ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Παροχή Υπηρεσιών Βασισμένων στην Θέση με χρήση της πλατφόρμας Google Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΣΤΑΥΡΟΥ Η. ΖΩΤΑΛΗ

Επιβλέπων: Νικόλαος Μήτρου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Παροχή Υπηρεσιών Βασισμένων στην Θέση με χρήση της πλατφόρμας Google Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΣΤΑΥΡΟΥ Η. ΖΩΤΑΛΗ

Επιβλέπων: Νικόλαος Μήτρου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την .....

.....	.....	.....
Μιχαήλ Θεολόγου Καθηγητής Ε.Μ.Π.	Νικόλαος Μήτρου Καθηγητής Ε.Μ.Π.	Συμεών Παπαβασιλείου Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009.

.....

ΣΤΑΥΡΟΣ Η. ΖΩΤΑΛΗΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Πρόλογος

Θα ήθελα να ευχαριστήσω θερμά τον κ. Νικόλαο Μήτρου, καθηγητή του τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής του Εθνικού Μετσόβιου Πολυτεχνείου, για την συμπαράσταση του και τη συνδρομή του στην εκπόνηση αυτής της εργασίας.

Επιπλέον θεωρώ υποχρέωση μου να ευχαριστήσω και τον υποψήφιο Διδάκτορα κ. Αναστάσιο Ζαφειρόπουλο για την πολύτιμη καθοδήγηση που μου παρείχε καθ' όλη την διάρκεια της διπλωματικής εργασίας

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για την συμπαράσταση που μου προσέφεραν όλα τα χρόνια των σπουδών μου.

Αθήνα, 15 Ιουλίου 2009
Σταύρος Ζώταλης

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη των Υπηρεσιών Βασισμένων στην Θέση του χρήστη και η υλοποίηση μιας τέτοιου είδους υπηρεσίας για την πλατφόρμα κινητών συσκευών Google Android.

Οι Υπηρεσίες Βασισμένες στην Θέση αποτελούν έναν αναπτυσσόμενο κλάδο του τομέα της πληροφορικής και των ασύρματων δικτύων και αναμένεται να παίξουν σημαντικό ρόλο στην αγορά παροχής ασύρματων υπηρεσιών. Οι αποφάσεις που λαμβάνει κανείς για την δημιουργία μιας τέτοιας υπηρεσίας κατά τον σχεδιασμό και την υλοποίηση της, δεν πρέπει να είναι τυχαίες. Αντιθέτως πρέπει να ληφθούν υπόψη ορισμένες παράμετροι που αφορούν το είδος της υπηρεσίας και τα επιθυμητά χαρακτηριστικά της. Αυτές τις παραμέτρους εξετάζουμε στα πλαίσια της παρούσας διπλωματικής εργασίας.

Άρρητα συνδεδεμένες με τις Υπηρεσίες Βασισμένες στην Θέση είναι και οι τεχνολογίες εντοπισμού θέσης, οι οποίες παρέχουν τον τρόπο για τον προσδιορισμό της θέσης μιας κινητής συσκευής. Σωστή επιλογή της υποκείμενης τεχνολογίας εντοπισμού θέσης παίζει μεγάλο ρόλο για την επιτυχημένη λειτουργία μιας Υπηρεσίας Βασισμένης στην Θέση. Για το λόγο αυτό στην παρούσα διπλωματική εργασία γίνεται μια σύντομη περιγραφή των κυριότερων τεχνικών ανίχνευσης θέσης με έμφαση στα πλεονεκτήματα και τα μειονεκτήματα της κάθε μιας προκειμένου μια τέτοια επιλογή να είναι ευκολότερη.

Τέλος γίνεται μια παρουσίαση της πλατφόρμας Google Android, πάνω στην οποία υλοποιήθηκε μια Υπηρεσία Βασισμένη στην Θέση με χρήση GPS. Η υπηρεσία αυτή αφορά τον εντοπισμό της θέσης μιας κινητής συσκευής, με βάση τις επιλογές του χρήστη την προβολή των κοντινότερων σημείων ενδιαφέροντος και συμπληρωματικών πληροφοριών σχετικών με αυτά τα σημεία.

Λέξεις Κλειδιά: εντοπισμός θέσης, υπηρεσίες βάσει θέσης, android, ασύρματες υπηρεσίες, τεχνολογίες εντοπισμού

Abstract

The purpose of this diploma thesis is the study of Location Based Services and the implementation of such a service on Google Android mobile phone platform.

The Location Based Services are a growing field of computer science and wireless networks. They are expected to play an important role in wireless services market. The decisions that are taken about the creation of such a service during the design and the implementation should not be random. On the contrary, some parameters regarding the type of service and the desired characteristics should be taken into account. These parameters are considered in the framework of this diploma thesis.

Strongly associated with the Location Based Services are the location detection technologies which provide the needed location information of a mobile device. The correct choice of the underlying location detection technology is essential for the successful operation of a Location Based Service. That is why in this diploma thesis, a brief presentation of the most known location detection technologies takes place, emphasizing on the advantages and the disadvantages of each one and making the correct choice easier.

Finally the Google Android platform, on which a GPS Location Based Service was developed and deployed, is presented. The service of this diploma thesis is about the tracking of a mobile phone, according to the end-user choices, the representation of the nearest points of interest and the display of additional information.

Keywords: location tracking, location based services, android, wireless services, location detection services

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Οι κινητές συσκευές και το Διαδίκτυο έφεραν επανάσταση στις επικοινωνίες και μαζί με αυτή αλλαγές στον τρόπο ζωής των ανθρώπων. Ο συνολικός αριθμός των χρηστών του Διαδικτύου έφτασε το 2007 τους 1.4 δισεκατομμύρια χρήστες, ενώ η διείσδυση των κινητών τηλεφώνων έφτασε στις 49 συσκευές ανά 100 κατοίκους παγκοσμίως και 97 συσκευές ανά 100 κατοίκους στις αναπτυγμένες χώρες [1] [2]. Οι χρήστες είναι πλέον σε θέση να έχουν πρόσβαση στο Διαδίκτυο από όπου και όποτε θέλουν και μέσω του Διαδικτύου μπορούν να αναζητήσουν πληροφορίες σχετικές με γεγονότα και τοποθεσίες.

Εκτός από το Διαδίκτυο και την κινητή τηλεφωνία, μεγάλη ανάπτυξη γνώρισαν και οι τεχνολογίες εντοπισμού θέσης. Σήμερα έχουν πλέον ωριμάσει σε τέτοιο βαθμό έτσι ώστε η εφαρμογή τους να είναι αποδοτική και εμπορικά εκμεταλλεύσιμη. Ένα από τα πιο διαδεδομένα συστήματα εντοπισμού θέσης για εξωτερικούς χώρους αποτελεί το Global Positioning System (GPS). Η διείσδυση συσκευών οι οποίες υποστηρίζουν την ανίχνευση θέσης μέσω GPS είναι μεγάλη και όλο και περισσότεροι κατασκευαστές κινητών συσκευών ενσωματώνουν στα προϊόντα τους δέκτες GPS. Πέραν του GPS όμως, το οποίο βρίσκει εφαρμογή σε εξωτερικούς χώρους κυρίως, τα τελευταία χρόνια παρατηρήθηκε και μια ραγδαία ανάπτυξη άλλων εναλλακτικών τεχνολογιών για χρήση κυρίως σε εσωτερικούς κλειστούς χώρους, όπου το GPS εμφανίζει σοβαρά μειονεκτήματα.

Η σύγκλιση αυτών των τριών τεχνολογιών (Διαδίκτυο, κινητή τηλεφωνία, εντοπισμός θέσης) έγινε εφικτή χάρις στην δημιουργία και στην μεγάλη διείσδυση στην αγορά κινητών συσκευών οι οποίες ενσωματώνουν και τις τρεις αυτές τεχνολογίες. Από την σύγκλιση αυτή δημιουργήθηκε η ιδέα των Υπηρεσιών Βασισμένων στην Θέση του χρήστη (Location Based Services ή LBS για συντομία). Στον ορισμό που δόθηκε από το Open Geospatial Consortium, μια Location Based Service είναι μια ασύρματη υπηρεσία η οποία χρησιμοποιεί πληροφορίες σχετικές με την γεωγραφική θέση για να εξυπηρετήσει έναν κινούμενο χρήστη. Δηλαδή κάθε εφαρμογή η οποία εκμεταλλεύεται την θέση ενός κινητού τερματικού για να παρέχει κάποια υπηρεσία [3].

Σκοπό της παρούσας διπλωματικής εργασίας αποτελεί η μελέτη των Υπηρεσιών Βα-

σισμένων στην Θέση του χρήστη, η μελέτη των βασικών τεχνολογιών εντοπισμού θέσης με έμφαση στο GPS και τελικά η ανάπτυξη μιας τέτοιας υπηρεσίας με βασική τεχνολογία εντοπισμού θέσης το GPS. Η Υπηρεσία Βασισμένη στην Θέση της παρούσας εργασίας έχει υλοποιηθεί πάνω στην πλατφόρμα Google Android, μια πλατφόρμα ανοικτού κώδικα για κινητές συσκευές η οποία αναπτύχθηκε από την εταιρία Google και τους συνεργάτες της. Στην επιλογή του Google Android συνηγόρησε το γεγονός οτι είναι μια πολλά υποσχόμενη πλατφόρμα, καθώς ενσωματώνει ένα σύνολο πλούσιων δυνατοτήτων και υποστηρίζει τεχνολογίες αιχμής στον τομέα των φορητών υπολογιστικών πολύ-συσκευών. Λαμβάνοντας υπόψη τα παραπάνω και δεδομένης της θέσης της Google ανάμεσα στις εταιρίες πληροφορικής και Διαδικτύου, το Android αναμένεται να παίξει πρωταγωνιστικό ρόλο στην αγορά κινητών τηλεφώνων προηγμένης γενιάς τα επόμενα χρόνια.

Επομένως η υλοποίηση LBS στην πλατφόρμα Google Android παρουσιάζει πολλές προκλήσεις, αφού συνδυάζονται δύο τεχνολογίες οι οποίες πρόκειται να απασχολήσουν στο μέλλον ένα μεγάλο ποσοστό της αγοράς κινητών τηλεφώνων και κινητών υπηρεσιών, τόσο από εμπορικής άποψης όσο και από ερευνητικής.

1.2 Οργάνωση του τόμου

Ο τόμος της παρούσας διπλωματικής εργασίας είναι οργανωμένος σε έξι κεφάλαια και τρία παραρτήματα. Σύμφωνα με την παρακάτω κατηγοριοποίηση:

- Το κεφάλαιο 1 είναι εισαγωγικό και περιγράφει σε γενικές γραμμές το αντικείμενο της διπλωματικής εργασίας.

- Στο κεφάλαιο 2 γίνεται εισαγωγή στις LBS και μια εκτενέστερη μελέτη τους. Αρχικά εξετάζονται συνοπτικά οι βασικές κατηγορίες μεθόδων ανίχνευσης θέσης, που χρησιμοποιούνται από τις LBS. Στην συνέχεια γίνεται κατηγοριοποίηση των LBS με βάση διάφορα κριτήρια και εξετάζονται ορισμένα επιθυμητά χαρακτηριστικά των LBS. Τέλος γίνεται μια αναφορά στο θέμα της ασφάλειας και της ιδιοτικότητας που ανακύπτει από την χρήση των LBS.

- Το κεφάλαιο 3 χωρίζεται σε δύο μέρη. Στο πρώτο μέρος εξετάζεται με αρκετή λεπτομέρεια η δομή και η λειτουργία του GPS, το οποίο χρησιμοποιείται και από την εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Στο δεύτερο μέρος εξετάζονται συνοπτικά άλλες τεχνολογίες ανίχνευσης θέσης, αναφέροντας κυρίως τα πλεονεκτήματα και τα μειονεκτήματα της κάθε τεχνολογίας.

- Το κεφάλαιο 4 περιέχει μια παρουσίαση της πλατφόρμας Google Android. Συγκεκριμένα εξετάζεται η αρχιτεκτονική της πλατφόρμας, γίνεται μια παρουσίαση των κύριων εργαλείων ανάπτυξης που παρέχονται από αυτήν, αναπτύσσεται το μοντέλο προγραμματισμού που αυτή ακολουθεί και τέλος περιγράφεται η διαδικασία εγκατάστασης τόσο για ανάπτυξη εφαρμογών όσο και για χρήση της πλατφόρμας.

- Το κεφάλαιο 5 περιέχει την παρουσίαση της λειτουργίας την υπηρεσίας η οποία αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Συγκεκριμένα γίνεται μια λεπτομερής περιγραφή όλων των λειτουργιών και των δυνατοτήτων τόσο της εφαρμογής που αναπτύχθηκε για να τρέχει σε συσκευές συμβατές με την πλατφόρμα του Android όσο και των εφαρμογών οι οποίες παρέχουν τις υπηρεσίες τους απομακρυσμένα στους χρήστες των συσκευών Android.
- Το κεφάλαιο 6 αποτελεί μια σύνοψη της παρούσας διπλωματικής εργασίας. Παρατίθενται συμπεράσματα που προέκυψαν καθώς επίσης και προτάσεις για πιθανή επέκταση του συστήματος που υλοποιήθηκε.
- Το παράρτημα Α περιέχει τα βασικά σημεία του κώδικα της εφαρμογής Android.
- Το παράρτημα Β περιέχει τα βασικά σημεία του κώδικα της Web εφαρμογής.
- Το παράρτημα Γ περιέχει τα CREATE statements της βάσης δεδομένων που χρησιμοποιείται ως μέσο αποθήκευσης των πληροφοριών της υπηρεσίας.

Κεφάλαιο 2

Υπηρεσίες Βασισμένες στην Θέση

2.1 Εισαγωγή

Οι Υπηρεσίες Βασισμένες στην Θέση (Location Based Services ή LBS) είναι εφαρμογές οι οποίες εκμεταλλεύονται την πληροφορία θέσης μιας τερματική συσκευής, προκειμένου να παρέχουν κάποιου είδους υπηρεσία στον κάτοχο της τερματικής συσκευής ή σε κάποιον τρίτο ενδιαφερόμενο.

Από ιστορικής άποψης η ιδέα για την παροχή Υπηρεσιών Βασισμένων στην Θέση δεν είναι καινούργια. Βασικές υπηρεσίες εντοπισμού και αυτόματης εύρεσης οχήματος (Automated Vehicle Location AVL) υπήρχαν από την δεκαετία του 80, όμως η βιομηχανία παροχής τέτοιου είδους υπηρεσιών επιβίωνε ως μια εξειδικευμένη αγορά της οποίας οι υψηλές τιμές τόσο των συσκευών ανίχνευσης όσο και των ίδιων των υπηρεσιών συμπίεζαν την ζήτηση. Μόνο τα τελευταία χρόνια έγινε δυνατή η πλήρης εμπορευματοποίηση των LBS χάρη στην τεχνολογική πρόοδο, στην αλλαγή των κανονισμών και στις νέες εμπορικές τάσεις.

Αυτήν την στιγμή οι Location Based Services αποτελούν ένα αναπτυσσόμενο κλάδο τόσο εμπορικά όσο και ερευνητικά ο οποίος εμπλουτίζεται συνεχώς με νέες ιδέες και στον οποίο ακόμα δεν έχουν επικρατήσει ορισμένα πρότυπα αλλά συνεχίζουν να διαμορφώνονται. Αξίζει λοιπόν να ασχοληθεί κανείς με τις LBS αφού αποτελούν έναν πολλά υποσχόμενο τομέα εμπορικής και ερευνητικής δραστηριότητας ο οποίος έχει ήδη αρχίσει να παίζει σημαντικό ρόλο στην καθημερινή ζωή.

2.2 Μέθοδοι ανίχνευσης θέσης

Μια η περισσότερες μέθοδοι μπορεί να χρησιμοποιηθούν για να καθοριστεί η θέση μιας κινητής συσκευής. Οι μέθοδοι ανίχνευσης λειτουργούν σε δύο φάσεις: μετρήσεις σήματος ή σημάτων και υπολογισμός της θέσης βάσει των μετρήσεων. Η ανίχνευση μπορεί να γίνει είτε από τον εξοπλισμό του χρήστη, είτε από το τηλεπικοινωνιακό δίκτυο. Τρείς είναι οι βασικές μέθοδοι ανίχνευσης θέσης η κάθε μια με τα πλεονεκτήματα και τα μειονεκτήματα της: Cell-id, προηγμένες τεχνικές ανίχνευσης με βάση το δίκτυο και τεχνικές

ανίχνευσης βασισμένες σε δορυφόρους.

2.2.1 Κυψέλη προέλευσης

Αυτή η τεχνική ανίχνευσης θέσης λειτουργεί αναγνωρίζοντας την κυψέλη του δικτύου μέσα στην οποία λειτουργεί η κινητή συσκευή. Η κυψέλη προέλευσης, η οποία αποκαλείται Cell-id, είναι η πρωταρχική τεχνολογία που χρησιμοποιείται στις μέρες μας στα ασύρματα δίκτυα. Για την χρησιμοποίηση της δεν απαιτείται καμία τροποποίηση ούτε στις κινητές συσκευές, ούτε στον εξοπλισμό του δικτύου αφού χρησιμοποιεί τους σταθμούς του δικτύου ως αναφορά της θέσης του χρήστη. Η εύρεση της θέσης του χρήστη γίνεται γρήγορα (χρειάζονται 3 δευτερόλεπτα περίπου) αλλά η ακρίβεια της θέσης είναι περιορισμένη. Πιο συγκεκριμένα η ακρίβεια της θέσης εξαρτάται από το μέγεθος της κυψέλης και από τις τεχνικές που χρησιμοποιούνται για την βελτίωση του υπολογισμού θέσης όπως αυτοπροσδιορισμός θέσης και μέτρηση του χρόνου διάδοσης. Ακρίβεια της τάξης των 150 μέτρων είναι μια συνηθισμένη τιμή για αστικές περιοχές, ενώ σε περιοχές με μικρό πληθυσμό η ακρίβεια μειώνεται ακόμα περισσότερο.

2.2.2 Προηγμένες τεχνικές ανίχνευσης με βάση το δίκτυο

Οι τεχνικές αυτές βασίζονται σε μετρήσεις σημάτων από παραπλήσιους σταθμούς οι οποίες λαμβάνονται από την συσκευή του χρήστη. Η θέση του χρήστη προκύπτει με τριγωνοποίηση, χρησιμοποιώντας τεχνικές όπως E-ODT (Enhanced Observed Time Difference) και OTDOA (Observed Time Difference Of Arrival) [4]. Η μέθοδος E-ODT λειτουργεί πάνω στο δίκτυο GSM (Global System for Mobile Communications). Παραλλαγές του E-ODT έχουν αναπτυχθεί όμως και για δίκτυα CDMA (Code Division Multiple Access) και WCDMA (Wide-band CDMA) δίκτυα όπως το AF-LT (Advanced Forward Link Triangulation) και IP-DL (Idle Period Downlink).

Η πληροφορία θέσης βρίσκεται στους σχετικούς χρόνους άφιξης σημάτων που αποστέλλονται από τους σταθμούς βάσης στην κινητή συσκευή και σε προκαθορισμένους δέκτες. Οι δέκτες θέσης η φάροι αναφοράς (Reference Beacons γνωστοί και ως Location Measuring Units LMU) τοποθετούνται στο κυψελωτό δίκτυο σε έναν αριθμό από γεωγραφικά σημεία. Η θέση μιας συσκευής μπορεί να υπολογιστεί χρησιμοποιώντας τις χρονικές διαφορές στους χρόνους άφιξης των σημάτων από τους σταθμούς βάσης στην κινητή συσκευή και στο LMU. Το E-ODT παρέχει ακρίβεια μεταξύ 50 και 125 μέτρων και έχει χρόνο απόκρισης γύρω στα 5 δευτερόλεπτα.

Αναλόγως λειτουργεί και το OTDOA υπολογίζοντας την χρονική διάφορα στην άφιξη σημάτων από τρεις διαφορετικούς σταθμούς βάσης στην κινητή συσκευή. Το υψηλό κόστος για τον συγχρονισμό του δικτύου δεν επιτρέπει μεγάλες βελτιώσεις σε σχέση με το cell-id σε αστικές περιοχές, ενώ ο χρόνος απόκρισης είναι σαφώς μεγαλύτερος

2.2.3 Τεχνικές ανίχνευσης βασισμένες σε σύστημα δορυφόρων

Σε μερικές περιπτώσεις, παγκόσμια δορυφορικά συστήματα πλοήγησης όπως το GPS μπορούν να χρησιμοποιηθούν για να βελτιωθεί η ακρίβεια της ράδιο-ανίχνευσης της θέσης. Η λειτουργία του GPS περιγράφεται με μεγαλύτερη λεπτομέρεια στο αντίστοιχο κεφάλαιο. Η ακρίβεια του GPS βρίσκεται μεταξύ των 10 και 20 μέτρων αλλά χρειάζονται 10-60 δευτερόλεπτα για την πρώτη ανίχνευση θέσης, δεν λειτουργεί ικανοποιητικά σε κλειστούς χώρους, ενώ προσθέτει στην κινητή συσκευή ένα υπολογίσιμο κόστος, πολυπλοκότητα και κατανάλωση ενέργειας [5].

Οι επιδόσεις του GPS μπορούν να βελτιωθούν αν χρησιμοποιηθεί σε συνδυασμό με κάποιο ασύρματο δίκτυο (AGPS). Ενσωματώνοντας έναν δέκτη GPS σε μια κινητή συσκευή μπορεί να γίνει ανίχνευση θέσης σε λιγότερο από 5 δευτερόλεπτα. Το δίκτυο μπορεί να βοηθά την κινητή συσκευή ώστε να μειωθεί η κατανάλωση ενέργειας και να αυξηθεί η ακρίβεια θέσης. Το AGPS μπορεί να χρησιμοποιηθεί και σε εσωτερικούς χώρους με ακρίβεια της τάξης των 50 μέτρων.

Ερευνητικές προσπάθειες γίνονται ώστε τεχνολογίες όπως το Bluetooth και το WiFi να χρησιμοποιηθούν στην ανίχνευση θέσης ώστε να επιτευχθεί μεγαλύτερη ακρίβεια σε εσωτερικούς χώρους (της τάξης των 10 μέτρων).

2.2.4 Συμπεράσματα

Γενικά οι δυνατότητες των διάφορων τεχνολογιών εντοπισμού θέσεις είναι συμπληρωματικές η μια ως προς την άλλη και δεν υπάρχει μια και μοναδική λύση. Σε εφαρμογές όπου τόσο η ακρίβεια όσο και η κάλυψη είναι αναγκαίες, υβριδικές τεχνολογίες μπορεί να μπορούν να δώσουν την βέλτιστη λύση. Επιπλέον, ανάλογα με την εφαρμογή, οι απαιτήσεις σε ακρίβεια και χρόνους απόκρισης ποικίλουν. Στον πίνακα 2.1 παρουσιάζουμε τις τεχνολογίες εντοπισμού συνοπτικά.

2.3 Κατηγοριοποίηση των LBS

2.3.1 Εμπλεκόμενοι στην υπηρεσία

Προκειμένου να προχωρήσουμε στην κατηγοριοποίηση των LBS πρέπει να λάβουμε υπόψη μας ποίοι εμπλέκονται κατά την παροχή μιας τέτοιας υπηρεσίας [6][7]. Ο κύριος εμπλεκόμενος μιας LBS είναι ο στόχος (Target). Ο στόχος αποτελεί την οντότητα της οποίας η θέση ανιχνεύεται και πάνω σε αυτήν την πληροφορία θέσης βασίζεται η υπηρεσία. Ο στόχος μπορεί να είναι για παράδειγμα ένα πρόσωπο, ένα ζώο, ένα όχημα. Από την άλλη πλευρά έχουμε τον εμπλεκόμενο ο οποίος ζητά πληροφορίες θέσης και ο οποίος αναφέρεται ως αιτών (Requestor). Ο αιτών μπορεί να είναι η ίδια φυσική οντότητα με τον στόχο ή διαφορετική (πχ ένα πρόσωπο, μια εταιρία, μια συσκευή). Ο πάροχος της υπηρεσίας (Service Provider) παρέχει μια υπηρεσία θέσης για τους τελικούς χρήστες. Τέλος ο πάροχος της υπηρεσίας ανίχνευσης θέσης (Location Provider) ο οποίος μπορεί να είναι ο

Τεχνολογία	Ακρίβεια	Χρόνος Απόκρισης	Μειονεκτήματα	Κόστος
Cell-id	Χαμηλή 150-10000m	Πολύ μικρός 3 secs	Πολύ μικρή ακρίβεια σε μεγάλες κυψέλες	Χαμηλό κόστος, δεν χρειάζονται μετατροπές στο δίκτυο και στις συσκευές
Προηγμένες τεχνικές βασισμένες στο δίκτυο	Μέτρια 50-125m	Μικρός 5 secs	Εξαρτώνται από την ορατότητα των σταθμών βάσης και τον αριθμώ των LMU	Μέσο κόστος, απαιτούνται κινητές συσκευές που μπορούν να τροποποιηθούν μέ- σω λογισμικού
Τεχνικές βασισμένες σε σύστημα δορυφόρων	Υψηλή 10-20m σε ανοικτούς χώρους 50m σε κλειστούς χώρους	Μεταβλητός 10-60 secs και 5 secs με AGPS	Υποβάθμιση του σήματος εντοπισμού και μειωμένη ακρίβεια σε συγκεκριμένα περιβάλλοντα	Υψηλό κόστος. Χρειάζονται νέες κινητές συσκευές

Πίνακας 2.1: Τεχνολογίες Εντοπισμού Θέσης

ίδιος ο στόχος (target) ο οποίος ανιχνεύει την θέση του, μπορεί να είναι ο ίδιος ο πάροχος της υπηρεσίας (Service Provider) ή μπορεί να είναι κάποιος τρίτος οργανισμός όπως πχ ένας τηλεπικοινωνιακός πάροχος.

2.3.2 Κατηγοριοποίηση βάσει της πληροφορίας θέσης

Μια πρώτη κατηγοριοποίηση των LBS μπορεί να γίνει βάσει του ποιος ζητά τις πληροφορίες θέσης και ποιον αφορούν αυτές οι πληροφορίες. Με βάση αυτό το κριτήριο έχουμε τρεις κατηγορίες pull, push και tracking [6].

Στις υπηρεσίες που ανήκουν στην κατηγορία pull, ο χρήστης (στόχος-αιτών) κάνει την αίτηση για την υπηρεσία σχετική με την θέση του και ταυτόχρονα δίνει την άδεια να χρησιμοποιηθεί η θέση του γιατί χωρίς αυτήν, η υπηρεσία που ζήτησε δεν μπορεί να ολοκληρωθεί. Ένα παράδειγμα pull υπηρεσίας: Ο χρήστης στέλνει μια αίτηση για ένα τοπικό δελτίο καιρού με την αποστολή ενός SMS.

Στις υπηρεσίες που ανήκουν στην κατηγορία push, η αίτηση για την χρήση των πληροφοριών θέσης γίνεται από τον πάροχο της υπηρεσίας. Και σε αυτήν την περίπτωση ο στόχος πρέπει να έχει δώσει την άδεια για την χρήση των πληροφοριών θέσης του. Ένα

παράδειγμα push υπηρεσίας: Ο χρήστης(στόχος-αιτών) ζητά από τον πάροχο να του λαμβάνει ένα τοπικό δελτίο καιρού κάθε μέρα σε μια συγκεκριμένη ώρα. Ο πάροχος της υπηρεσίας κάθε μέρα την συγκεκριμένη ώρα ανακτά την θέση του χρήστη και του αποστέλλει το αντίστοιχο δελτίο καιρού.

Στις υπηρεσίες που ανήκουν στην κατηγορία tracking ο χρήστης (αιτών) ζητάει πληροφορίες σχετικές με την θέση ενός τρίτου (στόχος). Ο πάροχος της υπηρεσίας δίδει τις ζητούμενες πληροφορίες μόνο αν ο στόχος επιτρέπει την χρήση της θέσης του. Ένα παράδειγμα tracking υπηρεσίας: Μια ομάδα ανθρώπων οι οποίοι θέλουν να ξέρουν που βρίσκεται ο καθένας από αυτούς. Ο καθένας δίνει την άδεια για την χρήση της θέσης του από τους υπόλοιπους.

2.3.3 Κατηγοριοποίηση βάσει του είδους της υπηρεσίας

Μια δεύτερη κατηγοριοποίηση των LBS μπορεί να γίνει βάση του είδους της παρεχόμενης υπηρεσίας προς τον χρήστη [8][9]. Έτσι προκύπτουν οι κατηγορίες που περιγράφονται με μεγαλύτερη λεπτομέρεια στην συνέχεια.

Υπηρεσίες ασφαλείας

Οι υπηρεσίες αυτές αφορούν την ικανότητα για την εύρεση της θέσης ενός ατόμου το οποίο είτε δεν γνωρίζει την ακριβή του θέση και βρίσκεται σε κίνδυνο ή δεν είναι σε θέση να την αποκαλύψει λόγο κάποιου περιστατικού έκτακτης ανάγκης (τραυματισμός, εγκληματική ενέργεια). Γνωρίζοντας την θέση του ατόμου το οποίο βρίσκεται σε κίνδυνο μπορεί να αποσταλεί άμεσα βοήθεια που σε πολλές περιπτώσεις ίσως του σώσει την ζωή. Λόγω της μεγάλης χρησιμότητας των υπηρεσιών αυτών στις Ηνωμένες Πολιτείες η Ομοσπονδιακή Επιτροπή Τηλεπικοινωνιών (FCC) πρότεινε και τελικά θεσμοθέτησε η προσθήκη υπηρεσιών εντοπισμού θέσης στο τηλέφωνο έκτακτης ανάγκης 911 (E911) [10].

Πέρα όμως από τις περιπτώσεις οπού απειλείται άμεσα η ασφάλεια κάποιου, υπάρχουν και άλλες καταστάσεις οπού κάποιος μπορεί να χρειάζεται βοήθεια όπως για παράδειγμα στην περίπτωση που το όχημα κάποιου υποστεί μηχανική βλάβη σε έναν μεγάλο αυτοκινητόδρομο. Στην περίπτωση αυτή οι υπηρεσίες εντοπισμού θα μπορούσαν να χρησιμοποιηθούν από την οδική βοήθεια.

Υπηρεσίες πλοήγησης

Οι υπηρεσίες πλοήγησης βασίζονται στην ανάγκη του χρήστη της κινητής συσκευής για οδηγίες πλοήγησης στην γεωγραφική περιοχή που βρίσκεται. Η ικανότητα ανίχνευσης θέσης χρησιμοποιείται προκειμένου να γνωρίζει ο χρήστης την ακριβή θέση του καθώς και λεπτομερείς οδηγίες για το πώς να μεταβεί στον επιθυμητό προορισμό. Αν ο πάροχος της υπηρεσίας έχει πληροφορίες για την κυκλοφοριακή κίνηση μπορεί να τις χρησιμοποιήσει προκείμενου να παρέχει δυναμικές οδηγίες πλοήγησης βάσει αυτών των πληροφοριών. Η δυνατότητα πλοήγησης μπορεί να επεκταθεί και σε μεγάλους εσωτερικούς χώρους όπως πολυκαταστήματα, μουσεία, νοσοκομεία κλπ ώστε κάποιος ο οποίος βρίσκεται μέσα στο κτήριο να μπορεί να μεταβεί στον τομέα που επιθυμεί.

Υπηρεσίες πληροφοριών

Οι υπηρεσίες παροχής πληροφοριών αφορούν την παροχή πληροφοριών οι οποίες επιλέγονται ανάλογα με την θέση, την χρονική στιγμή και τις επιλογές του χρήστη. Στις υπηρεσίες πληροφοριών συμπεριλαμβάνονται:

Ταξιδιωτικές υπηρεσίες όπως ξενάγηση (είτε αυτόματη, είτε υποβοηθούμενη). Οι τουρίστες συνήθως βρίσκονται σε μια περιοχή για την οποία ξέρουν ελάχιστα οπότε μπορούν να λαμβάνουν πληροφορίες μέσω ειδοποιήσεων για κοντινά σημεία ενδιαφέροντος όπως αρχαιολογικούς χώρους κλπ.

Υπηρεσίες καταλόγου οι οποίες παρέχουν στον χρήστη της κινητής συσκευής πληροφορίες σχετικές με προσφερόμενες υπηρεσίες που βρίσκονται κοντά στον χρήστη.

Υπηρεσίες πληροφόρησης για τοπικά επικείμενα γεγονότα όπως θεατρικές παραστάσεις, προβολές κινηματογράφου κλπ.

Υπηρεσίες διαφήμισης οι οποίες αφορούν την αποστολή διαφημιστικών μηνυμάτων προς τους χρήστες ανάλογα με την τοποθεσία που βρίσκονται. Λόγω της φύσης των υπηρεσιών αυτών ορισμένες φορές μπορεί να γίνουν ενοχλητικές προς τον χρήστη για αυτό πρέπει να λαμβάνεται η άδεια του.

Υπηρεσίες ανίχνευσης

Οι υπηρεσίες ανίχνευσης θέσης μπορούν να χρησιμοποιηθούν τόσο από απλούς χρήστες οι οποίοι θέλουν να γνωρίζουν που βρίσκεται κάποιο τρίτο πρόσωπο (πχ που ακριβώς βρίσκονται παιδιά ή ηλικιωμένοι), όσο και από εταιρίες και οργανισμούς οι οποίοι θέλουν να γνωρίζουν που ακριβώς βρίσκονται οι εργαζόμενοι τους, ο κινητός εξοπλισμός τους ή τα προϊόντα τους. Έχοντας γνώση αυτών των πληροφοριών μπορούν να διαχειριστούν με καλύτερο τρόπο τους πόρους τους. Για παράδειγμα μια εταιρία ταξί μπορεί να γνωρίζει που βρίσκεται κάθε ταξί της και να μπορεί να αποστέλλει στους πελάτες της το ταξί που βρίσκεται πιο κοντά. Η ανίχνευση θέσης μπορεί όμως να χρησιμοποιηθεί επίσης και στον τομέα της ψυχαγωγίας μέσω παιχνιδιών τα οποία προσφέρουν στους χρήστες διαφορετικές δυνατότητες αναλόγως της θέση τους.

Υπηρεσίες συναλλαγής

Οι υπηρεσίες συναλλαγής αφορούν κυρίως περιπτώσεις όπου ο πάροχος κάποιας υπηρεσίας πρέπει να χρεώνει δυναμικά τους χρήστες της ανάλογα με την θέση τους όταν την χρησιμοποιούν. Για παράδειγμα η χρέωση για την χρησιμοποίηση ενός δρόμου ανάλογα με τα χιλιόμετρα τα οποία διανύει ο χρήστης μπορεί να γίνει δυναμικά υπολογίζοντας τα συνολικά χιλιόμετρα που διένυσε και χρεώνοντας τον ανάλογα.

2.4 Απαιτήσεις μιας πλατφόρμας για LBS

Ανεξαρτήτως της κατηγορίας στην οποία ανήκει μια Υπηρεσία Βασισμένη στην Θέση, υπάρχουν ορισμένα επιθυμητά χαρακτηριστικά τα οποία αν πληρούνται από την πλατφόρμα ανάπτυξης αυξάνεται η απόδοση και η χρηστικότητα της υπηρεσίας [11]:

- **Ανεξαρτησία από την κινητή συσκευή:** Η χρήση της υπηρεσίας πρέπει να είναι ανεξάρτητη από το είδος της κινητής συσκευής. Πρέπει να μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα συσκευών: κινητά τηλέφωνα, συστήματα πλοήγησης, PDAs, φορητούς υπολογιστές κ.α. Παρόλο που οι υπηρεσίες που είναι εξαρτημένες από την υλοποίηση του υλικού, χρησιμοποιούν καλύτερα τους υπάρχοντες πόρους, δεν παρέχουν την απαιτούμενη ευελιξία όταν χρειάζεται να τροποποιηθεί ή να αναβαθμιστεί μια υπηρεσία και περιορίζουν τον αριθμό τον χρηστών της υπηρεσίας μόνο στους κατόχους του συγκεκριμένου υλικού.

  Η εφαρμογή που αναπτύξαμε βασίζεται στην πλατφόρμα Google Android. Η πλατφόρμα αυτή παρότι δεν είναι πλήρως ανεξάρτητη από την υλοποίηση του υλικού, υποστηρίζεται από ένα ευρύ φάσμα κινητών συσκευών οι οποίες πληρούν τις απαιτήσεις της, παρέχοντας ένα ενδιάμεσο στρώμα αφαίρεσης μεταξύ της LBS και του υλικού της κινητής συσκευής.

- **Απλή διαπροσωπεία χρήστη:** Η διαπροσωπεία της εφαρμογής μέσω της οποίας αλληλεπιδρούν οι χρήστες της πρέπει να είναι όσο πιο απλή γίνεται, αφού αφενός ο χρήστης πρέπει να θεωρηθεί οτι δεν έχει πρότερη εμπειρία στην χρήση ανάλογων εφαρμογών και αφετέρου μπορεί να χρειάζεται να την χρησιμοποιήσει κάτω από περιοριστικές συνθήκες (π.χ. οδηγώντας).

  Αυτή η απαίτηση λήφθηκε υπόψη κατά τον σχεδιασμό της εφαρμογής αυτής της διπλωματικής εργασίας. Η διαπροσωπεία χρήστη είναι όσο πιο αυτό-επεξηγηματική γίνεται με χρήση γραφικών, ενώ χρησιμοποιούνται οι κοινές συμβάσεις στην λειτουργία των πλήκτρων της συσκευής (π.χ. το πλήκτρο menu της συσκευής θα εμφανίσει ένα menu με τις δυνατές ενέργειες όπως θα περίμενε ο χρήστης).

- **Ελαχιστοποίηση της αποστολής δεδομένων διαμέσου του δικτύου κινητής τηλεφωνίας:** Παρότι οι ταχύτητες μετάδοσης διαμέσου του δικτύου κινητής τηλεφωνίας έχουν αυξηθεί σημαντικά με τις νέες τεχνολογίες (UMTS, GSM, GPRS) που αναπτύχθηκαν, το κόστος της επικοινωνίας αποτελεί ένα σημαντικό παράγοντα που πρέπει να ληφθεί υπόψη και οποίος δεν είναι αμελητέος. Άρα για διατηρηθεί το κόστος χρησιμοποίησης της υπηρεσίας σε χαμηλά επίπεδα πρέπει να αποφεύγεται άσκοπη ή περιττή αποστολή δεδομένων διαμέσου του δικτύου κινητής τηλεφωνίας.

  Η εφαρμογή αυτής της διπλωματικής εργασίας δεν βασίζεται στο δίκτυο κινητής τηλεφωνίας για την πραγματοποίηση της επικοινωνίας με τον εξυπηρετητή. Η επικοινωνία πραγματοποιείται διαμέσου της πρόσβασης στο Διαδίκτυο η οποία μπορεί να γίνει με διάφορους τρόπους (πχ WiFi). Ακόμα όμως και στην περίπτωση που η πρόσβαση στο Διαδίκτυο βασίζεται στο δίκτυο κινητής τηλεφωνίας το πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την μεταφορά δεδομένων αποστέλλει την ελάχιστη δυνατή ποσότητα δεδομένων για την λειτουργία της υπηρεσίας.

- **Ολοκλήρωση των υπαρχουσών υπηρεσιών Διαδικτύου:** Μερικές υπηρεσίες που προσφέρονται από LBS είναι ήδη υλοποιημένες ως υπηρεσίες Διαδικτύου (π.χ. Πληροφορίες οδικής κίνησης). Έτσι δεν είναι αναγκαία η ανάπτυξη αυτών των

υπηρεσιών από την αρχή. Αντιθέτως μπορούν να ολοκληρωθούν οι υπάρχουσες υπηρεσίες Διαδικτύου στις αναπτυσσόμενες LBS.

Η εφαρμογή που αναπτύσσεται στα πλαίσια αυτής της διπλωματικής εργασίας για παράδειγμα αντί να υλοποιήσει κάποια υπηρεσία παρουσίασης της θέσης χρησιμοποιεί την ήδη υπάρχουσα υπηρεσία Google Maps.

