνίας για την

αξιολόγηση της προσβασιμότητας στο διαδίκτυο. Αυτό το συμπέρασμα ενισχύει και η τάση που υπάρχει για ελαχιστοποίηση των Drive Test (MDT), όπως έχουμε αναφέρει προηγουμένως.

8.3 Επεκτάσεις

Οι επεκτάσεις που μπορούν να γίνουν στην εφαρμογή εξαρτώνται από το που θέλουμε να δώσουμε περισσότερη έμφαση. Μία προφανής επέκταση θα ήταν η υλοποίηση της εφαρμογής σε λειτουργικό σύστημα iOS για διεύρυνση του πλήθους των υποστηριζόμενων έξυπνων συσκευών.

Θα μπορούσαμε επίσης, να δώσουμε περισσότερη έμφαση στα στατιστικά μετρήσεων. Για να γίνει αυτό απαιτείται η υλοποίηση περισσότερων οθονών με γραφικά διαγράμματα. Επίσης, θα μπορούσαμε να δημιουργήσουμε διαγράμματα, στα οποία να γίνεται σύγκριση μετρήσεων στις ίδιες περιοχές σε διαφορετικές χρονικές περιόδους είτε ημερολογιακά είτε και μέσα στο 24ωρο. Θα ήταν πολύ ενδιαφέρον να διερευνήσουμε την απόδοση δικτύου και το φόρτο στη διάρκεια της ημέρας ή της εβδομάδας ή του μήνα. Επίσης, θα μπορούσαμε να δείχνουμε στοιχεία που αφορούν παλαιότερες ανενεργές καμπάνιες που έχουν γίνει στις ίδιες περιοχές και να τις συγκρίνουμε με τις αντίστοιχες ενεργές, ώστε να απορρέουν συμπεράσματα όσον αφορά το ιστορικό της περιοχής.

Όσον αφορά τις καμπάνιες, θα μπορούσαμε να προσθέσουμε κι άλλα κριτήρια για την επίτευξη των μετρήσεων. Για παράδειγμα, να υπάρχει η επιπλέον δυνατότητα μέτρησης με βάση την απόσταση που έχει διανύσει η συσκευή, κάτι όμως που αλλάζει τη λογική με την οποία έχει δημιουργηθεί η υπηρεσία παρασκήνιου (*Alarm Manager*). Επίσης, όσον αφορά την ανίχνευση της θέσης της συσκευής στα όρια μίας καμπάνιας, υπάρχει μία υπηρεσία της Google η οποία ονομάζεται *Geofencing*²⁴, με την οποία ορίζουμε γεωγραφικές κυκλικές περιοχές (συντεταγμένες σημείου και ακτίνα) και οι οποίες θα μπορούσαν να χρησιμοποιηθούν για τον γεωγραφικό ορισμό των καμπανιών. Οι συσκευές που βρίσκονται στο εσωτερικό αυτών των περιοχών, στην ουσία βρίσκονται μέσα στα όρια μίας καμπάνιας και θα μπορούσε να πραγματοποιηθεί μέτρηση.

Εάν θέλαμε να εστιάσουμε στην επέκταση των μετρικών που χρησιμοποιούμε για τις μετρήσεις, θεωρώ ότι έχουμε χρησιμοποιήσει τις σημαντικότερες όσον αφορά την ποιότητα απόδοσης των

²⁴ <https://developer.android.com/training/location/geofencing.html>

ασύρματων δικτύων. Θα μπορούσαμε να επεκτείνουμε το πεδίο των μετρικών χρησιμοποιώντας μεγέθη που λαμβάνονται από τους αισθητήρες των συσκευών, όπως για παράδειγμα τη μέτρηση της ηχορύπανσης.

Επίσης, όσον αφορά τις μετρήσεις κατά τη λειτουργία της υπηρεσίας παρασκηνίου, αυτές πάντα γίνονται στα πλαίσια μίας μόνο καμπάνιας. Θα μπορούσαμε να επεκτείνουμε τη δυνατότητα μετρήσεων στα πλαίσια περισσότερων καμπανιών ταυτόχρονα, εάν ικανοποιούνται οι συνθήκες. Αυτό θα γινόταν εάν ορίζαμε καμπάνια που τα χωρικά όριά της επικαλύπτονται από τα όρια κάποιας άλλης, με σχεδόν ίδια χαρακτηριστικά.