- **Υψηλή διαθεσιμότητα της υπηρεσίας ακόμα και σε υψηλή ζήτηση:** Μια LBS πρέπει να είναι διαθέσιμη οποιαδήποτε στιγμή. Αυτό σημαίνει οτι πρέπει να μπορεί να εξυπηρετεί αιτήσεις για εξυπηρέτηση κάτω από οποιεσδήποτε συνθήκες ακόμα και όταν η ζήτηση είναι υψηλή.

- **Επεκτασιμότητα:** Η δομή της διασύνδεσης των εξυπηρετητών της υπηρεσίας πρέπει να είναι τέτοια ώστε να μπορεί να επεκταθεί εύκολα, χωρίς όμως να υπάρχουν σημαντικές απώλειες στην απόδοση του όλου συστήματος.

  Η δομή των διακομιστών της υπηρεσίας που αναπτύσσεται στα πλαίσια αυτής της διπλωματικής εργασίας είναι πλήρως επεκτάσιμη αφού δεν τίθεται κανένας περιορισμός στην δομή της διασύνδεσης των εξυπηρετητών, απλά προτείνεται ένα πρωτόκολλο επικοινωνίας μεταξύ της εφαρμογής που τρέχει στην κινητή συσκευή και των διακομιστών. Η υλοποίηση του πρωτοκόλλου αυτού μπορεί να κρύβει οποιαδήποτε δομή διασύνδεσης από πίσω της.

- **Χαμηλό Κόστος:** Η υπηρεσία πρέπει να μπορεί να παρέχεται με χαμηλό κόστος. Αυτό συμπεριλαμβάνει τόσο το κόστος λειτουργίας των εξυπηρετητών, το κόστος της χρησιμοποίησης, το κόστος συντήρησης και το κόστος αναβάθμισης.

2.5 Ιδιωτικότητα – ασφάλεια – ανωνυμία

Με την ανάπτυξη των Υπηρεσιών Βασισμένων στην Θέση, παρά την χρησιμότητα τους, ανακύπτει ένα θέμα ηθικής φύσεως, η ασφάλεια των προσωπικών δεδομένων των χρηστών [12]. Η ικανότητα του προσδιορισμού της θέσης ενός χρήστη μιας υπηρεσίας και η αποκάλυψη της θέσης του σε τρίτους, χωρίς την ύπαρξη σαφώς ορισμένων κανόνων προφύλαξης των προσωπικών στοιχείων, μπορεί να οδηγήσει στην πλήρη καταγραφή των δραστηριοτήτων και των κινήσεων του, χάνοντας έτσι την ιδιωτικότητα και την ανωνυμία του.

Για τον παραπάνω λόγο πρέπει να μπορεί να τεθεί ένα σύνολο κανόνων για να οριοθετηθεί η ορθή ή μη χρήση της πληροφορίας θέσης και να συσταθεί κάποιος οργανισμός ο οποίος θα φροντίζει για την εφαρμογή αυτών των κανόνων. Ήδη στις Ηνωμένες Πολιτείες το 2003 με την υπογραφή του "CAN-SPAM Act" τέθηκε εκτός νόμου η αποστολή διαφημιστικών μηνυμάτων σε κινητές συσκευές χωρίς την ρητή άδεια του τελικού χρήστη [13]. Περαιτέρω εκτός από το ηθικό θέμα τις ασφάλειας των προσωπικών δεδομένων υπάρχει και το τεχνολογικό. Το σήμα ανίχνευσης θέσης μπορεί να είναι προσβάσιμο από οποιονδήποτε όπως το σήμα του GPS ή να υπάρχει κάποιος περιορισμός κρυπτογραφώ-

ντας το σήμα όπως πρόκειται να γίνει με το σύστημα Galileo ή με το ανάλογο “Selective Availability” που ίσχυε μέχρι τον Μάιο του 2000 για το GPS [14][15].

Επίσης οι πληροφορίες θέσης πρέπει να διασφαλιστεί ότι κρυπτογραφούνται όταν πρόκειται να αποσταλούν για να αποφευχθεί οποιαδήποτε προσπάθεια υποκλοπής τους. Η αποθήκευση τους πρέπει να γίνεται σε ασφαλείς βάσεις δεδομένων οι οποίες είναι προστατευμένες από ανεπιθύμητη πρόσβαση σε αυτές.

Κεφάλαιο 3

Τεχνολογίες Ανίχνευσης Θέσης

3.1 Global Positioning System

3.1.1 Εισαγωγή

Το Global Position System ή GPS αποτελεί ένα σύστημα εντοπισμού θέσης το οποίο αναπτύχθηκε από το υπουργείο αμύνης των Ηνωμένων Πολιτειών της Αμερικής. Για την λειτουργία του χρησιμοποιεί ένα σύνολο 24 μέχρι 32 δορυφόρων οι οποίοι μεταδίδουν ακριβή σήματα μικροκυμάτων τα οποία επιτρέπουν στους δέκτες GPS να καθορίσουν την τρέχουσα θέση τους, την τρέχουσα ώρα, και την τρέχουσα ταχύτητα τους. Το σύνολο των δορυφόρων διαχειρίζεται από την πολεμική αεροπορία των Ηνωμένων Πολιτειών αλλά το GPS μπορεί να χρησιμοποιηθεί και από απλούς πολίτες για απλή ανίχνευση της θέσης τους. [20].

Ένας δέκτης GPS υπολογίζει την τρέχουσα θέση του μετρώντας προσεκτικά τους χρόνους αφίξεων των σημάτων τα οποία στέλνονται από τους GPS δορυφόρους. Κάθε δορυφόρος μεταδίδει διαρκώς μηνύματα που περιέχουν πληροφορία για την ακριβή ώρα αποστολής του μηνύματος, για την ακριβή θέση του την στιγμή της αποστολής, για την συνολική κατάσταση του συστήματος και για τις τροχιές όλων των υπόλοιπων δορυφόρων. Ο δέκτης μετράει τον χρόνο που χρειάζεται κάθε σήμα να φτάσει σε αυτόν και υπολογίζει την απόσταση του από κάθε δορυφόρο. Στη συνέχεια με την γεωμετρική μέθοδο της τριγωνοποίησης (triangulation-trilateration) συνδυάζονται αυτές οι μετρούμενες αποστάσεις με τις θέσεις των δορυφόρων για να προσδιοριστεί η ακριβής θέση του δέκτη. Η θέση παρουσιάζεται είτε υπό την μορφή κινούμενου χάρτη, είτε υπό την μορφή γεωγραφικού πλάτους και γεωγραφικού μήκους και μερικές φορές γεωγραφικού υψόμετρου. Πολλοί δέκτες μπορούν να υπολογίσουν και επιπλέον απορρέουσες πληροφορίες όπως η διεύθυνση και η ταχύτητα βάσει των αλλαγών θέσης.

Τρεις δορυφόροι φαίνεται οτι θα ήταν αρκετοί για τον υπολογισμό της θέσης. Όμως ένα μικρό σφάλμα ρολογιού, δεδομένης της μεγάλης ταχύτητας του φωτός με την οποία ταξιδεύουν τα σήματα που στέλνουν οι δορυφόροι, συντελεί σε μεγάλα σφάλματα στην θέση. Για τον λόγο αυτό ο δέκτης χρησιμοποιεί και έναν τέταρτο δορυφόρο για να υπο-

λογίζει εκτός από τις τρεις χωρικές διαστάσεις και την διάσταση του χρόνου προκειμένου να είναι σε θέση να διορθώνει το ρολόι του.

Παρόλο που τέσσερις δορυφόροι απαιτούνται για την κανονική λειτουργία του συστήματος, σε ειδικές περιπτώσεις μπορούν να χρησιμοποιηθούν λιγότεροι. Αν μια από τις μεταβλητές του συστήματος είναι ήδη γνωστή (για παράδειγμα ένα πλοίο ή ένα αεροπλάνο μπορεί να γνωρίζει το υψόμετρο του) ένας δέκτης μπορεί να προσδιορίσει την θέση του χρησιμοποιώντας μόνο τρεις δορυφόρους. Μερικοί δέκτες μπορεί να χρησιμοποιήσουν περαιτέρω στοιχεία ή υποθέσεις για να δώσουν μια κάπως χειρότερη εκτίμηση της θέσης όταν είναι ορατοί λιγότεροι από τέσσερις δορυφόροι.

3.1.2 Δομή του GPS

Το GPS αποτελείται από τρία βασικά τμήματα. Αυτά είναι το Τμήμα Διαστήματος, το Τμήμα Ελέγχου και το Τμήμα Χρήστη [21].

Τμήμα διαστήματος

Το τμήμα διαστήματος αποτελείται από τους δορυφόρους ή διαστημικά οχήματα στην ορολογία του GPS. Οι δορυφόροι αυτοί περιφέρονται γύρω από τη γη, με ακτίνα περιστροφής 26.600 km, και ολοκληρώνουν δύο περιφορές σε μια αστρική ημέρα. Ο αρχικός σχεδιασμός του GPS απαιτούσε 24 διαστημικά οχήματα, από οκτώ σε τρία διαφορετικά κυκλικά τροχιακά επίπεδα. Στην συνέχεια το σύστημα τροποποιήθηκε μοιράζοντας τους δορυφόρους σε τέσσερις σε κάθε ένα από τα έξι διαφορετικά κυκλικά τροχιακά επίπεδα. Τα τροχιακά επίπεδα των δορυφόρων έχουν κέντρο την γη και δεν περιστρέφονται σε σχέση με τους απλανείς αστέρες. Οι τροχιές αυτές είναι διαμορφωμένες έτσι ώστε να είναι ορατοί τουλάχιστον 6 δορυφόροι από οποιοδήποτε σχεδόν σημείο της γης. Από τον Μάρτιο του 2008 προστέθηκαν άλλοι 7 δορυφόροι ώστε να βελτιωθεί η ακρίβεια των υπολογισμών. Με την αύξηση των δορυφόρων άλλαξαν και οι τροχιές τους σε μια ανομοιόμορφη διάταξη έτσι ώστε να βελτιωθεί η αξιοπιστία και η διαθεσιμότητα του συστήματος σε σχέση με μια ομοιόμορφη διάταξη, σε περίπτωση όπου πολλαπλοί δορυφόροι αποτυγχάνουν.

Τμήμα ελέγχου

Το τμήμα ελέγχου αποτελείται από ένα σύνολο σταθμών και κεραιών μέσω τον οποίων ελέγχεται η ορθή λειτουργία του συστήματος [22]. Οι τροχιές των δορυφόρων ελέγχονται από την αεροπορία των Ηνωμένων Πολιτειών μέσω σταθμών παρακολούθησης. Οι σταθμοί παρακολούθησης βρίσκονται στις παρακάτω περιοχές: Hawaii, Kwajalein, Ascension Island, Diego Garcia, και Colorado Springs. Εκτός από τους σταθμούς παρακολούθησης οι οποίοι ανήκουν στην αεροπορία των Ηνωμένων Πολιτειών, λειτουργούν και σταθμοί παρακολούθησης υπό την εποπτεία του National Geospatial-Intelligence Agency. Όλες οι πληροφορίες ανίχθευσης των δορυφόρων στέλνονται στον κεντρικό σταθμό ελέγχου στην αεροπορική βάση του Schriever. Ο κεντρικός σταθμός επικοινωνεί με κάθε δορυφόρο σε τακτά χρονικά διαστήματα στέλνοντας διορθώσεις στους δορυφόρους (χρησιμο-

ποιώντας επίγειες κεραίες οι οποίες βρίσκονται στις περιοχές: Ascension Island, Diego Garcia, Kwajalein, και Colorado Springs). Οι διορθώσεις συγχρονίζουν τα ατομικά ρολόγια των δορυφόρων (με σφάλμα μερικών nanosecond) καθώς επίσης τροποποιούν και την θέση τους. Η αλλαγή θέσης ενός δορυφόρου δεν είναι μια ακριβής διαδικασία σύμφωνα με το πρότυπο του GPS. Προκειμένου να αλλάξει η τροχιά ενός δορυφόρου αυτός πρέπει να χαρακτηριστεί μη-υγιής έτσι ώστε οι δέκτες να μην τον λαμβάνουν υπόψη στους υπολογισμούς τους. Στη συνέχεια γίνεται η διόρθωση της τροχιάς του δορυφόρου και η νέα τροχιά εξετάζεται από το έδαφος. Αν είναι σωστή, ο δορυφόρος χαρακτηρίζεται πάλι υγιής και μπορεί να ξαναχρησιμοποιηθεί από τους δέκτες.

Τμήμα χρήστη

Το τμήμα χρήστη αποτελείται από τους δέκτες GPS. Γενικά ένας δέκτης GPS αποτελείται από μια κεραία, συντονισμένη στις συχνότητες που εκπέμπουν οι δορυφόροι, έναν δέκτη-επεξεργαστή, ένα υψηλής σταθερότητας ρολόι (συνήθως έναν κρυσταλλικό ταλαντωτή). Μπορεί να περιλαμβάνει επίσης μια οθόνη για να παρέχει στον χρήστη πληροφορίες για την θέση, την ταχύτητα κ.α. Ένας δέκτης χαρακτηρίζεται από τον αριθμό των καναλιών του. Δηλαδή από τον αριθμό των δορυφόρων που μπορεί να παρακολουθεί ταυτόχρονα. Επίσης μπορεί να περιλαμβάνει μια είσοδο για διαφορικές διορθώσεις χρησιμοποιώντας το RTCM SC-104 format. Πολλοί δέκτες έχουν την δυνατότητα αποθήκευσης πληροφοριών θέσης σε Ηλεκτρονικό Υπολογιστή ή άλλες συσκευές χρησιμοποιώντας το NMEA protocol.

3.1.3 Μηνύματα πλοήγησης

Κάθε δορυφόρος μεταδίδει συνεχώς ένα μήνυμα πλοήγησης στα 50 bit/s [23]. Κάθε μήνυμα χωρίζεται σε πλαίσια το καθένα από τα οποία παίρνει 30 δευτερόλεπτα για να μεταδοθεί (1500bit). Η μετάδοση κάθε ενός από τα πλαίσια ξεκινάει ακριβώς στο λεπτό και στο μισό λεπτό όπως υποδεικνύεται από το ατομικό ρολόι του κάθε δορυφόρου. Κάθε πλαίσιο περιέχει 5 υποπλαίσια καθένα μήκους 6 δευτερολέπτων και μεγέθους 300 bits. Κάθε υποπλαίσιο περιέχει 10 λέξεις των 30 bits χρονικής διάρκειας 0.6 δευτερολέπτων το καθένα.

Οι λέξεις 1 και 2 κάθε υποπλαισίου περιέχουν τον ίδιο τύπο δεδομένων. Η πρώτη λέξη ονομάζεται λέξη τηλεμετρίας, υποδεικνύει την έναρξη του υποπλαισίου και χρησιμοποιείται από τους δέκτες για τον συγχρονισμό με το σήμα πλοήγησης. Η δεύτερη λέξη ονομάζεται HOW (handover word) και περιέχει πληροφορίες χρονισμού οι οποίες επιτρέπουν στον δέκτη να αναγνωρίσει το υποπλαίσιο.

Οι λέξεις 3 μέχρι 10 του υποπλαισίου 1 περιέχουν δεδομένα τα οποία περιγράφουν το δορυφορικό ρολόι και την σχέση του με τον GPS χρόνο. Οι λέξεις 3 μέχρι 10 του υποπλαισίου 2 και 3, περιέχουν πληροφορίες για την ephemeris, η οποία περιέχει πληροφορίες για την ακριβή τροχιά του δορυφόρου και τον ακριβή χρόνο αποστολής του μηνύματος. Η ephemeris ανανεώνεται κάθε 2 ώρες και γενικά είναι έγκυρη για 4 ώρες, με μέριμνα να ανανεώνεται κάθε 6 ώρες σε μη φυσιολογικές συνθήκες λειτουργίας. Ο χρόνος ανάκτησης

της ephemeris είναι ένα σημαντικό στοιχείο καθυστέρησης για την πρώτη ανίχνευσης θέσης. Ενώ το υλικό γίνεται όλο και πιο ικανό, ο χρόνος που χρειάζεται για να κλειδώσει στο σήμα ενός δορυφόρου μικραίνει, παρόλα αυτά η λήψη των πληροφοριών της ephemeris χρειάζεται 30 δευτερόλεπτα (στην χειρότερη περίπτωση) , λόγω του χαμηλού ρυθμού μετάδοσης δεδομένων.

Το ημερολόγιο (almanac) αποτελείται από πληροφορίες για την τροχιά και την κατάσταση κάθε δορυφόρου του συστήματος, ένα ιονοσφαιρικό μοντέλο και πληροφορίες για την συσχέτιση του χρόνου GPS με τον UTC (Coordinated Universal Time). Οι λέξεις 3 μέχρι 10 των υποπλαισίων 4 και 5 περιέχουν ένα νέο μέρος του ημερολογίου. Κάθε πλαίσιο περιέχει το 1/25 του ημερολογίου, έτσι χρειάζονται 12.5 λεπτά για την λήψη ολόκληρου του ημερολογίου από έναν μόνο δορυφόρο. Το ημερολόγιο εξυπηρετεί πολλαπλούς σκοπούς. Ο πρώτος είναι να βοηθά στην ανάκτηση των δορυφόρων κατά την εκκίνηση του δέκτη επιτρέποντας την δημιουργία μιας λίστας ορατών δορυφόρων βασισμένη σε αποθηκευμένες πληροφορίες, ενώ χρειάζεται μια ephemeris από κάθε δορυφόρο για τον υπολογισμό θέσης χρησιμοποιώντας αυτόν τον δορυφόρο. Σε παλιότερες συσκευές, η έλλειψη ημερολογίου σε έναν καινούργιο δέκτη προκαλούσε μεγάλες καθυστερήσεις πριν μπορέσει να δώσει την πρώτη έγκυρη θέση, επειδή η ανίχνευση των δορυφόρων ήταν μια αργή διαδικασία. Η εξέλιξη στο υλικό έκανε την διαδικασία ανάκτησης των δορυφόρων πολύ γρηγορότερη, έτσι η μη ύπαρξη ημερολογίου δεν αποτελεί πλέον θέμα. Ο δεύτερος σκοπός του ημερολογίου είναι να συσχετίζει τον παραγόμενο χρόνο από το GPS με το διεθνές πρότυπο UTC. Τέλος το ημερολόγιο επιτρέπει σε έναν δέκτη μιας μόνο συχνότητας να διορθώνει το ιονοσφαιρικό σφάλμα περιέχοντας ένα ιονοσφαιρικό μοντέλο.

Όλοι οι δορυφόροι εκπέμπουν στις δύο ίδιες συχνότητες, 1.57542 GHz (L1 signal) και 1.2276 GHz (L2 signal). Ο δέκτης μπορεί να ξεχωρίσει το σήμα του κάθε δορυφόρου από τους υπολοίπους επειδή το GPS χρησιμοποιεί μια τεχνική CDMA (Code Division Multiple Access). Πριν την εκπομπή στο μήνυμα πλοήγησης προστίθενται 2 PNR ακολουθίες που ονομάζονται Coarse/Acquisition code (C/A) και Precise Code (P), ο δορυφόρος διαμορφώνει το παραγόμενο μήνυμα με ένα L-Band φέρων σήμα δημιουργώντας ένα σήμα διευρυμένου φάσματος το οποίο και εκπέμπεται. Ο κάθε C/A κωδικός είναι μοναδικός και υλοποιεί τον μηχανισμό που επιτρέπει την αναγνώριση του κάθε δορυφόρου. Ο P κωδικός είναι συνήθως κρυπτογραφημένος και χρησιμοποιείται από στρατιωτικές υπηρεσίες μόνο.

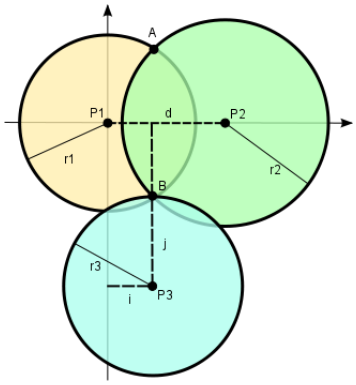
3.1.4 Εισαγωγή στον υπολογισμό θέσης

Υπολογισμός θέσης χωρίς σφάλματα

Σε αυτήν την παράγραφο προκειμένου να εξετάσουμε το πως λειτουργεί ένας δέκτης GPS, η ύπαρξη σφαλμάτων θα αγνοηθεί. Χρησιμοποιώντας τα μηνύματα που λαμβάνει από τουλάχιστον τέσσερις ορατούς δορυφόρους, ένας δέκτης GPS μπορεί να προσδιορίσει τις θέσεις αυτών των δορυφόρων και τον χρόνο που στάλθηκαν αυτά τα μηνύματα. Οι συνιστώσες x, y, z και t της θέσης και του χρόνου ορίζονται ως $[x_i, y_i, z_i, t_i]$ όπου i είναι ο αριθμός του δορυφόρου και παίρνει την τιμή 1, 2, 3, ή 4. Γνωρίζοντας τον χρόνο που λήφθηκε το μήνυμα tr_i, ο GPS δέκτης μπορεί να υπολογίσει τον χρόνο μετάδοσης, $(tr_i -$

t_i) του. Θεωρώντας οτι το μήνυμα ταξίδεψε με την ταχύτητα του φωτός, c, η απόσταση που διένυσε, p_i μπορεί να υπολογιστεί από τον τύπο: $(tr_i - t_i)c$. Γνωρίζοντας την απόσταση μεταξύ του δέκτη και ενός δορυφόρου και την θέση του δορυφόρου, μπορεί να υποτεθεί οτι ο δέκτης βρίσκεται πάνω στην επιφάνεια μιας σφαίρας με κέντρο τον δορυφόρο και ακτίνα την απόσταση. Έτσι ο δέκτης βρίσκεται πάνω ή κοντά στην τομή τεσσάρων σφαιρών. Στην ιδανική περίπτωση που δεν υπάρχουν σφάλματα, ο δέκτης θα βρίσκεται πάνω στην τομή των τεσσάρων σφαιρών. Τα σημεία τομής των επιφανειών δύο σφαιρών οι οποίες τέμνονται σε περισσότερα από ένα σημείο αποτελούν κύκλο.

Ένας κύκλος και μια σφαίρα στις περισσότερες περιπτώσεις πρακτικού ενδιαφέροντος τέμνονται σε 2 σημεία, αν και είναι εύκολα κατανοητό οτι μπορεί να τέμνονται σε ένα μόνο σημείο ή σε κανένα. Η σωστή θέση του δέκτη GPS είναι αυτό το σημείο τομής το οποίο βρίσκεται εγγύτερα στην επιφάνεια της γης αν ο δέκτης βρίσκεται κοντά στην επιφάνεια της γης. Η σωστή θέση του δέκτη επίσης είναι το σημείο τομής το οποίο βρίσκεται εγγύτερα στην επιφάνεια της τέταρτης σφαίρας.



Σχήμα 3.1: Δισδιάστατη Τριγωνοποίηση

Τριγωνοποίηση

Ο τρόπος υπολογισμού που περιγράφτηκε παραπάνω θεωρητικά αποδεικνύεται μαθηματικά μέσω του θεωρήματος της τριγωνοποίησης. Θεωρούμε τις εξισώσεις τριών σφαιρών. Για να απλοποιήσουμε τους υπολογισμούς, εφαρμόζουμε περιορισμό ως προς την θέση του κέντρου των σφαιρών χωρίς βλάβη της γενικότητας. Υποθέτουμε οτι και οι τρεις σφαίρες έχουν κέντρο στο επίπεδο z=0. Η μια έχει κέντρο την αρχή των αξόνων, η δεύτερη έχει κέντρο πάνω στον άξονα των x. Είναι δυνατόν να μετασχηματίσουμε κάθε σύνολο τριών δοσμένων σημείων ώστε να ικανοποιούν αυτούς τους περιορισμούς, για να βρούμε το σημείο τομής, και στην συνέχεια να αντιστρέψουμε τον μετασχηματισμό για να βρούμε το σημείο τομής στο αρχικό σύστημα συντεταγμένων.

Οι εξισώσεις των τριών σφαιρών:

$$r_1^2 = x^2 + y^2 + z^2$$

$$r_2^2 = (x-d)^2 + y^2 + z^2,$$

$$r_3^2 = (x-i)^2 + (y-j)^2 + z^2$$

Πρέπει να βρούμε ένα σημείο (x, y, z) το οποίο ικανοποιεί και τις τρεις εξισώσεις. Αρχικά αφαιρούμε την δεύτερη εξίσωση από την πρώτη και λύνουμε ως προς x:

$$x = \frac{r_1^2 - r_2^2 + d^2}{2d}$$

Θεωρούμε οτι οι δύο πρώτες σφαίρες τέμνονται σε περισσότερα από ένα σημεία, και έτσι d-r1 < r2 < d+r1. Σε αυτήν την περίπτωση αντικαθιστώντας στην εξίσωση της πρώτης σφαίρας παίρνουμε την εξίσωση ενός κύκλου, ο οποίος είναι η λύση για την τομή των δύο σφαιρών:

$$y^2 + z^2 = r_1^2 - \frac{(r_1^2 - r_2^2 + d^2)^2}{4d^2}$$

Αντικαθιστώντας:

$$y^2 + z^2 = r_1^2 - x^2$$

στην εξίσωση της τρίτης σφαίρας και λύνοντας ως προς y έχουμε το αποτέλεσμα:

$$y = \frac{r_1^2 - r_3^2 - x^2 + (x-i)^2 + j^2}{2j} = \frac{r_1^2 - r_3^2 + i^2 + j^2}{2j} - \frac{i}{j}x$$

Τώρα έχοντας τις x και y συντεταγμένες του σημείου τομής, μπορούμε απλά να αναδιατάξουμε την εξίσωση για την πρώτη σφαίρα έτσι ώστε να βρούμε τον τύπο για την συντεταγμένη z:

$$z = \pm\sqrt{r_1^2 - x^2 - y^2}$$

Έτσι έχουμε και τις τρεις συντεταγμένες της λύσης. Επειδή η z μπορεί να είναι θετική ή αρνητική τετραγωνική ρίζα, είναι δυνατόν να έχουμε μηδέν. μια ή δύο λύσεις για το πρόβλημα.

3.1.5 Προηγμένος υπολογισμός θέσης

Υπολογισμός θέσης με σφάλματα

Έστω οτι b συμβολίζει το σφάλμα ρολογιού, δηλαδή την ποσότητα κατά την οποία το ρολόι του δέκτη είναι χρονικά πίσω. Ο δέκτης GPS έχει τέσσερις αγνώστους, τις τρεις συνιστώσες της θέσης του δέκτη και το σφάλμα ρολογιού $[x, y, z, b]$. Οι εξισώσεις των σφαιρών δίδονται από τον τύπο:

$$(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2 = \big((tr_i + b - t_i)c\big)^2,\ i = 1, 2, 3, 4$$

Άλλη μια χρήσιμη μορφή για αυτές τις εξισώσεις δίνεται αν θέσουμε: $p_i = (tr_i - t_i)\, c$. Τότε η εξισώσεις γίνονται:

$$p_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} - bc,\ i = 1, 2, 3, 4$$

Δύο από τους σημαντικότερους τρόπους υπολογισμού της θέσης του δέκτη και του σφάλματος ρολογιού είναι (1) τριγωνοποίηση και (2) πολυδιάστατοι υπολογισμοί Newton-Raphson.

- Ο δέκτης μπορεί να χρησιμοποιήσει τριγωνοποίηση και εν συνεχεία αριθμητική εύρεση ρίζας μιας διάστασης. Με αυτήν την μέθοδο χρησιμοποιείται τριγωνοποίηση για την εύρεση των σημείων τομής τριών σφαιρών. Τα σημεία τομής των τριών σφαιρών μπορεί να είναι 0, 1 ή 2 σημεία. Στις συνήθεις περιπτώσεις που τα σημεία τομής είναι δύο, η λύση που βρίσκεται εγγύτερα στην επιφάνεια της σφαίρας που αντιστοιχεί στον τέταρτο δορυφόρο επιλέγεται. Η επιφάνεια της γης χρησιμοποιείται μερικές φορές στην θέση της τέταρτης σφαίρας. Το σφάλμα ρολογιού, b υπολογίζεται ως συνάρτηση της απόστασης της λύσης από την επιφάνεια της σφαίρας που αντιστοιχεί στον τέταρτο δορυφόρο. Χρησιμοποιώντας τον χρόνο βασισμένο στο σφάλμα ρολογιού που υπολογίστηκε, νέες σφαίρες υπολογίζονται και η διαδικασία επαναλαμβάνεται. Αυτή η επανάληψη συνεχίζεται μέχρι η απόσταση από την λύση της τριγωνοποίησης να είναι αρκετά κοντά στην επιφάνεια της σφαίρας που αντιστοιχεί στον τέταρτο δορυφόρο.

- Ο δέκτης μπορεί να χρησιμοποιήσει κάποια πολυδιάστατη μέθοδο εύρεσης ρίζας όπως η μέθοδος Newton-Raphson. Γραμμικοποιεί γύρω από μια προσεγγιστική λύση,

  $$[x^{(k)}, y^{(k)}, z^{(k)}, b^{(k)}]$$

  στην k-στή επανάληψη, και στην συνέχεια λύνει τέσσερις γραμμικές εξισώσεις που παράγονται από τις δευτεροβάθμιες εξισώσεις για τον υπολογισμό του

  $$[x^{(k+1)}, y^{(k+1)}, z^{(k+1)}, b^{(k+1)}]$$

  Οι ακτίνες είναι μεγάλες και έτσι οι επιφάνειες των σφαιρών μπορούν να θεωρηθούν επίπεδες. Αυτή η θεώρηση μπορεί να κάνει την επαναληπτική μέθοδο να συγκλίνει πολύ γρήγορα στην περίπτωση που το b είναι κοντά στην σωστή τιμή και οι κύριες αλλαγές συντελούνται στις τιμές x, y, και z. Όταν όμως το b αλλάζει σημαντικά, η σύγκλιση της μεθόδου δεν είναι γρήγορη. Αυτή η πιθανή ταχύτητα σύγκλισης της μεθόδου αποτελεί το κύριο πλεονέκτημα της.

Παραπάνω από τέσσερις δορυφόροι μπορούν να χρησιμοποιούνται όταν είναι διαθέσιμοι. Έτσι προκύπτει ένα σύστημα εξισώσεων που δεν έχει μοναδική λύση και το οποίο πρέπει να λυθεί με την μέθοδο ελαχίστων τετραγώνων ή κάποια παρόμοια τεχνική. Αν όλοι οι ορατοί δορυφόροι χρησιμοποιηθούν, τα αποτελέσματα είναι πάντα τουλάχιστον τόσο ακριβή όσο ακριβή θα ήταν τα αποτελέσματα αν χρησιμοποιούταν οι τέσσερις καλύτεροι δορυφόροι ή ακόμα περισσότερο.

3.1.6 Πηγές σφάλματος και παρεμβολών

Διάδοση του Σφάλματος Μέτρησης των Χρόνων Διάδοσης

Η θέση που υπολογίζεται από τον δέκτη GPS προαπαιτεί γνώση της τρέχουσας ώρας, της θέσης του δορυφόρου και την μετρούμενη καθυστέρηση του λαμβανόμενου σήματος. Η ακρίβεια θέσης εξαρτάται κυρίως στην θέση του δορυφόρου και στην καθυστέρηση του σήματος.

Για να μετρήσει την καθυστέρηση, ο δέκτης συγκρίνει την ακολουθία bits από τον δορυφόρο με μια εσωτερικά δημιουργούμενη εκδοχή της. Από την σύγκριση των ανόδων και των καθόδων των δύο παλμών κατά τις εναλλαγές των bits, τα σύγχρονα ηλεκτρονικά στοιχεία μπορούν να μετρήσουν την διαφορά μεταξύ των δύο σημάτων με ακρίβεια περίπου 10 nanoseconds για τον C/A κώδικα. Αφού τα σήματα του GPS διαδίδονται με την ταχύτητα του φωτός, αυτό το σφάλμα στην μέτρηση της χρονικής καθυστέρησης αντιστοιχεί σε σφάλμα 3 περίπου μέτρων στον χώρο.

Η ακρίβεια της θέσης μπορεί να βελτιωθεί χρησιμοποιώντας ένα σήμα P υψηλότερης συχνότητας. Έτσι μπορεί να επιτευχθεί ακρίβεια της τάξης των 30 εκατοστών. Τα σφάλματα λόγω των ηλεκτρονικών είναι μια από τις πολλές αιτίες υποβάθμισης της ακρίβειας από αυτά που φαίνονται στον πίνακα 3.1:

Πηγή Σφάλματος	Σφάλμα
Σφάλματα Ιονόσφαιρας	± 5 m
Σφάλματα στην Ephemeris	± 2.5 m
Σφάλματα ρολογιού δορυφόρου	± 2 m
Παραμόρφωση πολλαπλών μονοπατιών	± 1 m
Σφάλματα Τροπόσφαιρας	± 0.5 m
Αριθμητικά Σφάλματα	± 1 m

Πίνακας 3.1: Σφάλματα GPS

Σφάλματα λόγω της ατμόσφαιρας

Η ανομοιογένεια στις ατμοσφαιρικές συνθήκες επηρεάζει την ταχύτητα των σημάτων GPS καθώς αυτά περνάνε διαμέσου της γήινης ατμόσφαιρας, ειδικά διαμέσου της ιονόσφαιρας. Η διόρθωση αυτών των σφαλμάτων αποτελεί πρόκληση για την βελτίωση της ακρίβειας θέσης που παρέχει το GPS. Τα σφάλματα αυτά είναι μικρότερα όταν ο δορυφόρος βρίσκεται ακριβώς από πάνω από τον δέκτη και γίνονται μεγαλύτερα όταν ο δορυφόρος βρίσκεται κοντά στον ορίζοντα, καθώς η διαδρομή που διανύει το σήμα δια μέσω της ατμόσφαιρας είναι μεγαλύτερη. Όταν η εκτιμώμενη θέση του δέκτη είναι γνωστή, ένα μαθηματικό μοντέλο μπορεί να χρησιμοποιηθεί για τον υπολογισμό και την διόρθωση αυτών των λαθών.

Η υγρασία στην τροπόσφαιρα επίσης προκαλεί μια μεταβλητή καθυστέρηση, η οποία προκαλεί σφάλματα παρόμοια με την ιονοσφαιρική καθυστέρηση. Αυτά τα σφάλματα εί-

ναι περισσότερο τοπικά και αλλάζουν πολύ πιο γρήγορα από τα ιονοσφαιρικά σφάλματα, κάτι που κάνει την ακριβή μέτρηση και διόρθωση τους πολύ πιο δύσκολη από οτι για την διόρθωση των ιονοσφαιρικών σφαλμάτων.

Με τις αλλαγές στο υψόμετρο του δέκτη αλλάζει η καθυστέρηση καθώς το σήμα διανύει μεταβαλλόμενου μεγέθους διαδρομή διαμέσου της ατμόσφαιρας. Από την στιγμή που ο δέκτης GPS υπολογίζει το υψόμετρο, αυτό το σφάλμα είναι σχετικά απλό να διορθωθεί.

Σφάλματα λόγω πολλαπλών διαδρομών

Τα GPS σήματα μπορούν να επηρεαστούν από πολλαπλές διαδρομές, όπου τα ραδιοσήματα ανακλώνται από το έδαφος, κτίρια, φαράγγια, κλπ. Αυτά τα καθυστερημένα σήματα μπορούν να προκαλέσουν μείωση της ακρίβειας. Ένα πλήθος τεχνικών έχουν αναπτυχθεί προκειμένου να διορθωθούν αυτού του είδους τα σφάλματα. Για μεγάλες καθυστερήσεις πολλαπλών διαδρομών, ο δέκτης μπορεί από μόνος του να αναγνωρίσει το καθυστερημένο σήμα και να το απορρίψει. Τα μικρότερης καθυστέρησης σήματα πολλαπλών διαδρομών είναι δυσκολότερο να φιλτραριστούν γιατί παρεμβάλλονται με το πραγματικό σήμα. Για να απορριφθούν μικρότερης καθυστέρησης σήματα πολλαπλών διαδρομών που ανακλώνται από το έδαφος, μπορούν να χρησιμοποιηθούν ειδικές κεραίες. Τέτοιου είδους σφάλματα είναι πολύ λιγότερο σημαντικά σε κινούμενα οχήματα. Όταν η κεραία κινείται, οι ψευδείς λύσεις που προέρχονται από ανακλώμενα σήματα αποτυγχάνουν να συγκλίνουν σε αντίθεση με το πραγματικό σήμα.

Επιλεκτική διαθεσιμότητα

Το GPS συμπεριλαμβάνει μια (προς το παρόν απενεργοποιημένη) δυνατότητα η οποία ονομάζεται επιλεκτική διαθεσιμότητα (Selective Availability) και η οποία προσθέτει σκόπιμα χρονομεταβλητά σφάλματα της τάξης μέχρι και των 100 μέτρων στο GPS σήμα το οποίο είναι δημοσίως ελεύθερο προς χρήση. Ο σκοπός αυτής της δυνατότητας ήταν η μη ικανότητα χρήσης του σήματος GPS από εχθρικούς ή πολιτικούς δέκτες για ακριβή καθοδήγηση οπλικών συστημάτων (π.χ. πυραυλικών συστημάτων).

Τα σφάλματα της επιλεκτικής διαθεσιμότητας στην πραγματικότητα είναι ψευδοτυχαία και δημιουργούνται από έναν κρυπτογραφικό αλγόριθμο με την χρήση ενός κλειδιού το οποίο είναι προσβάσιμο μόνο από εξουσιοδοτημένους χρήστες (τον στρατό των Ηνωμένων Πολιτειών, τους συμμάχους του, και μερικούς άλλους φορείς κυρίως κυβερνητικούς) με ειδικούς στρατιωτικούς δέκτες. Η κατοχή του ειδικού δέκτη από μόνη της είναι ανεπαρκής, αφού χρειάζεται το αυστηρώς ελεγχόμενο κλειδί το οποίο αλλάζει καθημερινά.

Πριν απενεργοποιηθεί το 2000, τυπικά σφάλματα της ΕΔ ήταν της τάξης των 10 μέτρων οριζοντίως και των 10 μέτρων καθέτως. Επειδή όμως η ΕΔ επηρεάζει κάθε δέκτη GPS σε μια δοσμένη περιοχή με σχεδόν τον ίδιο τρόπο, ένας σταθερός σταθμός με γνωστή την θέση του με μεγάλη ακρίβεια μπορεί να μετρήσει τα σφάλματα λόγω ΕΔ και να τα εκπέμψει στους τοπικούς GPS δέκτες ώστε να διορθώσουν τις υπολογιζόμενες θέσεις. Αυτό ονομάζεται Διαφορικό GPS (Differential GPS ή DGPS). Το DGPS διορθώνει και άλλα πολλά και σημαντικά σφάλματα που προέρχονται από άλλες πηγές, και συγκεκριμένα την ιονοσφαιρική καθυστέρηση, έτσι συνεχίζει να χρησιμοποιείται ακόμα και σήμερα που η

ΕΔ είναι απενεργοποιημένη. Η ανικανότητα της ΕΔ να αντιμετωπίσει το ευρέος διαδεδομένο DGPS ήταν το κύριο επιχείρημα για την απενεργοποίηση της, η οποία έγινε τελικά του 2000.

Η Επιλεκτική Διαθεσιμότητα αποτελεί ακόμα μια δυνατότητα του συστήματος GPS , και θεωρητικά μπορεί να ενεργοποιηθεί ξανά. Στην πράξη όμως λόγω τον κινδύνων και του κόστους είναι σχετικά απίθανο να ενεργοποιηθεί πάλι. Στις 19 Σεπτεμβρίου 2007, το υπουργείο αμύνης των Ηνωμένων Πολιτειών ανακοίνωσε οτι οι μελλοντικοί GPS III δορυφόροι δεν θα υλοποιούν την ΕΔ.

Σχετικότητα

Σύμφωνα με την θεωρεία της σχετικότητας, λόγω της συνεχούς κίνησης και το σχετικού ύψους τους ως προς το σύστημα αναφοράς της γης, τα ρολόγια των δορυφόρων επηρεάζονται από την ταχύτητα τους (ειδική σχετικότητα) και από το βαρυτικό δυναμικό (γενική σχετικότητα). Για τους δορυφόρους του GPS, η γενική σχετικότητα προβλέπει οτι τα ατομικά τους ρολόγια θα λειτουργούν πιο γρήγορα, περίπου 45.9 μs την ημέρα. Η ειδική σχετικότητα προβλέπει οτι τα ατομικά ρολόγια, που κινούνται στην τροχιά των GPS δορυφόρων, θα λειτουργούν πιο αργά κατά περίπου 7.2 μs την ημέρα. Λαμβάνοντας υπόψη τις προβλέψεις αυτές, η συχνότητα λειτουργίας κάθε ρολογιού πάνω σε κάθε δορυφόρο τίθεται λίγο μικρότερη από την επιθυμητή συχνότητα στην επιφάνεια της γης [24].

Παραμόρφωση Sagnac

Η παρατήρηση του σήματος GPS πρέπει να λαμβάνει υπόψη της το φαινόμενο Sagnac. Η χρονική κλίμακα του GPS ορίζεται σε ένα αδρανειακό σύστημα αλλά οι παρατηρήσεις γίνονται σε ένα γεωκεντρικό σύστημα, στο οποίο η έννοια του ταυτόχρονου δεν είναι μοναδικώς ορισμένη. Έτσι πρέπει να εφαρμοστεί ο μετασχηματισμός Lorentz για την μετατροπή από το αδρανειακό σύστημα στο γεωκεντρικό σύστημα. Αγνοώντας αυτό το φαινόμενο θα προέκυπταν σφάλματα εκατοντάδων nanoseconds ή δεκάδων μέτρων αντίστοιχα.

Παρεμβολές

Τα δορυφορικά σήματα είναι πολύ αδύναμα. Το πιο αδύναμο σήμα με το οποίο μπορεί να λειτουργήσει ένα κινητό τηλέφωνο είναι περίπου 1,000 φορές ισχυρότερο από το δυνατότερο δορυφορικό σήμα GPS, ενώ το πιο αδύναμο ψηφιακό τηλεοπτικό σήμα με το οποίο μια συσκευή τηλεόρασης μπορεί να λειτουργήσει είναι περίπου 100,000 φορές ισχυρότερο από το ισχυρότερο σήμα GPS. Όταν κάποιο εμπόδιο διακόπτει την άμεση οπτική επαφή με τον δορυφόρο, το σήμα γίνεται ακόμα πιο αδύναμο και πολύ ευαίσθητοι δέκτες απαιτούνται προκειμένου να το μετρήσουν.

Επομένως αφού το σήμα GPS όταν φτάνει στους επίγειους δέκτες τείνει να είναι αδύναμο, είναι εύκολο άλλες πηγές ηλεκτρομαγνητικής ακτινοβολίας να κάνουν την ανάκτηση και παρακολούθηση των δορυφορικών σημάτων δύσκολή ή ακόμα και αδύνατη.

Οι ηλιακές εκρήξεις στην επιφάνεια του ήλιου είναι ένα τέτοιο φυσικό φαινόμενο εκπομπής ηλεκτρομαγνητικής ακτινοβολίας που μπορεί να υποβαθμίσει την λήψη του σή-

ματος GPS, προκαλώντας επιπτώσεις σε περισσότερο από τη μισή επιφάνεια της γης η οποία βρίσκεται προς την μεριά του ήλιου. Τα σήματα GPS μπορούν να παρεμβληθούν από φυσικές γεωμαγνητικές καταιγίδες, οι οποίες μπορούν να συναντηθούν κυρίως κοντά στους πόλους της γης.

Σε δέκτες οι οποίοι βρίσκονται μέσα σε οχήματα, μεταλλικές κατασκευές μπορούν να λειτουργήσουν σαν κλωβοί Faraday, υποβαθμίζοντας την λήψη μόνο μέσα στο όχημα. Ηλεκτρομαγνητική Παρεμβολή ανθρώπινης προέλευσης μπορεί να παρενοχλεί ή να εμποδίζει το GPS σήμα.

Μερικές χώρες επιτρέπουν την χρήση επαναληπτών του σήματος GPS ώστε να είναι δυνατή η λήψη του μέσα σε κλειστούς χώρους. Παρόλα αυτά στο Ηνωμένο Βασίλειο και στην Ευρωπαϊκή Ένωση απαγορεύεται χρήση επαναληπτών λόγω της πιθανής παρεμβολής του μεταδιδόμενου σήματος με το αρχικό σε ορισμένες συσκευές.

3.1.7 Υποβοηθούμενο GPS

Σε συγκεκριμένες συνθήκες, όταν το σήμα είναι αδύναμο, το συμβατικό σύστημα GPS συναντά δυσκολία στο να παρέχει αξιόπιστα αποτελέσματα [26]. Για παράδειγμα όταν ο δέκτης βρίσκεται περιτριγυρισμένος από ψηλά κτίρια (κάτι που προκαλεί σφάλματα πολλαπλών διαδρομών), ή όταν το δορυφορικό σήμα είναι αδύναμο λόγω έλλειψης οπτικής επαφής με τον δορυφόρο (π.χ. σε εσωτερικούς κλειστούς χώρους).

Επιπλέον όταν βρίσκονται σε τέτοιες συνθήκες μερικοί μη υποβοηθούμενοι δέκτες μπορεί να μην μπορούν να ανακτήσουν το ημερολόγιο ή την ephemeris από κάποιον GPS δορυφόρο, καθιστώντας τον εαυτό τους ανίκανο να λειτουργήσει μέχρι να ληφθεί ένα καθαρό σήμα για ένα συνεχόμενο χρονικό διάστημα μέχρι και 40 δευτερολέπτων.

Ένας υποβοηθούμενος GPS δέκτης (Assisted-GPS) μπορεί να λύσει αυτά τα προβλήματα με πολλούς τρόπους, χρησιμοποιώντας τα στοιχεία κάποιου δικτύου όπως εξυπηρετητές βοήθειας (assistance servers) ή άλλα δεδομένα από κάποιο δίκτυο. Η βοήθεια χωρίζεται σε δύο κατηγορίες:

- Πληροφορίες που χρησιμοποιούνται για την γρήγορη ανάκτηση της θέσης των δορυφόρων
- Υπολογισμοί οι οποίοι γίνονται απομακρυσμένα:

    1. Ο βοηθητικός εξυπηρετητής μπορεί να εκτιμήσει την θέση της συσκευής η οποία συνδέεται σε κάποιο κυψελωτό δίκτυο από την κυψέλη στην οποία βρίσκεται η συσκευή.
    2. Ο βοηθητικός εξυπηρετητής λαμβάνει δυνατό δορυφορικό σήμα και διαθέτει μεγάλη υπολογιστική ισχύ. Έτσι μπορεί να συγκρίνει σήματα που στέλνονται από κινητές συσκευές με το δορυφορικό σήμα που λαμβάνει απευθείας και στην συνέχεια να πληροφορήσει τις κινητές συσκευές.
    3. Ο βοηθητικός εξυπηρετητής μπορεί να παρέχει πληροφορίες για τις τροχιές η το ημερολόγιο στις κινητές συσκευές, επιτρέποντας σε αυτές να κλειδώνουν γρηγορότερα στους δορυφόρους.

4. Το δίκτυο μπορεί να παρέχει τον ατομικό χρόνο (Accurate Time Assistance).
5. Ο βοηθητικός εξυπηρετητής μπορεί να έχει καλύτερη γνώση για τις ιονοσφαιρικές συνθήκες και άλλους παράγοντες που προκαλούν σφάλματα στην περιοχή των πύργων των κυψελών από ότι το κινητό τηλέφωνο από μόνο του.

Ένας τυπικός δέκτης A-GPS χρησιμοποιεί συνδέσεις (μέσω Διαδικτύου ή μέσω άλλων τρόπων) για να επικοινωνήσει με τον βοηθητικό εξυπηρετητή ή με το δίκτυο που παρέχει A-GPS πληροφορίες. Εάν επιπλέον έχει έναν αυτόνομο λειτουργικό δέκτη συμβατικού GPS, μπορεί να τον χρησιμοποιήσει έχοντας μια καθυστέρηση στην πρώτη προσπάθεια ανίχνευσης θέσης χωρίς όμως τις αρνητικές συνέπειες της εξάρτησης από ένα εξωτερικό δίκτυο, όπως αποτυχία σύνδεσης ή αλλαγές στην κίνηση δεδομένων του δικτύου. Παρόλα αυτά ορισμένες A-GPS συσκευές δεν έχουν την δυνατότητα να χρησιμοποιήσουν μόνο το συμβατικό αυτόνομο GPS.

3.2 Λοιπές τεχνικές ανίχνευσης θέσης

3.2.1 Εισαγωγή

Εκτός από το GPS το οποίο είναι το σύστημα ανίχνευσης θέσης το οποίο χρησιμοποιείται στην ανάπτυξη της Location Based Service αυτής της εργασίας και το οποίο αποτελεί μια από τις κυρίαρχες τεχνολογίες ανίχνευσης θέσης σε εξωτερικούς χώρους, υπάρχει και ένα πλήθος άλλων τεχνολογιών ανίχνευσης θέσης. Κάθε μια από αυτές τις τεχνολογίες έχει τα δικά της πλεονεκτήματα και οι περισσότερες υπερτερούν του GPS στην ανίχνευση θέσης σε εσωτερικούς χώρους. Μερικές από τις πιο διαδεδομένες τεχνολογίες ανίχνευσης θέσης θα εξετάσουμε στην συνέχεια.

3.2.2 Συστήματα ανίχνευσης θέσης βασισμένα σε υπέρυθρες ακτίνες

Η χρήση των υπέρυθρων ακτίνων για εντοπισμό θέσης είναι από τις πιο κοινές τεχνικές, αφού η υπέρυθρη τεχνολογία είναι ενσωματωμένη σε ένα πλήθος ενσύρματων και ασύρματων συσκευών. Προκειμένου να λειτουργήσει ένα σύστημα ανίχνευσης θέσης βασισμένο σε υπέρυθρες ακτίνες πρέπει να υπάρχει οπτική επαφή μεταξύ του πομπού και του δέκτη. Έτσι η κάλυψη που μπορεί να παρέχει ένα τέτοιο σύστημα περιορίζεται σε κλειστούς και όχι πολύ μεγάλους χώρους (π.χ. ένα δωμάτιο).

Στα πλεονεκτήματα αυτής της μεθόδου συγκαταλέγονται η ακρίβεια των υπολογισμών θέσης, το μικρό μέγεθος των πομπών ώστε να μεταφέρονται εύκολα από ένα άτομο και η απλότητα της αρχιτεκτονικής των συστημάτων που την υλοποιούν κάτι που μειώνει το κόστος και χρόνο εγκατάστασης και συντήρησης.

Παρόλα αυτά η μέθοδος έχει και μειονεκτήματα. Υπάρχουν ορισμένοι περιορισμοί στον εντοπισμό της θέσης κυρίως λόγω της παρεμβολής από ισχυρές πηγές φωτός (π.χ. ηλιακό φως). Το πρόβλημα αυτό μπορεί να λυθεί με χρήση οπτικών και ηλεκτρονικών

φίλτρων ή με την υλοποίηση αλγορίθμων μείωσης του θορύβου στους δέκτες. Όλες αυτές οι λύσεις όμως αυξάνουν το κόστος του συστήματος εντοπισμού. Άλλο ένα μειονέκτημα τέτοιων συστημάτων αποτελεί το υψηλό κόστος του συνολικού εξοπλισμού. Παρότι οι πομποί IR είναι σχετικά φτηνοί, το όλο σύστημα της συστοιχίας των δεκτών και των καλωδίων διασύνδεσης μεταξύ τους είναι ακριβό σε σχέση με το μέγεθος της περιοχής κάλυψης. Τέλος αν κάποια συσκευή IR καλυφθεί από τα ρούχα του ατόμου το οποίο τον φέρει (ή κάποια παρόμοια κατάσταση) το σύστημα αδυνατεί να λειτουργήσει καθώς οι υπέρυθρες ακτίνες δεν μπορούν να διαπεράσουν μη διαφανή υλικά.

3.2.3 Συστήματα ανίχνευσης θέσης βασισμένα σε υπέρηχους

Η χρήση υπέρηχων είναι ακόμα ένας τρόπος εντοπισμού θέσης. Οι υπέρηχοι χρησιμοποιούνται από τις νυχτερίδες για την καθοδήγηση τους στο σκοτάδι, κάτι που ενέπνευσε την δημιουργία ενός παρόμοιου συστήματος εντοπισμού θέσης. Έτσι δημιουργήθηκε ένα πλήθος συστημάτων ανίχνευσης θέσης βασισμένων στους υπέρηχους τα οποία παρέχουν μια κάπως φτηνή λύση σε αυτό το πρόβλημα. Συνήθως τα υπερηχητικά σήματα χρησιμοποιούνται σε συνδυασμό με RF σήματα τα οποία χρησιμεύουν για συγχρονισμό και συντονισμό. Με αυτόν τον τρόπο αυξάνεται η περιοχή κάλυψης του συστήματος εντοπισμού θέσης. Παρόλα αυτά συστήματα βασισμένα σε υπέρηχους έχουν μικρότερη ακρίβεια εντοπισμού από αντίστοιχα IR συστήματα, ενώ επηρεάζονται αρνητικά από πηγές ηχητικού θορύβου και ανακλάσεις.

3.2.4 Συστήματα ανίχνευσης θέσης βασισμένα σε ραδιοκύματα

Η χρήση ραδιοκυμάτων (RF ή Radio Frequency) στον εντοπισμό θέσης κληρονομεί ορισμένα από τα πλεονεκτήματα τους. Συγκεκριμένα τα ραδιοκύματα μπορούν να περνούν μέσα από τοίχους και από το ανθρώπινο σώμα ευκολότερα από άλλου είδους κύματα και έτσι συστήματα εντοπισμού θέσης που τα χρησιμοποιούν πετυχαίνουν μεγαλύτερη κάλυψη και άρα χρήση λιγότερου υλικού εξοπλισμού σε σχέση με τα υπόλοιπα συστήματα. Επιπλέον συστήματα βασισμένα σε RF μπορούν να χρησιμοποιήσουν υπαρκτές τεχνολογίες όπως τα AP (Access Points), WLAN (Wireless Lan π.χ. WiFi κλπ) και WPAN (Wireless Personal Area Network π.χ. Bluetooth). Οι πιο διαδεδομένες τεχνικές που χρησιμοποιούνται σε τέτοια συστήματα είναι αυτές της τριγωνοποίησης και του δακτυλικού αποτυπώματος.

Αναγνώριση ραδιοσυχνότητας (RFID)

Η RFID (Radio Frequency Identification) αποτελεί μια μέθοδο αποθήκευσης και ανάκτησης δεδομένων μέσω της ηλεκτρομαγνητικής μετάδοσης σε ένα RF συμβατό ολοκληρωμένο κύκλωμα. Τα συστήματα εντοπισμού θέσης αυτού του είδους χρησιμοποιούνται συνήθως σε περίπλοκους εσωτερικούς χώρους όπως γραφεία, νοσοκομεία κλπ. Η RFID ως ασύρματη τεχνολογία επιτρέπει επίσης την φτηνή αναγνώριση ξεχωριστών ατόμων ή συσκευών και μπορεί να αντικαταστήσει άλλες μεθόδους αναγνώρισης όπως για παράδειγμα

τα barcodes.

Υπάρχουν δύο είδη της RFID τεχνολογίας: παθητική RFID και ενεργητική RFID. Με την παθητική RFID, η συσκευή που ανιχνεύεται είναι ένας δέκτης. Έτσι συσκευές ανίχνευσης παθητικής RFID είναι σχετικά μικρές και φτηνές. Το μειονέκτημα τους είναι οτι η εμβέλεια χρήσης τους είναι περιορισμένη. Οι συσκευές ανίχνευσης ενεργητικής RFID είναι πομποδέκτες, οι οποίοι μεταδίδουν την ταυτότητα τους και άλλες πληροφορίες. Το κόστος τέτοιων συσκευών είναι υψηλότερο από το αντίστοιχο κόστος των παθητικών συσκευών. Από την άλλη πλευρά όμως η εμβέλεια τους είναι μεγαλύτερη.

Ασύρματα τοπικά δίκτυα (WLAN)

Η τεχνολογία WLAN έχει γίνει πολύ δημοφιλής τα τελευταία χρόνια και χρησιμοποιείται σε ένα πλήθος δημόσιων χώρων. Συστήματα ανίχνευσης θέσης βασισμένα σε WLAN χρησιμοποιούν τις υπάρχουσες εγκατεστημένες δομές του WLAN μειώνοντας έτσι το κόστος του συστήματος ανίχνευσης. Η ακρίβεια στον εντοπισμό της θέσης που βασίζεται στην μέτρηση της ισχύος του σήματος του WLAN επηρεάζεται από διάφορα στοιχεία όπως η κίνηση και ο προσανατολισμός του ανθρώπινου σώματος, η αλληλοκάλυψη των AP, οι τοίχοι, οι πόρτες κ.α. Όλα αυτά τα στοιχεία μειώνουν την ακρίβεια εντοπισμού, η οποία μπορεί να βελτιωθεί με χρήση αποθηκευμένης πληροφορίας και την τεχνική του δακτυλικού αποτυπώματος με κόστος την μεγαλύτερη πολυπλοκότητα και το μεγαλύτερο κόστος των συστημάτων εντοπισμού, ειδικά εάν ο αριθμός των χρηστών μεγαλώσει σημαντικά.

Bluetooth

Το Bluetooth (IEEE 802.15.1) αποτελεί ένα πρότυπο για ασύρματα προσωπικά δίκτυα (Wireless Personal Area Network ή WPAN). Επιτρέπει την επικοινωνία σε μια ακτίνα των περίπου 100 μέτρων και προορίζεται να αντικαταστήσει τους ενσωματωμένους IR πομποδέκτες στις κινητές συσκευές, ενώ ήδη έχει ενσωματωθεί σε ένα μεγάλο μέρος κινητών συσκευών. Επιπλέον οι πομποδέκτες Bluetooth είναι σχετικά φτηνοί επιτρέποντας την δημιουργία, φτηνών συστημάτων εντοπισμού θέσης. Σε τέτοια συστήματα οι χρήστες οργανώνονται σε συστάδες και η θέση μιας κινητής συσκευής υπολογίζεται από τις υπόλοιπες συσκευές της συστάδας. Τα μειονεκτήματα της χρήσης του Bluetooth είναι η μικρή ακρίβεια (μεταξύ 2 και 3 μέτρων) και η μεγάλη καθυστέρηση (της τάξης των 20 δευτερολέπτων).

Δίκτυα αισθητήρων

Οι αισθητήρες είναι συσκευές εκτεθειμένες σε κάποιες περιβαλλοντολογικές συνθήκες συμπεριλαμβανομένων του ήχου, της πίεσης, της θερμοκρασίας, του φωτός κλπ οι οποίες παράγουν εξόδους ανάλογες των μεταβολών αυτών των συνθηκών. Οι αισθητήρες χωρίζονται σε δύο κατηγορίες, στους ενεργούς και στους παθητικούς. Οι ενεργοί αισθητήρες μπορούν να αλληλεπιδρούν με το περιβάλλον (πχ ραντάρ). Οι παθητικοί μπορούν μόνο να λαμβάνουν πληροφορίες από το περιβάλλον τους. Τα συστήματα εντοπισμού θέσης που

βασίζονται σε αισθητήρες αποτελούνται από ένα μεγάλο αριθμό αισθητήρων τοποθετημένων σε προκαθορισμένες θέσεις. Από τις μετρήσεις που λαμβάνονται από τις θέσεις αυτές μπορεί να υπολογιστεί η θέση ενός ατόμου η μιας κινητής συσκευής. Έτσι αποτελούν μια αποτελεσματική μέθοδο εντοπισμού θέσης με βάση το κόστος και την ευκολία ανίχνευσης της θέσης, λόγω του όλο και μικρότερου κόστους και μεγέθους των αισθητήρων. Από την άλλη πλευρά όμως οι χαμηλού κόστους αισθητήρες έχουν περιορισμένη επεξεργαστική ικανότητα και ισχύ μπαταρίας σε σχέση με άλλες ασύρματες συσκευές.

Τεχνολογία υπέρ-ευρείας ζώνης UWB

Τα συστήματα που βασίζονται σε RF σήματα για την ανίχνευση θέσης αντιμετωπίζουν το πρόβλημα της παραμόρφωσης λόγω πολλών μονοπατιών από ανακλάσεις του σήματος. Οι παλμοί υπέρ-ευρείας ζώνης (Ultra-Wide Band ή UWB) έχοντας πολύ μικρή διάρκεια καθιστούν δυνατό το φιλτράρισμα των ανακλώμενων σημάτων από το αρχικό σήμα, το οποίο αυξάνει την ακρίβεια του εντοπισμού θέσης. Η οπτική επαφή μεταξύ πομπού και δέκτη δεν είναι απαραίτητη, δεν υπάρχει παραμόρφωση λόγω πολλαπλών μονοπατιών, μειώνονται οι παρεμβολές και αυξάνεται η διεισδυτική ικανότητα του σήματος όταν χρησιμοποιείται UWB. Έτσι με την τεχνολογία UWB επιτυγχάνεται μεγαλύτερη ακρίβεια. Τέλος η μεγάλη περιοχή κάλυψης που προσφέρουν συστήματα βασισμένα σε UWB τα καθιστούν εύκολα επεκτάσιμα.

3.2.5 Συστήματα ανίχνευσης θέσης βασισμένα σε μαγνητικά σήματα

Η χρήση μαγνητικών σημάτων αποτελεί έναν παλιό τρόπο ανίχνευσης θέσης. Συστήματα βασισμένα σε μαγνητικά πεδία προσφέρουν υψηλή ακρίβεια ενώ δεν υποφέρουν από τα μειονεκτήματα της έλλειψης οπτικής επαφής, όπου παρεμβάλλεται ένα εμπόδιο μεταξύ του πομπού και του δέκτη. Τέτοια συστήματα μπορούν επιπλέον να καταγράψουν ακόμα και κινήσεις του ανθρώπινου σώματος με χρήση πολλαπλών αισθητήρων τοποθετημένων πάνω σε αυτό, αφού έχουν την δυνατότητα παρακολούθησης πολλαπλών αισθητήρων ταυτόχρονα σε πραγματικό χρόνο. Οι αισθητήρες αυτοί είναι μικροί σε μέγεθος και φτηνοί. Το βασικό μειονέκτημα αυτών των συστημάτων είναι η μικρή περιοχή κάλυψης που μπορούν να επιτύχουν και χρειάζεται περαιτέρω έρευνα για την ανάπτυξη συστημάτων που θα παρέχουν μεγαλύτερες περιοχές κάλυψης.

3.2.6 Συστήματα ανίχνευσης θέσης βασισμένα σε τεχνητή όραση

Συστήματα ανίχνευσης τα οποία βασίζονται στην τεχνολογία τεχνητής όρασης μπορούν να χρησιμοποιηθούν για ανίχνευση θέσης και αναγνώριση ατόμων και συσκευών σε πολύπλοκα εσωτερικά περιβάλλοντα. Στα συστήματα αυτά τα άτομα ή τα αντικείμενα που ανιχνεύονται δεν χρειάζεται να φέρουν πάνω τους κάποια συσκευή ανίχνευσης. Σε ένα σύστημα βασισμένο στην τεχνολογία τεχνητής όρασης μπορεί να χρησιμοποιήσει μια σχετικά φτηνή κάμερα για να καλύψει μια σχετικά μεγάλη περιοχή. Ένα μειονέκτημα των

συστημάτων αυτών είναι η έλλειψη της ιδιοτικότητας των ατόμων που ανιχνεύονται. Ένα δεύτερο μειονέκτημα είναι η αδυναμία προσαρμογής σε περιβάλλοντα τα οποία αλλάζουν δυναμικά. Οι υπολογισμοί θέσης βασίζονται σε στατικές εικόνες οι οποίες βρίσκονται αποθηκευμένες σε βάσεις δεδομένων, οι οποίες πρέπει να αλλάζουν αν το περιβάλλον αλλάξει για κάποιον λόγο. Επιπλέον η τεχνητή όραση επηρεάζεται αρνητικά από τις συνθήκες που επικρατούν στο περιβάλλον όπως ο καιρός ή το επίπεδο της φωτεινής ακτινοβολίας. Τέλος η ανίχνευση πολλαπλών ατόμων αποτελεί ακόμη μια πρόκληση για συστήματα τεχνητής όρασης.

Κεφάλαιο 4

Λογισμικό Android

4.1 Γενικά στοιχεία για το Android

Το Android αποτελεί μια στοίβα λογισμικού για κινητές συσκευές η οποία περιλαμβάνει ένα λειτουργικό σύστημα, μεσισμικό και βασικές εφαρμογές. Εφαρμογές για αυτήν την πλατφόρμα μπορούν να αναπτυχθούν χρησιμοποιώντας το Android SDK και την γλώσσα προγραμματισμού Java. Οι εφαρμογές αυτές τρέχουν στην Dalvik Virtual Machine, μια τροποποιημένη Virtual Machine σχεδιασμένη για χρήση σε ενσωματωμένα συστήματα η οποία τρέχει πάνω σε έναν πυρήνα Linux έκδοση 2.6.

Το Android SDK παρέχει APIs για την χρήση web browser, εμφάνιση δισδιάστατων και τρισδιάστατων γραφικών, δομημένη αποθήκευση δεδομένων σε βάση δεδομένων, εμφάνιση πολυμεσικού υλικού (ήχος, βίντεο, εικόνες), χρήση των τεχνολογιών GSM, Bluetooth, EDGE, 3G και WiFi, χρήση συσκευών όπως φωτογραφική μηχανή, GPS, πυξίδα, επιταχυνσιόμετρο.

4.2 Αρχιτεκτονική του Android

Η πλατφόρμα του Android αποτελείται από μια στοίβα λογισμικού. Τα επίπεδα της στοίβας του Android από το υψηλότερο προς το χαμηλότερο περιγράφονται παρακάτω:

- Επίπεδο Εφαρμογών (Application). Στο επίπεδο αυτό περιλαμβάνεται ένα σύνολο από βασικές εφαρμογές μερικές από τις οποίες είναι e-mail client, πρόγραμμα SMS, ημερολόγιο, χάρτες, browser, επαφές κ.α. Όλες οι εφαρμογές είναι γραμμένες με χρήση της γλώσσας προγραμματισμού Java.
- Επίπεδο Πλαισίου Εφαρμογών (Application Framework). Βρίσκεται κάτω από το επίπεδο Εφαρμογών και αποτελείται ένα σύνολο συστημάτων και υπηρεσιών:
  - Ένα σύνολο από γραφικά στοιχεία (Views) για την δημιουργία γραφικού περιβάλλοντος συμπεριλαμβανομένων λιστών (lists), πλεγμάτων (grids), κουτιών κειμένου (text boxes), κουμπιών (buttons) κ.α.

  - Ένα διαχειριστή περιεχομένου (Content Manager) ο οποίος επιτρέπει στις εφαρμογές την πρόσβαση σε δεδομένα άλλων εφαρμογών ή τον διαμοιρασμό των δικών τους δεδομένων με άλλες εφαρμογές.
  - Ένα διαχειριστή πόρων (Resource Manager) για την πρόσβαση στους πόρους όπως strings, εικόνες, layout files.
  - Έναν διαχειριστή ειδοποιήσεων (Notification Manager) ο οποίος επιτρέπει την προβολή ειδοποιήσεων στην μπάρα κατάστασης (status bar).
  - Έναν διαχειριστή δραστηριοτήτων (Activity Manager) ο οποίος διαχειρίζεται τον κύκλο ζωής των εφαρμογών.

- Επίπεδο Βιβλιοθηκών (Libraries). Το οποίο περιλαμβάνει ένα σύνολο από βιβλιοθήκες γραμμένες σε C/C++ οι οποίες χρησιμοποιούνται από διάφορα στοιχεία του συστήματος του Android. Οι δυνατότητες που προσφέρουν αυτές οι βιβλιοθήκες είναι προσβάσιμες στους προγραμματιστές δια μέσου του επιπέδου πλαισίου εφαρμογής.

- Επίπεδο Εκτέλεσης (Android Runtime). Το οποίο αποτελείται από ένα σύνολο από βασικές βιβλιοθήκες και την Dalvik Virtual Machine.

- Ο Πυρήνας του Linux. Το Android βασίζεται στον πυρήνα Linux έκδοση 2.6 για βασικές υπηρεσίες συστήματος όπως ασφάλεια, διαχείριση μνήμης, διαχείριση διεργασιών, στοίβα δικτύου, και οδηγούς συσκευών. Ο πυρήνας λειτουργεί επίσης ως ένα ενδιάμεσο επίπεδο αφαίρεσης μεταξύ της στοίβας λογισμικού και του υλικού.

4.3 Εργαλεία ανάπτυξης εφαρμογών

4.3.1 Περιβάλλοντα ανάπτυξης εφαρμογών

Η ανάπτυξη και αποσφαλμάτωση των εφαρμογών μπορούν να γίνουν με την χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης εφαρμογών Eclipse και του Android Deve lopment Tools Plug-in το οποίο δίνει την δυνατότητα πρόσβασης στα εργαλεία ανάπτυξης του Android SDK μέσα από το περιβάλλον του Eclipse.

Εναλλακτικά η ανάπτυξη και αποσφαλμάτωση των εφαρμογών μπορούν να γίνουν με χρήση κάποιου άλλου περιβάλλοντος ανάπτυξης εφαρμογών και των εργαλείων που παρέχει το Android SDK.

4.3.2 Ο εξομοιωτής

Προκειμένου να γίνει ευκολότερη η διαδικασία της ανάπτυξης και αποσφαλμάτωσης μιας εφαρμογής το Android SDK περιλαμβάνει έναν εξομοιωτή μιας εικονικής κινητής συσκευής η οποία τρέχει το λειτουργικό του Android. Έτσι δεν είναι η αναγκαία η ύπαρξη

πραγματικής κινητής συσκευής για την εκτέλεση και δοκιμή των εφαρμογών. Ο εξομοιωτής προσομοιώνει ένα μεγάλο πλήθος λειτουργιών μιας τυπικής συσκευής η οποία τρέχει το Android:

- Παρέχει μια ποικιλία πλήκτρων πλοήγησης και ελέγχου.
- Παρέχει μια οθόνη για την προβολή των εφαρμογών που τρέχουν στον εξομοιωτή.
- Επιτρέπει στις εφαρμογές την χρήση των υπηρεσιών που προσφέρει η πλατφόρμα του Android δηλαδή την κλήση άλλων εφαρμογών, την πρόσβαση στο δίκτυο, την αναπαραγωγή ήχου και βίντεο, την αποθήκευση και επαναφορά δεδομένων, την ειδοποίηση χρήστη, το γραφικό περιβάλλον του Android.

Επίσης παρέχει ένα πλήθος λειτουργιών για την ευκολότερη αποσφαλμάτωση:

- Κονσόλα για την καταγραφή της εξόδου του πυρήνα.
- Προσομοίωση διακοπών (όπως η άφιξη SMS μηνύματος ή τηλεφωνικής κλήσης)
- Προσομοίωση καθυστέρησης και απώλειας στο κανάλι δεδομένων
- Προσομοίωση λήψης δεδομένων θέσης από την συσκευή GPS.

4.3.3 Άλλα εργαλεία του Android

Το Android SDK περιλαμβάνει μερικά ακόμη εργαλεία για την ανάπτυξη εφαρμογών:

- Το Dalvik Debug Monitor Service (DDMS) το οποίο επιτρέπει την διαχείριση των διεργασιών στον εξομοιωτή ή στην συσκευή. Συγκεκριμένα δίνεται η δυνατότητα port-forwarding υπηρεσιών, λήψη screenshots, εμφάνιση πληροφοριών για τον σωρό και τα νήματα, logcat, εμφάνιση πληροφοριών ράδιο και πληροφοριών διεργασιών, προσομοίωση εισερχόμενων κλήσεων και μηνυμάτων, προσομοίωση δεδομένων θέσης κ.α.
- Την Android Debug Bridge (adb) η οποία επιτρέπει την διαχείριση της κατάστασης του εξομοιωτή ή της συσκευής. Μέσω του adb είναι δυνατή η εκτέλεση εντολών φλοιού, η διαχείριση της προώθησης θυρών και η αντιγραφή από και προς την συσκευή ή τον εξομοιωτή.
- Το Android Asset Packaging Tool (aapt) το οποίο δίνει την δυνατότητα δημιουργίας .apk αρχείων τα οποία περιέχουν τα εκτελέσιμα και τους πόρους μιας εφαρμογής.
- Την Android Interface Description Language (aidl) η οποία επιτρέπει την δημιουργία κώδικα που επιτρέπει σε δύο διεργασίες σε μια συσκευή βασισμένη στο Android να συνομιλούν χρησιμοποιώντας διαδιεργασιακή επικοινωνία.
- Το sqlite3 το οποίο επιτρέπει την πρόσβαση στα δεδομένα της SQLite που δημιουργούνται από τις διάφορες εφαρμογές.

- Το Traceview το οποίο επιτρέπει την γραφική προβολή της ανάλυσης των trace log data που δημιουργούν οι διάφορες εφαρμογές.

- Το mksdcard το οποίο βοηθά στην δημιουργία εικονικού δίσκου ο οποίος μπορεί να χρησιμοποιηθεί από τον εξομοιωτή για την προσομοίωση της παρουσίας εξωτερικής αποθηκευτικής κάρτας (όπως η SD card).

- Το dx tool το οποίο μετατρέπει τα αρχεία .class από java bytecode σε Android bytecode.

- Το UI/Application Exerciser Monkey το οποίο είναι ένα πρόγραμμα που τρέχει στον εξομοιωτή και παράγει ψευδό-τυχαίες σειρές από συμβάντα χρήστη όπως clicks, touches, gestures καθώς επίσης και έναν αριθμό από συμβάντα συστήματος.

- Το activitycreator το οποίο είναι ένα script που δημιουργεί Ant build αρχεία τα οποία μπορούν να χρησιμοποιηθούν για την μεταγλώττιση των εφαρμογών.

4.4 Μοντέλο προγραμματισμού

4.4.1 Μοντέλο εφαρμογών

Στα περισσότερα συστήματα υπάρχει μια στενή συσχέτιση μεταξύ του εκτελέσιμου αρχείου στο οποίο υπάρχει μια εφαρμογή, της διεργασίας στην οποία τρέχει και στο περιβάλλον μέσα στο οποίο ο χρήστης αλληλεπιδρά με αυτήν. Στο Android αυτή η συσχέτιση είναι πιο χαλαρή λόγω της ευέλικτης φύσης των εφαρμογών οι οποίες είναι γραμμένες για αυτό. Για μια εφαρμογή Android γίνεται ο εξής διαχωρισμός:

- Ένα android package (.apk) είναι το αρχείο το οποίο περιέχει τον εκτελέσιμο κώδικα και τους πόρους μιας εφαρμογής. Είναι το αρχείο που διαμοιράζεται και χρησιμοποιούν οι χρήστες προκειμένου να εγκαταστήσουν την εφαρμογή στην συσκευή τους.

- Ένα task είναι αυτό που αντιλαμβάνεται ο χρήστης σαν μια εφαρμογή η οποία μπορεί να εκκινήσει. Συνήθως ένα task έχει το δικό του εικονίδιο μέσω του οποίου ο χρήστης έχει πρόσβαση σε αυτό και είναι προσβάσιμο ως αντικείμενο του πιο υψηλού επιπέδου που μπορεί έρθει στο προσκήνιο μπροστά από άλλα tasks.

- Μια process είναι μια χαμηλού επιπέδου διεργασία του πυρήνα στην οποία τρέχει ο κώδικας της εφαρμογής. Συνήθως όλος ο κώδικας που περιέχεται σε ένα .apk εκτελείται μέσα σε μια διεργασία, αφιερωμένη για το συγκεκριμένο .apk. Παρόλα αυτά η ετικέτα διεργασίας μπορεί να χρησιμοποιηθεί για να τροποποιηθεί το που θα εκτελεστεί ο κώδικας είτε για ολόκληρο το .apk είτε για ένα μόνο στοιχείο του package.

4.4.2 Δομικά στοιχεία μιας εφαρμογής

Τα βασικά δομικά στοιχεία μιας εφαρμογής Android:

1. Δραστηριότητα (Activity)
2. Δέκτης Εκπεμπόμενων Προθέσεων (Broadcast Intent Receiver)
3. Υπηρεσία (Service)
4. Πάροχος Περιεχομένου (Content Provider)
5. Πρόθεση και Φίλτρο Προθέσεως (Intent και Intent Filter)
6. Ειδοποίηση (Notification)
7. Όψη (View)
8. AndroidManifest.xml

Δεν είναι απαραίτητο κάθε εφαρμογή να περιέχει όλα αυτά στοιχεία, αλλά μια εφαρμογή είναι ένας συνδυασμός μερικών από αυτά. Παρακάτω γίνεται μια σύντομη περιγραφή του κάθε ενός.

Δραστηριότητα

Η Δραστηριότητα (Activity) αποτελεί το πιο κοινό από τα δομικά στοιχεία μιας εφαρμογής. Συνήθως είναι μια μόνο ξεχωριστή οθόνη σε μια εφαρμογή. Κάθε Δραστηριότητα υλοποιείται μέσα σε μια ξεχωριστή κλάση που επεκτείνει την κλάση Activity. Παρουσιάζει την διαπροσωπεία χρήστη (User Interface) και ανταποκρίνεται σε διάφορα συμβάντα.

Δέκτης Εκπεμπόμενων Προθέσεων

Ένας Δέκτης Εκπεμπόμενων Προθέσεων (BroadcastReceiver) μπορεί να χρησιμοποιηθεί όταν χρειάζεται να εκτελεστεί κώδικας μιας εφαρμογής ως αποτέλεσμα ενός εξωτερικού συμβάντος. Ο receiver δεν εμφανίζει user interface αν και μπορεί να χρησιμοποιήσει τον Διαχειριστή Ειδοποιήσεων (NotificationManager) για να ειδοποιήσει τον χρήστη. Δεν είναι απαραίτητο μια εφαρμογή η οποία περιέχει έναν Δέκτη Εκπεμπόμενων Προθέσεων να τρέχει για να ενεργοποιηθεί ο receiver. Το σύστημα θα ενεργοποιήσει μια εφαρμογή, αν είναι απαραίτητο, όταν ενεργοποιηθεί κάποιος Δέκτης Εκπεμπόμενων Προθέσεων. Τέλος μια εφαρμογή μπορεί να στείλει τα δικά της intent broadcasts σε άλλες εφαρμογές.

Υπηρεσία

Μια Υπηρεσία (Service) είναι κώδικας ο οποίος τρέχει χωρίς user interface. Άλλα στοιχεία μιας εφαρμογής μπορούν να συνδεθούν με μια Υπηρεσία και να επικοινωνήσουν μαζί του διαμέσου μιας διαπροσωπείας που παρέχεται από την Υπηρεσία.

Πάροχος Περιεχομένου

Οι εφαρμογές μπορούν να αποθηκεύουν τα δεδομένα τους σε αρχεία, βάσεις δεδομένων SQLite ή με κάποιον άλλον μηχανισμό. Ένας Πάροχος Περιεχομένου (Content Provider) όμως είναι χρήσιμος όταν χρειάζεται μια εφαρμογή να μοιράζεται τα δεδομένα της με άλλες εφαρμογές. Ο Πάροχος Περιεχομένου είναι μια κλάση που υλοποιεί ένα αριθμό από μεθόδους που επιτρέπουν στις εφαρμογές να αποθηκεύουν και να επαναφέρουν δεδομένα του συγκεκριμένου τύπου που χειρίζεται ο Πάροχος Περιεχομένου.

Πρόθεση και Φίλτρο Προθέσεως

Ένα αντικείμενο Πρόθεσης (Intent) είναι ένα αντικείμενο το οποίο περιγράφει τι θέλει να κάνει μια εφαρμογή. Τα βασικά στοιχεία μιας Πρόθεσης είναι ποια ενέργεια θέλει η εφαρμογή να εκτελεστεί και τα δεδομένα πάνω στα οποία θα εκτελεστεί η συγκεκριμένη ενέργεια. Ενώ ένα αντικείμενο Πρόθεσης αποτελεί μια πρόθεση να γίνει κάτι ένα αντικείμενο Φίλτρο Προθέσεως (Intent Filter) αποτελεί μια περιγραφή του τι είδους Προθέσεις είναι δυνατόν να εξυπηρετηθούν.