Όσον αφορά τη βάση δεδομένων, υπήρξε μία τεχνική δυσκολία στη δημιουργία του πίνακα *measurements*. Ο συγκεκριμένος πίνακας απαιτεί τη δημιουργία ενός ευρετηρίου (index) τύπου SPATIAL, για την ομαλή σύνδεσή του με τον GeoServer. Όμως η συγκεκριμένη έκδοση της MySQL (5.5.41) δεν υποστηρίζει πίνακες τύπου InnoDB με index τύπου SPATIAL, οπότε ο συγκεκριμένος πίνακας δημιουργήθηκε ως τύπου MyISAM. Το αποτέλεσμα είναι να μην δημιουργηθούν ξένα κλειδιά (πχ *camp_id*) στον συγκεκριμένο πίνακα για σύνδεση με τους άλλους πίνακες της βάσης, όπως φαίνεται και στην Εικόνα 6.28. Βέβαια, αυτή η παράλειψη δεν έχει κάποια επίπτωση στη λειτουργία της Android εφαρμογής. Παρ'όλα αυτά, εάν θέλουμε να διορθώσουμε την παράλειψη, θα πρέπει να εγκαταστήσουμε νέα έκδοση της MySQL (5.7.4), η οποία πλέον υποστηρίζει ευρετήρια τύπου SPATIAL σε InnoDB πίνακες²⁵.

Τέλος, εάν θέλαμε να προσωποποιήσουμε περισσότερο την πληροφορία από τις μετρήσεις, θα μπορούσαμε να δημιουργήσουμε λογαριασμούς χρηστών μέσα από τα μέσα κοινωνικής δικτύωσης, όπως πχ το facebook ή μέσω gmail, ώστε να μην έχουμε και την απαίτηση για άδεια χρήσης των προσωπικών δεδομένων (χρήση IMEI συσκευής).

²⁵ <http://mysqlserverteam.com/innodb-spatial-indexes-in-5-7-4-lab-release/>

Βιβλιογραφία - Αναφορές

- [1] Κορωνιώτης Νικόλαος, "Ανάπτυξη εφαρμογής και δικτυακής πλατφόρμας μέτρησης απόδοσης ασύρματων δικτύων για συσκευές Android," Χαροκόπειο Πανεπιστήμιο, Αθήνα, Bsc Thesis openarchives.gr/visit/2509464, 2014.
- [2] wikipedia.org. (2014, Δεκέμβριος) Smartphones. [Online].
<http://en.wikipedia.org/wiki/Smartphone>
- [3] Adriano Faggiani, Enrico Gregori, Luciano Lenzini, Valerio Luconi, and Alessio Vecchio, *Smartphone-based crowdsourcing for network monitoring: opportunities, challenges, and a case study*. University of Pisa, Italy: IEEE, 2014.
- [4] wikipedia.org. (2014, Δεκέμβριος) Crowdsourcing. [Online].
<http://en.wikipedia.org/wiki/Crowdsourcing>
- [5] Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Pérez, and and Nevine Labib Georggi, *A General Architecture in Support of Interactive, Multimedia, Location-Based Mobile Applications*. University of South Florida: IEEE, 2006.
- [6] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz, *Location-based Crowdsourcing: Extending Crowdsourcing to the Real World*. University of Duisburg-Essen: ACM, 2010.
- [7] Amazon. (2015, January) Mechanical Turk. [Online]. <https://www.mturk.com>
- [8] Getty Images. (2015, January) iStock. [Online]. <http://www.istockphoto.com/>
- [9] Nathan Eagle, "txteagle: Mobile Crowdsourcing," in *Internationalization, Design and Global Development*, Nuray Aykin, Ed. San Diego, USA: Springer Berlin Heidelberg, 2009, ch. 5, pp. 447-456.
- [10] Fedor Chernogorov and Jani Puttonen, *User Satisfaction Classification for Minimization of Drive Tests QoS Verification*. Jyväskylä, Finland: IEEE, 2013.
- [11] 3GPP. (2014, December) 3GPP. [Online]. <http://www.3gpp.org/>
- [12] en.wikipedia.org. (2014, Δεκέμβριος) LTE. [Online].
[http://en.wikipedia.org/wiki/LTE_\(telecommunication\)](http://en.wikipedia.org/wiki/LTE_(telecommunication))
- [13] wikipedia.org. (2014, Δεκέμβριος) Wikipedia(k-nearest neighbors algorithm). [Online].
http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- [14] Google. (2015, Φεβρουάριος) Google Maps. [Online]. <https://www.google.gr/maps>
- [15] Google. (2015, Φεβρουάριος) GCM (Google Cloud Messaging). [Online].
<https://developer.android.com/google/gcm/index.html>

- [16] Boundless. (2015, Ιανουάριος) OpenGeo Suite/Barnes Surface. [Online].
<http://suite.opengeo.org/4.1/cartography/rt/barnes.html>
- [17] el.wikipedia.org. (2015, Απρίλιος) Google. [Online]. <http://el.wikipedia.org/wiki/Google>
- [18] open handset alliance. (2015, Απρίλιος) android. [Online].
http://www.openhandsetalliance.com/android_overview.html
- [19] el.wikipedia.org. (2015, Απρίλιος) Android. [Online]. <http://el.wikipedia.org/wiki/Android>
- [20] tutorialspoint. (2015, Απρίλιος) Android Architecture. [Online].
http://www.tutorialspoint.com/android/android_architecture.htm
- [21] Wikipedia. (2015, Απρίλιος) Android version history. [Online].
http://en.wikipedia.org/wiki/Android_version_history
- [22] Android Open Source Project. (2015, Απρίλιος) Android Activities. [Online].
<http://developer.android.com/guide/components/activities.html>
- [23] Android Open Source Project. (2015, Απρίλιος) Managing the Activity Lifecycle. [Online].
<http://developer.android.com/training/basics/activity-lifecycle/index.html>
- [24] Android Open Source Project. (2015, Απρίλιος) Activity. [Online].
<http://developer.android.com/reference/android/app/Activity.html>
- [25] Vogella. (2015, Απρίλιος) Android Broadcast Receiver - Tutorial. [Online].
<http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html>
- [26] Android Open Source Project. (2015, Απρίλιος) Manipulating Broadcast Receivers On Demand. [Online]. <http://developer.android.com/training/monitoring-device-state/manifest-receivers.html>
- [27] Android Open Source Project. (2015, Απρίλιος) Services. [Online].
<http://developer.android.com/guide/components/services.html>
- [28] Android Open Source Project. (2015, Απρίλιος) AsyncTask. [Online].
<http://developer.android.com/reference/android/os/AsyncTask.html>
- [29] Android Open Source Project. (2015, Απρίλιος) Intents and Intent Filters. [Online].
<http://developer.android.com/guide/components/intents-filters.html>
- [30] Android Open Source Project. (2015, Απρίλιος) Scheduling Repeating Alarms. [Online].
<https://developer.android.com/training/scheduling/alarms.html>
- [31] Android Open Source Project. (2015, Απρίλιος) Making Your App Location-Aware. [Online]. <https://developer.android.com/training/location/index.html>
- [32] Vogella. (2014, Δεκέμβριος) Android Location API - Tutorial. [Online].