Ειδοποίηση

Μια Ειδοποίηση (Notification) αποτελεί ένα μικρό εικονίδιο που εμφανίζεται στην μπάρα καταστάσεων. Ο χρήστης μπορεί να αλληλεπιδράσει με το εικονίδιο αυτό για να λάβει περισσότερες πληροφορίες.

Όψη

Η Όψη (View) είναι ένα αντικείμενο το οποίο εμφανίζεται στην οθόνη. Το user interface δημιουργείται με χρήση Όψεων.

AndroidManifest.xml

Το AndroidManifest.xml αρχείο είναι το αρχείο ελέγχου το οποίο υπάρχει στον κεντρικό φάκελο κάθε εφαρμογής και μέσα στο οποίο περιγράφονται καθολικές ιδιότητες της εφαρμογής

4.4.3 Ο κύκλος ζωής μιας εφαρμογής Android

Στις περισσότερες περιπτώσεις, κάθε εφαρμογή Android τρέχει στην δική της ξεχωριστή διεργασία. Η διεργασία αυτή δημιουργείται όταν χρειάζεται να εκτελεστεί κάποιο μέρος του κώδικα της εφαρμογής, συνεχίζει να υπάρχει μέχρι ωσότου δεν είναι χρήσιμη πλέον και το σύστημα χρειάζεται να ελευθερώσει την μνήμη που καταλαμβάνει για χρησιμοποίηση από άλλες εφαρμογές.

Ένα βασικό χαρακτηριστικό του Android είναι ότι μια εφαρμογή δεν ελέγχει άμεσα τον κύκλο ζωής της. Αντιθέτως το σύστημα αποφασίζει για τον κύκλο ζωής μιας εφαρμογής αναλόγως με το ποια μέρη της εφαρμογής τρέχουν, πόσο σημαντικά είναι αυτά για τον χρήστη και πόση είναι η διαθέσιμη μνήμη του συστήματος.

Η απόφαση του συστήματος για το ποια διεργασία πρέπει να τερματιστεί σε περίπτωση έλλειψης μνήμης βασίζεται σε μια ιεράρχηση των διεργασιών. Τα επίπεδα αυτής της ιεράρχησης είναι τα εξής:

1. Διεργασία Προσκηνίου (foreground process). Μια διεργασία θεωρείται διεργασία προσκηνίου όταν ισχύει κάποια από τις εξής συνθήκες:
    - Περιέχει μια Δραστηριότητα (Activity) στο υψηλότερο επίπεδο της οθόνης και ο χρήστης αλληλεπιδρά με αυτήν.
    - Εκτελείται ένας Δέκτη Εκπομπής (BroadcastReceiver) που περιέχεται στην διεργασία
    - Εκτελείται κώδικας κάποιου Service από κάποια από τις συναρτήσεις δημιουργίας, εκκίνησης, καταστροφής του.
2. Ορατή Διεργασία (visible process). Μια διεργασία που περιέχει μια Δραστηριότητα (Activity) που είναι ορατή στην οθόνη αλλά δεν βρίσκεται στο προσκήνιο.
3. Διεργασία Υπηρεσίας (service process). Μια διεργασία που περιέχει μια Υπηρεσία (Service) η οποία έχει εκκινήσει.
4. Διεργασία Παρασκηνίου (background process). Μια διεργασία που περιέχει μια δραστηριότητα (Activity) η οποία δεν είναι ορατή στον χρήστη.
5. Άδεια Διεργασία (empty process). Μια διεργασία που δεν περιέχει κανένα ενεργό στοιχείο.

4.4.4 Ασφάλεια στο Android

Το μεγαλύτερο μέρος της διασφάλισης της ασφάλειας μεταξύ των εφαρμογών και του συστήματος το αναλαμβάνει ο πυρήνας του Linux. Ένας επιπλέον μηχανισμός ασφάλειας που παρέχεται από το Android είναι τα Permissions (Άδειες) τα οποία θέτουν περιορισμούς σε συγκεκριμένες ενέργειες που μπορεί να εκτελέσει μια διεργασία.

Η φιλοσοφία του μηχανισμού αυτού είναι ότι καμία εφαρμογή δεν έχει δικαίωμα να εκτελέσει οποιαδήποτε ενέργεια που επηρεάζει δυσμενώς κάποια άλλη εφαρμογή, το λειτουργικό σύστημα ή τον χρήστη. Μερικές από αυτού του είδους της ενέργειες είναι η ανάγνωση και εγγραφή των δεδομένων του χρήστη, η ανάγνωση και εγγραφή αρχείων άλλων εφαρμογών, πρόσβαση στο δίκτυο, διατήρηση της συσκευής ανοικτής κ.α. Προκειμένου μια εφαρμογή να εκτελέσει μια τέτοιου είδους ενέργεια πρέπει να έχει δηλώσει ότι απαιτεί το αντίστοιχο permission. Αυτή η δήλωση γίνεται στατικά μέσα στην εφαρμογή (συγκεκριμένα μέσα στο AndroidManifest.xml) έτσι ώστε είναι γνωστή κατά την εγκατάσταση της εφαρμογής και δεν αλλάζει. Τα permissions που δηλώνει ότι απαιτεί μια εφαρμογή

διαχειρίζονται από το λειτουργικό σύστημα με διάφορους τρόπους. Συνήθως είτε γίνονται αυτομάτως αποδεκτά ή μη αποδεκτά βάσει πιστοποιητικών, είτε ζητείται από τον χρήστη να αποφασίσει για την αποδοχή τους.

4.4.5 Φιλοσοφία σχεδίασης εφαρμογών Android

Η φιλοσοφία της σχεδίασης μιας εφαρμογής android είναι η διατήρηση τριών βασικών στοιχείων: της ταχύτητας, της αποκρισιμότητας και της συνοχής.

Μια εφαρμογή Android πρέπει να είναι γρήγορη. Ο όρος γρήγορη χρησιμοποιείται με την έννοια της αποδοτικότητας. Υπάρχει η τάση στην επιστήμη των υπολογιστών αυτές τις μέρες να θεωρείται ότι τελικά ο νόμος του Moore θα λύσει όλα τα προβλήματα. Όσο αναφορά όμως τα ενσωματωμένα συστήματα είναι πιο πολύπλοκος στην εφαρμογή του. Δεν εφαρμόζεται με τον ίδιο τρόπο στις κινητές συσκευές όπως στους υπολογιστές γραφείου ή στους διακομιστές.

Ο νόμος του Moore αφορά την πυκνότητα των τρανζίστορ που σημαίνει ότι μπορούν να πακεταριστούν περισσότερα κυκλώματα στο ίδιου μεγέθους chip με το πέρασμα του χρόνου. Για τους υπολογιστές γραφείου ή για τους διακομιστές αυτό σημαίνει ότι μπορεί να πακεταριστεί περισσότερη «ταχύτητα» σε ένα chip του ίδιου μεγέθους με αποτέλεσμα την αύξηση της επίδοσης. Στην περίπτωση των κινητών συσκευών ο νόμος του Moore χρησιμοποιείται για την κατασκευή μικρότερων chip. Με την χρήση μικρότερων chip μειώνεται η κατανάλωση ενέργειας με αποτέλεσμα την μείωση του μεγέθους της κινητής συσκευής και την αύξηση της διάρκειας της μπαταρίας. Άρα οι επιδόσεις των ενσωματωμένων συστημάτων, όπως οι κινητές συσκευές, αυξάνονται με πολύ πιο αργό ρυθμό από τους υπολογιστές γραφείου.

Για αυτόν τον λόγο είναι σημαντικό ο κώδικας μιας εφαρμογής android να είναι αποδοτικός. Αυτό σημαίνει την όσο το δυνατόν μικρότερη δέσμευση μνήμης και την αποφυγή ορισμένων προγραμματιστικών ιδιωμάτων τα οποία μπορούν να καταστρέψουν την απόδοση.

Μια εφαρμογή android πρέπει να έχει αποκρισιμότητα. Μπορεί μια εφαρμογή να είναι αποδοτική και παρόλα αυτά να μην είναι φιλική προς τον χρήστη. Υπάρχουν εφαρμογές οι οποίες δεν ανταποκρίνονται αρκετά, δηλαδή φαίνεται πως λειτουργούν αργά ή παγώνουν για σημαντικά χρονικά διαστήματα ή κάνουν πολύ χρόνο να επεξεργαστούν την είσοδο από τον χρήστη.

Μια εφαρμογή android η οποία δεν έχει επαρκή αποκρισιμότητα θα προκαλεί συχνά την εμφάνιση του μηνύματος “Application Not Responding” από το σύστημα. Αυτό γενικά συμβαίνει όταν μια εφαρμογή δεν μπορεί να ανταποκριθεί σε κάποια είσοδο χρήστη. Για παράδειγμα η εφαρμογή σταματάει σε κάποια I/O διεργασία (συχνά μια πρόσβαση στο δίκτυο). Τότε το κύριο νήμα της εφαρμογής δεν θα μπορεί να χειριστεί τα όποια συμβάντα προκαλεί ο χρήστης. Μετά από κάποιο χρονικό διάστημα το σύστημα θα θεωρήσει ότι η εφαρμογή έχει κολλήσει και δίνει την δυνατότητα του τερματισμού της.

Παρομοίως αν μια εφαρμογή χρειάζεται πολύ χρόνο για την εκτέλεση υπολογισμών τότε πάλι το σύστημα θα θεωρήσει ότι η εφαρμογή έχει κολλήσει. Για τον λόγο αυτόν τέτοιοι υπολογισμοί πρέπει να γίνονται όσο το δυνατόν αποδοτικότερα. Αλλά ακόμα και

ο πιο αποδοτικός υπολογισμός ορισμένες φορές χρειάζεται χρόνο για να εκτελεστεί. Σε τέτοιες περιπτώσεις συνήθως προτιμάται η δημιουργία ενός νήματος-παιδιού το οποίο εκτελεί την χρονοβόρα διεργασία-υπολογισμό. Με αυτόν τον τρόπο το κύριο νήμα το οποίο διαχειρίζεται τα συμβάντα χρήστη συνεχίζει να τρέχει και αποτρέπει το σύστημα από το να θεωρήσει οτι η εφαρμογή έχει κολλήσει.

Τέλος ακόμα και αν μια εφαρμογή είναι γρήγορη και έχει αποκρισιμότητα μπορεί να μην είναι φιλική προς τους χρήστες. Μια εφαρμογή πρέπει να συνεργάζεται ομαλά με το σύστημα και με τις υπόλοιπες εφαρμογές και να ολοκληρώνεται πλήρως στο σύστημα.

4.4.6 Πακέτα του Android

Android Location API

Το Android SDK περιλαμβάνει δύο πακέτα τα οποία παρέχουν την απαραίτητη υποστήριξη για την δημιουργία location-based υπηρεσιών: το πακέτο android.location και το com.google.android.maps.

android.location Αυτό το πακέτο περιέχει μια πληθώρα κλάσεων που σχετίζονται με τις υπηρεσίες θέσης της πλατφόρμας του Android. Η πιο σημαντική κλάση που παρέχει είναι η LocationManager η οποία παρέχει μια διαπροσωπεία για τον καθορισμό της θέσης εάν η υποκείμενη συσκευή υποστηρίζει αυτήν την λειτουργία. Ο LocationManager δεν πρέπει να αρχικοποιείται από τον χρήστη. Αντιθέτως πρέπει να λαμβάνεται μια αναφορά με την χρήση της μεθόδου getSystemService μιας και αποτελεί μια από τις υπηρεσίες του συστήματος. Αν μια εφαρμογή έχει μια αναφορά στον LocationManager μπορεί να κάνει τα παρακάτω τρία πράγματα:

1. Ερώτηση για όλους τους παρόχους θέσης (LocationProviders) οι οποίοι είναι γνωστοί στον LocationManager για την τελευταία γνωστή περιοχή.

2. Λήψη περιοδικών ενημερώσεων για την τρέχουσα θέση από έναν πάροχο θέσης (LocationProvider) (είτε με βάση κάποια κριτήρια, είτε με βάση το όνομα του)

3. Δυνατότητα για μια δοσμένη Πρόθεση (Intent) να ενεργοποιηθεί εάν η συσκευή εισέλθει σε μια ορισμένη περιοχή.

Παρακάτω παραθέτονται όλες οι classes και τα interface που περιέχονται στο πακέτο και μια σύντομη περιγραφή του καθενός:

LocationListener. Ένα interface το οποίο λαμβάνει ειδοποιήσεις από τον LocationManager όταν υπάρχει κάποια αλλαγή στην θέση της συσκευής.

Address. Μια class η οποία αναπαριστά μια διεύθυνση σε μια απλοποιημένη μορφή της xAL (eXtensive Address Language).

Criteria. Μια class η οποία αναπαριστά τα κριτήρια με βάση τα οποία επιλέγεται ένας πάροχος θέσης (Location Provider).

Geocoder. Μια class η οποία χειρίζεται την μετατροπή γεωγραφικών συντεταγμένων σε διευθύνσεις και το αντίστροφο.

Location. Μια class η οποία αναπαριστά μια τοποθεσία.

LocationManager. Η class που παρέχει μια διαπροσωπεία για τον καθορισμό θέσης.

LocationProvider. Μια abstract class για περιγραφή παρόχων θέσης (Location Provider).

com.google.android.maps Αυτό το πακέτο περιέχει ένα σύνολο από κλάσεις που σχετίζονται με την παρουσίαση, και τον έλεγχο ενός Google Map σε μια Activity. Η πιο σημαντική κλάση του πακέτου είναι η MapView η οποία εμφανίζει αυτόματα τον βασικό Google Map. Για την ορθή χρήση αυτού του πακέτου πρέπει να αποκτηθεί ένα κλειδί MapView API από την υπηρεσία Google Maps ώστε να γίνεται δυνατή η φόρτωση των χαρτών. Μετά την δημιουργία ενός αντικειμένου MapView μπορεί να γίνει η ανάκτηση ενός MapController για τον έλεγχο και την κίνηση του χάρτη και η προσθήκη πληροφοριών στον χάρτη.

Android WiFi API

Το WiFi API περιέχεται στο πακέτο android.net.wifi. Αποτελείται από ένα σύνολο κλάσεων με βασικότερη την WifiManager η οποία παρέχει μια διαπροσωπεία για τον χειρισμό όλων των πτυχών μιας WiFi σύνδεσης. Ο WifiManager όπως και ο LocationManager δεν πρέπει να αρχικοποιείται από τον χρήστη αλλά ως μια υπηρεσία συστήματος πρέπει να λαμβάνεται μέσω της μεθόδου getSystemService. Μέσω του WifiManager μπορεί να γίνει:

1. Διαχείριση της λίστας των δικτύων. Αυτή η λίστα μπορεί απλά να διαβαστεί, να ανανεωθεί καθώς επίσης ιδιότητες μεμονωμένων εγγραφών της μπορούν να αλλαχτούν.

2. Εύρεση του τρέχοντος ενεργού WiFi δικτύου, εάν υπάρχει. Μπορεί να γίνει σύνδεση ή αποσύνδεση από αυτό και να ανακτηθούν δυναμικές πληροφορίες για την κατάσταση του δικτύου

3. Διαχείριση των αποτελεσμάτων της αναζήτησης για σημεία πρόσβασης (access points), τα οποία περιέχουν αρκετές πληροφορίες ώστε να ληφθεί η απόφαση σε ποιο από τα σημεία πρόσβασης θα γίνει σύνδεση.

4. Εκπομπή συγκεκριμένων Προθέσεων (Intents) σε περίπτωση κάποιας αλλαγής στην κατάσταση του WiFi.

   Παρακάτω παραθέτονται όλες οι κλάσεις που περιέχονται στο πακέτο και μια σύντομη περιγραφή της κάθε μιας:

   Scan Result. Μια κλάση που περιέχει πληροφορίες για ένα σημείο πρόσβασης (Access Point) που έχει ανιχνευθεί.

WifiConfiguration. Μια κλάση που αναπαριστά ένα WiFi δίκτυο

WifiConfiguration.AuthAlgorithm. Μια κλάση που περιέχει σταθερές που αναπαριστούν αλγόριθμους πιστοποίησης.

WifiConfiguration.GroupCipher. Μια κλάση που περιέχει σταθερές που αναπαριστούν group ciphers.

WifiConfiguration.KeyMgmt. Μια κλάση που περιέχει σταθερές που αναπαριστούν σχέδια αναγνώρισης κλειδιού.

WifiConfiguration.PairwiseCipher. Μια κλάση που περιέχει σταθερές που αναπαριστούν pairwise ciphers.

WifiConfiguration.Protocol. Μια κλάση που περιέχει σταθερές που αναπαριστούν πρωτόκολλα ασφαλείας.

WifiConfiguration.Status. Μια κλάση που αναπαριστά την κατάσταση του δικτύου.

WifiInfo. Μια κλάση που περιέχει πληροφορίες για οποιαδήποτε σύνδεση WiFi η οποία είναι ενεργή ή στην διαδικασία της ενεργοποίησης της.

WifiManager. Η κλάση που παρέχει την διαπροσωπεία για την διαχείριση όλων των πτυχών του WiFi.

WifiManager.WifiLock. Μια κλάση για να διατηρείται η συσκευή WiFi ενεργή.

Άλλα πακέτα

Το Android API περιέχει αρκετά ακόμα πακέτα. Μερικά από αυτά είναι:

- Το Media API που περιέχεται στο android.media το οποίο παρέχει υποστήριξη για την αναπαραγωγή ήχου και βίντεο, καθώς επίσης και streaming ήχο ή βίντεο.
- Το OpenGl API που περιέχεται στο android.opengl και αποτελεί μια διαπροσωπεία για την χρήση της βιβλιοθήκης OpenGl.

4.5 Διαδικασία εγκατάστασης

4.5.1 Εγκατάσταση του Android

Ο χρήστης κάποιας κινητής συσκευής βασισμένης στο android δεν χρειάζεται να κάνει τίποτα για να το εγκαταστήσει στην συσκευή του αφού βρίσκεται προεγκατεστημένο σε αυτή. Ο προγραμματιστής εφαρμογών βασισμένων στο android από την άλλη χρειάζεται να κατεβάσει τις βιβλιοθήκες και τα εργαλεία του android από το επίσημο site (τρέχουσα έκδοση την στιγμή των γραφομένων είναι η Android

SDK 1.0). Με την βοήθεια αυτών των εργαλείων καθώς και με την χρήση του ολοκληρωμένου περιβάλλοντος eclipse όπως αναφέρθηκε και παραπάνω η διαδικασία συγγραφής μιας android εφαρμογής απλοποιείται κατά πολύ.

4.5.2 Εγκατάσταση εφαρμογών Android

Για να μπορέσει να διαμοιραστεί μια εφαρμογή γραμμένη σε android πακετάρεται σε ένα αρχείο *.apk (android package). Το πακέτο αυτό περιέχει τόσο τις κλάσεις της εφαρμογής μεταγλωττισμένες για να τρέχουν στην Dalvic Virtual Machine, όσο και τους πόρους της εφαρμογής (γραφικά, ήχο κλπ). Ο κάθε χρήστης που θέλει να χρησιμοποιήσει την εφαρμογή στην κινητή συσκευή του πρέπει να αντιγράψει το αρχείο-πακέτο της εφαρμογής στην συσκευή του έτσι ώστε να εγκατασταθεί σε αυτήν. Μετά την αντιγραφή ο χρήστης μπορεί άμεσα να χρησιμοποιήσει την εφαρμογή επιλέγοντας το εικονίδιο της από το menu των εφαρμογών.

Κεφάλαιο 5

Η εφαρμογή

5.1 Περιγραφή κινητής συσκευής

Σχήμα 5.1: Τυπική Συσκευή Android

Προκειμένου ο χρήστης να αλληλεπιδράσει με κάθε εφαρμογή android που έχει εγκατεστημένη στην συσκευή του πρέπει να έχει μια γενική γνώση των βασικών πλήκτρων με τα οποία είναι εφοδιασμένη μια κινητή συσκευή android. Μια τυπική συσκευή android φαίνεται στο σχήμα 5.1 και μια τυπική περιγραφή δίνεται παρακάτω.

Το μεγαλύτερο μέρος της συσκευής το καταλαμβάνει η οθόνη της συσκευής. Η οθόνη λειτουργεί με δύο τρόπους. Όπως κάθε οθόνη χρησιμεύει ως μια συσκευή εξόδου για να παρουσιάζεται ένα γραφικό περιβάλλον στον χρήστη. Ταυτόχρονα όμως αποτελεί και μια συσκευή εισόδου αφού είναι ευαίσθητη στην αφή (touchscreen) και μπορεί να ανιχνεύει συμβάντα όπως clicks, touches, gestures τα οποία παράγει ο χρήστης με απλή επαφή του με την οθόνη.

Στο κέντρο της συσκευής κάτω από την οθόνη βρίσκεται το D-pad το οποίο αποτελεί στην ουσία έναν σταυρό τεσσάρων κατευθύνσεων με ένα επιπλέον πλήκτρο στο κέντρο του. Με τα πλήκτρα κατευθύνσεων ο χρήστης είναι σε θέση να μετακινείται ανάμεσα στα διάφορα menu της εκάστοτε εφαρμογής. Με το κεντρικό πλήκτρο του D-Pad μπορεί να ενεργήσει έτσι ώστε να ενεργοποιήσει ένα επιλεγμένο αντικείμενο (πχ να πατήσει το επιλεγμένο πλήκτρο). Πρέπει να σημειωθεί οτι αυτή είναι η συνήθης λειτουργία αυτών των πλήκτρων και από εφαρμογή σε εφαρμογή μπορεί να διαφοροποιείται. Επιπλέον η κινητή συσκευή περιλαμβάνει ένα πλήκτρο με την ετικέτα MENU το οποίο όταν πατηθεί εμφανίζει στην οθόνη ένα αναδυόμενο menu αν αυτό είναι διαθέσιμο από την εκάστοτε εφαρμογή. Εκτός από τα πλήκτρα που φαίνονται στο σχήμα κάθε συσκευή android είναι εφοδιασμένη με ένα πληκτρολόγιο QERTY για να μπορεί ο χρήστης να εισάγει χαρακτήρες.

5.2 Γενική περιγραφή της εφαρμογής

Σκοπό αυτής της διπλωματικής εργασίας αποτελεί η συγγραφή μιας location based εφαρμογής για την πλατφόρμα Google Android. Η ανάπτυξη αυτής της εφαρμογής αποτελείται από τρία στάδια:

1. Την ανάπτυξη της εφαρμογής η οποία τρέχει στην κινητή συσκευή και χρησιμοποιείται από τον απλό χρήστη.

2. Την ανάπτυξη μιας web service εφαρμογής η οποία μπορεί να παρέχει τα δεδομένα που της ζητούνται από εφαρμογές οι οποίες τρέχουν σε κινητές συσκευές καθώς και να αποθηκεύει τα όποια δεδομένα έχουν σταλθεί. Η εφαρμογή αυτή χρησιμοποιείται από έναν κεντρικό πάροχο της υπηρεσίας ο οποίος πρέπει να εξασφαλίζει την ομαλή και απρόσκοπτη λειτουργία της.

3. Την δημιουργία μιας βάσης δεδομένων στην οποία βρίσκονται αποθηκευμένα τα δεδομένα τα οποία αποστέλλει η απομακρυσμένη εφαρμογή. Η βάση δεδομένων μπορεί να βρίσκεται στο ίδιο υπολογιστή με την απομακρυσμένη εφαρμογή οπότε να υπάρχει κοινή διαχείριση τους από τον ίδιο χρήστη. Μπορεί ακόμη να βρίσκεται και σε διαφορετικό υπολογιστή ώστε η διαχείριση της να γίνεται από διαφορετικό χρήστη από τον χρήστη της απομακρυσμένης εφαρμογής.

5.2.1 Android based application

Η συγκεκριμένη εφαρμογή εκμεταλλεύεται την δυνατότητα που παρέχεται από το Android API για λήψη πληροφοριών σχετικών με την θέση της υποκείμενης συσκευής. Μπορεί να αναπαραστήσει γραφικά αυτές τις πληροφορίες και παρέχει στον χρήστη την δυνατότητα της επεξεργασίας και αποθήκευσης τους. Οι πληροφορίες αυτές παρέχονται με την βοήθεια ενός GPS δέκτη ο οποίος βρίσκεται ενσωματωμένος στην συσκευή.

Συγκεκριμένα η εφαρμογή μπορεί να παρακολουθεί την τρέχουσα θέση της κινητής συσκευής. Με βάση το γεωγραφικό πλάτος και το γεωγραφικό μήκος της θέσης αυτής την αναπαριστά ως ένα σημείο πάνω σε έναν παγκόσμιο χάρτη. Ταυτόχρονα αναζητά από μια τοπική ή απομακρυσμένη βάση δεδομένων ορισμένα σημεία ενδιαφέροντος (Points of Interest). Το κριτήριο επιλογής τους είναι το αν αυτά βρίσκονται εντός μια ακτίνας αναζήτησης, προκαθορισμένη από τον χρήστη. Τα σημεία που πληρούν το παραπάνω κριτήριο εμφανίζονται πάνω στον παγκόσμιο χάρτη. Εκτός από την θέση τους στην υδρόγειο τα σημεία ενδιαφέροντος έχουν προσαρτημένες μερικές ακόμη χρήσιμες πληροφορίες στις οποίες μπορεί να έχει άμεση πρόσβαση ο χρήστης από τον χάρτη.

Τα σημεία αυτά όπως αναφέρθηκε παραπάνω μπορεί να είναι αποθηκευμένα είτε τοπικά στην κινητή συσκευή του χρήστη σε μια τοπική βάση δεδομένων, είτε σε κάποια απομακρυσμένη βάση δεδομένων σε έναν κεντρικό server. Ο χρήστης έχει την δυνατότητα να προσθέσει, να διαγράψει και να επεξεργαστεί τα σημεία ενδιαφέροντος που ορίζει ο ίδιος. Έχει επίσης την δυνατότητα να συσχετιστεί με άλλους χρήστες της ίδιας απομακρυσμένης υπηρεσίας. Έτσι μπορεί να έχει πρόσβαση σε σημεία ενδιαφέροντος που έχουν ορίσει άλλοι χρήστες αλλά και οι συσχετισμένοι με αυτόν χρήστες να μπορούν να έχουν πρόσβαση στα δικά του σημεία ενδιαφέροντος.

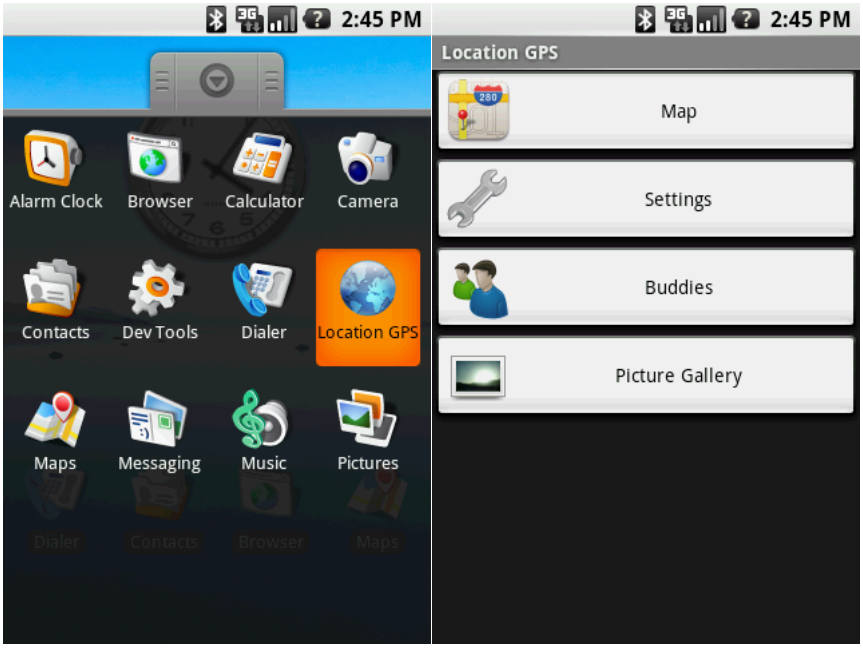
Τέλος η εφαρμογή μπορεί να ελαχιστοποιηθεί έτσι ώστε ο χρήστης να μπορεί να εκτελέσει κάποια άλλη λειτουργία στην κινητή του συσκευή. Παρότι όμως η εφαρμογή είναι ελαχιστοποιημένη δεν σταματά να δέχεται πληροφορίες θέσης. Αντιθέτως συνεχίζει να εμφανίζει μηνύματα στο χρήστη, με την μορφή ειδοποιήσεων (Notifications), όταν βρίσκεται κοντά σε κάποιο σημείο ενδιαφέροντος προσφέροντας του την δυνατότητα να δει περισσότερες πληροφορίες για αυτό. Περισσότερες λεπτομέρειες για την λειτουργία της εφαρμογής δίνονται παρακάτω.

5.2.2 Αναλυτική περιγραφή της android εφαρμογής

Αρχικά για να τρέξει ο χρήστη την εφαρμογή πρέπει να επιλέξει το εικονίδιο της εφαρμογής που γράφει “Location GPS” από το menu εφαρμογών και να κάνει click πάνω του όπως φαίνεται και στο σχήμα 5.2α’.

Στην συνέχεια θα εκκινήσει η εφαρμογή Location GPS και θα βρεθεί σε ένα εισαγωγικό μενού, το οποίο φαίνεται στο σχήμα 5.2β’, από όπου μπορεί να πλοηγηθεί σε όλες τις λειτουργίες της εφαρμογής. Όπως φαίνεται και από το σχήμα ο χρήστης μπορεί να επιλέξει από ένα πλήθος λειτουργιών:

1. Map
2. Settings



(α') Έναρξη Εφαρμογής (β') Κύριο Menu

Σχήμα 5.2: Αναλυτική περιγραφή της λειτουργίας της εφαρμογής android

3. Buddies

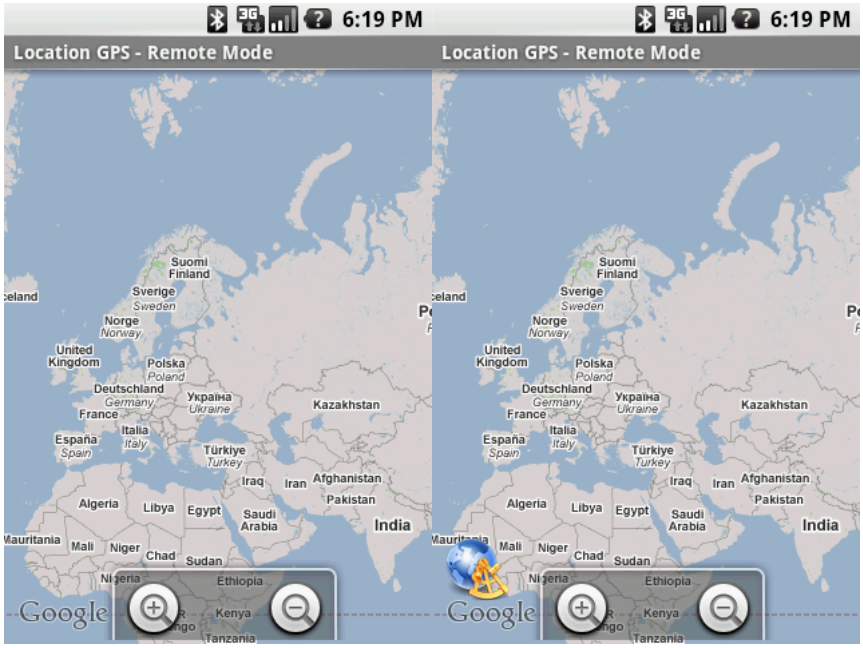
4. Picture Gallery

Map

Η επιλογή Map (Χάρτης) μεταφέρει τον χρήστη σε μια νέα οθόνη η οποία αποτελεί και την κύρια οθόνη της εφαρμογής (φαίνεται στο σχήμα 5.3β' και 5.3α'). Με τα δύο πλήκτρα του σχήματος 5.3 μπορεί να κάνει zoom in/out στον χάρτη. Επίσης μπορεί να σύρει τον χάρτη απλά ακουμπώντας το δάκτυλο του στην οθόνη και σύροντας το πάνω σε αυτή. Στην οθόνη αυτή ο χρήστης μπορεί να επιλέξει ανάμεσα σε δύο menus, PoIs και Modes, το καθένα από τα οποία έχει με την σειρά του ένα σύνολο από submenus. Η λειτουργίες του κάθε συνόλου submenus περιγράφονται παρακάτω.

Το menu Modes

Για να ενεργοποιηθεί η λειτουργία λήψεως ενημερώσεων θέσης ο χρήστης πρέπει να επιλέξει από το menu Modes το submenu Toggle Position Listening. Αντιστοίχως αν ή λειτουργία λήψεως ενημερώσεων θέσης είναι ενεργοποιημένη μπορεί να την απενεργοποιήσει από το ίδιο submenu. Όταν η λειτουργία είναι ενεργοποιημένη εμφανίζεται στην κάτω αριστερά περιοχή της οθόνης ένα εικονίδιο το οποίο σηματοδοτεί την λήψη ενημερώσεων θέσης όπως φαίνεται και στο σχήμα 5.3β'.



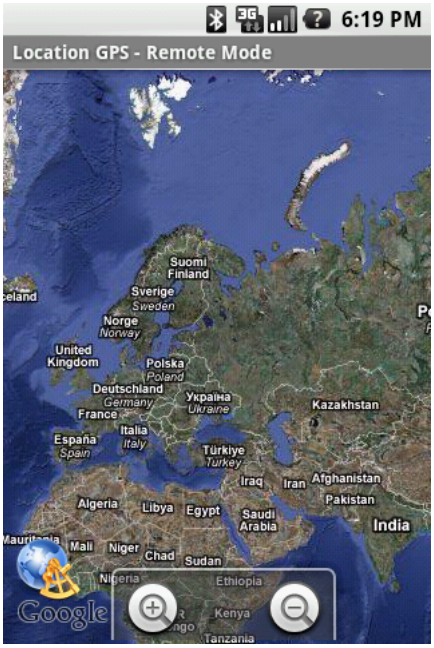
(α') Map Listening Off (β') Map Listening On

Σχήμα 5.3: Η οθόνη Map

Επιπλέον ο χρήστης μπορεί να επιλέξει το χρονικό και χωρικό διάστημα μεταξύ της λήψης δύο διαδοχικών ενημερώσεων θέσης. Η προκαθορισμένη τιμή είναι 0 και για τις δύο αυτές παραμέτρους που σημαίνει οτι δεν μεσολαβεί καθόλου χρονικό ή χωρικό διάστημα μεταξύ των ενημερώσεων και μόλις γίνει λήψη μιας καινούργιας θέσης αυτή παρουσιάζεται στον χρήστη. Η τιμή του χρονικού διαστήματος μεταξύ της λήψης δύο διαδοχικών ενημερώσεων θέσης δεν είναι δεσμευτική για την συσκευή, αλλά είναι η μέγιστη επιτρεπτή προκειμένου να γίνει εξοικονόμηση ενέργειας. Η αλλαγή των παραμέτρων μπορεί να γίνει από τον χρήστη επιλέγοντας το από το menu Modes το υπομενού Listening Parameters.

Ο χρήστης μπορεί να εναλλάσσει τον χάρτη μεταξύ δύο προβολών. Η μια προβολή είναι αυτή που φαίνεται και στα προηγούμενα σχήματα. Η δεύτερη προβολή όπως φαίνεται και στο σχήμα 5.4 είναι η δορυφορική. Προκειμένου να γίνει εναλλαγή μεταξύ των δύο αυτών προβολών ο χρήστης πρέπει να επιλέξει από το menu Modes το submenu Toggle View Mode.

Ο χρήστης μπορεί να εναλλάσσει μεταξύ δύο λειτουργιών ανάκτησης σημείων ενδιαφέροντος, της τοπικής και της απομακρυσμένης. Η τοπική λειτουργία του επιτρέπει να έχει πρόσβαση στα σημεία ενδιαφέροντος που έχει αποθηκεύσει στην κινητή συσκευή του (Local Mode), ενώ η απομακρυσμένη λειτουργία του επιτρέπει να έχει πρόσβαση στα σημεία ενδιαφέροντος που είναι αποθηκευμένα στον απομακρυσμένο server (Remote Mode). Η τρέχουσα λειτουργία φαίνεται στον τίτλο της εφαρμογής. Προκειμένου να εναλλάσσεται η εφαρμογή μεταξύ αυτών των δύο λειτουργιών ο χρήστης πρέπει να επιλέξει από το menu Modes το submenu Toggle Point Mode. Στην περίπτωση που ο χρήστης θέλει να μεταβεί από την απομακρυσμένη λειτουργία στην τοπική δεν υπάρχει κανένα πρόβλημα

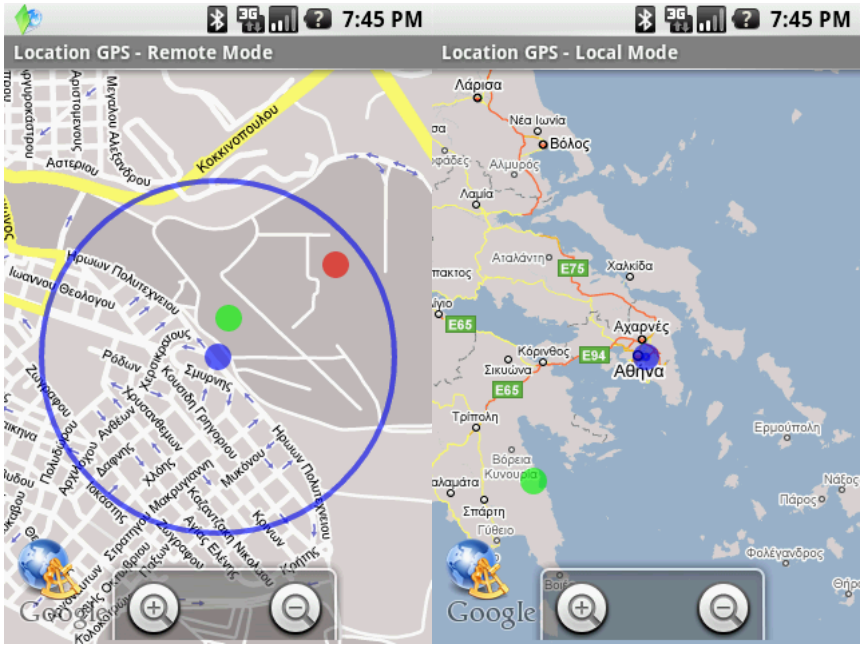


Σχήμα 5.4: Δορυφορική Προβολή

γιατί πάντα είναι εφικτή μια τέτοια μετάβαση. Στην περίπτωση όμως που ο χρήστης θέλει να μεταβεί από την τοπική λειτουργία στην απομακρυσμένη λειτουργία δεν είναι πάντα δυνατή αυτή η μετάβαση. Για να μπορέσει να γίνει η εφαρμογή πρέπει να είναι ρυθμισμένη σωστά όπως περιγράφεται στο κεφάλαιο 5.2.2 και ο απομακρυσμένος server να ανταποκρίνεται μέσα στο προκαθορισμένο timeout. Αν για οποιοδήποτε λόγο δεν είναι εφικτή μια τέτοια μετάβαση ο χρήστης ενημερώνεται με κατάλληλο μήνυμα.

Ο χρήστης μπορεί να αλλάξει την ακτίνα αναζήτησης σημείων ενδιαφέροντος επιλέγοντας από το menu Modes το submenu Change Radius. Η ακτίνα αυτή είναι η απόσταση σε μέτρα μέσα στην οποία ψάχνει η εφαρμογή για σημεία ενδιαφέροντος. Αν ένα σημείο ενδιαφέροντος απέχει λιγότερο από την απόσταση αυτή παρουσιάζεται στον χάρτη. Επίσης μπορεί να επιλέξει αν στην αναζήτηση των σημείων ενδιαφέροντος θα συμπεριλαμβάνονται τα σημεία ενδιαφέροντος που οι φίλοι του έχουν ορίσει ως δημόσια μέσω της επιλογής από το menu Modes του submenu Include Buddies.