- <http://www.vogella.com/tutorials/AndroidLocationAPI/article.html>
- [33] Google. (2015, Φεβρουάριος) Implementing GCM Server. [Online].
<https://developer.android.com/google/gcm/server.html>
- [34] Nurka Gerti, "Διαδικτυακή Πλατφόρμα Συντονισμού για Σύντηξη Γεωχωρικών Δεδομένων από Μετρήσεις Απόδοσης Ασύρματων Δικτύων," Χαροκόπειο Πανεπιστήμιο, Αθήνα, Msc Thesis 2015.
- [35] Google. (2015, Φεβρουάριος) Implementing GCM Client on Android. [Online].
<https://developer.android.com/google/gcm/client.html>
- [36] Eclipse. (2014, Δεκέμβριος) ADT Plugin Release Notes. [Online].
<http://developer.android.com/tools/sdk/eclipse-adt.html>
- [37] JetBrains. (2015, Μάρτιος) Android Studio. [Online].
<https://developer.android.com/sdk/index.html>
- [38] Google. (2014, Δεκέμβριος) Google Play Services. [Online].
<https://developer.android.com/google/play-services/index.html>
- [39] Google. (2014, Δεκέμβριος) Android SDK Manager. [Online].
<https://developer.android.com/tools/help/sdk-manager.html>
- [40] Oracle. (2015, Ιανουάριος) NetBeans IDE. [Online]. <https://netbeans.org/downloads/>
- [41] Google. (2015, Φεβρουάριος) GCM Overview. [Online].
<https://developer.android.com/google/gcm/gcm.html>
- [42] Android Open Project. (2014, Δεκέμβριος) Google Services - Getting Started on Android. [Online]. <https://developer.android.com/google/gcm/gs.html#create-proj>
- [43] wikipedia. (2015, Απρίλιος) Representational state transfer (REST). [Online].
http://en.wikipedia.org/wiki/Representational_state_transfer
- [44] Pearson eCollege, @tfredrich Todd Fredrich. (2013, August) REST API Tutorial. [Online].
<http://www.restapitutorial.com/resources.html>
- [45] MYSQL AB. (2015, Απρίλιος) MySQL.com. [Online]. <http://www.mysql.com/products/>
- [46] wikipedia. (2015, Απρίλιος) MySQL. [Online]. <http://en.wikipedia.org/wiki/MySQL>
- [47] phpMyAdmin Team. (2015, Απρίλιος) Bringing MySQL to the web. [Online].
http://www.phpmyadmin.net/home_page/index.php
- [48] bitnami. (2014, Δεκέμβριος) LAMP Stack Installers. [Online].
<https://bitnami.com/stack/lamp/installer>
- [49] Android Open Source Project. (2015, Απρίλιος) Accessing Google APIs. [Online].

<https://developer.android.com/google/auth/api-client.html>

[50] Google. (2015, Απρίλιος) Google Maps Android API. [Online].

<https://developers.google.com/maps/documentation/android/start>

[51] Apache. (2004, Ιανουάριος) Apache License. [Online].

<http://www.apache.org/licenses/LICENSE-2.0>

[52] Philipp Jahoda. (2015, Μάρτιος) PhilJay/MPAndroidChart. [Online].

<https://github.com/PhilJay/MPAndroidChart>

[53] Open Source Geospatial Foundation. (2014, Νοέμβριος) GeoServer. [Online].

<http://geoserver.org/>

[54] Amey Raut. (2012, Σεπτέμβριος) 50 Awesome & useful Android custom button style : Set 1.

[Online]. <http://www.mindfreakerstuff.com/2012/09/50-useful-android-custom-button-style-set-1/#button-set1>

[55] Android Open Source Project. (2014, Δεκέμβριος) PreferenceActivity. [Online].

<http://developer.android.com/reference/android/preference/PreferenceActivity.html>

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα Α: Android Project's manifest

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="gr.soulis.huaservicetester2"
    android:versionCode="2"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="21" />

    <permission
        android:name="gr.soulis.huaservicetester2.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <!-- Use OpenGL ES version 2 for Google Maps -->
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />
    <uses-feature android:name="android.hardware.wifi" />

    <!-- Use permission for Google Maps (ref:
https://developers.google.com/maps/documentation/android/start#getting\_the\_google\_maps\_android\_api\_v2) -->
    <uses-permission android:name="gr.soulis.huaservicetester2.permission.MAPS_RECEIVE" />
    <!-- Permission for accessing the Internet -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <!-- Permission for writing to external storage -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <!-- Use permission for Google Maps -->
    <uses-permission
        android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <!-- Permissions for user's location -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <!-- Permissions for deviceId -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <!-- GCM requires a Google account. (ref:
https://code.google.com/p/gcm/source/checkout) -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />

    <!-- Keeps the processor from sleeping when a message is received. -->
    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <!--
    Creates a custom permission so only this app can receive its messages.
    NOTE: the permission *must* be called PACKAGE.permission.C2D_MESSAGE,
        where PACKAGE is the application's package name.
    -->
    <permission
        android:name="gr.soulis.huaservicetester2.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />
```

```

<uses-permission android:name="gr.soulis.huaservicetester2.permission.C2D_MESSAGE" />

<!-- This app has permission to register and receive data message. -->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".MainActivity"
        android:configChanges="orientation|keyboardHidden"
        android:label="@string/app_name"
        android:screenOrientation="portrait" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".TracerActivity"
        android:label="@string/title_activity_tracer" >
    </activity>
    <activity
        android:name=".SettingsActivity"
        android:label="@string/action_settings" >
    </activity>