Αν η εφαρμογή βρίσκεται σε λειτουργία λήψης ενημερώσεων θέσης σε κάθε αλλαγή θέσης που λαμβάνει από την υποκείμενη GPS συσκευή παρουσιάζει σε έναν παγκόσμιο χάρτη την τρέχουσα θέση με ένα μπλε κυκλικό σημείο. Η τρέχουσα θέση βρίσκεται πάντα στο κέντρο του χάρτη. Επίσης με έναν μπλε κύκλο με κέντρο την τρέχουσα θέση φαίνεται η ακτίνα αναζήτησης σημείων ενδιαφέροντος. Ο χάρτης έχει υποστεί τόσο zoom έτσι ώστε να φαίνεται ολόκληρος ο κύκλος της ακτίνας αναζήτησης αν υπάρχει κάποιο σημείο ενδιαφέροντος εντός του κύκλου αυτού. Στην αντίθετη περίπτωση το zoom ρυθμίζεται έτσι ώστε να φαίνεται το κοντινότερο σημείο ενδιαφέροντος. Όταν λαμβάνεται μια νέα τρέχουσα θέση το πρόγραμμα ανακτά όλα εκείνα τα σημεία ενδιαφέροντος τα οποία



(α') Σημεία μέσα στην ακτίνα (β') Σημεία εκτός της ακτίνας

Σχήμα 5.5: Εμφάνιση Σημείων Ενδιαφέροντος στον χάρτη

βρίσκονται μέσα στην ακτίνα αναζήτησης. Αυτά τα σημεία παρουσιάζονται με κόκκινα σημεία πάνω στον παγκόσμιο χάρτη. Το κοντινότερο από αυτά τα σημεία ενδιαφέροντος όμως δεν παρουσιάζεται με κόκκινο χρώμα αλλά με πράσινο. Σε περίπτωση που δεν υπάρχει κανένα σημείο μέσα στην ακτίνα αναζήτησης τότε η εφαρμογή αναζητά το κοντινότερο σημείο ενδιαφέροντος εκτός της ακτίνας και το εμφανίζει στον παγκόσμιο χάρτη. Ορισμένα παραδείγματα σημείων εντός της ακτίνας αναζήτησης και εκτός της ακτίνας αναζήτησης φαίνονται στα σχήματα 5.5α' και 5.5β' αντίστοιχα.

Στην περίπτωση που το κοντινότερο σημείο είναι συνδεδεμένο και με κάποιο url θα εμφανιστεί στην taskbar του κινητού ένα notification από το οποίο ο χρήστης μπορεί να έχει άμεση πρόσβαση στην ιστοσελίδα αυτή. Επιπλέον παρέχεται η δυνατότητα στον χρήστης να επιλέξει ένα σημείο ενδιαφέροντος (κάνοντας click πάνω στον χάρτη). Τότε θα εμφανιστεί ένα αναδυόμενο παράθυρο με περισσότερες πληροφορίες για αυτό το σημείο όπως φαίνεται και στο σχήμα 5.7. Όπως φαίνεται στο παράθυρο εμφανίζεται μια μικρή λεκτική περιγραφή του σημείου και η απόσταση του σε μέτρα από την τρέχουσα θέση.

Επιλέγοντας από το menu Modes το submenu Recalculate Position ξαναγίνεται η αναζήτηση σημείων ενδιαφέροντος με βάση τις τρέχουσες ρυθμίσεις.

Αν η εφαρμογή βρίσκεται σε απομακρυσμένη λειτουργία ο χρήστης μπορεί να ενεργοποιήσει επίσης την ελαχιστοποιημένη λειτουργία επιλέγοντας από το menu Modes το submenu Minimized Mode. Σε αυτή την λειτουργία, όπως φαίνεται και στο σχήμα 5.6, δεν φαίνεται ο χάρτης της εφαρμογής αλλά η κινητή συσκευή του χρήστη συνεχίζει να λαμβάνει πληροφορίες θέσης και να εμφανίζει notification για το κοντινότερο σημείο ενδιαφέροντος αν αυτό είναι συνδεδεμένο με κάποιο url. Ανάλογή είναι και η λειτουργία ξε-

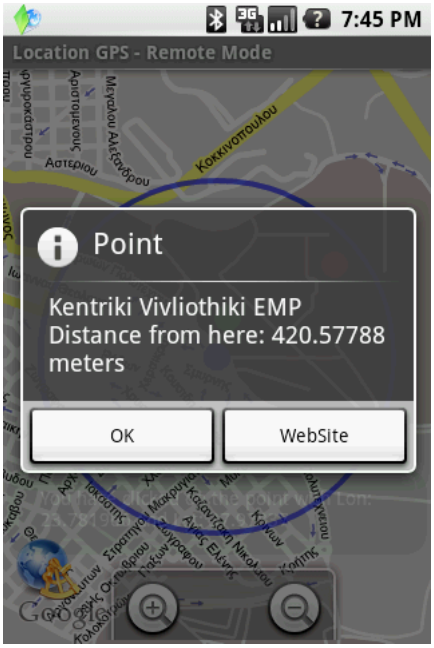
Σχήμα 5.6: Minimized Mode - Trip Mode

νάγησης η οποία ενεργοποιείται επιλέγοντας από το menu Modes το submenu Trip Mode. Η λειτουργία ξενάγησης κάνει ότι ακριβώς κάνει και η ελαχιστοποιημένη λειτουργία μόνο που ανοίγει ταυτόχρονα και το url που συνδέεται με το κοντινότερο σημείο ενδιαφέροντος στον φυλλομετρητή. Και στις δύο περιπτώσεις ο χρήστης μπορεί μεταφερθεί γρήγορα στην οθόνη του χάρτη επιλέγοντας το notification της εφαρμογής το οποίο μάλιστα δεν επηρεάζεται από τις εκκαθαρίσεις των ειδοποιήσεων.

Όλες οι αλλαγές που γίνονται από το menu Modes αποθηκεύονται κατά την έξοδο από την οθόνη του χάρτη και επαναφέρονται αυτόματα στην επόμενη είσοδο (πχ ο χρήστης έχει απενεργοποιήσει την λήψη ενημερώσεων θέσης ή έχει αλλάξει την τιμή της ακτίνα αναζήτησης). Στην περίπτωση που ο χρήστης εγκαταλείψει την οθόνη του χάρτη ενώ βρίσκεται σε απομακρυσμένη λειτουργία στην επόμενη είσοδο του στην οθόνη του χάρτη η εφαρμογή θα προσπαθήσει να συνδεθεί με τον απομακρυσμένο server. Αν τα καταφέρει θα μεταβεί σε απομακρυσμένη λειτουργία ακριβώς όπως ήταν πριν ο χρήστης εγκαταλείψει την οθόνη του χάρτη. Στην αντίθετη περίπτωση θα παραμείνει σε τοπική λειτουργία.

Το menu PoIs

Ο χρήστης κάνοντας κλικ πάνω στον χάρτη μπορεί να δει τις συντεταγμένες του σημείου αυτού στην οθόνη της κινητής συσκευής. Αυτό το σημείο μπορεί να αποθηκευτεί ως σημείο ενδιαφέροντος τοπικά ή απομακρυσμένα (ανάλογα με την λειτουργία στην οποία βρίσκεται η εφαρμογή). Για να αποθηκευτεί αρκεί ο χρήστης να επιλέξει από το μενού PoIs το υπομενού Last Click Location. Στην συνέχεια θα μεταφερθεί σε μια νέα οθόνη-φόρμα η οποία φαίνεται στο σχήμα 5.8α’ και οπού μπορεί να συμπληρώσει επιπλέον στοιχεία για

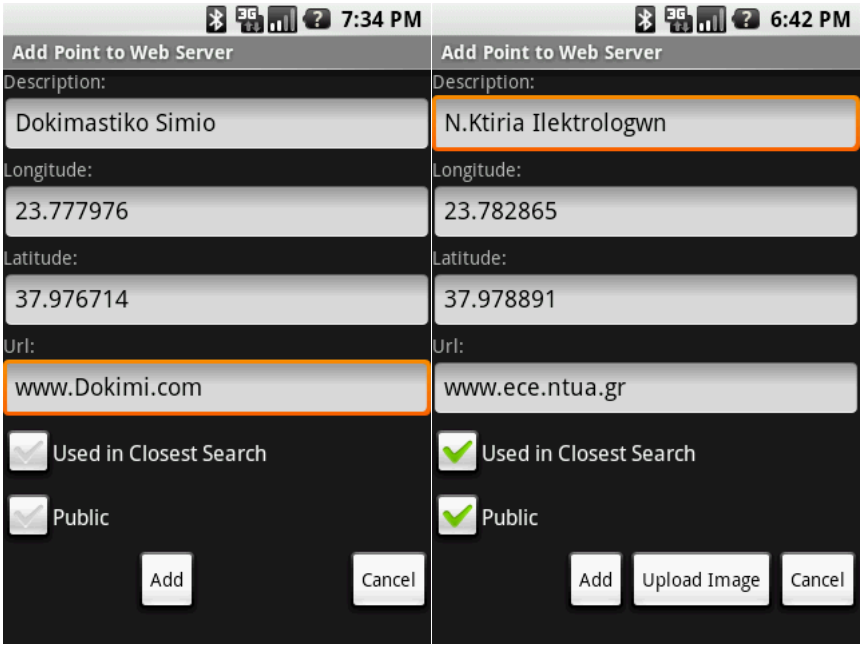


Σχήμα 5.7: Dialog Πληροφοριών Σημείου Ενδιαφέροντος

το σημείο. Τελικά μπορεί να επιβεβαιώσει την αποθήκευση του πατώντας στο πλήκτρο Add. Συγκεκριμένα μπορεί να προσθέσει μια σύντομη περιγραφή (Description), ένα url και να επιλέξει αν θα χρησιμοποιείται το σημείο στις αναζητήσεις κοντινότερου σημείου ενδιαφέροντος τόσο από τον ίδιο όσο και από τους φίλους του.

Ανάλογή δυνατότητα αποθήκευσης του προσφέρεται και για την πιο πρόσφατη ενημέρωση θέσης (τρέχουσα θέση της κινητής συσκευής). Για να αποθηκευτεί αρκεί ο χρήστης να επιλέξει από το menu PoIs το submenu Current Location. Στην συνέχεια θα μεταφερθεί σε μια νέα οθόνη-φόρμα ανάλογη της οθόνης που περιγράφηκε στην προηγούμενη παράγραφο.

Επιλέγοντας από το menu PoIs το submenu PoI List ο χρήστης μεταβαίνει σε μια νέα οθόνη η οποία φαίνεται στο σχήμα 5.9α'. Η νέα οθόνη ουσιαστικά αποτελεί μια λίστα όλων των σημείων ενδιαφέροντος τα οποία έχει ορίσει ο χρήστης. Ανάλογα με την λειτουργία στην οποία βρισκόταν η εφαρμογή πριν την επιλογή του PoI List η λίστα περιέχει είτε τα τοπικά αποθηκευμένα σημεία, είτε τα απομακρυσμένα αποθηκευμένα σημεία. Κάθε εγγραφή της λίστας περιέχει το Id του σημείου (Ο αριθμός στο μπλε φόντο), μια λεκτική περιγραφή του σημείου, τις συντεταγμένες του σημείου (Longitude, Latitude), μια ένδειξη για την ύπαρξη η μη ιστοσελίδας συνδεδεμένης με το σημείο και δύο ενδείξεις για το αν το σημείο θα χρησιμοποιείται στην αναζήτηση του κοντινότερου σημείου τόσο από τον χρήστη όσο και από τους φίλους του. Εκτελώντας κλικ πάνω στην ένδειξη WebPage μεταφερόμαστε αυτόματα στην σελίδα που είναι συνδεδεμένη με το αντίστοιχό σημείο. Αν δεν υπάρχει η ένδειξη WebPage αλλά η ένδειξη No WebPage σημαίνει ότι το σημείο δεν είναι συνδεδεμένο με κάποια ιστοσελίδα.



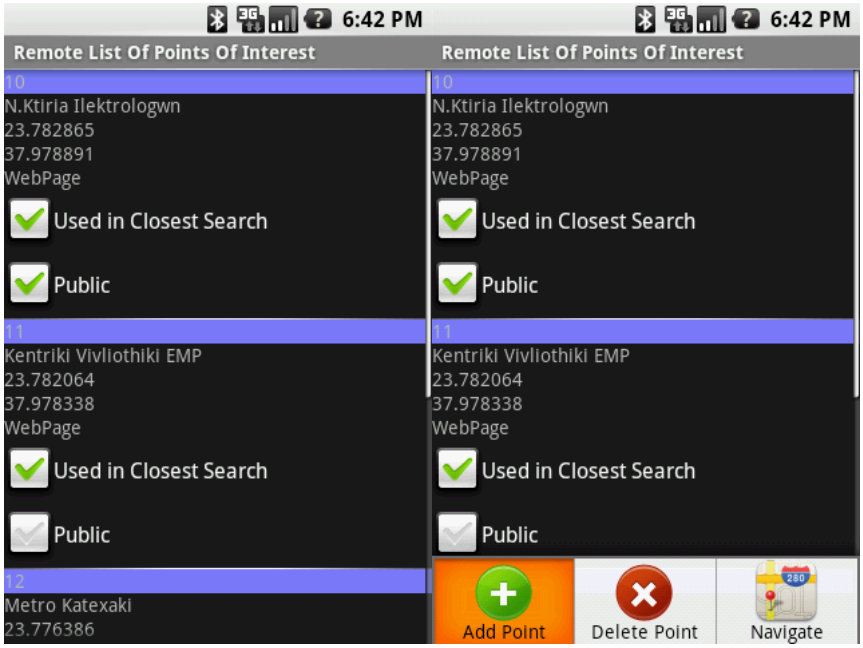
(α') Προσθήκη Σημείου Ενδιαφέροντος (β') Επεξεργασία Σημείου

Σχήμα 5.8: Προσθήκη-Επεξεργασία Σημείου Ενδιαφέροντος

Από το menu της εφαρμογής το οποίο φαίνεται στο σχήμα 5.9β' ο χρήστης μπορεί να επιλέξει να προσθέσει ένα στοιχείο. Τότε θα μεταφερθεί σε μια οθόνη-φόρμα όπως του σχήματος 5.8α' της οποίας η λειτουργία περιγράφτηκε σε προηγούμενη παράγραφο. Επίσης μπορεί να επιλέξει να διαγράψει το επιλεγμένο σημείο ενδιαφέροντος. Για να το επιτύχει πρέπει με τα πλήκτρα κατεύθυνσης του D-pad να επιλέξει το σημείο ενδιαφέροντος προς διαγραφή και στην συνέχεια να επιλέξει το menu Delete Point. Τελευταία δυνατότητα στην διαχείριση των σημείων ενδιαφέροντος αποτελεί η επεξεργασία ενός υπάρχοντος σημείου ενδιαφέροντος. Για να το επιτύχει αυτό ο χρήστης πρέπει να επιλέξει το σημείο ενδιαφέροντος προς επεξεργασία ακριβώς όπως και στην διαδικασία για διαγραφή και να πατήσει το μεσαίο πλήκτρο του D-Pad. Τότε θα μεταφερθεί σε μια νέα οθόνη-φόρμα όπως στο σχήμα 5.8β'. Η οθόνη αυτή είναι πανομοιότυπη με την οθόνη του σχήματος 5.8α' με την προσθήκη ενός πλήκτρου, του Upload Image. Με το πλήκτρο αυτό θα μεταφερθεί σε μια νέα οθόνη την Picture Gallery η οποία περιγράφεται στο κεφάλαιο 5.2.2. Από αυτήν μπορεί να επιλέξει μια εικόνα η οποία θα γίνει upload στον απομακρυσμένο server. Αφού επιλεγεί η εικόνα από τον χρήστη και γίνει με επιτυχία το upload στον server το url της τοποθετείται στο textbox με επιγραφή Url. Μόνο μια εικόνα ανά σημείο ενδιαφέροντος είναι δυνατόν να αποθηκευτεί στον server και όχι παραπάνω.

Τέλος ο χρήστης μπορεί να εντοπίσει ένα επιλεγμένο σημείο ενδιαφέροντος στον παγκόσμιο χάρτη επιλέγοντας το menu Navigate. Τότε ο χρήστης θα επιστρέψει στον χάρτη όπου στο κέντρο του θα έχει σημειωθεί το σημείο που επέλεξε με ένα μπλε κυκλικό σημείο, όπως φαίνεται και στο σχήμα 5.10.

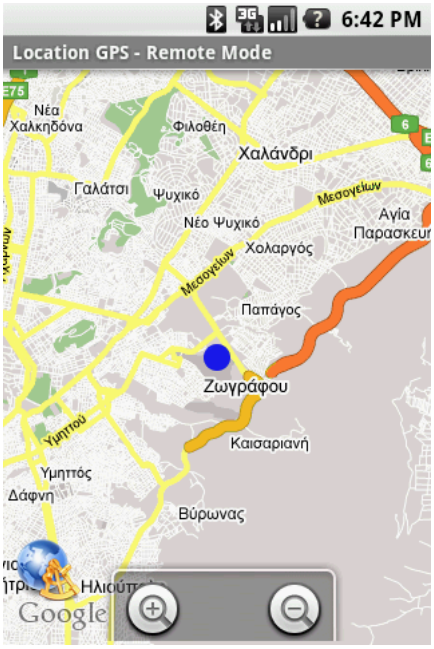
Σε οποιαδήποτε στιγμή και αν χαθεί η επικοινωνία με τον server ενώ η εφαρμογή βρί-



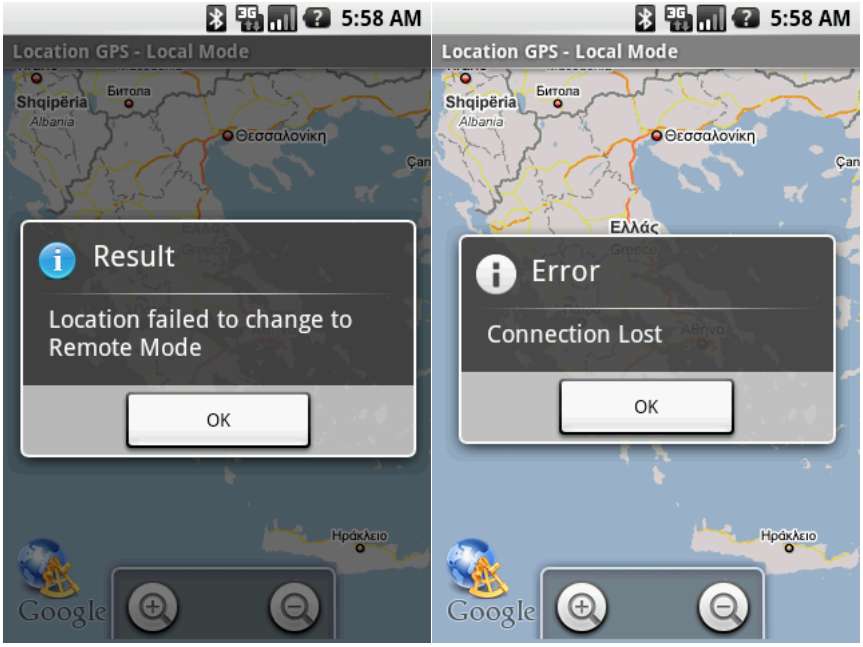
(α') Η λίστα των σημείων (β') Menu της λίστας των σημείων

Σχήμα 5.9: Λίστα Σημείων Ενδιαφέροντος

σκεται σε απομακρυσμένη λειτουργία ο χρήστης θα επιστρέψει στην οθόνη του χάρτη σε τοπική λειτουργία (με εξαιρέσεις την ελαχιστοποιημένη λειτουργία και την λειτουργία ξενάγησης οι οποίες απλά θα σταματήσουν να αναζητούν σημεία ενδιαφέροντος). Πρέπει να σημειωθεί οτι με τον όρο οποιαδήποτε στιγμή εννοούμε σε κάποια χρονική στιγμή που η εφαρμογή προσπαθήσει να έρθει σε επικοινωνία με τον server και αυτός δεν απαντήσει, χωρίς αυτό να σημαίνει οτι η εφαρμογή ανά τακτά χρονικά διαστήματα ερωτά τον server για να βεβαιώνεται οτι υπάρχει ακόμη η σύνδεση. Μερικές περιπτώσεις αποτυχίας φαίνονται στο σχήμα 5.11. Συγκεκριμένα στο σχήμα 5.11α' φαίνεται μια αποτυχημένη προσπάθεια για μετάβαση σε απομακρυσμένη λειτουργία, ενώ στο σχήμα 5.11β' φαίνεται η απώλεια σύνδεσης κατά την προσπάθεια ανάκτησης της απομακρυσμένης λίστας σημείων ενδιαφέροντος. Στο σχήμα 5.12 φαίνεται μια επιτυχημένη μετάβαση σε απομακρυσμένη λειτουργία.



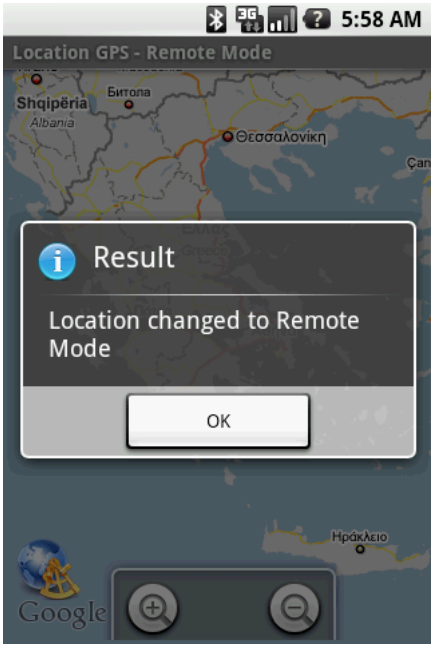
Σχήμα 5.10: Πλοήγηση σε Σημείο Ενδιαφέροντος



(α') Αποτυχία μετάβασης σε απομακρυσμένη λειτουργία

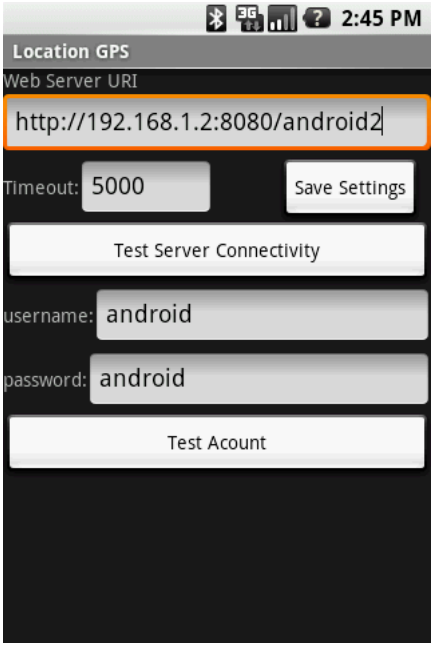
(β') Απώλεια σύνδεσης

Σχήμα 5.11: Περιπτώσεις αποτυχίας σύνδεσης



Σχήμα 5.12: Επιτυχημένη μετάβαση σε απομακρυσμένη λειτουργία

Settings



Σχήμα 5.13: Settings

Η επιλογή Settings (Ρυθμίσεις) μεταφέρει τον χρήστη σε μια νέα οθόνη η οποία φαίνεται στο σχήμα 5.13. Σε αυτήν την οθόνη ο χρήστης μπορεί να ελέγξει ένα πλήθος παραμέτρων του προγράμματος:

Μπορεί να ορίσει το Web Server URI. Το Web Server URI αποτελεί την διεύθυνση του web server με τον οποίο θα επικοινωνεί η εφαρμογή προκειμένου να κάνει αιτήσεις για λήψη δεδομένων σχετικών με τα σημεία ενδιαφέροντος που είναι αποθηκευμένα στην απομακρυσμένη βάση και αιτήσεις για αποστολή δεδομένων προκειμένου να αποθηκευτούν νέα σημεία ενδιαφέροντος ή να τροποποιηθούν τα ήδη υπάρχοντα.

Μπορεί να ορίσει το Timeout το οποίο είναι ο χρόνος σε milliseconds τον οποίο θα περιμένει στην χειρότερη περίπτωση η εφαρμογή μια απόκριση από τον web server. Το Timeout δεν μπορεί να ξεπερνάει τα 20000 milliseconds (20 secs).

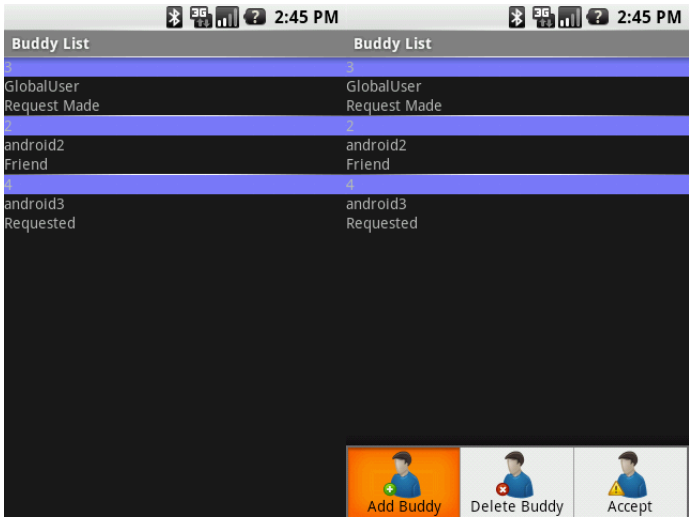
Μπορεί να ορίσει το username και password τα οποία είναι το όνομα χρήστη και ο κωδικός χρήστη τα οποία χρησιμεύουν στην ταυτοποίηση του χρήστη από τον web server. Το username και το password παρέχονται από τον κάτοχο του web server και της βάσης δεδομένων.

Με το πλήκτρο Save Settings μπορεί να αποθηκεύσει τις τρέχουσες ρυθμίσεις όπως φαίνονται στην οθόνη. Αν κάποια είσοδος είναι εσφαλμένη (πχ το Timeout δεν είναι αριθμός ή είναι μεγαλύτερο του 20000) τότε οι ρυθμίσεις δεν θα αποθηκευτούν και ο χρήστης θα ενημερωθεί μέσω μηνύματος στην οθόνη της συσκευής του.

Με το πλήκτρο Test Server Connectivity μπορεί να διαπιστώσει αν η τρέχουσα διεύθυνση του web server όπως φαίνεται στην οθόνη αντιστοιχεί σε κάποιο web service το οποίο είναι σχεδιασμένο έτσι ώστε να μπορεί να εξυπηρετεί της αιτήσεις της εφαρμογής για αποστολή και λήψη δεδομένων. Όταν πατηθεί στέλνεται ένα μήνυμα στον web server και αναμένεται μια έγκυρη απάντηση μέσα στο χρονικό πλαίσιο που θέτει το Timeout.

Αν ο web server απαντήσει με ένα έγκυρο μήνυμα μέσα στον προκαθορισμένο χρόνο τότε ο χρήστης θα ειδοποιηθεί με μήνυμα στην οθόνη του ότι υπάρχει διαθεσιμότητα του web server. Στην αντίθετη περίπτωση ο χρήστης θα ειδοποιηθεί ότι δεν υπάρχει διαθεσιμότητα του web server. Κάτι τέτοιο μπορεί να συμβεί για έναν από τους εξής λόγους: Ο web server δεν απάντησε καθόλου οπότε έληξε το timeout. Ο web server απάντησε αλλά η απάντηση έφτασε μετά την λήξη του timeout. Ο web server απάντησε μέσα στα χρονικά όρια αλλά το μήνυμα απάντησης δεν ήταν έγκυρο.

Με το πλήκτρο Test Account μπορεί να διαπιστώσει αν τα τρέχοντα username και password είναι έγκυρα και αναγνωρίζονται από τον web server. Προϋπόθεση για να πετύχει το Test Account είναι να υπάρχει ο web server με το σωστό web service και να αποκριθεί πριν λήξει το timeout κάτι που μπορεί να διαπιστωθεί με το πλήκτρο Server Connectivity.



(α') Buddies Screen (β') Buddies Menu

Σχήμα 5.14: Buddies

Buddies

Η επιλογή Buddies μεταφέρει τον χρήστη σε μία νέα οθόνη η οποία φαίνεται στο σχήμα 5.14α'. Στην οθόνη αυτή παρουσιάζεται μια λίστα με όλους τους απομακρυσμένους φίλους (Buddies) του χρήστη. Κάθε εγγραφή σε αυτήν την λίστα περιέχει το Id του φίλου

(Ο αριθμός στο μπλε φόντο), το username του φίλου και την κατάσταση (status) του φίλου. Οι καταστάσεις στις οποίες μπορεί να βρίσκεται μια εγγραφή της λίστας είναι:

Request Made: Ο χρήστης της εφαρμογής έκανε μια αίτηση αποδοχής στον χρήστη της λίστας με κατάσταση Request Made την οποία δεν έχει κάνει ακόμη δεκτή.

Requested: Ο χρήστης με κατάσταση Requested έκανε μια αίτηση αποδοχής στον χρήστη της εφαρμογής την οποία δεν έχει κάνει ακόμη δεκτή.

Friend: Μια αίτηση αποδοχής των δύο παραπάνω περιπτώσεων έγινε δεκτή και πλέον οι δύο χρήστες είναι φίλοι.

Για να αποδεχτεί ο χρήστης της εφαρμογής έναν φίλο στην κατάσταση Requested πρέπει να τον επιλέξει χρησιμοποιώντας τα πλήκτρα κατευθύνσεως του D-Pad και στην συνέχεια να επιλέξει το menu Accept(Το menu αυτό εμφανίζεται μόνο αν γίνει highlighted ένας χρήστης στην κατάσταση Requested, σε κάθε άλλη περίπτωση δεν εμφανίζεται).

Για να διαγράψει ο χρήστης της εφαρμογής έναν φίλο σε όποια κατάσταση και αν βρίσκεται αυτός πρέπει να επιλέξει πρώτα τον φίλο προς διαγραφή χρησιμοποιώντας τα πλήκτρα κατευθύνσεων του D-Pad και στην συνέχεια να επιλέξει το menu Delete Buddy.

Για να στείλει ο χρήστης της εφαρμογής μια αίτηση αποδοχής σε έναν άλλον χρήστη πρέπει να επιλέξει το menu Add Buddy. Στην συνέχεια θα του ζητηθεί να δώσει τον αριθμό Id του χρήστη που θέλει να στείλει την αίτηση αποδοχής. Όλες οι επιλογές του menu φαίνονται στο σχήμα 5.14β'.

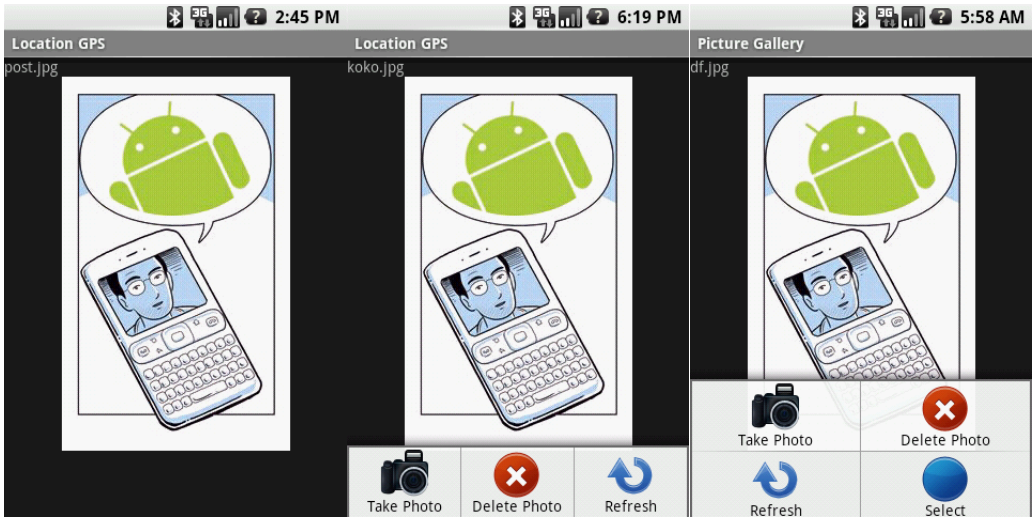
Τέλος αν ο χρήστης κάνει click πάνω σε κάποιον φίλο ο οποίος βρίσκεται στην κατάσταση Friend θα μεταφερθεί σε μια νέα οθόνη όπου μπορεί να δει τα σημεία ενδιαφέροντος του συγκεκριμένου φίλου τα οποία έχουν οριστεί από αυτόν τον φίλο ως δημόσια. Η λίστα σημείων είναι πανομοιότυπη με την λίστα σημείων που περιγράφηκε στο κεφάλαιο 5.2.2. Η μόνη διαφορά της είναι ότι έχει περιορισμένη λειτουργικότητα μιας και τα σημεία που περιέχει δεν "ανήκουν" στον χρήστη αλλά στον φίλο του. Έτσι περιορίζεται στην απλή παρουσίαση των σημείων αυτών.

Πρέπει να σημειωθεί πως αν γίνει μια αίτηση αποδοχής στον χρήστη κατά την διάρκεια που αυτός βρίσκεται στην παρούσα οθόνη δεν θα προστεθεί αυτόματα μια καινούργια εγγραφή στην λίστα. Αντιθέτως την επόμενη φορά που θα επιστρέψει την ίδια οθόνη και θα ξαναφορτωθεί η λίστα με τους φίλους θα εμφανιστεί κανονικά. Σε περίπτωση που για κάποιον λόγο χαθεί η επικοινωνία με τον server η οθόνη με την λίστα των φίλων ή με την λίστα των δημόσιων σημείων των φίλων θα κλίσει και ο χρήστης θα επιστρέψει στο κεντρικό menu της εφαρμογής.

Picture Gallery

Είτε η επιλογή Picture Gallery από το κεντρικού menu, είτε η επιλογή Upload Image από την οθόνη επεξεργασίας ενός σημείου ενδιαφέροντος μεταφέρει τον χρήστη σε μια νέα οθόνη η οποία φαίνεται στο σχήμα 5.15α'.

Η οθόνη αυτή αποτελεί ουσιαστικά έναν περιηγητή στις φωτογραφίες που έχει τραβήξει ο χρήστης μέσα από την εφαρμογή. Η τρέχουσα φωτογραφία και το filename της



(α') Picture Gallery (β') Menu 1 (γ') Menu 2

Σχήμα 5.15: Η οθόνη Picture Gallery

φαίνονται στην οθόνη. Ενώ με σύρσιμο δεξιά και αριστερά πάνω στην τρέχουσα φωτογραφία μπορούμε να δούμε την προηγούμενη και την επόμενη φωτογραφία στον φάκελο. Ο χρήστης μπορεί να σβήσει την τρέχουσα φωτογραφία επιλέγοντας το menu Delete. Επίσης μπορεί επιλέγοντας το menu Refresh να ελέγξει για ύπαρξη νέων εικόνων η οποίες δεν φαίνονται στον περιηγητή. Τέλος μπορεί να επιλέξει από το menu την επιλογή Take Photo προκειμένου να τραβήξει μια νέα φωτογραφία με την συσκευή του (Όλες οι επιλογές των menu φαίνονται στο σχήμα 5.15β'). Μετά την επιλογή Take Photo ζητείται από τον χρήστη να δώσει ένα όνομα για το αρχείο στο οποίο θα αποθηκευτεί η εικόνα που πρόκειται να τραβήξει. Το όνομα πρέπει να τελειώνει με την κατάληξη .jpg ή .JPG για να γίνει δεκτό. Αν δοθεί ένα δεκτό όνομα ο χρήστης είναι σε θέση να τραβήξει μια φωτογραφία με την συσκευή του πατώντας στο μεσαίο πλήκτρο το D-pad. Η φωτογραφία που τράβηξε θα αποθηκευτεί με το όνομα αρχείου που έδωσε αρχικά και το πρόγραμμα θα επιστρέψει στην οθόνη Picture Gallery. Εκτός από τα 3 αυτά κύρια menu (Take Photo, Delete Photo, Refresh) αν ο χρήστης έχει φτάσει σε αυτήν την οθόνη μέσω της επιλογής Upload Image εμφανίζεται μια ακόμη επιλογή menu, η Select, η οποία φαίνεται και στο σχήμα 5.15γ'. Μέσω αυτής της επιλογής ο χρήστης μπορεί να επιλέξει την τρέχουσα φωτογραφία για να την αποθηκεύσει στον webserver.

Σημείωση: Για να μπορεί ο χρήστης να χρησιμοποιήσει την επιλογή Picture Gallery θα πρέπει η συσκευή του να είναι εφοδιασμένη με SD Memory Card προκειμένου να γίνεται σε αυτήν η αποθήκευση των εικόνων.

5.2.3 Μέθοδος υλοποίησης της εφαρμογής Android

Η εφαρμογή της παρούσας διπλωματικής εργασίας δημιουργήθηκε στην γλώσσα προγραμματισμού Java και συγκεκριμένα είναι συμβατή με την έκδοση 1.6 του Java Develop ment Kit (JDK). Για την συγγραφή της χρησιμοποιήθηκε το Android SDK 1.0 καθώς επίσης και οι βιβλιοθήκες Apache Mime4j (http://james.apache.org/mime4j/index.html) 0.5, από την HttpComponents Client το Http Mime module (http://hc.apache.org/index.html) και την Commons IO (http://commons.apache.org/io/).

Το Android SDK έχει περιγραφεί με λεπτομέρεια σε προηγούμενο κεφάλαιο και αποτελεί την βάση πάνω στην οποία αναπτύσσεται η εφαρμογή. Οι υπόλοιπες βιβλιοθήκες έχουν βοηθητικό ρόλο και σκοπός τους είναι η ευκολότερη επικοινωνία με τον web server. Παρέχουν όλες τις απαραίτητες διαπροσωπείες προκειμένου να μπορέσει το πρόγραμμα να στέλνει και να δέχεται διαφόρων ειδών http μηνύματα (πχ multipart http message).

Ο πηγαίος κώδικας της εφαρμογής οργανώνεται σε 4 packages και για περισσότερες λεπτομέρειες σχετικές με την υλοποίηση ο αναγνώστης μπορεί να ανατρέξει στο παράρτημα στο οποίο βρίσκεται ο πλήρης πηγαίος κώδικας της εφαρμογής. Τα τέσσερα πακέτα είναι:

1. gr.ntua.cn.GPS
2. gr.ntua.cn.GPS.Auxiliary
3. gr.ntua.cn.GPS.ImageManagement
4. gr.ntua.cn.GPS.Services

gr.ntua.cn.GPS

Το package gr.ntua.cn.GPS αποτελεί το κύριο πακέτο της εφαρμογής. Περιέχει τις κλάσεις οι οποίες δημιουργούν το σύνολο σχεδόν του User Interface καθώς και την λειτουργικότητα που κρύβεται πίσω από αυτό.

gr.ntua.cn.GPS.Auxiliary

Το package gr.ntua.cn.GPS.Auxiliary περιέχει ένα σύνολο από βοηθητικές κλάσεις οι οποίες εκτελούν ένα πλήθος λειτουργιών. Συγκεκριμένα περιέχει μια κλάση για την πρόσβαση στην τοπική βάση δεδομένων η οποία βρίσκεται αποθηκευμένη στο κινητό (DbAdapter), μια κλάση η οποία περιέχει πληροφορίες για μια περιοχή (DescribedLocation), μια κλάση η οποία αναλαμβάνει την επικοινωνία με τον απομακρυσμένο web server (HttpConnector), μια κλάση η οποία "ζωγραφίζει" τα διάφορα σημεία ενδιαφέροντος στον χάρτη (myOverlay) και τέλος μια κλάση η οποία χρησιμεύει στην αποθήκευση και ανάκτηση των ρυθμίσεων της εφαρμογής (SettingManager).

gr.ntua.cn.GPS.ImageManagement

Το package gr.ntua.cn.GPS.ImageManagement περιέχει ένα σύνολο από κλάσεις οι οποίες ασχολούνται με τα θέματα διαχείρισης εικόνων. Συγκεκριμένα περιέχει μια κλάση η οποία εμφανίζει το User Interface της Picture Gallery (picGallery), μια κλάση η οποία εμφανίζει μια οθόνη προεπισκόπησης λήψης φωτογραφίας (CameraPreview) και μια κλάση η οποία χρησιμοποιεί τις δυνατότητες της φωτογραφικής μηχανής της υποκείμενης συσκευής προκειμένου να τραβήξει μια φωτογραφία (Preview).

gr.ntua.GPS.Services

Το package gr.ntua.GPS.Services περιέχει τρείς κλάσεις οι οποίες αποτελούν Services τα οποία τρέχουν χωρίς την εμφάνιση κάποιου User Interface. Συγκεκριμένα ένα service το οποίο τρέχει διαρκώς και το οποίο ανιχνεύει την διαγραφή των notification (Notification DeleteService), ένα service το οποίο τρέχει όταν η εφαρμογή βρίσκεται σε minimized mode και ένα service το οποίο τρέχει όταν η εφαρμογή βρίσκεται σε trip mode.