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyAYDLBS0MiT4Nb01Ezo09kf8v04ZCNk2k" />

    <activity
        android:name=".GoogleMapsActivity"
        android:configChanges="orientation|keyboardHidden"
        android:label="@string/title_activity_google_maps"
        android:parentActivityName=".SubMainActivity"
        android:screenOrientation="portrait"
        android:uiOptions="splitActionBarWhenNarrow" >
        <meta-data
            android:name="android.support.UI_OPTIONS"
            android:value="splitActionBarWhenNarrow" />
        <!-- Parent activity meta-data to support API level 7+ -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=".SubMainActivity" />
    </activity>
    <activity
        android:name=".SubMainActivity"
        android:configChanges="orientation|keyboardHidden"
        android:label="@string/button_map"
        android:screenOrientation="portrait" >
    </activity>
    <activity
        android:name=".OnDemandActivity"
        android:configChanges="orientation|keyboardHidden"
        android:label="@string/title_activity_on_demand"
        android:parentActivityName=".MainActivity"
        android:screenOrientation="portrait" >

        <!-- Parent activity meta-data to support API level 7+ -->

```

```

        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=".MainActivity" />
    </activity>
    <activity
        android:name=".MessageActivity"
        android:label="@string/title_activity_message" >
    </activity>
    <activity
        android:name=".CampaignsActivity"
        android:label="@string/title_activity_campaigns"
        android:parentActivityName=".MainActivity" >

        <!-- Parent activity meta-data to support API level 7+ -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=".MainActivity" />
    </activity>

    <activity
        android:name=".stats.PieChartActivity"
        android:label="@string/title_activity_pie_chart"
        android:parentActivityName="gr.soulis.huaservicetester2.SubMainActivity"
    >

        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="gr.soulis.huaservicetester2.SubMainActivity" />
    </activity>
    <activity
        android:name=".stats.PieOpNetworkActivity"
        android:label="@string/title_activity_pie_NetworkTypes"
        android:parentActivityName="gr.soulis.huaservicetester2.SubMainActivity"
    >

        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="gr.soulis.huaservicetester2.SubMainActivity" />
    </activity>
    <activity
        android:name=".stats.MultiBarChartActivity"
        android:label="@string/title_activity_multi_bar_chart"
        android:configChanges="orientation|keyboardHidden"
        android:screenOrientation="Landscape"
        android:parentActivityName="gr.soulis.huaservicetester2.SubMainActivity"
        android:uiOptions="splitActionBarWhenNarrow" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="gr.soulis.huaservicetester2.SubMainActivity" />
        <meta-data
            android:name="android.support.UI_OPTIONS"
            android:value="splitActionBarWhenNarrow" />
    </activity>
    <activity
        android:name=".stats.MultiBarChartOpActivity"
        android:label="@string/title_activity_multi_bar_chart_op"
        android:configChanges="orientation|keyboardHidden"
        android:screenOrientation="Landscape"
        android:parentActivityName="gr.soulis.huaservicetester2.SubMainActivity"
        android:uiOptions="splitActionBarWhenNarrow" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="gr.soulis.huaservicetester2.SubMainActivity" />
        <meta-data
            android:name="android.support.UI_OPTIONS"
            android:value="splitActionBarWhenNarrow" />
    </activity>

    <service
        android:exported="false"

```

```

        android:enabled="true"
        android:name=".TestService">

</service>

<!--
(ref: https://code.google.com/p/gcm/source/checkout)
WakefulBroadcastReceiver that will receive intents from GCM
services and hand them to the custom IntentService.

The com.google.android.c2dm.permission.SEND permission is necessary
so only GCM services can send data messages for the app.

-->
<receiver
    android:name=".GcmBroadcastReceiver"
    android:enabled="true"
    android:exported="true"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>

        <!-- Receives the actual messages. -->
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />

        <category android:name="gr.soulis.huaservicetester2" />
    </intent-filter>
</receiver>

<service android:name=".GcmIntentService" />

<receiver
    android:name=".ActiveConnReceiver"
    android:enabled="true" >
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        <action android:name="android.net.wifi.WIFI_STATE_CHANGED" />
    </intent-filter>
</receiver>

</application>

</manifest>

```

Παράρτημα Β: Κώδικας υπηρεσίας παρασκηνίου

TestService.java

```
package gr.soulis.huaservicetester2;

import java.io.IOException;
import java.text.ParseException;

import org.json.JSONException;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;

import android.annotation.SuppressLint;
import android.app.Service;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.os.Bundle;
import android.os.IBinder;
import android.os.PowerManager;
import android.os.PowerManager.WakeLock;
import android.provider.Settings;
import android.util.Log;
import android.widget.Toast;

public class TestService extends Service implements GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener, com.google.android.gms.location.LocationListener{

    LocationRequest mLocationRequest = null;
    GoogleApiClient mGoogleApiClient = null;

    boolean registeredUpdates = false;
    boolean runningService = true;

    String activeConn = null;

    public String prefs_network_Type;
    public int prefs_mobile_limit;
    public int prefs_mobile_freq;
    public int prefs_wifi_freq;

    // the selected campaign
    private int camp_id = 0;
    private boolean found_campaign = false;
    private boolean start_button = true;

    final static String TAG = "TestService";

    private CampaignJson campJson = null;

    String mes="def";
    static boolean wifiSleepPolicyChanged = false;
    int sStartId = 0;

    PowerManager powerManager = null;
    WakeLock wakeLock = null;
```

```

@Override
public void onCreate() {
    // TODO Auto-generated method stub
    super.onCreate();
    Log.i(TAG, "Service is creating!! ");
    /// FOR VERIFY THAT THE SERVICE WILL STOP IF STOP BUTTON PUSHED
    TestServiceFile tsf = new TestServiceFile(getApplicationContext());
    try {
        if(!tsf.isMyServiceBtnRunning()){
            start_button = false;
            Log.i(TAG, "SERVICE: start_button is FALSE!! ");
        }
    }catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        Toast.makeText(getApplicationContext(), "Cannot open File for checking
if Service is running!! - "+e.getStackTrace().toString(), Toast.LENGTH_LONG).show();
        Log.e(TAG, "Cannot open File for checking if Service is running!! -
"+e.getStackTrace().toString());
    }
}

@SuppressWarnings("deprecation")
@SuppressLint("InlinedApi")
@Override
public int onStartCommand(Intent intent, int flags, int startId) {

    sStartId = startId;

    if(!start_button){
        runningService = false;
        mes = "Pushed Stop Button. The Service is stopping!! - ";
        Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
        Log.i(TAG, "Pushed Stop Button. The Service is stopped!! ");
        stopSelf();
    }

    @SuppressWarnings("unused")
    Bundle sharedPrefBundle = MyPreferences.getUserPrefs(getApplicationContext());

    prefs_network_Type = MyPreferences.prefs_network_Type;
    prefs_mobile_limit = MyPreferences.prefs_mobile_limit;
    prefs_mobile_freq = MyPreferences.prefs_mobile_freq;
    prefs_wifi_freq = MyPreferences.prefs_wifi_freq;

    // check if current networkType meets the preferences
    try {
        if(checkNetwork() == false){
            runningService = false;
            mes = "No appropriate Network Type for Service!!";
            Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
            Log.i(TAG, "No appropriate Network Type for Service");
            stopSelf();
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        mes = e.getMessage();
        Log.i(TAG, mes);
        Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
        stopSelf(startId);
    }

    // if Service is running in WiFi mode, Wifi must be always ON
    if(runningService){
        if(!activeConn.equals("mobile")){

```

```

        if(Settings.System.getInt(getContentResolver(), Settings.Global.WIFI_SLEEP_POLICY, 0)
!= 0){ // for API >= 17
            if(Settings.System.getInt(getContentResolver(),
Settings.Global.WIFI_SLEEP_POLICY, Settings.Global.WIFI_SLEEP_POLICY_NEVER) !=
Settings.Global.WIFI_SLEEP_POLICY_NEVER){
                Settings.System.putInt(getContentResolver(),
Settings.Global.WIFI_SLEEP_POLICY, Settings.Global.WIFI_SLEEP_POLICY_NEVER);
                wifiSleepPolicyChanged = true;
            }
        }
        else{ // for API < 17
            if(Settings.System.getInt(getContentResolver(),
Settings.System.WIFI_SLEEP_POLICY, Settings.System.WIFI_SLEEP_POLICY_NEVER) !=
Settings.System.WIFI_SLEEP_POLICY_NEVER){
                Settings.System.putInt(getContentResolver(),
Settings.System.WIFI_SLEEP_POLICY, Settings.System.WIFI_SLEEP_POLICY_NEVER);
                wifiSleepPolicyChanged = true;
            }
        }
    }