5.2.4 Web Εφαρμογή

Η Web εφαρμογή είναι σχεδιασμένη για να μπορεί να δέχεται μηνύματα που παράγονται από μια Android based εφαρμογή και να ενεργεί κατάλληλα είτε απαντώντας σε αυτά, είτε αποθηκεύοντας δεδομένα στην βάση δεδομένων, είτε αποθηκεύοντας φωτογραφίες σε κάποιο τοπικό ή απομακρυσμένο για αυτήν αποθηκευτικό μέσο.

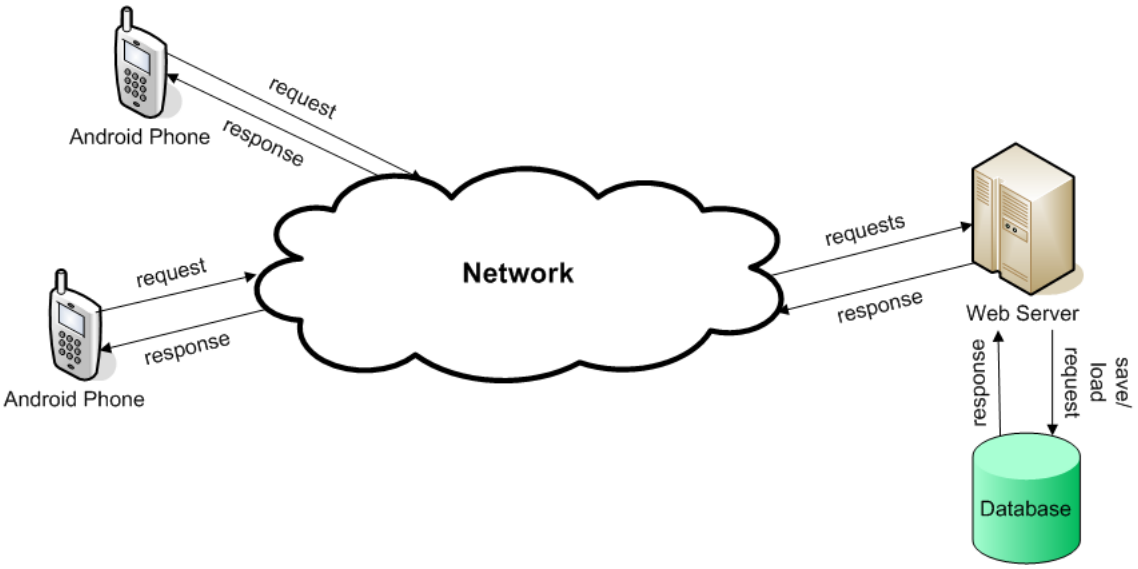
Η Android based εφαρμογή όταν στέλνει ένα μήνυμα προς κάποιον απομακρυσμένο server δεν γνωρίζει τίποτα για την υλοποίηση της απομακρυσμένης εφαρμογής στην οποία απευθύνεται. Το μόνο το οποίο γνωρίζει είναι ένα υποτυπώδες πρωτόκολλο το οποίο περιμένει να τηρηθεί από την πλευρά της απομακρυσμένης εφαρμογής. Έτσι δεν έχει σημασία σε ποια γλώσσα είναι γραμμένη η web εφαρμογή αρκεί να υλοποίει το υποτυπώδες πρωτόκολλο επικοινωνίας και να χρησιμοποιεί το πρωτόκολλο HTTP για την αποστολή και λήψη μηνυμάτων. Η συγκεκριμένη web εφαρμογή αυτής της διπλωματικής εργασίας αποτελείται από ένα σύνολο JSP σελίδων (Java Server Pages) τις οποίες φιλοξενεί ένας webserver Apache Jacarta Tomcat 3.3.2 (χωρίς να παίζει ιδιαίτερο ρόλο σε ποιόν webserver θα φιλοξενήσουμε τις σελίδες αρκεί αυτός να υποστηρίζει JSP).

Στην συνέχεια γίνεται μια σύντομη περιγραφή του πρωτοκόλλου το οποίο υλοποιούν τόσο η Android based εφαρμογή όσο και η web εφαρμογή. Για περισσότερες λεπτομέρειες στα θέματα της υλοποίησης ο αναγνώστης μπορεί να ανατρέξει στο παράρτημα στο οποίο μπορεί να βρει τον πλήρη πηγαίο κώδικα της εφαρμογής.

Το πρωτόκολλο επικοινωνίας

Η επικοινωνία μεταξύ μιας Android based εφαρμογής και της web εφαρμογής πάντα ξεκινά από την μεριά της Android based εφαρμογής. Αυτή κάνει μια αίτηση για κάποια υπηρεσία στην web εφαρμογή και η web εφαρμογή αναλόγως της αιτήσεως αυτής πράττει ανάλογα. Κάθε αίτηση της Android based εφαρμογής γίνεται προς συγκεκριμένο url του

webserver οπού αυτός περιμένει τις αιτήσεις αυτού του είδους. Οι διάφορες αιτήσεις και λειτουργίες που επιτελούνται παρουσιάζονται παρακάτω (Για την ευκολότερη παρουσίαση θα αναφερόμαστε σε μια Android based εφαρμογή ως A και στην web εφαρμογή ως S):



Σχήμα 5.16: Πρωτόκολλο επικοινωνίας

Πιστοποίηση Χρήστη. Ο A στέλνει ένα HTTP Post στο URL "/login.jsp" με παραμέτρους "user" & "pass" το username και password προς πιστοποίηση. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει το μήνυμα AUTH OK <userId> αν τα στοιχεία που έστειλε ο A πιστοποιήθηκαν ή AUTH FAILED αν η πιστοποίηση απέτυχε.

Έλεγχος Σύνδεσης. Ο A στέλνει ένα HTTP Post στο URL "/test.jsp". Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει το μήνυμα TEST OK. Ο A περιμένει για την απάντηση μέχρι να λήξει ένα προκαθορισμένο timeout.

Λήψη Απομακρυσμένων Σημείων Ενδιαφέροντος. Ο A στέλνει ένα HTTP Post στο URL "/getpoint.jsp" με παράμετρο "userId" το Id του χρήστη του οποίου θέλει να λάβει τα σημεία ενδιαφέροντος. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει ένα xml αρχείο το οποίο περιέχει μια λίστα με σημεία ενδιαφέροντος. Η μορφή του xml δίδεται παρακάτω:

```
<Message>
        <Status>OK</Status>
        <PointList>
        <Point>
                <pointId>pointId</pointId>
                <Longitude>Longitude</Longitude>
```

```
                <Latitude>Latitude</Latitude>
                <Description>Description</Description>
                <Used>Used</Used>
                <Public>Public</Public>
                <Url>urlStr</Url>
        </Point>
        <Point>
                        ...
        </Point>
           ...
        </PointList>
</Message>
```

Αποστολή Σημείου Ενδιαφέροντος προς αποθήκευση. Ο Α στέλνει ένα HTTP Post στο URL "/addpoint.jsp" με παραμέτρους "userId", "Longitude", "Latitude", "Description", "Mode", "Used", "Public" και "Url" των οποίων τα ονόματα υποδηλώνουν και το περιεχόμενο τους. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει το μήνυμα OK αν πέτυχε η αποθήκευση του σημείου ή το μήνυμα FAIL αν αποτύχει η αποθήκευση.

Εύρεση Σημείων Ενδιαφέροντος μέσα στην ακτίνα αναζήτησης. Ο Α στέλνει ένα HTTP Post στο URL "/closestlocations.jsp" με παραμέτρους "Longitude", "Latitude", "Radius", "myuserId" και "userId" των οποίων τα ονόματα υποδηλώνουν και το περιεχόμενο τους. Παράμετροι με κλειδί "userId" μπορούν να υπάρχουν παραπάνω από μια και αναπαριστούν τα ids των φίλων των οποίων τα δημόσια σημεία θα συμπεριληφθούν στην αναζήτηση. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει ένα xml αρχείο το οποίο περιέχει μια λίστα με σημεία ενδιαφέροντος. Η μορφή του xml δίδεται παρακάτω:

```
<PointList>
        <Point>
        <Longitude>longitude</Longitude>
        <Latitude>latitude</Latitude>
        <Description>description</Description>
        <Distance>tempDist</Distance>
        <HasUrl>Yes</HasUrl>
        <Url>UrlStr</Url>
        </Point>

          ...
        <Point>
        <Longitude>longitude</Longitude>
        <Latitude>latitude</Latitude>
        <Description>description</Description>
        <Distance>tempDist</Distance>
        <HasUrl>No</HasUrl>
        </Point>
```

```
</PointList>
```

Εύρεση του κοντινότερου Σημείου Ενδιαφέροντος. Ο Α στέλνει ένα HTTP Post στο URL "/closestlocation.jsp" με παραμέτρους "Longitude", "Latitude", "myuserId" και "userId" των οποίων τα ονόματα υποδηλώνουν και το περιεχόμενο τους. Παράμετροι με κλειδί "userId" μπορούν να υπάρχουν παραπάνω από μια και αναπαριστούν τα ids των φίλων των οποίων τα δημόσια σημεία θα συμπεριληφθούν στην αναζήτηση. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει ένα xml αρχείο το οποίο περιέχει ένα σημείο ενδιαφέροντος. Η μορφή του xml δίδεται παρακάτω:

```
<Point>
        <Longitude>longitude</Longitude>
        <Latitude>latitude</Latitude>
        <Description>description</Description>
        <Distance>tempDist</Distance>
        <HasUrl>No</HasUrl>
</Point>
```

Λήψη λίστας φίλων. Ο Α στέλνει ένα HTTP Post στο URL "/getbuddy.jsp" με παράμετρο "userId" η οποία είναι το id του χρήστη του οποίου τους φίλους πρόκειται να αποστείλει ο webserver. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει ένα xml αρχείο το οποίο περιέχει την λίστα με τους φίλους. Η μορφή του xml δίδεται παρακάτω:

```
<Message>
<Status>OK</Status>
<BuddyList>
        <Status>OK</Status>
        <Buddy>
        <buddyId>friendUserId</buddyId>
        <Username>buddyusername</Username>
        <Status>status</Status>
        </Buddy>
   ...
</BuddyList>
</Message>
```

Λήψη λίστας δημόσιων σημείων ενδιαφέροντος φίλου. Ο Α στέλνει ένα HTTP Post στο URL "/getbuddypoint.jsp" με παράμετρο "userId" η οποία είναι το id του φίλου του οποίου τα σημεία πρόκειται να αποστείλει ο webserver. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει ένα xml αρχείο το οποίο περιέχει την λίστα με τα σημεία του φίλου. Η μορφή του xml δίδεται παρακάτω:

```
<Message>
        <Status>OK</Status>
        <PointList>
```

```
        <Point>
                <pointId>pointId</pointId>
                <Longitude>Longitude</Longitude>
                <Latitude>Latitude</Latitude>
                <Description>Description</Description>
                <Used>Used</Used>
                <Public>Public</Public>
                <Url>urlStr</Url>
        </Point>
        <Point>
                    ...
        </Point>
           ...
        </PointList>
</Message>
```

Επεξεργασία της λίστας φίλων. Ο Α στέλνει ένα HTTP Post στο URL "/addbuddy.jsp" με παραμέτρους "userId", "buddyUserId" και "Mode". Τα ονόματα των δύο πρώτων παραμέτρων περιγράφουν το περιεχόμενο τους. Η παράμετρος "Mode" μπορεί να πάρει τις τιμές 1,2,3 και σηματοδοτεί την προσθήκη, αποδοχή και διαγραφή ενός φίλου αντίστοιχα. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει το μήνυμα OK αν πέτυχε η αλλαγή στην λίστα ή το μήνυμα FAIL αν απέτυχε.

Αποστολή Εικόνας στον webserver. Ο Α στέλνει ένα multipart HTTP Post στο URL "/uploadimage.jsp" με παραμέτρους "userId", "pointId" και "image". Τα ονόματα των δύο πρώτων παραμέτρων περιγράφουν το περιεχόμενο τους και είναι απλά Strings. Η παράμετρος "image" είναι ένα InputStream της εικόνας προς αποστολή. Ο S απαντάει με μια HTTP σελίδα η οποία περιέχει το μήνυμα Upload Successful!! αν πέτυχε η αποστολή.

5.2.5 Βάση Δεδομένων

Στην βάση δεδομένων αποθηκεύονται οι πληροφορίες που στέλνουν χρήστες οι οποίοι χρησιμοποιούν την android based εφαρμογή, καθώς επίσης και πληροφορίες για τους ίδιους τους χρήστες. Αποτελεί μια σχεσιακή βάση δεδομένων που για την υλοποίηση της χρησιμοποιήθηκε ο mySQL Server. Στην συνέχεια δίδεται μια σύντομη περιγραφή του κάθε πίνακα της βάσης, ενώ στο παράρτημα υπάρχουν όλα τα CREATE statements για την κατασκευή της. Η βάση αποτελείται από τρεις πίνακες: Ο πίνακας users περιέχει πληροφορίες σχετικές με την πιστοποίηση των χρηστών (userId,username,password). Ο πίνακας buddies περιέχει πληροφορίες για τις σχέσεις των χρηστών μεταξύ τους και το είδος αυτής της σχέσης (budy1Id,budy2Id,Status1,Status2). Τέλος ο πίνακας points είναι ο πυρήνας της βάσης δεδομένων στον οποίον αποθηκεύονται πληροφορίες σχετικές με τα σημεία ενδιαφέροντος του κάθε χρήστη της εφαρμογής.

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη και Συμπεράσματα

Στην παρούσα διπλωματική εργασία έγινε μελέτη του μηχανισμού παροχής Υπηρεσιών Βασισμένων στην Θέση του χρήστη, των διαθέσιμων τεχνολογιών εντοπισμού θέσης και αναπτύχθηκε μια εφαρμογή παροχής υπηρεσιών θέσης βασισμένη στην πλατφόρμα Google Android.

Από την μελέτη των Location Based Services βγήκε το συμπέρασμα οτι αποτελούν έναν πολλά υποσχόμενο κλάδο στην αγορά της πληροφορικής, τηλεπικοινωνιών και παροχής υπηρεσιών. Οι τεχνολογίες πάνω στις οποίες στηρίζονται οι LBS έχουν ωριμάσει σε τέτοιο βαθμό ώστε να τις καθιστούν υλοποιήσιμες και οικονομικά συμφέρουσες για εμπορική εκμετάλλευση. Από την άλλη δεν έχει πρωτοτυποποιηθεί ακόμα ένα πλαίσιο με βάση το οποίο θα γίνεται η ανάπτυξη, η λειτουργία και η συντήρηση ενός LBS, αλλά υπάρχουν διάφορες ξεχωριστές προτάσεις.

Οι τεχνολογίες εντοπισμού θέσης, οι οποίες παρέχουν την βασική πληροφορία της θέσης για ένα LBS, μπορούν πλέον να δώσουν σε μικρό χρονικό διάστημα και με αρκετά μεγάλη ακρίβεια υπολογισμούς θέσης για διάφορες κινητές συσκευές με σχετικά μικρό κόστος. Ανάμεσα στις τεχνολογίες ανίχνευσης θέσης κυρίαρχη θέση έχει το GPS το οποίο φαίνεται να έχει επικρατήσει στην ανίχνευση θέσης σε ανοικτό χώρο (για το λόγω αυτό επιλέχτηκε ως η τεχνολογία ανίχνευσης που χρησιμοποιεί η εφαρμογή της παρούσας διπλωματικής εργασίας). Εκτός από το GPS, στα πλαίσια αυτής της διπλωματικής εργασίας, μελετήθηκαν συνοπτικά και άλλες τεχνολογίες εντοπισμού θέσης. Κάθε μια από τις οποίες έχει τα δικά της πλεονεκτήματα και μειονεκτήματα, τα οποία παρουσιάζονται αναλυτικά.

Η πλατφόρμα Google Android παρέχει όλες τις δυνατότητες για την ανάπτυξη ενός LBS. Υποστηρίζει GPS, σύνδεση στο Διαδίκτυο μέσω WiFi, Web Browsing, εμφάνιση πολυμεσικού περιεχομένου και γενικά τις περισσότερες από τις τεχνολογίες που ενσωματώνουν όλες οι σύγχρονες συσκευές κινητής τηλεφωνίας.

Το κύριο μέρος της διπλωματικής εργασίας αφορά την ανάπτυξη μιας LBS για το Google Android, στο οποίο ο χρήστης μπορεί να βλέπει την τρέχουσα θέση του σε έναν παγκόσμιο χάρτη καθώς και να ορίζει διάφορα σημεία ενδιαφέροντος τα οποία εμφανίζο-

νται μόνο όταν βρίσκεται κοντά σε αυτά. Επιπλέον ένα σύνολο βοηθητικών λειτουργιών και επιλογών τελικού χρήστη προστέθηκαν με σκοπό την αύξηση της χρηστικότητας της υπηρεσίας.

6.2 Μελλοντικές Επεκτάσεις

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας σχεδιάστηκε και οργανώθηκε έτσι ώστε να είναι πλήρως επεκτάσιμη. Έχει χωριστεί σε διαφορετικά modules ανάλογα με τις παρεχόμενες λειτουργίες, παρέχοντας έτσι την δυνατότητα στους προγραμματιστές να επεκτείνουν συγκεκριμένες λειτουργίες χωρίς να επηρεάζονται τα υπόλοιπα τμήματα της εφαρμογής. Μελλοντικές επεκτάσεις της εφαρμογής αφορούν:

- **Υλοποίηση ασφαλούς μεταφοράς.** Ένα θέμα που δεν υλοποιήθηκε πλήρως είναι αυτό της ασφαλούς μεταφοράς των δεδομένων μεταξύ του server και της κινητής συσκευής. Παρόλο που χρησιμοποιήθηκαν όλες εκείνες οι προγραμματιστικές μέθοδοι για την ασφάλεια της εφαρμογής, δεν γίνεται κρυπτογράφηση των http μηνυμάτων. Η απόφαση αυτή πάρθηκε έτσι ώστε να απλοποιηθεί η υλοποίηση της εφαρμογής. Έτσι σε μελλοντικές εκδόσεις της εφαρμογής μπορεί να συμπεριληφθεί η δυνατότητα κρυπτογράφησης με κάποιο πρωτόκολλο όπως τα SSL, TSL κα.

- **Επιπλέον τεχνολογίες εντοπισμού θέσης.** Η εφαρμογή θα μπορούσε επίσης να επεκταθεί με χρήση επιπλέον τεχνολογιών ανίχνευσης. Συγκεκριμένα σε μελλοντικές εκδόσεις θα μπορούσε να χρησιμοποιηθεί το WiFi για τον εντοπισμό θέσης, το οποίο ήδη υποστηρίζεται από την τρέχουσα έκδοση του SDK του Android. Σε επόμενες εκδόσεις του Android SDK αναμένεται να υποστηριχτεί και το Bluetooth, οπότε μπορεί να χρησιμοποιηθεί και αυτό για εντοπισμό θέσης.

- **Μεγαλύτερη αλληλεπίδραση με σημεία ενδιαφέροντος.** Τέλος θα μπορούσαν να προστεθούν ορισμένες επιπλέον δυνατότητες σχετικές με την αλληλεπίδραση του χρήστη με τα σημεία ενδιαφέροντος που αυτός ορίζει. Για παράδειγμα η προσθήκη επιπλέον πληροφοριών ή η εμφάνισή πληροφοριών πλοήγησης μεταξύ σημείων ενδιαφέροντος.

Βιβλιογραφία

[1] International Telecommunication Union, ”Telecommunication Statistics”, [Online Statistics], (2008 Jul), [2009 Jun 16], Available at HTTP: http://www.itu.int/ITU-D/ict/statistics/ict/graphs/mobile.jpg

[2] International Telecommunication Union, ”Telecommunication Statistics”, [Online Statistics], (2008 Jul), [2009 Jun 16], Available at HTTP: http://www.itu.int/ITU-D/icteye/Indicators/Indicators.aspx

[3] Martin Klopfer, ”Interoperability & Open Architectures: An Analysis of Existing Standardisation Processes & Procedures”, Open Geospatial Consortium, 2005 May

[4] Gidon Lissai, ”Assisted GPS Solution in Cellular Networks”, Rochester Institute of Technology, Nov 2006

[5] Goran M. Djuknic, Robert E. Richton, ”Geolocation and Assisted GPS”, Lucent Technologies IEEE Computer, vol. 34, no. 2, pp. 123 - 125, 2001 Feb

[6] GSM Association, ”Location Based Services”, 3.1.0 ver. , 2003 Jan

[7] V. Joy, S. Sridevi, P. V. Laxman, ”Location Based Services- Enterprise Mobility”, Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE, pp. 3087-3092, 2008 Apr

[8] G. M. Giaglis, P. Kourouthanassis, A. Tsamakos, ”Towards a classification framework for mobile location services”, Mobile commerce: technology, theory, and applications, Hersley PA., USA: IGI Publishing, 2003, pp. 67 - 85.

[9] S. J. Barnes , ”Location-Based Services: The State of the Art”, e-Service Journal, vol. 2, no. 3, pp. 59-70 , 2006 Nov 29

[10] Federal Communication Commission, ”9-1-1 Service”, [2009 June 16], Available at HTTP: http://www.fcc.gov/pshs/services/911-services/

[11] S. Herden, A. Mkrtchyan, C. Rautenstrauch, A. Zwanziger, M. Schenk, ”Personal information guide - a platform with location based service for mobile powered e-commerce”, Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on, Magdeburg, Germany: 2003 Sept, pp. 895 - 900

[12] Center for Democracy and Technology, [2009 Jun 16], Available at HTTP: http://www.cdt.org/privacy/guide/

[13] CAN-SPAM Act: (15 U.S.C. 7701, et seq., Public Law No. 108-187, was S.877 of the 108th United States Congress)

[14] Federal Aviation Administration, ”GPS Policy - Selective Availability”, (2007 Jun), [2009 Jun 16], Available at HTTP: http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/policy/availability/index.cfm

[15] J. E. Dobson, P. F. Fisher, ”Geoslavery”, IEEE Technology and Society Magazine, vol. 22, no. 1, pp. 47-52, 2003 Spring

[16] C. Ciavarella, F. Paternò, ”Design Criteria for Location-aware, Indoor, PDA Applications”, of Lecture Notes in Computer Science, Pisa, Italy: Springer, 2003, pp. 131-144

[17] A. M. Bernardos, J. R. Casar, P. Tarrio , ”Building a framework to characterize location-based services”, Next Generation Mobile Applications, Services and Technologies, 2007. NGMAST ’07. The 2007 International Conference on, Madrid, Spain: Springer, 2007 Sept, pp. 110-118

[18] M. Porretta, P. Nepa, G. Manara, F. Giannetti, ”Location, Location, Location”, Vehicular Technology Magazine, IEEE, vol. 3, no. 2, pp. 20-29, 2008 June

[19] Y. Gu, A. Lo and I. Niemegeers, ”A Survey of Indoor Positioning Systems for Wireless Personal Networks,”IEEE Communications Surveys and Tutorials”, vol. 11, no. 1, pp. 13-32, 2009 First Quarter

[20] Global Positioning System Site, ”The Global Positioning System”, [2009 Jun 16] Available at HTTP: http://www.gps.gov/systems/gps/index.html

[21] Global Security Site, ”Space:GPS III Operational Control Segment (OCX)”, [2009 Jun 16] Available at HTTP: http://www.globalsecurity.org/space/systems/gps_3-ocx.htm

[22] Los Angeles Airforce Base, ”Global Positioning System”, [2009 Jun 16], Available at HTTP: http://www.losangeles.af.mil/library/factsheets/factsheet.asp?id=5325

[23] E. Weston, ”GPS Navigation Satellite message format and protocol details”, (1999 Mar), [2009 Jun 16], Available at HTTP: http://gpsinformation.net/gpssignal.htm

[24] T. V. Flandern, ”What the Global Positioning System Tells Us about Relativity”, Open Questions in Relativistic Physics, Montreal, Canada: Apeiron, 1998, pp. 81-90

[25] kowoma.de, ”Sources of Errors in GPS”, (2009 Mar), [2009 jun 2006], Available at HTTP: http://www.kowoma.de/en/gps/errors.htm

[26] J. LaMance, J. Jarvinen, J. DeSalas, ”Assisted GPS: A Low-Infrastructure Approach”, (2002 Mar), [2009 Jun 16], Available at HTTP: http://www.gpsworld.com/gpsworld/article/articleDetail.jsp?id=12287

[27] Android Site, ”Android SDK Reference”, [2009 Jun 16], Available at HTTP: http://www.android.com/

Παράρτημα Α'

Κώδικας Εφαρμογής Android

Πακέτο gr.ntua.cn.GPS

GPS.java

```
1  package gr.ntua.cn.GPS;
2
3  import gr.ntua.cn.GPS.Auxiliary.DbAdapter;
4  import gr.ntua.cn.GPS.Auxiliary.DescribedLocation;
5  import gr.ntua.cn.GPS.Auxiliary.HttpConnector;
6  import gr.ntua.cn.GPS.Auxiliary.SettingManager;
7  import gr.ntua.cn.GPS.Auxiliary.myOverlay;
8  import gr.ntua.cn.GPS.Auxiliary.HttpConnector.BuddyGetter;
9  import gr.ntua.cn.GPS.Auxiliary.HttpConnector.ClosestLocations;
10 import gr.ntua.cn.GPS.Services.LocationService;
11 import gr.ntua.cn.GPS.Services.LocationTripService;
12 import gr.ntua.cn.GPS.Services.NotificationDeleteService;
13
14 import java.io.IOException;
15 import java.util.Iterator;
16 import java.util.List;
17 import java.util.Vector;
18
19 import javax.xml.parsers.DocumentBuilder;
20 import javax.xml.parsers.DocumentBuilderFactory;
21 import javax.xml.parsers.ParserConfigurationException;
22
23 import org.w3c.dom.Document;
24 import org.w3c.dom.Node;
25 import org.w3c.dom.NodeList;
26 import org.xml.sax.SAXException;
27
28 import android.app.Activity;
29 import android.app.AlertDialog;
30 import android.app.Notification;
31 import android.app.NotificationManager;
32 import android.app.PendingIntent;
33 import android.app.ProgressDialog;
34 import android.content.ComponentName;
35 import android.content.Context;
36 import android.content.DialogInterface;
37 import android.content.Intent;
38 import android.content.ServiceConnection;
39 import android.content.SharedPreferences;
40 import android.database.Cursor;
41 import android.graphics.Paint.Style;
42 import android.location.Location;
43 import android.location.LocationListener;
44 import android.location.LocationManager;
45 import android.net.Uri;
46 import android.os.Bundle;
47 import android.os.IBinder;
48 import android.view.Menu;
49 import android.view.MenuItem;
```

```
50  import android.view.MotionEvent;
51  import android.view.SubMenu;
52  import android.view.View;
53  import android.view.View.OnClickListener;
54  import android.view.View.OnTouchListener;
55  import android.widget.EditText;
56  import android.widget.FrameLayout;
57  import android.widget.ImageView;
58  import android.widget.LinearLayout;
59  import android.widget.TextView;
60  import android.widget.Toast;
61  import android.widget.ZoomControls;
62
63  import com.google.android.maps.GeoPoint;
64  import com.google.android.maps.MapActivity;
65  import com.google.android.maps.MapController;
66  import com.google.android.maps.MapView;
67  import com.google.android.maps.Overlay;
68  import com.google.android.maps.Projection;
69
70  /**
71   * The main class of the application. It contains a map which displays the
72   * current position and the closest PoI (Point of Interest)
73   *
74   * @author Stavros Zotalis
75   */
76  public class GPS extends MapActivity {
77    private ImageView gpsStatus;
78
79    private boolean listeningLocations = true;
80    /**
81     * The Location Manager of the activity
82     */
83    private LocationManager lm;
84    /**
85     * The Notification Manager of the activity
86     */
87    private NotificationManager nm;
88    /**
89     * The Notification Id
90     */
91    private int NOTIFICATION_ID = 1;
92
93    /**
94     * The Location Listener for the Location Manager of the activity
95     */
96    private LocationListener locationListener;
97    /**
98     * A DbAdapter object for the retrieving of the PoIs
99     */
100   private DbAdapter dbadapter;
101   /**
102    * A ZoomControl View for zooming in/out
103    */
104   private ZoomControls zc;
105   /**
106    * The last clicked GeoPoint on the map
107    */
108   private GeoPoint lastPointClicked = null;
109   /**
110    * A List for storing all the overlays that are displayed on the map
111    */
112   private List<Overlay> overlaylist = null;
113   /**
114    * The overlay that displays the current location
115    */
116   private myOverlay myLocOver = null;
117
118   private myOverlay myLocOverRadius = null;
119   /**
120    * The overlay that displays the closest PoI
121    */
122   private myOverlay poiOver = null;
123
124   /**
125    * Constant showing the Local Mode
126    */
127   public final static int LOCAL_MODE = 1;
128   /**
129    * Constant showing the Remote Mode
130    */
131   public final static int REMOTE_MODE = 2;
132   /**
```

```
133       * Variable holding the Mode Status: Local/Remote
134       */
135      private int pointsMode = LOCAL_MODE;
136
137      /**
138       * A remote connector
139       */
140      private HttpConnector remoteConnector;
141
142      /**
143       * SettingManager for retrieving the settings
144       */
145      private SettingManager settingManager;
146
147      /**
148       * A vector that saves the buddy ids
149       */
150      private Vector<String> buddies;
151      /**
152       * Include buddies or not
153       */
154      private boolean includebuddies;
155      /**
156       * The Radius of the search
157       */
158      private float Radius;
159      /**
160       * the Frame Layout which is used for the Radius input
161       */
162      private FrameLayout fl;
163      private EditText input;
164      private AlertDialog ad;
165
166      /**
167       * The Second Frame Layout wich is used for Listening Parameters input
168       */
169      private FrameLayout fl2;
170      private EditText timeInput;
171      private EditText distInput;
172      private TextView labTime;
173      private TextView labDist;
174      private AlertDialog ad2;
175      /**
176       * The minimum time between updates
177       */
178      private long minTimeForUpdates = 0;
179      /**
180       * The minimum distance between updates
181       */
182      private float minDistanceForUpdates = 0;
183
184      /**
185       * Creation of the menus
186       */
187      @Override
188      public boolean onCreateOptionsMenu(Menu menu) {
189        SubMenu sm = menu.addSubMenu("PoIs");
190        sm.setIcon(R.drawable.poiadd);
191        sm.add(1, 1, 1, "PoI␣List");
192        sm.add(2, 2, 1, "Current␣Location");
193        sm.add(3, 3, 1, "Last␣Clicked␣Location");
194        sm = menu.addSubMenu("Modes").setIcon(R.drawable.modes);
195        sm.add(4, 4, 1, "Toggle␣View␣Mode");
196        sm.add(5, 5, 1, "Toggle␣Point␣Mode");
197        sm.add(7, 7, 1, "Minimized␣Mode");
198        sm.add(11, 11, 1, "Trip␣Mode");
199        sm.add(8, 8, 1, "Change␣Radius");
200        sm.add(9, 9, 1, "Recalculate␣Position");
201        sm.add(10, 10, 1, "Toggle␣Position␣Listening");
202        sm.add(12, 12, 1, "Listening␣Parameters");
203        sm.add(6, 6, 1, "Include␣Buddies").setCheckable(true);
204
205        return super.onCreateOptionsMenu(menu);
206      }
207
208      /**
209       * This function is called every time the menu is shown
210       */
211      @Override
212      public boolean onPrepareOptionsMenu(Menu menu) {
213        menu.findItem(6).setChecked(this.includebuddies);
214        return super.onPrepareOptionsMenu(menu);
215      }
```

```
216
217     /**
218      * Actions taken when one of the menu items selected
219      */
220     @Override
221     public boolean onOptionsItemSelected(MenuItem item) {
222       Intent i;
223       switch (item.getItemId()) {
224       case 12:
225         ad2.show();
226         break;
227       case 10: // Toggle between Listening Location Modes
228         if (listeningLocations) {
229           listeningLocations = false;
230           gpsStatus.setBackgroundDrawable(null);
231           lm.removeUpdates(this.locationListener);
232           lm.requestLocationUpdates("gps", minTimeForUpdates,
233               minDistanceForUpdates, this.secondaryListener);
234         } else {
235           listeningLocations = true;
236           gpsStatus.setBackgroundResource(R.drawable.gps);
237           lm.removeUpdates(this.secondaryListener);
238           lm.requestLocationUpdates("gps", minTimeForUpdates,
239               minDistanceForUpdates, this.locationListener);
240         }
241         break;
242       case 9: // Search for closest PoIs based on the Last Known Location
243         // (Recalculate Position)
244         Location loc = this.lm.getLastKnownLocation("gps");
245         if (loc != null)
246           this.locationListener.onLocationChanged(loc);
247         break;
248       case 8:
249         ad.setTitle("Radius: " + Radius + " meters. Change Radius:");
250         ad.show();
251         break;
252       case 11:
253         if (this.pointsMode == GPS.REMOTE_MODE)
254           tripMode();
255         break;
256       case 7:
257         if (this.pointsMode == GPS.REMOTE_MODE)
258           minimizeMode();
259         break;
260
261       case 6: // Include Buddies
262         this.includebuddies = !this.includebuddies;
263         item.setChecked(this.includebuddies);
264         break;
265       case 1: // Show the Local/Remote PointList
266         if (pointsMode == GPS.LOCAL_MODE) {
267           i = new Intent(GPS.this, PointList.class);
268           startActivityForResult(i, 1);
269         } else {
270           i = new Intent(GPS.this, RemotePointList.class);
271           i.putExtra("settingmanager", settingManager);
272           startActivityForResult(i, 1);
273         }
274         break;
275       case 2: // Add the current location
276         loc = null;
277         if ((loc = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER)) == null)
278           return false;
279         double longt = loc.getLongitude();
280         double lat = loc.getLatitude();
281         if (pointsMode == GPS.LOCAL_MODE) {
282           i = new Intent(GPS.this, AddPoint.class);
283           i.setAction("gr.ntua.cn.GPS.ADD");
284           i.putExtra(DbAdapter.KEY_LONG, longt);
285           i.putExtra(DbAdapter.KEY_LAT, lat);
286           startActivity(i);
287         } else {
288           i = new Intent(GPS.this, AddRemotePoint.class);
289           i.setAction("gr.ntua.cn.GPS.ADD");
290           i.putExtra(DbAdapter.KEY_LONG, longt);
291           i.putExtra(DbAdapter.KEY_LAT, lat);
292           i.putExtra("settingmanager", settingManager);
293           startActivityForResult(i, 1);
294         }
295         break;
296       case 3: // Add the last clicked location
297         if (lastPointClicked == null)
298           return false;
```

```
299         longt = lastPointClicked.getLongitudeE6() / (1000000.0);
300         lat = lastPointClicked.getLatitudeE6() / (1000000.0);
301         if (pointsMode == GPS.LOCAL_MODE) {
302
303           i = new Intent(GPS.this, AddPoint.class);
304           i.setAction("gr.ntua.cn.GPS.ADD");
305           i.putExtra(DbAdapter.KEY_LONG, longt);
306           i.putExtra(DbAdapter.KEY_LAT, lat);
307           startActivity(i);
308         } else {
309           i = new Intent(GPS.this, AddRemotePoint.class);
310           i.setAction("gr.ntua.cn.GPS.ADD");
311           i.putExtra(DbAdapter.KEY_LONG, longt);
312           i.putExtra(DbAdapter.KEY_LAT, lat);
313           i.putExtra("settingmanager", settingManager);
314           startActivityForResult(i, 1);
315         }
316         break;
317       case 4: // Change View
318         mapView.setSatellite(!mapView.isSatellite());
319         mapView.invalidate();
320         break;
321       case 5: // Change Point Mode
322
323         // Create a progress Dialog
324         final ProgressDialog pd = ProgressDialog.show(GPS.this,
325             "Please␣Wait", "Connecting␣To␣Server...");
326         pd.setCancelable(false);
327         // Try to change Mode in a new Thread because changing modes may
328         // take time
329         new Thread() {
330           @Override
331           public void run() {
332             final String msg = changeMode();
333             pd.dismiss();
334             GPS.this.runOnUiThread(new Runnable() {
335
336               @Override
337               public void run() {
338
339                 new AlertDialog.Builder(GPS.this)
340                     .setMessage(msg)
341                     .setTitle("Result")
342                     .setNeutralButton(
343                         "OK",
344                         new DialogInterface.OnClickListener() {
345
346                           @Override
347                           public void onClick(
348                               DialogInterface arg0,
349                               int arg1) {
350
351                             arg0.dismiss();
352                           }
353
354                         }).setIcon(R.drawable.knobinfo)
355                     .show();
356               }
357
358             });
359
360           }
361         }.start();
362
363         break;
364       }
365       return super.onOptionsItemSelected(item);
366     }
367
368     private void tripMode() {
369       // Create a notification that can restart the application
370       Notification appnot = new Notification(R.drawable.map,
371           "GPS␣Location␣Minimized", System.currentTimeMillis());
372
373       PendingIntent pi = PendingIntent.getActivity(this, 1, new Intent(
374           GPS.this, GPS.class).putExtra("pointmode", GPS.REMOTE_MODE)
375           .putExtra("includebuddies", GPS.this.includebuddies), 0);
376
377       appnot.setLatestEventInfo(this, "GPS␣Location", "Restart", pi);
378
379       // Set Delete Intent
380       appnot.deleteIntent = PendingIntent.getService(this, 1, (new Intent(
381           GPS.this, LocationService.class))
```

```
382             .setAction("gr.ntua.cn.GPS.ServiceNotificationDeleted"), 1);
383       nm.notify(this.NOTIFICATION_ID + 1, appnot);
384
385       // Start Service and stop this
386       Intent i = new Intent(GPS.this, LocationTripService.class)
387           .setAction("gr.ntua.cn.GPS.StartService");
388       i.putExtra("settingmanager", settingManager);
389       i.putExtra("includebuddies", this.includebuddies);
390       GPS.this.startService(i);
391       GPS.this.setResult(3);
392       GPS.this.finish();
393
394     }
395
396     private void minimizeMode() {
397       // Create a notification that can restart the application
398       Notification appnot = new Notification(R.drawable.map,
399           "GPS␣Location␣Minimized", System.currentTimeMillis());
400
401       PendingIntent pi = PendingIntent.getActivity(this, 1, new Intent(
402           GPS.this, GPS.class).putExtra("pointmode", GPS.REMOTE_MODE)
403           .putExtra("includebuddies", GPS.this.includebuddies), 0);
404
405       appnot.setLatestEventInfo(this, "GPS␣Location", "Restart", pi);
406
407       // Set Delete Intent
408       appnot.deleteIntent = PendingIntent.getService(this, 1, (new Intent(
409           GPS.this, LocationService.class))
410           .setAction("gr.ntua.cn.GPS.ServiceNotificationDeleted"), 1);
411       nm.notify(this.NOTIFICATION_ID + 1, appnot);
412
413       // Start Service and stop this
414       Intent i = new Intent(GPS.this, LocationService.class)
415           .setAction("gr.ntua.cn.GPS.StartService");
416       i.putExtra("settingmanager", settingManager);
417       i.putExtra("includebuddies", this.includebuddies);
418       GPS.this.startService(i);
419       GPS.this.setResult(3);
420       GPS.this.finish();
421     }
422
423     @Override
424     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
425       switch (requestCode) {
426       case 1:
427
428         if (resultCode == Activity.RESULT_CANCELED) {
429           new AlertDialog.Builder(GPS.this).setMessage("Connection␣Lost")
430               .setTitle("Error").setNeutralButton("OK",
431                   new DialogInterface.OnClickListener() {
432
433                     @Override
434                     public void onClick(DialogInterface arg0,
435                         int arg1) {
436
437                       arg0.dismiss();
438                     }
439
440                   }).show();
441
442           if (this.pointsMode == GPS.REMOTE_MODE)
443             changeMode(); // Don't need new thread no communication
444           // takes place
445
446         }
447         if (resultCode == 3) {
448           double longitude = Double.parseDouble(data
449               .getStringExtra("Longitude"));
450           double latitude = Double.parseDouble(data
451               .getStringExtra("Latitude"));
452           String descr = data.getStringExtra("Description");
453           String url = data.getStringExtra("Url");
454           myOverlay mo = new myOverlay(10, true, this);
455           Location loc = new Location("gps"), loc2 = lm
456               .getLastKnownLocation("gps");
457           ;
458           loc.setLatitude(latitude);
459           loc.setLongitude(longitude);
460           float distance = 0;
461           if (loc2 != null) {
462             distance = loc2.distanceTo(loc);
463           }
464
```

```
465           mo.setCoord(longitude, latitude);
466           mo.setDescription(descr);
467           mo.setUrl(url);
468
469           mo.setDistance(distance);
470           this.overlaylist.add(mo);
471           this.mc.animateTo(new GeoPoint((int) (latitude * 1E6),
472               (int) (longitude * 1E6)));
473           this.mapView.invalidate();
474           mc.setZoom(mapView.getMaxZoomLevel() - 5);
475         }
476         break;
477       }
478
479       super.onActivityResult(requestCode, resultCode, data);
480     }
481
482     /**
483      * changes the modes between LOCAL_MODE and REMOTE_MODE <br>
484      * REMOTE_MODE----->LOCAL_MODE executed asap <br>
485      * LOCAL_MODE----->REMOTE_MODE executed after checking the connection
486      * availability
487      */
488
489     /**
490      * Changes the modes between LOCAL and Remote Mode.<br>
491      * While doing that the location listener is turned off. <br>
492      * This method can be used on a different Thread from the UI Thread<br>
493      * as the operations that require to be executed in UI Thread are <br>
494      * executed in UI thread.
495      *
496      */
497     private String changeMode() {
498       // Unregister the Location Listener
499       lm.removeUpdates(locationListener);
500
501       // The return string and the title string
502       String retStr = "Connection␣Attempt␣Finished";
503       String title = null;
504       if (pointsMode == LOCAL_MODE) {
505
506         // Testing the existance of web server
507         try {
508           HttpConnector.TestConnection tc = remoteConnector
509               .createTestConnection();
510           tc.testConnection();
511
512           if (tc.succeed) {  // If webserver is online
513
514             // Identify the user
515             HttpConnector.Authenticator auth = remoteConnector
516                 .createAuthenticator(settingManager.getUsername(),
517                     settingManager.getPassword());
518             auth.logUser();
519
520             if (auth.isAuthenticated()) {  // if user is authenticated
521               settingManager.setUserId(auth.userId);
522               pointsMode = GPS.REMOTE_MODE;
523               // retrieve buddylist
524               if (!this.retrieveBuddies())
525                 return "Location␣failed␣to␣change␣to␣Remote␣Mode";
526               title = "Location␣GPS␣-␣Remote␣Mode";
527               retStr = "Location␣changed␣to␣Remote␣Mode";
528             } else {
529               retStr = "Location␣failed␣to␣change␣to␣Remote␣Mode";
530             }
531           } else {
532             retStr = "Location␣failed␣to␣change␣to␣Remote␣Mode";
533           }
534         } catch (Exception e) {
535           retStr = "Location␣failed␣to␣change␣to␣Remote␣Mode";
536         }
537
538       } else {
539         pointsMode = GPS.LOCAL_MODE;
540         retStr = "Location␣changed␣to␣Local␣Mode";
541         title = "Location␣GPS␣-␣Local␣Mode";
542       }
543       final String title2 = title;
544       runOnUiThread(new Runnable() {
545
546         @Override
547         public void run() {
```

```
548           // Register the Location Listener
549           if (GPS.this.listeningLocations)
550             lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
551                 minTimeForUpdates, minDistanceForUpdates,
552                 locationListener);
553           // Change the title
554           if (title2 != null)
555             GPS.this.setTitle(title2);
556         }
557
558       });
559       return retStr;
560
561     }
562
563     private boolean retrieveBuddies() {
564       this.buddies = null;
565       BuddyGetter bg = remoteConnector.createBuddyGetter(settingManager
566           .getUserId(), settingManager.getUsername(), settingManager
567           .getPassword());
568       bg.getBuddies();
569
570       if (bg.xmlBuddyStream != null) {
571
572         try {
573           DocumentBuilderFactory dbf = DocumentBuilderFactory
574               .newInstance();
575           DocumentBuilder db = dbf.newDocumentBuilder();
576           Document doc = db.parse(bg.xmlBuddyStream);
577           if (!doc.getElementsByTagName("Status").item(0).getFirstChild()
578               .getNodeValue().equals("OK")) {
579             this.changeMode();
580             return false;
581           }
582
583           NodeList nodelist = doc.getElementsByTagName("Buddy");
584           this.buddies = new Vector<String>();
585           Node iNode, icNode;
586           int length = nodelist.getLength();
587           for (int i = 0; i < length; i++) {
588
589             iNode = nodelist.item(i);
590             NodeList children = iNode.getChildNodes();
591
592             int cLength = children.getLength();
593             for (int j = 0; j < cLength; j++) {
594               icNode = children.item(j);
595               if (icNode.getNodeType() == Node.ELEMENT_NODE) {
596                 NodeList texts = icNode.getChildNodes();
597                 for (int k = 0; k < texts.getLength(); k++) {
598                   String val = texts.item(k).getNodeValue();
599                   String name = icNode.getNodeName();
600                   if (name.equals("buddyId"))
601                     this.buddies.add(val);
602                 }
603               }
604             }
605
606           }
607         } catch (ParserConfigurationException e) {
608           this.changeMode();
609           return false;
610         } catch (IOException e) {
611           this.changeMode();
612           return false;
613         } catch (SAXException e) {
614           this.changeMode();
615           return false;
616         } catch (Exception e) {
617           this.changeMode();
618           return false;
619         }
620         return true;
621       } else {
622         this.changeMode();
623         return false;
624       }
625
626     }
627
628     private MapView mapView;
629     private MapController mc;
630
```

```
631     /** Called when the activity is first created. */
632     @Override
633     public void onCreate(Bundle savedInstanceState) {
634       super.onCreate(savedInstanceState);
635
636       // Set the gui
637       setContentView(R.layout.main);
638       // Get location and notification services
639       lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
640       nm = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
641
642       // create AlertDialog for radius input
643       input = new EditText(this);
644
645       fl = new FrameLayout(this);
646       fl.addView(input, new FrameLayout.LayoutParams(
647           FrameLayout.LayoutParams.FILL_PARENT,
648           FrameLayout.LayoutParams.WRAP_CONTENT));
649
650       ad = new AlertDialog.Builder(this).setView(fl)
651           .setTitle("Choose␣Radius").setCancelable(false)
652           .setPositiveButton("OK", new DialogInterface.OnClickListener() {
653
654             @Override
655             public void onClick(DialogInterface arg0, int arg1) {
656               // TODO Auto-generated method stub
657               try {
658                 Radius = Float.parseFloat(input.getText()
659                     .toString());
660               } catch (Exception e) {
661
662               }
663             }
664
665           }).create();
666
667       // End of AlertDialog
668
669       // create AlertDialog for listening input
670       timeInput = new EditText(this);
671       distInput = new EditText(this);
672
673       labTime = new TextView(this);
674       labTime.setText("Suggested␣Time:");
675       labDist = new TextView(this);
676       labDist.setText("Dist:");
677       fl2 = new FrameLayout(this);
678       LinearLayout linLayout = new LinearLayout(this);
679       linLayout.setOrientation(LinearLayout.VERTICAL);
680
681       linLayout.addView(labTime, 0, new FrameLayout.LayoutParams(
682           FrameLayout.LayoutParams.FILL_PARENT,
683           FrameLayout.LayoutParams.WRAP_CONTENT));
684       linLayout.addView(timeInput, 1, new FrameLayout.LayoutParams(
685           FrameLayout.LayoutParams.FILL_PARENT,
686           FrameLayout.LayoutParams.WRAP_CONTENT));
687       linLayout.addView(labDist, 2, new FrameLayout.LayoutParams(
688           FrameLayout.LayoutParams.FILL_PARENT,
689           FrameLayout.LayoutParams.WRAP_CONTENT));
690       linLayout.addView(distInput, 3, new FrameLayout.LayoutParams(
691           FrameLayout.LayoutParams.FILL_PARENT,
692           FrameLayout.LayoutParams.WRAP_CONTENT));
693       fl2.addView(linLayout);
694       ad2 = new AlertDialog.Builder(this).setView(fl2).setTitle("Parameters")
695           .setCancelable(false).setPositiveButton("OK",
696               new DialogInterface.OnClickListener() {
697                 public void onClick(DialogInterface arg0, int arg1) {
698                   try {
699                     minTimeForUpdates = Long
700                         .parseLong(timeInput.getText()
701                             .toString());
702                     minDistanceForUpdates = Float
703                         .parseFloat(distInput.getText()
704                             .toString());
705                     GPS.this.runOnUiThread(new Runnable() {
706
707                       @Override
708                       public void run() {
709                         if (GPS.this.listeningLocations == true) {
710                         lm.removeUpdates(locationListener);
711                         lm.requestLocationUpdates("gps",
712                             minTimeForUpdates,
713                             minDistanceForUpdates,
```

```
                            locationListener);
                    }
                    }

                  });
                } catch (Exception e) {

                }
              }
            }).create();
    // End AlertDialog

    // Bind to Notification DeleteService which detects the removal of
    // notification by the user
    this.bindService(new Intent(GPS.this, NotificationDeleteService.class),
        this.mConnection, Context.BIND_AUTO_CREATE);

    // Create a Location Listener
    locationListener = new MyLocationListener();

    // Set Touch Event for MapVew
    mapView = (MapView) findViewById(R.id.mapview1);
    mapView.setOnTouchListener(new OnTouchListener() {

      @Override
      public boolean onTouch(View arg0, MotionEvent arg1) {
        if (arg1.getAction() == MotionEvent.ACTION_DOWN) {

          int x = (int) arg1.getX();
          int y = (int) arg1.getY();
          lastPointClicked = mapView.getProjection().fromPixels(x, y);
          Toast.makeText(
              GPS.this,
              "You␣have␣clicked␣on␣the␣point␣with␣Lon:␣"
                  + lastPointClicked.getLongitudeE6()
                  / ((double) 1E6) + "␣and␣Lat:␣"
                  + lastPointClicked.getLatitudeE6()
                  / ((double) 1E6), Toast.LENGTH_SHORT)
              .show();
        }
        return false;
      }

    });
    // Get map Controller
    mc = mapView.getController();

    // get imageView for displaying the status
    gpsStatus = (ImageView) this.findViewById(R.id.gpsImage);

    // Get the overlay list
    overlaylist = mapView.getOverlays();

    // Create an overlay for the current location
    myLocOver = new myOverlay(10, true, this);
    myLocOver.setDescription("Current␣Location");
    myLocOver.setDistance(0);
    myLocOver.setUrl(null);

    // Create an overlay for the Radius Circle
    myLocOverRadius = new myOverlay(Radius, false, this);
    myLocOverRadius.setDescription("Radius␣of␣Poi␣Search");
    myLocOverRadius.setDistance(Radius);
    myLocOverRadius.setUrl(null);

    myLocOverRadius.getPaint().setStyle(Style.STROKE);

    poiOver = new myOverlay(10, true, this);
    poiOver.getPaint().setARGB(101, 0, 255, 0);
    // overlaylist.add(myLocOver);
    // overlaylist.add(poiOver);

    // Get The ZoomControls and set the click listeners
    zc = (ZoomControls) this.findViewById(R.id.zoomcontrols);

    zc.setOnZoomInClickListener(new OnClickListener() {

      @Override
      public void onClick(View arg0) {
        mc.zoomIn();
        mapView.invalidate();
      }