    // if running in mobile mode check usage limit and STOP Service if data usage exceeded the
limit
    if(activeConn.equals("mobile")){

        try {
            }{
            if(Float.valueOf(MobileUsageFile.readMobileUsageFile(getApplicationContext()).floatVal
ue()*TracerActivity.DATA_TEST_CAP > prefs_mobile_limit){
                mes = "The Mobile Data Usage exceeded the limit of
"+Integer.toString(prefs_mobile_limit)+" MB!! The Service stopped!";
                Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
                stopSelf(startId);
            }

        } catch (NumberFormatException | IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            mes = e.getMessage();
            Log.e(TAG, mes);
            Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
            stopSelf(startId);
        }
    }

    if(mGoogleApiClient == null){
        mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addApi(LocationServices.API)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .build();
    }

    if(mLocationRequest == null){
        // Create the LocationRequest object
        mLocationRequest = LocationRequest.create();
        // Use high accuracy
        mLocationRequest.setPriority(
            LocationRequest.PRIORITY_HIGH_ACCURACY);
        // Set the update interval according to preferences
        if(activeConn.equals("mobile"))
            mLocationRequest.setInterval(prefs_mobile_freq*1000);
        else
            mLocationRequest.setInterval(prefs_wifi_freq*1000);

        // Set the fastest update interval to 5 minutes
        mLocationRequest.setFastestInterval(300000);
    }

```

```

        if(!mGoogleApiClient.isConnected())
            mGoogleApiClient.connect();
    }
    else
    {
        Toast.makeText(getApplicationContext(), "Your Preferences doesn't meet
the current Network Type!", Toast.LENGTH_SHORT).show();
        stopSelf(startId);
    }

    return Service.START_REDELIVER_INTENT;
}

@SuppressWarnings("deprecation")
@SuppressWarnings("InlinedApi")
@Override
public void onDestroy() {
    Log.d(TAG, "onDestroyService");

    if(wifiSleepPolicyChanged){
        if(Settings.System.getInt(getContentResolver(),
Settings.Global.WIFI_SLEEP_POLICY, 0) != 0){ // for API >= 17
            Settings.System.putInt(getContentResolver(),
Settings.Global.WIFI_SLEEP_POLICY, Settings.Global.WIFI_SLEEP_POLICY_DEFAULT);
        }
        else{ // for API < 17
            Settings.System.putInt(getContentResolver(),
Settings.System.WIFI_SLEEP_POLICY, Settings.System.WIFI_SLEEP_POLICY_DEFAULT);
        }
    }

    /*
    * Remove location updates for a listener.
    * The current Activity is the listener, so
    * the argument is "this".
    */
    if(mGoogleApiClient != null && registeredUpdates == true){
        LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        registeredUpdates = false;
    }
    // If the client is connected
    if(mGoogleApiClient != null){
        if (mGoogleApiClient.isConnected()) {
            /*
            * After disconnect() is called, the client is
            * considered "dead".
            */
            mGoogleApiClient.disconnect();
        }
    }

    runningService = false;

    super.onDestroy();
}

@Override
public IBinder onBind(Intent intent) {
    // TODO Auto-generated method stub
    return null;
}

```

```

@Override
public void onLocationChanged(Location location) {
    // TODO Auto-generated method stub
    Log.d(TAG, "Location changed");
    Log.i(TAG, "Lat:"+location.getLatitude()+"-Long:"+location.getLongitude()+" -
Accuracy:"+location.getAccuracy());

    /// CHECK FOR THE APPROPRIATE CAMPAIGN -----
    try {
        Bundle bundle = campJson.findCampaign(activeConn, location);
        if(bundle != null){
            found_campaign = true;
            registeredUpdates = true;
            camp_id = bundle.getInt("camp_id", 0);
        }
    } catch (IOException | JSONException | ParseException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
        found_campaign = false;
        registeredUpdates = false;
        mes = e1.getMessage();
        Log.e(TAG, mes);
        Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
        stopSelf(sStartId);
        return;
    }

    if(found_campaign){
        System.gc();
        mes = "Service found Campaign. Proceed to Measurements!";
        Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
        try{
            SpeedServiceTester st=new SpeedServiceTester(getApplicationContext());
            st.setLocationTest(location.getLatitude(), location.getLongitude(),
location.getAccuracy(), location.getSpeed(), camp_id);
            /// START MEASUREMENTS -----
            -----
            st.startCommand();
        }
        catch(IOException ie){
            ie.printStackTrace();
            mes = ie.getMessage();
            Log.e(TAG, mes);
            Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
            stopSelf(sStartId);
            return;
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            mes = e.getMessage();
            Log.e(TAG, mes);
            Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
            stopSelf(sStartId);
            return;
        }
    }
    else{
        mes = "Service NOT found any appropriate active Campaign!";
        Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
        stopSelf();
    }

    stopSelf();
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {

```

```

        // TODO Auto-generated method stub
        Log.d(TAG, "onStatusChanged");
    }

    @Override
    public void onConnected(Bundle arg0) {
        // TODO Auto-generated method stub
        Log.d(TAG, "onConnected");

        // retrieve Campaigns
        campJson = new CampaignJson(getApplicationContext());
        try {
            if(!campJson.createJsonObj()){
                mGoogleApiClient.disconnect();
                return;
            }
        } catch (IOException | JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            mes = e.getMessage();
            Log.i(TAG, mes);
            //showToast();
            Toast.makeText(getApplicationContext(), mes, Toast.LENGTH_SHORT).show();
            stopSelf(sStartId);
            return;
        }

        if(mGoogleApiClient != null){
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
            registeredUpdates = true;
        }
    }

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {
        // TODO Auto-generated method stub
        Log.d(TAG, "onConnectionFailed");

        /*
        * Google Play services can resolve some errors it detects.
        * If the error has a resolution, try sending an Intent to
        * start a Google Play services activity that can resolve
        * error.
        */
        if (connectionResult.hasResolution()) {
            if (connectionResult.hasResolution()) {
                if(mGoogleApiClient == null){
                    mGoogleApiClient = new GoogleApiClient.Builder(this)
                        .addApi(LocationServices.API)
                        .addConnectionCallbacks(this)
                        .addOnConnectionFailedListener(this)
                        .build();
                }
                mGoogleApiClient.connect();
            }
        } else {
            /*
            * If no resolution is available, display a dialog to the
            * user with the error.
            */
            Intent i = new Intent();

```

```

        i.setClassName("gr.soulis.huaservicetester2",
"gr.soulis.huaservicetester2.MessageActivity");
        startActivity(i);

        stopSelf();
    }
}

@Override
public void onConnectionSuspended(int arg0) {
    // TODO Auto-generated method stub
    Log.d(TAG, "onConnectionSuspended");

    if(mGoogleApiClient != null && registeredUpdates == true){
        LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        registeredUpdates = false;
    }

    stopSelf();
}

@Override
public void onProviderEnabled(String provider) {
    // TODO Auto-generated method stub
    Toast.makeText(getApplicationContext(), "onProviderEnabled",
Toast.LENGTH_SHORT).show();
    Log.d(TAG, "onProviderEnabled");
}

@Override
public void onProviderDisabled(String provider) {
    // TODO Auto-generated method stub
    Toast.makeText(getApplicationContext(), "onProviderDisabled",
Toast.LENGTH_SHORT).show();
    Log.d(TAG, "onProviderDisabled");
    /*
    * Remove location updates for a listener.
    * The current Activity is the listener, so
    * the argument is "this".
    */
    if(mGoogleApiClient != null){
        if(registeredUpdates == true){
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
            registeredUpdates = false;
        }

        if (mGoogleApiClient.isConnected()) {
            /*
            * After disconnect() is called, the client is
            * considered "dead".
            */
            mGoogleApiClient.disconnect();
        }
    }
}

private boolean checkNetwork() throws IOException{
    TestServiceFile tsf = new TestServiceFile(getApplicationContext());
    activeConn = tsf.activeConnection();
}

```

```

// if network not exists
if(activeConn == null || activeConn.isEmpty()){
    return false;
}

if(!prefs_network_Type.equalsIgnoreCase("both")){
    if(prefs_network_Type.equalsIgnoreCase(activeConn))
        return true;
    else
        return false;
}

return true;
}
}

```