```

```
797       });
798       zc.setOnZoomOutClickListener(new OnClickListener() {
799
800         @Override
801         public void onClick(View arg0) {
802           mc.zoomOut();
803           mapView.invalidate();
804         }
805
806       });
807
808       // Create a dbadapter for local database access
809       dbadapter = new DbAdapter(this);
810       dbadapter.open();
811
812       // Initializing the setting manager
813       settingManager = new SettingManager(this);
814
815       // Initializing the connector
816       remoteConnector = new HttpConnector(0);
817
818       // Setting the title
819       this.setTitle("Location␣GPS␣-␣Local␣Mode");
820
821     }
822
823     @Override
824     protected boolean isRouteDisplayed() {
825
826       return false;
827     }
828
829     /**
830      * A Extension of the LocationListener class that takes action on location
831      * changes
832      *
833      * @author Stavros
834      *
835      */
836
837     LocationListener secondaryListener = new LocationListener() {
838       @Override
839       public void onLocationChanged(Location loc) {
840
841       }
842
843       @Override
844       public void onProviderDisabled(String provider) {
845
846       }
847
848       @Override
849       public void onProviderEnabled(String provider) {
850
851       }
852
853       @Override
854       public void onStatusChanged(String provider, int status, Bundle extras) {
855
856       }
857     };
858
859     private class MyLocationListener implements LocationListener {
860       /**
861        * Called when new location data is available
862        */
863       DescribedLocation lastLoc = null;
864
865       @Override
866       public void onLocationChanged(Location loc) {
867         DescribedLocation dl = findClosestPointOfInterest(loc);
868         if (dl.loc == null)
869           dl.loc = loc; // If no point of interest found
870         // If we have a new Location or notifications have been deleted
871         if (!dl.equals(lastLoc)
872             || ((notificationBinder != null) && (notificationBinder
873                 .hasBeenDeleted()))) {
874           GPS.this.nm.cancel(GPS.this.NOTIFICATION_ID);
875           if (dl.url != null) {
876             GPS.this.showNotification(dl.url, dl.descr);
877           }
878         }
879         lastLoc = dl;
```

```
880         String closestPointOfInterest = dl.descr;
881         if (closestPointOfInterest == null)
882           closestPointOfInterest = "No␣Point␣Of␣Interest";
883         // Set the overlay coord for the my position circle to be drawn
884
885         myLocOver.setCoord(loc.getLongitude(), loc.getLatitude());
886         myLocOverRadius.setCoord(loc.getLongitude(), loc.getLatitude());
887         myLocOverRadius.setRad(Radius);
888
889         overlaylist.add(myLocOver);
890         overlaylist.add(myLocOverRadius);
891         // Show a message containing info
892         Toast.makeText(
893             GPS.this,
894             "Location␣changed␣:␣Lat:␣" + loc.getLatitude() + "␣Lng:␣"
895                 + loc.getLongitude()
896                 + "\nThe␣closest␣point␣of␣interest␣is␣"
897                 + closestPointOfInterest, Toast.LENGTH_LONG).show();
898
899         // Center the map at the middle of the current location and the
900         // PoI location
901         // If PoI location does not exist center the map to the
902         // currentlocation
903         GeoPoint p = new GeoPoint((int) (loc.getLatitude() * 1E6),
904             (int) (loc.getLongitude() * 1E6));
905         final Location l1 = loc;
906         final Location l2 = dl.loc;
907         mc.animateTo(p, new Runnable() {
908
909           @Override
910           public void run() {
911             calculateZoom(l1, l2); // Calculate and apply the
912             // correct zoom
913           }
914
915         });
916
917         mapView.invalidate();
918
919       }
920
921       @Override
922       public void onProviderDisabled(String provider) {
923
924       }
925
926       @Override
927       public void onProviderEnabled(String provider) {
928
929       }
930
931       @Override
932       public void onStatusChanged(String provider, int status, Bundle extras) {
933
934       }
935     }
936
937     /**
938      * This function calculates and applies the correct zoom level in order both
939      * loc and loc2 points to be visible
940      *
941      * @param loc:
942      *            The first location
943      * @param loc2:
944      *            The second location
945      */
946     private void calculateZoom(Location loc, Location loc2) {
947       // calculate the radius of the zoom
948       float Radius;
949       if (loc2.distanceTo(loc) > this.Radius)
950         Radius = loc2.distanceTo(loc);
951       else
952         Radius = this.Radius;
953       // Find the maximum zoomlevel and set the map to it
954       int zoomlevel = mapView.getMaxZoomLevel();
955       mc.setZoom(zoomlevel);
956
957       // This loop finds if the points are in the mapView window
958       // If one of the isn't then we Zoom out one level
959       do {
960
961         // Get the projection for the calculation of the points
962         Projection proj = mapView.getProjection();
```

```
963         int maxDim = mapView.getWidth();
964
965         if (proj.metersToEquatorPixels(Radius)
966             * (1 / (float) Math.cos(Math.toRadians(loc.getLatitude())))) > (maxDim / 2)) {
967
968           mc.setZoom(--zoomlevel);
969           continue;
970         }
971         break;
972         /*
973          * // Find the Points from GeoPoints Point p1 = proj.toPixels(gp1,
974          * null); Point p2 = proj.toPixels(gp2, null); // Determine if they
975          * can fit in width if ((p1.x > maxDim) || (p1.x < 0)) {
976          * mc.setZoom(--zoomlevel); continue; } if ((p2.x > maxDim) || (p2.x <
977          * 0)) { mc.setZoom(--zoomlevel); continue; } // Determine if they
978          * can fit in height maxDim = mapView.getHeight(); if ((p1.y >
979          * maxDim) || (p1.y < 0)) { mc.setZoom(--zoomlevel); continue; } if
980          * ((p2.y > maxDim) || (p2.y < 0)) { mc.setZoom(--zoomlevel);
981          * continue; } break;
982          */
983
984       } while (true);
985       // invalidate the map in order to force redraw
986       mapView.invalidate();
987     }
988
989     /**
990      * A simple class just to return both a string and a Location from one
991      * function
992      *
993      * @author Stavros
994      *
995      */
996
997     /**
998      * This function finds and returns the closest local PoI to the given
999      * Location loc
1000     *
1001     * @param loc:
1002     *            The given location
1003     * @return A DescribedLocation object containing all the info for the
1004     *         closest PoI
1005     */
1006    private DescribedLocation findClosestLocalPointOfInterest(Location loc) {
1007
1008      // get all the points from the database
1009      Cursor cur = dbadapter.fetchAllPoints();
1010
1011      // Check if the database has elements
1012      if (!cur.moveToFirst()) {
1013        overlaylist.clear();
1014        poiOver.setCoord(null, null); // if no PoI found set its overlay
1015        // coord to null
1016        return new DescribedLocation(null, "no␣Point␣of␣Interest", 0, null);
1017      }
1018
1019      // Some declarations
1020      Location poi;
1021      String provider = loc.getProvider();
1022      String retDesc = "no␣Point␣of␣Interest";
1023      Location retLoc = null;
1024      float distance = -1, tempDist;
1025      double longt, lat;
1026      String descr;
1027      int i = 0, pos = -1;
1028      // Examine each PoI retrieved from the database
1029      // Save the closest to the given location
1030      // create overlays for the rests within the radius
1031      overlaylist.clear();
1032      while (!cur.isAfterLast()) {
1033        longt = cur.getDouble(cur.getColumnIndex(DbAdapter.KEY_LONG));
1034        lat = cur.getDouble(cur.getColumnIndex(DbAdapter.KEY_LAT));
1035        descr = cur.getString(cur.getColumnIndex(DbAdapter.KEY_DESC));
1036        poi = new Location(provider);
1037        poi.setLatitude(lat);
1038        poi.setLongitude(longt);
1039        tempDist = loc.distanceTo(poi);
1040
1041        if ((tempDist < distance) || (distance == -1)) {
1042          distance = tempDist;
1043          retDesc = descr;
1044          retLoc = poi;
1045          pos = i;
```

```
1046          }
1047
1048          if (tempDist <= this.Radius) {
1049            poiOver = new myOverlay(10, true, this);
1050            poiOver.getPaint().setARGB(101, 255, 0, 0);
1051            poiOver.setCoord(longt, lat);
1052            poiOver.setDescription(descr);
1053            poiOver.setDistance(loc.distanceTo(poi));
1054            overlaylist.add(poiOver);
1055            i++;
1056          }
1057
1058          cur.moveToNext();
1059        }
1060
1061        if (distance > Radius) {
1062          poiOver.setCoord(retLoc.getLongitude(), retLoc.getLatitude());
1063          poiOver.setDescription(retDesc);
1064          poiOver.setDistance(distance);
1065          poiOver.setUrl(null);
1066          overlaylist.add(poiOver);
1067        }
1068
1069        if (pos >= 0)
1070          ((myOverlay) overlaylist.get(pos)).getPaint().setARGB(101, 0, 255,
1071              0);
1072        cur.close();
1073        return new DescribedLocation(retLoc, retDesc, distance, null);
1074
1075      }
1076
1077      private DescribedLocation findClosestPointOfInterest(Location loc) {
1078        if (this.pointsMode == GPS.LOCAL_MODE)
1079          return findClosestLocalPointOfInterest(loc);
1080        else
1081          return findClosestRemotePointOfInterest(loc);
1082      }
1083
1084      /**
1085       * Finds the closest location (on the web server) and returns it
1086       *
1087       * @param loc
1088       * @return
1089       */
1090      private DescribedLocation findClosestRemotePointOfInterest(Location loc) {
1091        // Some Communication takes place here
1092        Vector<String> params = new Vector<String>();
1093
1094        params.add(Integer.toString(settingManager.getUserId()));
1095        if (this.includebuddies) {
1096          Iterator<String> it = buddies.iterator();
1097          while (it.hasNext()) {
1098            params.add(it.next());
1099          }
1100        }
1101        ClosestLocations cl = remoteConnector.createClosestLocations(params,
1102            settingManager.getUsername(), settingManager.getPassword(),
1103            Double.toString(loc.getLongitude()), Double.toString(loc
1104                .getLatitude()), Float.toString(Radius));
1105
1106        cl.findClosestLocations();
1107
1108        if (cl.connected == false) {
1109          changeMode();
1110          new AlertDialog.Builder(GPS.this).setMessage("Connection␣Lost")
1111              .setNeutralButton("OK",
1112                  new DialogInterface.OnClickListener() {
1113                    @Override
1114                    public void onClick(DialogInterface arg0,
1115                        int arg1) {
1116
1117                      arg0.dismiss();
1118                    }
1119                  }).setTitle("Error").show();
1120          return this.findClosestLocalPointOfInterest(loc);
1121        }
1122
1123        Iterator<DescribedLocation> it = cl.results.iterator();
1124        DescribedLocation dl = null;
1125        overlaylist.clear();
1126        int pos = -1, i = 0;
1127        DescribedLocation retdl = new DescribedLocation(null, null, -1, null);
1128        while (it.hasNext()) { // here we can find the closest
```

```
1129        dl = it.next();
1130        poiOver = new myOverlay(10, true, this);
1131        poiOver.getPaint().setARGB(101, 255, 0, 0);
1132        if ((retdl.distance == -1) || (dl.distance < retdl.distance)) {
1133          retdl = dl;
1134          pos = i;
1135        }
1136        poiOver.setCoord(dl.loc.getLongitude(), dl.loc.getLatitude());
1137        poiOver.setDescription(dl.descr);
1138        poiOver.setDistance(dl.distance);
1139        poiOver.setUrl(dl.url);
1140        overlaylist.add(poiOver);
1141        i++;
1142      }
1143      if (pos >= 0)
1144        ((myOverlay) overlaylist.get(pos)).getPaint().setARGB(101, 0, 255,
1145            0);
1146      return retdl;
1147    }
1148
1149    @Override
1150    protected void onPause() {
1151      lm.removeUpdates(locationListener); // Stop listening to the
1152      // LocationListener
1153
1154      // Save the current state of the GPS-Map
1155      // That includes Radius,REMOTE-LOCAL_MODE, Include Buddies
1156      SharedPreferences.Editor editor = getPreferences(0).edit();
1157      editor.putFloat("Radius", this.Radius);
1158      editor.putBoolean("includeBuddies", this.includebuddies);
1159      editor.putInt("pointsMode", this.pointsMode);
1160      editor.putBoolean("listeningLocations", listeningLocations);
1161      editor.putLong("minTimeForUpdates", minTimeForUpdates);
1162      editor.putFloat("minDistanceForUpdates", minDistanceForUpdates);
1163      editor.commit();
1164      super.onPause();
1165    }
1166
1167    @Override
1168    protected void onResume() {
1169      // Stop any running Services
1170      stopService(new Intent(GPS.this, LocationService.class));
1171      stopService(new Intent(GPS.this, LocationTripService.class));
1172
1173      if (!settingManager.retrieveSettings())
1174        Toast.makeText(this, "Could␣not␣open␣the␣configuration␣file",
1175            Toast.LENGTH_SHORT).show();
1176
1177      remoteConnector.setTimeout(settingManager.getTimeout());
1178      remoteConnector.setBaseUrl(settingManager.getWebserver());
1179
1180      // retrieve Saved Settings
1181      SharedPreferences prefs = getPreferences(0);
1182      int pointsModeRetrieved;
1183      if (prefs != null) {
1184        this.Radius = prefs.getFloat("Radius", 500);
1185        this.includebuddies = prefs.getBoolean("includeBuddies", false);
1186        this.listeningLocations = prefs.getBoolean("listeningLocations",
1187            true);
1188        pointsModeRetrieved = prefs.getInt("pointsMode", GPS.LOCAL_MODE);
1189        this.minTimeForUpdates = prefs.getLong("minTimeForUpdates", 0);
1190        this.minDistanceForUpdates = prefs.getFloat(
1191            "minDistanceForUpdates", 0);
1192      } else {
1193        this.Radius = 500;
1194        this.includebuddies = false;
1195        this.listeningLocations = true;
1196        this.minTimeForUpdates = 0;
1197        this.minDistanceForUpdates = 0;
1198        pointsModeRetrieved = GPS.LOCAL_MODE;
1199      }
1200
1201      timeInput.setText("" + minTimeForUpdates);
1202      distInput.setText("" + minDistanceForUpdates);
1203
1204      if (!listeningLocations)
1205        this.gpsStatus.setBackgroundDrawable(null);
1206      final boolean ll = listeningLocations;
1207      if (this.pointsMode != pointsModeRetrieved)
1208        new Thread() {
1209
1210          @Override
1211          public void run() {
```

```
1212            GPS.this.changeMode();
1213            // request the updates here
1214            if (ll)
1215              GPS.this.runOnUiThread(new Runnable() {
1216
1217                @Override
1218                public void run() {
1219                  // TODO Auto-generated method stub
1220                  lm
1221                      .requestLocationUpdates(
1222                          LocationManager.GPS_PROVIDER,
1223                          minTimeForUpdates,
1224                          minDistanceForUpdates,
1225                          locationListener);
1226                }
1227
1228              });
1229            // Start listening
1230            // to the
1231            // LocationListener
1232
1233          }
1234
1235        }.start();
1236      else if (ll) {
1237        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
1238            minTimeForUpdates, minDistanceForUpdates, locationListener); // Start
1239        // listening
1240        // to
1241        // the
1242
1243      }
1244      super.onResume();
1245    }
1246
1247    @Override
1248    protected void onDestroy() {
1249      dbadapter.close();
1250      unbindService(mConnection);
1251      super.onDestroy();
1252    }
1253
1254    private void showNotification(String url, String msg) {
1255      Notification notification = new Notification(R.drawable.webpage,
1256          "Location␣Web␣Info", System.currentTimeMillis());
1257
1258      if (!url.startsWith("http://"))
1259        url = "http://" + url;
1260
1261      PendingIntent pi = PendingIntent.getActivity(this, 1, new Intent(
1262          Intent.ACTION_VIEW, Uri.parse(url)), 0);
1263
1264      notification.setLatestEventInfo(this, "Location␣Web␣Data", msg, pi);
1265
1266      notification.deleteIntent = PendingIntent.getService(this, 1,
1267          (new Intent(GPS.this, NotificationDeleteService.class))
1268              .setAction("gr.ntua.cn.GPS.NotificationDeleted"), 1);
1269      nm.notify(this.NOTIFICATION_ID, notification);
1270    }
1271
1272    // Code for binding to the NotificationDeleteService
1273    private NotificationDeleteService.customBinder notificationBinder = null;
1274    private ServiceConnection mConnection = new ServiceConnection() {
1275
1276      @Override
1277      public void onServiceConnected(ComponentName name, IBinder service) {
1278        GPS.this.notificationBinder = (NotificationDeleteService.customBinder) service;
1279      }
1280
1281      @Override
1282      public void onServiceDisconnected(ComponentName arg0) {
1283        GPS.this.notificationBinder = null;
1284
1285      }
1286    };
1287
1288  }
```

AddRemotePoint.java-AddPoint.java

Παραθέτουμε το AddRemotePoint.java, το AddPoint.java είναι ανάλογο:

```
1  package gr.ntua.cn.GPS;
2
3  import gr.ntua.cn.GPS.Auxiliary.DbAdapter;
4  import gr.ntua.cn.GPS.Auxiliary.HttpConnector;
5  import gr.ntua.cn.GPS.Auxiliary.SettingManager;
6  import gr.ntua.cn.GPS.Auxiliary.HttpConnector.ImageSender;
7  import gr.ntua.cn.GPS.Auxiliary.HttpConnector.PointAdder;
8  import gr.ntua.cn.GPS.ImageManagement.picGallery;
9  import android.app.Activity;
10 import android.app.ProgressDialog;
11 import android.content.Intent;
12 import android.os.Bundle;
13 import android.view.View;
14 import android.view.View.OnClickListener;
15 import android.widget.Button;
16 import android.widget.CheckBox;
17 import android.widget.EditText;
18
19 /**
20  * The AddPoint class is used to collect the required data for a <b>PoI</b> to
21  * be created or updated
22  *
23  * @author Stavros
24  */
25 public class AddRemotePoint extends Activity {
26   private EditText descr, longt, lat, urltext;
27   private Button bAdd;
28   private Button bCancel;
29   private Button bUploadImage;
30   private int editRowId;
31   private HttpConnector remoteConnector;
32   private SettingManager settingManager;
33   private CheckBox used;
34   private CheckBox Public;
35
36   @Override
37   protected void onCreate(Bundle savedInstanceState) {
38     super.onCreate(savedInstanceState);
39     this.setContentView(R.layout.addremotepoint);
40     this.setResult(Activity.RESULT_OK);
41     this.setTitle("Edit␣Point␣Of␣Interest");
42     descr = (EditText) findViewById(R.id.description);
43     longt = (EditText) findViewById(R.id.longt);
44     lat = (EditText) findViewById(R.id.latt);
45     urltext = (EditText) findViewById(R.id.urltext);
46     used = (CheckBox) findViewById(R.id.usedcheckbox);
47     Public = (CheckBox) findViewById(R.id.publiccheckbox);
48     bAdd = (Button) findViewById(R.id.bAdd);
49     bCancel = (Button) findViewById(R.id.bCancel);
50     bUploadImage = (Button) findViewById(R.id.bUpload);
51     // Open the database
52
53     // Set the onClick Listeners for the buttons
54     bAdd.setOnClickListener(new OnClickListener() {
55
56       @Override
57       public void onClick(View arg0) {
58
59         // Some code for input checking must be added here
60         final Intent i = AddRemotePoint.this.getIntent();
61
62         // The communication may take time.
63         // So display message
64         // and run the addition in a new thread
65         final ProgressDialog pd = ProgressDialog.show(
66             AddRemotePoint.this, "Please␣Wait", "Sending␣Data...");
67         pd.setCancelable(false);
68
69         new Thread() {
70
71           @Override
72           public void run() {
73
74             // Check the action of the intent
75             // If it is ADD createPoint
76             if (i.getAction().equals("gr.ntua.cn.GPS.ADD")) {
77               try {
78                 String urlStr = urltext.getText().toString()
79                     .equalsIgnoreCase("") ? null : urltext
80                     .getText().toString();
81                 PointAdder pa = remoteConnector
82                     .createPointAdder(null, Integer
83                         .toString(settingManager
```

```
                        .getUserId()),
                    settingManager.getUsername(),
                    settingManager.getPassword(),
                    longt.getText().toString(), lat
                        .getText().toString(),
                    descr.getText().toString(),
                    Boolean.toString(used
                        .isChecked()), Boolean
                        .toString(Public
                            .isChecked()),
                    urlStr, PointAdder.ADD_MODE);
            pa.addPoint();

            if (pa.connected == false) {
              AddRemotePoint.this
                  .setResult(Activity.RESULT_CANCELED);
              AddRemotePoint.this.finish();

            } else {
              AddRemotePoint.this
                  .setResult(Activity.RESULT_OK);
              AddRemotePoint.this.finish();
            }
          } catch (Exception e) {

          }

        }
        // Else if it is EDIT updatePoint
        else if (i.getAction().equals("gr.ntua.cn.GPS.EDIT")) {

          try {
            String urlStr = urltext.getText().toString()
                .equalsIgnoreCase("") ? null : urltext
                .getText().toString();
            PointAdder pa = remoteConnector
                .createPointAdder(Integer
                    .toString(editRowId), Integer
                    .toString(settingManager
                        .getUserId()),
                    settingManager.getUsername(),
                    settingManager.getPassword(),
                    longt.getText().toString(), lat
                        .getText().toString(),
                    descr.getText().toString(),
                    Boolean.toString(used
                        .isChecked()), Boolean
                        .toString(Public
                            .isChecked()),
                    urlStr, PointAdder.UPDATE_MODE);
            pa.addPoint();

            if (pa.connected == false) {
              AddRemotePoint.this
                  .setResult(Activity.RESULT_CANCELED);
              AddRemotePoint.this.finish();

            } else {
              AddRemotePoint.this
                  .setResult(Activity.RESULT_OK);
              AddRemotePoint.this.finish();
            }
          } catch (Exception e) {

          }

        }
        // Else if it is a simple addition
        else {
          try {
            String urlStr = urltext.getText().toString()
                .equalsIgnoreCase("") ? null : urltext
                .getText().toString();
            PointAdder pa = remoteConnector
                .createPointAdder(null, Integer
                    .toString(settingManager
                        .getUserId()),
                    settingManager.getUsername(),
                    settingManager.getPassword(),
                    longt.getText().toString(), lat
                        .getText().toString(),
                    descr.getText().toString(),
                    Boolean.toString(used
```

```
                                .isChecked()), Boolean
                                .toString(Public
                                    .isChecked()),
                            urlStr, PointAdder.ADD_MODE);
                    pa.addPoint();

                    if (pa.connected == false) {
                      AddRemotePoint.this
                          .setResult(Activity.RESULT_CANCELED);
                      AddRemotePoint.this.finish();

                    } else {
                      AddRemotePoint.this
                          .setResult(Activity.RESULT_OK);
                      AddRemotePoint.this.finish();
                    }
                  } catch (Exception e) {

                  }

                }

                pd.dismiss();
              }

            }.start();

          }

        });

        bCancel.setOnClickListener(new OnClickListener() {

          @Override
          public void onClick(View arg0) {
            // Finish this activity
            AddRemotePoint.this.finish();
          }

        });

        bUploadImage.setOnClickListener(new OnClickListener() {

          @Override
          public void onClick(View arg0) {
            Intent i = new Intent(AddRemotePoint.this, picGallery.class);
            i.putExtra("returnvalue", true);
            startActivityForResult(i, 1);

          }

        });
        // Check the source of the Intent and initialize the text fields
        // accordingly
        Intent i = getIntent();

        // First Get some things that are surely in the intent
        settingManager = (SettingManager) i
            .getParcelableExtra("settingmanager");
        remoteConnector = new HttpConnector(settingManager.getTimeout());
        remoteConnector.setBaseUrl(settingManager.getWebserver());

        this.setTitle("Add␣Point␣to␣Web␣Server");
        editRowId = -1;
        if (i.getAction().equals("gr.ntua.cn.GPS.ADD")
            || i.getAction().equals("gr.ntua.cn.GPS.EDIT")) {
          Double longt = i.getDoubleExtra(DbAdapter.KEY_LONG, 0);
          Double lat = i.getDoubleExtra(DbAdapter.KEY_LAT, 0);
          boolean used = i.getBooleanExtra("Used", false);
          boolean Public = i.getBooleanExtra("Public", false);
          String descr = i.getStringExtra(DbAdapter.KEY_DESC);
          String urlStr = i.getStringExtra("Url");
          editRowId = i.getIntExtra(DbAdapter.KEY_ROWID, -1);
          if (descr == null)
            descr = "";
          this.longt.setText(longt.toString());
          this.lat.setText(lat.toString());
          this.descr.setText(descr);
          this.used.setChecked(used);
          this.Public.setChecked(Public);
          if (urlStr == null)
            urlStr = "";
          if (!urlStr.equalsIgnoreCase("null"))
```

```
250           this.urltext.setText(urlStr);
251         else
252           this.urltext.setText("");
253       }
254       if (editRowId == -1)
255         bUploadImage.setVisibility(View.INVISIBLE);
256       else
257         bUploadImage.setVisibility(View.VISIBLE);
258     }
259
260     @Override
261     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
262       if ((requestCode == 1) && (resultCode == Activity.RESULT_OK)) {
263         final String filename = data.getStringExtra("filename");
264
265         final ProgressDialog pd = ProgressDialog.show(AddRemotePoint.this,
266             "Please␣Wait", "Uploading␣To␣Server...");
267         pd.setCancelable(false);
268
269         new Thread() {
270
271           @Override
272           public void run() {
273             ImageSender is = remoteConnector.createImageSender(
274                 filename, Integer.toString(settingManager
275                     .getUserId()),settingManager.getUsername(),settingManager.getPassword(), Integer.toString(
                            editRowId));
276             is.sendImage();
277
278             AddRemotePoint.this.runOnUiThread(new Runnable() {
279
280               @Override
281               public void run() {
282                 AddRemotePoint.this.urltext.setText(settingManager
283                     .getWebserver()
284                     + "/userdata/"
285                     + settingManager.getUserId()
286                     + "/" + editRowId + ".jpg");
287
288               }
289
290             });
291             pd.dismiss();
292           }
293
294         }.start();
295
296       }
297     }
298
299     @Override
300     protected void onDestroy() {
301
302       super.onDestroy();
303     }
304
305 }
```

RemotePointList.java-PointList.java-BuddyPointList.java

Παραθέτουμε το RemotePointList.java, τα PointList.java και BuddyPointList.java είναι ανάλογα:

```
1  package gr.ntua.cn.GPS;
2
3  import gr.ntua.cn.GPS.Auxiliary.DbAdapter;
4  import gr.ntua.cn.GPS.Auxiliary.HttpConnector;
5  import gr.ntua.cn.GPS.Auxiliary.SettingManager;
6  import gr.ntua.cn.GPS.Auxiliary.HttpConnector.PointAdder;
7  import gr.ntua.cn.GPS.Auxiliary.HttpConnector.PointGetter;
8
9  import java.io.IOException;
10 import java.util.HashMap;
11 import java.util.List;
12 import java.util.Map;
13 import java.util.Vector;
14
15 import javax.xml.parsers.DocumentBuilder;
```

```
16  import javax.xml.parsers.DocumentBuilderFactory;
17  import javax.xml.parsers.ParserConfigurationException;
18
19  import org.w3c.dom.Document;
20  import org.w3c.dom.Node;
21  import org.w3c.dom.NodeList;
22  import org.xml.sax.SAXException;
23
24  import android.app.Activity;
25  import android.app.ListActivity;
26  import android.app.ProgressDialog;
27  import android.content.Intent;
28  import android.net.Uri;
29  import android.os.Bundle;
30  import android.view.Menu;
31  import android.view.MenuItem;
32  import android.view.View;
33  import android.view.View.OnClickListener;
34  import android.widget.CheckBox;
35  import android.widget.ListView;
36  import android.widget.SimpleAdapter;
37  import android.widget.TextView;
38  import android.widget.SimpleAdapter.ViewBinder;
39
40  /**
41   * An activity that extends the ListActivity. Actually it is a list of all the
42   * PoIs with a menu for manipulation of them
43   *
44   * @author Stavros
45   *
46   */
47  public class RemotePointList extends ListActivity {
48    /**
49     * The ListView of the activity
50     */
51    private ListView lv;
52    /**
53     * A list that saves the data
54     */
55    private List<Map<String, String>> data;
56
57    private HttpConnector remoteConnector;
58    private SettingManager settingManager;
59    private ViewBinder viewBinder;
60
61    @Override
62    protected void onCreate(Bundle savedInstanceState) {
63
64      setContentView(R.layout.poi);
65      this.setResult(Activity.RESULT_OK);
66      lv = (ListView) findViewById(android.R.id.list);
67
68      // initialize list
69
70      this.setTitle("Remote␣List␣Of␣Points␣Of␣Interest");
71
72      // initializing the HttpConnector
73      remoteConnector = new HttpConnector(0);
74      super.onCreate(savedInstanceState);
75
76      // new View Binder
77      viewBinder = new ViewBinder() {
78        public boolean setViewValue(View view, Object data,
79            String textRepresentation) {
80          switch (view.getId()) {
81          case R.id.listcheckbox:
82            ((CheckBox) view).setChecked(Boolean
83                .parseBoolean((String) data));
84            return true;
85          case R.id.publiccheckbox:
86            ((CheckBox) view).setChecked(Boolean
87                .parseBoolean((String) data));
88            return true;
89          case R.id.urltext:
90            if (!((String)data).equalsIgnoreCase("null")) {
91            ((TextView) view).setText("WebPage");
92            ((TextView) view).setTag(data);
93            final String data2 = (String) data;
94            view.setOnClickListener(new OnClickListener() {
95
96              @Override
97              public void onClick(View arg0) {
98                String data3 = data2;
```

```
99                if (!((String) data3).startsWith("http://"))
100                 data3 = "http://" + data3;
101
102               Intent i = new Intent(
103                   Intent.ACTION_VIEW, Uri.parse(data3));
104               startActivity(i);
105
106             }
107
108           });
109
110           }
111           else {
112             ((TextView) view).setText("No WebPage");
113           }
114           return true;
115         default:
116           return false;
117         }
118
119       }
120     };
121
122   }
123
124   /**
125    * this function is called to populate the List with Elements containing the
126    * PoI info
127    */
128   private void populatelist() {
129     // Get all of the rows from the remote database and create the item list
130
131     // New List
132     data = new Vector<Map<String, String>>();
133
134     // Get The Points
135     PointGetter pg = remoteConnector.createPointGetter(settingManager
136         .getUserId(),settingManager.getUsername(),settingManager.getPassword());
137     pg.getPoints();
138
139     // If nothing went wrong
140     if (pg.xmlPointStream != null) {
141
142       try {
143         DocumentBuilderFactory dbf = DocumentBuilderFactory
144             .newInstance();
145         DocumentBuilder db = dbf.newDocumentBuilder();
146         Document doc = db.parse(pg.xmlPointStream);
147         if (!doc.getElementsByTagName("Status").item(0).getFirstChild()
148             .getNodeValue().equals("OK")) {
149           RemotePointList.this.setResult(Activity.RESULT_CANCELED);
150           RemotePointList.this.finish();
151           return;
152         }
153
154         NodeList nodelist = doc.getElementsByTagName("Point");
155         Node iNode, icNode;
156         int length = nodelist.getLength();
157         for (int i = 0; i < length; i++) {
158           HashMap<String, String> tempMap = new HashMap<String, String>();
159           iNode = nodelist.item(i);
160           NodeList children = iNode.getChildNodes();
161
162           int cLength = children.getLength();
163           for (int j = 0; j < cLength; j++) {
164             icNode = children.item(j);
165             if (icNode.getNodeType() == Node.ELEMENT_NODE) {
166               NodeList texts = icNode.getChildNodes();
167               for (int k = 0; k < texts.getLength(); k++) {
168                 String val = texts.item(k).getNodeValue();
169
170                 String name = icNode.getNodeName();
171                 tempMap.put(name, val);
172               }
173             }
174           }
175           data.add(tempMap);
176         }
177       } catch (ParserConfigurationException e) {
178         RemotePointList.this.setResult(Activity.RESULT_CANCELED);
179         RemotePointList.this.finish();
180         return;
181       } catch (IOException e) {
```

```
182           RemotePointList.this.setResult(Activity.RESULT_CANCELED);
183           RemotePointList.this.finish();
184           return;
185         } catch (SAXException e) {
186           RemotePointList.this.setResult(Activity.RESULT_CANCELED);
187           RemotePointList.this.finish();
188           return;
189         } catch (Exception e) {
190           RemotePointList.this.setResult(Activity.RESULT_CANCELED);
191           RemotePointList.this.finish();
192           return;
193         }
194
195       } else {
196         RemotePointList.this.setResult(Activity.RESULT_CANCELED);
197         RemotePointList.this.finish();
198         return;
199       }
200       // Create an array to specify the fields we want to display in the list
201
202       String[] from = new String[] { "pointId", "Longitude", "Latitude",
203           "Description", "Used", "Public","Url" };
204
205       // and an array of the fields we want to bind those fields to
206       // text4-id text2-Longitude text3 - Latitude text1 - Description
207       int[] to = new int[] { R.id.text4, R.id.text2, R.id.text3, R.id.text1,
208           R.id.listcheckbox, R.id.publiccheckbox, R.id.urltext };
209
210       // Now create a simple cursor adapter and set it to display
211       final SimpleAdapter points = new SimpleAdapter(this, data,
212           R.layout.remoterow, from, to);
213
214       points.setViewBinder(viewBinder);
215
216       RemotePointList.this.runOnUiThread(new Runnable() {
217
218         @Override
219         public void run() {
220
221           setListAdapter(points);
222         }
223
224       });
225
226     }
227
228     /**
229      * Creates the menu
230      */
231     @Override
232     public boolean onCreateOptionsMenu(Menu menu) {
233
234       menu.add(1, 1, 1, "Add␣Point").setIcon(R.drawable.addpoi);
235       menu.add(1, 2, 1, "Delete␣Point").setIcon(R.drawable.removepoi);
236       menu.add(1,3,1,"Navigate").setIcon(R.drawable.map);
237       return super.onCreateOptionsMenu(menu);
238     }
239
240     /**
241      * Called when a menu item selected
242      */
243     @SuppressWarnings("unchecked")
244     @Override
245     public boolean onOptionsItemSelected(MenuItem item) {
246
247       switch (item.getItemId()) {
248       case 3: // Navigate To Point
249         if (getSelectedItemPosition() < 0) break;
250         HashMap<String, String> y = (HashMap<String, String>) lv
251         .getAdapter().getItem(getSelectedItemPosition());
252         String longitude = y.get("Longitude");
253         String latitude = y.get("Latitude");
254         String descr = y.get("Description");
255         String url = y.get("Url");
256         Intent data = new Intent();
257         data.putExtra("Longitude", longitude);
258         data.putExtra("Latitude", latitude);
259         data.putExtra("Description", descr);
260         data.putExtra("Url", url);
261         this.setResult(3, data);
262         this.finish();
263         break;
264
```

```
265       case 1: // ADD POINT
266         Intent i = new Intent(RemotePointList.this, AddRemotePoint.class);
267         i.setAction("gr.ntua.cn.GPS.NEW");
268         i.putExtra("settingmanager", settingManager);
269         startActivityForResult(i, 1);
270         break;
271       case 2: // DELETE POINT
272         // Deleting may take time
273         // so lets display a message
274         // and run the deleting commands in a new thread
275         final ProgressDialog pd = ProgressDialog.show(RemotePointList.this,
276             "Please␣Wait", "Sending␣Data...");
277         pd.setCancelable(false);
278
279         new Thread() {
280
281           @Override
282           public void run() {
283             HashMap<String, String> y = (HashMap<String, String>) lv
284                 .getAdapter().getItem(getSelectedItemPosition());
285
286             String pointId = y.get("pointId");
287             PointAdder pa = remoteConnector.createPointAdder(pointId,Integer.toString(settingManager.getUserId
                    ()),settingManager.getUsername(),
288                 settingManager.getPassword(), null, null, null, null, null, null,
289                 PointAdder.DELETE_MODE);
290
291             pa.addPoint();
292
293             if (pa.connected == true)
294               populatelist();
295             else {
296               RemotePointList.this
297                   .setResult(Activity.RESULT_CANCELED);
298               RemotePointList.this.finish();
299             }
300             pd.dismiss();
301           }
302
303         }.start();
304
305         break;
306
307       }
308       return super.onOptionsItemSelected(item);
309     }
310
311     @Override
312     protected void onPause() {
313
314       super.onPause();
315     }
316
317     /**
318      * Overridden method which is called when a list item is clicked
319      */
320     @SuppressWarnings( { "unchecked", "unchecked" })
321     @Override
322     protected void onListItemClick(ListView l, View v, int position, long id) {
323       try {
324
325         HashMap<String, String> cur = (HashMap<String, String>) getListView()
326             .getAdapter().getItem(position);
327         int rowid = Integer.parseInt(cur.get("pointId"));
328         double longt = Double.parseDouble(cur.get("Longitude"));
329         double lat = Double.parseDouble(cur.get("Latitude"));
330         String descr = (String) cur.get("Description");
331         boolean used = Boolean.parseBoolean(cur.get("Used"));
332         boolean Public = Boolean.parseBoolean(cur.get("Public"));
333         String urlStr = cur.get("Url");
334         Intent i = new Intent(RemotePointList.this, AddRemotePoint.class);
335         i.setAction("gr.ntua.cn.GPS.EDIT");
336         i.putExtra(DbAdapter.KEY_LONG, longt);
337         i.putExtra(DbAdapter.KEY_LAT, lat);
338         i.putExtra(DbAdapter.KEY_DESC, descr);
339         i.putExtra(DbAdapter.KEY_ROWID, rowid);
340         i.putExtra("Used", used);
341         i.putExtra("Public", Public);
342         i.putExtra("Url", urlStr);
343         i.putExtra("settingmanager", settingManager);
344         startActivityForResult(i, 1);
345       } catch (Exception e) {
346         e.printStackTrace();
```

```
347       }
348       super.onListItemClick(l, v, position, id);
349
350     }
351
352     @Override
353     protected void onDestroy() {
354
355       super.onDestroy();
356     }
357
358     @Override
359     protected void onResume() {
360       Intent i = this.getIntent(); // Intent has the proper data for the
361       // httpconnector
362       settingManager = (SettingManager) i
363           .getParcelableExtra("settingmanager");
364       remoteConnector.setTimeout(settingManager.getTimeout());
365       remoteConnector.setBaseUrl(settingManager.getWebserver());
366
367       // Population may take some time
368       // its a good idea to do it in a separate thread
369       final ProgressDialog pd = ProgressDialog.show(RemotePointList.this,
370           "Please␣Wait", "Retrieving␣Data...");
371       pd.setCancelable(false);
372       new Thread() {
373
374         @Override
375         public void run() {
376           // TODO Auto-generated method stub
377           populatelist();
378           pd.dismiss();
379         }
380
381       }.start();
382
383       super.onResume();
384     }
385
386     @Override
387     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
388       switch (requestCode) {
389       case 1:
390         if (resultCode == Activity.RESULT_CANCELED) {
391           RemotePointList.this.setResult(Activity.RESULT_CANCELED);
392           RemotePointList.this.finish();
393         }
394         break;
395       }
396       super.onActivityResult(requestCode, resultCode, data);
397     }
398
399   }
```

Buddies.java

```
1   package gr.ntua.cn.GPS;
2
3   import gr.ntua.cn.GPS.Auxiliary.HttpConnector;
4   import gr.ntua.cn.GPS.Auxiliary.SettingManager;
5   import gr.ntua.cn.GPS.Auxiliary.HttpConnector.Authenticator;
6   import gr.ntua.cn.GPS.Auxiliary.HttpConnector.BuddyAdder;
7   import gr.ntua.cn.GPS.Auxiliary.HttpConnector.BuddyGetter;
8
9   import java.io.IOException;
10  import java.util.HashMap;
11  import java.util.List;
12  import java.util.Map;
13  import java.util.Vector;
14
15  import javax.xml.parsers.DocumentBuilder;
16  import javax.xml.parsers.DocumentBuilderFactory;
17  import javax.xml.parsers.ParserConfigurationException;
18
19  import org.w3c.dom.Document;
20  import org.w3c.dom.Node;
21  import org.w3c.dom.NodeList;
22  import org.xml.sax.SAXException;
23
```

```
24  import android.app.Activity;
25  import android.app.ListActivity;
26  import android.app.ProgressDialog;
27  import android.content.Intent;
28  import android.os.Bundle;
29  import android.util.Log;
30  import android.view.Menu;
31  import android.view.MenuItem;
32  import android.view.View;
33  import android.widget.ListView;
34  import android.widget.SimpleAdapter;
35
36  public class Buddies extends ListActivity {
37
38    private List<Map<String, String>> data;
39    private HttpConnector remoteConnector;
40    private SettingManager settingManager;
41    private int userId;
42
43    @Override
44    protected void onCreate(Bundle savedInstanceState) {
45      this.setResult(Activity.RESULT_OK);
46      this.setContentView(R.layout.buddies);
47
48      // initialize list
49
50      this.setTitle("Buddy␣List");
51
52      // initializing the HttpConnector
53      settingManager = new SettingManager(Buddies.this);
54      settingManager.retrieveSettings();
55      remoteConnector = new HttpConnector(settingManager.getTimeout());
56
57      super.onCreate(savedInstanceState);
58    }
59
60    private void populatelist() {
61      // Get all of the rows from the remote database and create the item list
62
63      // New List
64      data = new Vector<Map<String, String>>();
65
66      BuddyGetter bg = remoteConnector.createBuddyGetter(settingManager
67          .getUserId(), settingManager.getUsername(), settingManager
68          .getPassword());
69      bg.getBuddies();
70
71      if (bg.xmlBuddyStream != null) {
72
73        try {
74          DocumentBuilderFactory dbf = DocumentBuilderFactory
75              .newInstance();
76          DocumentBuilder db = dbf.newDocumentBuilder();
77          Document doc = db.parse(bg.xmlBuddyStream);
78          if (!doc.getElementsByTagName("Status").item(0).getFirstChild()
79              .getNodeValue().equals("OK")) {
80            Buddies.this.setResult(Activity.RESULT_CANCELED);
81            Buddies.this.finish();
82            return;
83          }
84
85          NodeList nodelist = doc.getElementsByTagName("Buddy");
86          Node iNode, icNode;
87          int length = nodelist.getLength();
88          for (int i = 0; i < length; i++) {
89            HashMap<String, String> tempMap = new HashMap<String, String>();
90            iNode = nodelist.item(i);
91            NodeList children = iNode.getChildNodes();
92
93            int cLength = children.getLength();
94            for (int j = 0; j < cLength; j++) {
95              icNode = children.item(j);
96              if (icNode.getNodeType() == Node.ELEMENT_NODE) {
97                NodeList texts = icNode.getChildNodes();
98                for (int k = 0; k < texts.getLength(); k++) {
99                  String val = texts.item(k).getNodeValue();
100                 String name = icNode.getNodeName();
101                 tempMap.put(name, val);
102               }
103             }
104           }
105           data.add(tempMap);
106         }
```

```
107         } catch (ParserConfigurationException e) {
108           Buddies.this.setResult(Activity.RESULT_CANCELED);
109           Buddies.this.finish();
110           return;
111         } catch (IOException e) {
112           Buddies.this.setResult(Activity.RESULT_CANCELED);
113           Buddies.this.finish();
114           return;
115         } catch (SAXException e) {
116           Buddies.this.setResult(Activity.RESULT_CANCELED);
117           Buddies.this.finish();
118           return;
119         } catch (Exception e) {
120           Buddies.this.setResult(Activity.RESULT_CANCELED);
121           Buddies.this.finish();
122           return;
123         }
124
125       } else {
126         Buddies.this.setResult(Activity.RESULT_CANCELED);
127         Buddies.this.finish();
128         return;
129       }
130
131       String[] from = new String[] { "buddyId", "Username", "Status" };
132       int[] to = new int[] { R.id.buddyid, R.id.buddyusername, R.id.status };
133
134       final SimpleAdapter buddies = new SimpleAdapter(this, data,
135           R.layout.buddyrow, from, to);
136
137       Buddies.this.runOnUiThread(new Runnable() {
138
139         @Override
140         public void run() {
141           setListAdapter(buddies);
142         }
143
144       });
145
146     }
147
148     @Override
149     public boolean onCreateOptionsMenu(Menu menu) {
150
151       menu.add(1, 1, 1, "Add␣Buddy").setIcon(R.drawable.addbuddy);
152       menu.add(1, 2, 1, "Delete␣Buddy").setIcon(R.drawable.removebuddy);
153       return super.onCreateOptionsMenu(menu);
154
155     }
156
157     @SuppressWarnings("unchecked")
158     @Override
159     public boolean onPrepareOptionsMenu(Menu menu) {
160       menu.removeItem(3);
161       int index = this.getSelectedItemPosition();
162       if (index < 0)
163         return true;
164
165       HashMap<String, String> cur = (HashMap<String, String>) this
166           .getListAdapter().getItem(index);
167
168       if ((cur.get("Status").equalsIgnoreCase("Requested")))
169         menu.add(1, 3, 1, "Accept").setIcon(R.drawable.acceptbuddy);
170
171       return super.onPrepareOptionsMenu(menu);
172     }
173
174     @SuppressWarnings("unchecked")
175     @Override
176     public boolean onOptionsItemSelected(MenuItem item) {
177       int index = this.getSelectedItemPosition();
178       if ((index < 0) && (item.getItemId() != 1))
179         return false;
180
181       HashMap<String, String> cur;
182
183       switch (item.getItemId()) {
184       case 1: // Add a Friend
185         Intent i = new Intent(Buddies.this, AddBuddy.class);
186         i.putExtra("settingmanager", settingManager);
187         startActivityForResult(i, 1);
188
189         break;
```

```
190       case 2: // Delete Friend
191         cur = (HashMap<String, String>) this.getListAdapter()
192             .getItem(index);
193         final String buddyId = cur.get("buddyId");
194
195         final ProgressDialog pd2 = ProgressDialog.show(Buddies.this,
196             "Please␣Wait", "Retrieving␣Data...");
197         pd2.setCancelable(false);
198
199         new Thread() {
200
201           @Override
202           public void run() {
203             BuddyAdder ba = remoteConnector.createBuddyAdder(Integer
204                 .toString(settingManager.getUserId()),
205                 settingManager.getUsername(), settingManager
206                     .getPassword(), buddyId,
207                 BuddyAdder.DELETE_MODE);
208             ba.addBuddy();
209             if (ba.connected == false) {
210               Buddies.this.setResult(Activity.RESULT_CANCELED);
211               Buddies.this.finish();
212               return;
213             }
214             Buddies.this.populatelist();
215             pd2.dismiss();
216           }
217         }.start();
218         break;
219       case 3: // If Accept Friend Clicked
220         cur = (HashMap<String, String>) this.getListAdapter()
221             .getItem(index);
222         final String buddyId2 = cur.get("buddyId");
223
224         // Progress Bar and Communication
225         final ProgressDialog pd3 = ProgressDialog.show(Buddies.this,
226             "Please␣Wait", "Retrieving␣Data...");
227         pd3.setCancelable(false);
228         new Thread() {
229
230           @Override
231           public void run() {
232             BuddyAdder ba = remoteConnector.createBuddyAdder(Integer
233                 .toString(settingManager.getUserId()),
234                 settingManager.getUsername(), settingManager
235                     .getPassword(), buddyId2,
236                 BuddyAdder.ACCEPT_MODE);
237             ba.addBuddy();
238             if (ba.connected == false) {
239               Log.d("Blim:", "MPIKA␣EDW␣MESA");
240               Buddies.this.setResult(Activity.RESULT_CANCELED);
241               Buddies.this.finish();
242               return;
243             }
244
245             Buddies.this.populatelist();
246             pd3.dismiss();
247           }
248
249         }.start();
250
251         break;
252       }
253       return false;
254     }
255
256     @SuppressWarnings("unchecked")
257     @Override
258     protected void onListItemClick(ListView l, View v, int position, long id) {
259       HashMap<String, String> cur = (HashMap<String, String>) this
260           .getListAdapter().getItem(position);
261       if (!cur.get("Status").equalsIgnoreCase("Friend"))
262         return;
263       Intent i = new Intent(Buddies.this, BuddyPointList.class);
264       String buddyId = cur.get("buddyId");
265       i.putExtra("buddyId", buddyId);
266       i.putExtra("settingmanager", settingManager);
267       startActivityForResult(i, 1);
268
269     }
270
271     @Override
272     protected void onResume() {
```

```
273
274       // httpconnector
275       if (!settingManager.retrieveSettings()) {
276         Buddies.this.setResult(Activity.RESULT_CANCELED);
277         Buddies.this.finish();
278       }
279
280       remoteConnector.setTimeout(settingManager.getTimeout());
281       remoteConnector.setBaseUrl(settingManager.getWebserver());
282
283       // Population may take some time
284       // its a good idea to do it in a separate thread
285       final ProgressDialog pd = ProgressDialog.show(Buddies.this,
286           "Please␣Wait", "Retrieving␣Data...");
287       pd.setCancelable(false);
288       new Thread() {
289
290         @Override
291         public void run() {
292           // Authenticate the user
293           Authenticator auth = remoteConnector.createAuthenticator(
294               settingManager.getUsername(), settingManager
295                   .getPassword());
296           auth.logUser();
297
298           if (!auth.isAuthenticated()) {
299             Buddies.this.finish();
300             return;
301           }
302
303           Buddies.this.userId = auth.userId;
304           settingManager.setUserId(userId);
305
306           populatelist();
307           pd.dismiss();
308         }
309
310       }.start();
311
312       super.onResume();
313     }
314
315   }
```

EntryMenu.java

```
1   package gr.ntua.cn.GPS;
2
3   import gr.ntua.cn.GPS.ImageManagement.picGallery;
4
5   import java.io.File;
6
7   import android.app.Activity;
8   import android.content.Intent;
9   import android.os.Bundle;
10  import android.view.View;
11  import android.view.View.OnClickListener;
12  import android.widget.Button;
13  import android.widget.Toast;
14
15  public class EntryMenu extends Activity {
16    private Button mapmenu;
17    private Button webservermenu;
18    private Button buddies;
19    private Button takepicture;
20
21    @Override
22    protected void onCreate(Bundle savedInstanceState) {
23      this.setContentView(R.layout.entry);
24      checkForDirectories();
25      super.onCreate(savedInstanceState);
26      mapmenu = (Button) this.findViewById(R.id.mapmenu);
27      mapmenu.setOnClickListener(new OnClickListener() {
28
29        @Override
30        public void onClick(View arg0) {
31          Intent i = new Intent(EntryMenu.this,GPS.class);
32          EntryMenu.this.startActivityForResult(i,3);
33        }
34
35      });
```

```
36
37      webservermenu = (Button) this.findViewById(R.id.webservermenu);
38      webservermenu.setOnClickListener(new OnClickListener() {
39
40        @Override
41        public void onClick(View arg0) {
42          Intent i = new Intent(EntryMenu.this,ServerMenu.class);
43          EntryMenu.this.startActivity(i);
44
45        }
46
47      });
48
49      buddies = (Button) this.findViewById(R.id.bBuddies);
50      buddies.setOnClickListener(new OnClickListener() {
51
52        @Override
53        public void onClick(View arg0) {
54          // TODO Auto-generated method stub
55          Intent i = new Intent(EntryMenu.this, Buddies.class);
56          startActivityForResult(i,1);
57        }
58
59      });
60
61      takepicture = (Button) this.findViewById(R.id.bTakePicture);
62      takepicture.setOnClickListener(new OnClickListener() {
63
64        @Override
65        public void onClick(View arg0) {
66          Intent i = new Intent(EntryMenu.this, picGallery.class);
67          startActivity(i);
68
69
70        }
71
72      });
73    }
74
75    private void checkForDirectories() {
76      File file = new File("/sdcard/android");
77      if (!file.exists()) {
78        Toast.makeText(this, "Creating␣Directories", Toast.LENGTH_SHORT).show();
79        if (!file.mkdir())
80          Toast.makeText(this, "Could␣not␣create␣directory", Toast.LENGTH_SHORT).show();
81      }
82
83    }
84
85    @Override
86    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
87      if ((requestCode == 3) && (resultCode == 3))
88        finish();
89      super.onActivityResult(requestCode, resultCode, data);
90    }
91
92
93
94
95
96  }
```

ServerMenu.java

```
1   package gr.ntua.cn.GPS;
2
3   import gr.ntua.cn.GPS.Auxiliary.HttpConnector;
4   import gr.ntua.cn.GPS.Auxiliary.SettingManager;
5
6   import java.io.FileNotFoundException;
7   import java.io.FileOutputStream;
8   import java.io.IOException;
9
10  import android.app.Activity;
11  import android.app.ProgressDialog;
12  import android.content.Context;
13  import android.os.Bundle;
14  import android.view.View;
15  import android.view.View.OnClickListener;
16  import android.widget.Button;
17  import android.widget.EditText;
```

```
18  import android.widget.Toast;
19
20  public class ServerMenu extends Activity {
21    private EditText uritext;
22    private EditText timeouttext;
23    private EditText usertext;
24    private EditText passtext;
25    private Button bSetUri;
26    private Button bTestServer;
27    private Button bTestAcount;
28
29    @Override
30    protected void onCreate(Bundle savedInstanceState) {
31      this.setContentView(R.layout.webservermenu);
32
33      uritext = (EditText) this.findViewById(R.id.webserverUri);
34      timeouttext = (EditText) this.findViewById(R.id.timeouttext);
35      usertext = (EditText) this.findViewById(R.id.usernametext);
36      passtext = (EditText) this.findViewById(R.id.passwordtext);
37
38      bSetUri = (Button) this.findViewById(R.id.bseturi);
39      bSetUri.setOnClickListener(new OnClickListener() {
40        FileOutputStream webserverini = null;
41
42        @Override
43        public void onClick(View arg0) {
44
45          try {
46            int timeout = Integer.parseInt(timeouttext.getText()
47                .toString());
48            if (timeout > 20000)
49              throw new Exception();
50            webserverini = ServerMenu.this.openFileOutput(
51                "webserver.ini", Context.MODE_PRIVATE);
52            String conffile = null;
53            conffile = "serveruri=" + uritext.getText().toString()
54                + "\ntimeout=" + timeouttext.getText().toString()
55                + "\nuser=" + usertext.getText().toString()
56                + "\npass=" + passtext.getText().toString();
57
58            webserverini.write(conffile.getBytes());
59            Toast.makeText(ServerMenu.this, "Settings␣saved",
60                Toast.LENGTH_LONG).show();
61          } catch (FileNotFoundException e) {
62            Toast.makeText(ServerMenu.this,
63                "Settings␣failed␣to␣be␣saved", Toast.LENGTH_LONG)
64                .show();
65
66          } catch (IOException e) {
67            Toast.makeText(ServerMenu.this,
68                "Settings␣failed␣to␣be␣saved", Toast.LENGTH_LONG)
69                .show();
70
71          } catch (NumberFormatException e) {
72            Toast.makeText(ServerMenu.this,
73                "Timeout␣not␣an␣integer␣value", Toast.LENGTH_LONG)
74                .show();
75          } catch (Exception e) {
76            Toast.makeText(ServerMenu.this,
77                "Timeout␣cannot␣be␣over␣20000", Toast.LENGTH_LONG)
78                .show();
79          }
80
81        }
82
83      });
84
85      bTestServer = (Button) this.findViewById(R.id.btestserver);
86      bTestServer.setOnClickListener(new OnClickListener() {
87
88        @Override
89        public void onClick(View arg0) {
90          // TODO Auto-generated method stub
91
92          final ProgressDialog pd = ProgressDialog.show(ServerMenu.this,
93              "Please␣Wait", "Connecting␣To␣Server...");
94          pd.setCancelable(false);
95          new Thread() {
96
97            @Override
98            public void run() {
99
100             HttpConnector testConnector = new HttpConnector(
```

```
101                     Integer.parseInt(timeouttext.getText()
102                         .toString()));
103                 testConnector.setBaseUrl(uritext.getText().toString());
104                 HttpConnector.TestConnection tc = testConnector
105                     .createTestConnection();
106 
107                 final String status = tc.testConnection();
108 
109                 pd.dismiss();
110                 ServerMenu.this.runOnUiThread(new Runnable() {
111 
112                   @Override
113                   public void run() {
114                     // TODO Auto-generated method stub
115                     Toast.makeText(ServerMenu.this, status,
116                         Toast.LENGTH_LONG).show();
117                   }
118 
119                 });
120               }
121 
122             }.start();
123 
124           }
125 
126         });
127 
128         bTestAcount = (Button) this.findViewById(R.id.btestacount);
129         bTestAcount.setOnClickListener(new OnClickListener() {
130 
131           @Override
132           public void onClick(View arg0) {
133             final ProgressDialog pd = ProgressDialog.show(ServerMenu.this,
134                 "Please␣Wait", "Connecting␣To␣Server...");
135             pd.setCancelable(false);
136 
137             new Thread() {
138 
139               @Override
140               public void run() {
141                 HttpConnector testConnector = new HttpConnector(Integer
142                     .parseInt(timeouttext.getText().toString()));
143                 testConnector.setBaseUrl(uritext.getText().toString());
144 
145                 HttpConnector.Authenticator auth = testConnector
146                     .createAuthenticator(usertext.getText()
147                         .toString(), passtext.getText()
148                         .toString());
149                 auth.logUser();
150                 final int userIdt = auth.userId;
151                 pd.dismiss();
152                 ServerMenu.this.runOnUiThread(new Runnable() {
153 
154                   @Override
155                   public void run() {
156                     if (userIdt == -1)
157                       Toast.makeText(ServerMenu.this,
158                           "User␣could␣not␣be␣identified.",
159                           Toast.LENGTH_LONG).show();
160                     else
161                       Toast.makeText(
162                           ServerMenu.this,
163                           "User␣identified␣with␣userId:"
164                               + userIdt + ".",
165                           Toast.LENGTH_LONG).show();
166                   }
167 
168                 });
169 
170               }
171 
172             }.start();
173 
174           }
175 
176         });
177         super.onCreate(savedInstanceState);
178       }
179 
180       @Override
181       protected void onResume() {
182 
183         SettingManager setMan = new SettingManager(this);
```

```
184       if (setMan.retrieveSettings()) {
185         uritext.setText(setMan.getWebserver());
186         timeouttext.setText(setMan.getTimeout() + "");
187         usertext.setText(setMan.getUsername());
188         passtext.setText(setMan.getPassword());
189       } else
190         Toast.makeText(ServerMenu.this, "Settings␣could␣not␣be␣retrieved",
191             Toast.LENGTH_SHORT).show();
192
193       // get the configurations
194
195       super.onResume();
196     }
197
198   }
```

Πακέτο gr.ntua.cn.GPS.Auxiliary

HttpConnector.java

```
1   package gr.ntua.cn.GPS.Auxiliary;
2
3   // TODO
4   // Create a postFunction which will do the actuall post
5
6   import java.io.BufferedReader;
7   import java.io.File;
8   import java.io.FileInputStream;
9   import java.io.IOException;
10  import java.io.InputStream;
11  import java.io.InputStreamReader;
12  import java.nio.charset.Charset;
13  import java.util.ArrayList;
14  import java.util.Iterator;
15  import java.util.Vector;
16
17  import javax.xml.parsers.DocumentBuilder;
18  import javax.xml.parsers.DocumentBuilderFactory;
19
20  import org.apache.http.HttpResponse;
21  import org.apache.http.client.ClientProtocolException;
22  import org.apache.http.client.entity.UrlEncodedFormEntity;
23  import org.apache.http.client.methods.HttpPost;
24  import org.apache.http.entity.mime.MultipartEntity;
25  import org.apache.http.entity.mime.content.InputStreamBody;
26  import org.apache.http.entity.mime.content.StringBody;
27  import org.apache.http.impl.client.DefaultHttpClient;
28  import org.apache.http.message.BasicNameValuePair;
29  import org.w3c.dom.Document;
30  import org.w3c.dom.Node;
31  import org.w3c.dom.NodeList;
32
33  import android.location.Location;
34  import android.location.LocationManager;
35
36  public class HttpConnector {
37    private String baseurl = "http://192.168.1.2:8080/android";
38    public final static String AUTH_URL = "/login.jsp";
39    public final static String TEST_URL = "/test.jsp";
40    public final static String GETPOINT_URL = "/getpoint.jsp";
41    public final static String ADDPOINT_URL = "/addpoint.jsp";
42    public final static String CLOSESTLOCATION_URL = "/closestlocation.jsp";
43    public final static String CLOSESTLOCATIONS_URL = "/closestlocations.jsp";
44    public final static String GETBUDDY_URL = "/getbuddy.jsp";
45    public final static String GETBUDDYPOINT_URL = "/getbuddypoint.jsp";
46    public final static String ADDBUDDY_URL = "/addbuddy.jsp";
47    public final static String UPLOADIMAGE_URL = "/uploadimage.jsp";
48    private DefaultHttpClient httpclient;
49    private HttpPost httppost;
50    private ArrayList<BasicNameValuePair> pairs;
51    private int timeout;
52
53    /**
54     * The Constructor of the HttpConnector
55     *
56     * @param ctx
57     *            the Context in which the connector is created
58     * @param timeout
```

```
59      *            the timeout time for the connection
60      */
61     public HttpConnector(int timeout) {
62       this.timeout = timeout;
63     }
64
65     /**
66      * The setter for the Base Url
67      *
68      * @param baseurl
69      *            The base url of the web server application
70      */
71     public void setBaseUrl(String baseurl) {
72       this.baseurl = baseurl;
73     }
74
75     public void setTimeout(int timeout) {
76       this.timeout = timeout;
77     }
78
79     /**
80      * Creates a new Authenticator Object for authenticating user. <br>
81      * After the creation use <b>start()</b> method.
82      *
83      * @param user:
84      *            The username for authentication
85      * @param pass:
86      *            The password for authentication
87      * @return an Authenticator Object
88      */
89     public Authenticator createAuthenticator(String user, String pass) {
90       return new Authenticator(user, pass);
91     }
92
93     /**
94      * Authenticator Class
95      *
96      * @author Stavros
97      *
98      */
99     public class Authenticator {
100      String user, pass;
101      boolean athenticated;
102      public int userId;
103
104      Authenticator() {
105
106        this.athenticated = false;
107      }
108
109      Authenticator(String user, String pass) {
110        this.user = user;
111        this.pass = pass;
112
113        this.athenticated = false;
114      }
115
116      void setLogInfo(String user, String pass) {
117        this.user = user;
118        this.pass = pass;
119        this.athenticated = false;
120
121      }
122
123      public boolean isAuthenticated() {
124        return this.athenticated;
125      }
126
127      public void logUser() {
128
129        pairs = new ArrayList<BasicNameValuePair>();
130        pairs.add(new BasicNameValuePair("user", user));
131        pairs.add(new BasicNameValuePair("pass", pass));
132
133        try {
134
135          InputStream is = HttpConnector.this.makeConnection(pairs,
136              AUTH_URL);
137          String strRes = convertStreamToString(is);
138          if ((strRes != null) && (strRes.contains("AUTH␣OK␣"))) {
139            this.athenticated = true;
140            strRes = strRes.replace("AUTH␣OK␣", "");
141            strRes = strRes.replace("\n", "");
```

```
142             userId = Integer.parseInt(strRes);
143             return;
144           } else {
145             this.athenticated = false;
146             userId = -1;
147             return;
148           }
149         } catch (ClientProtocolException e) {
150           this.athenticated = false;
151           userId = -1;
152           return;
153
154         } catch (IOException e) {
155           this.athenticated = false;
156           userId = -1;
157           return;
158         }
159
160       }
161     }
162
163     /**
164      * Creates a new TestConnection Object for testing the existance of
165      * webserver <br>
166      * After the creation use <b>start()</b> method
167      *
168      * @return a TestConnection Object
169      */
170     public TestConnection createTestConnection() {
171       return new TestConnection();
172     }
173
174     /**
175      * Test the availability of the server <br>
176      * <b>Thread Free</b>
177      *
178      * @author Stavros
179      *
180      */
181     public class TestConnection {
182       public boolean succeed;
183
184       public String testConnection() {
185         String resString = "Failed␣to␣Connect";
186         succeed = false;
187         try {
188
189           InputStream is = HttpConnector.this.makeConnection(null,
190               TEST_URL);
191           resString = convertStreamToString(is);
192           succeed = true;
193           if (!resString.equalsIgnoreCase("Test␣OK\n")) {
194             resString = "Failed␣to␣Connect";
195             succeed = false;
196           }
197           resString = resString.replaceAll("\n", "");
198         } catch (ClientProtocolException e) {
199
200         } catch (IOException e) {
201
202         } catch (Exception e) {
203
204         }
205         return resString;
206       }
207
208     }
209
210     /**
211      * Creates a new PointGetter Object for downloading the point list for the
212      * specified userId
213      *
214      * @param userId:
215      *            the userId number that indicates a user
216      * @return a PointGetter Object
217      */
218     public PointGetter createPointGetter(int userId, String user, String pass) {
219       return new PointGetter(userId, user, pass);
220     }
221
222     /**
223      * Gets the list of the points for the specified userId <br>
224      * <b>Thread Free</b>
```

```
225       *
226       * @author Stavros
227       *
228       */
229      public class PointGetter {
230        private int userId;
231        private String user, pass;
232        public InputStream xmlPointStream;
233
234        PointGetter(int userId, String user, String pass) {
235          this.userId = userId;
236          this.user = user;
237          this.pass = pass;
238
239        }
240
241        public void getPoints() {
242          xmlPointStream = null;
243          pairs = new ArrayList<BasicNameValuePair>();
244          pairs
245              .add(new BasicNameValuePair("userId", Integer
246                  .toString(userId)));
247          pairs.add(new BasicNameValuePair("user", user));
248          pairs.add(new BasicNameValuePair("pass", pass));
249          try {
250            xmlPointStream = HttpConnector.this.makeConnection(pairs,
251                GETPOINT_URL);
252            return;
253          } catch (ClientProtocolException e) {
254
255          } catch (IOException e) {
256
257          }
258          xmlPointStream = null;
259        }
260
261      }
262
263      /**
264       * Creates a new PointAdder Object for adding-eddiding-deleting a point
265       *
266       * @param pointId:
267       *            pointId
268       * @param userId:
269       *            userId
270       * @param Longi:
271       *            Longitude
272       * @param Lati:
273       *            Latitude
274       * @param descr:
275       *            Description
276       * @param used
277       *            Used
278       * @param Public
279       *            Public
280       * @param urlStr
281       *            Url String
282       * @param mode:
283       *            mode can be PointAdder.ADD_MODE, PointAdder.EDIT_MODE,
284       *            PointAdder.DELETE_MODE
285       * @return a PointAdder Object
286       */
287      public PointAdder createPointAdder(String pointId, String userId,
288          String user, String pass, String Longi, String Lati, String descr,
289          String used, String Public, String urlStr, int mode) {
290        return new PointAdder(pointId, userId, user, pass, Longi, Lati, descr,
291            used, Public, urlStr, mode);
292      }
293
294      /**
295       * Adds-Edits-Deletes the specified point <br>
296       * <b>Thread Free</b>
297       *
298       * @author Stavros
299       *
300       */
301      public class PointAdder {
302        public final static int ADD_MODE = 1;
303        public final static int UPDATE_MODE = 2;
304        public final static int DELETE_MODE = 3;
305        private int mode;
306        private String Longitude, Latitude;
307        private String Description;
```

```
308       private String userId;
309       private String user, pass;
310       private String pointId;
311       private String Used;
312       private String Public;
313       private String urlStr;
314       public boolean connected;
315
316       PointAdder(String pointId, String userId, String user, String pass,
317           String Longi, String Lati, String desc, String Used,
318           String Public, String urlString, int mode) {
319         this.pointId = pointId;
320         this.userId = userId;
321         this.Longitude = Longi;
322         this.Latitude = Lati;
323         if (desc != null)
324           this.Description = desc.replaceAll("<", "").replaceAll(">", "");
325         if (user != null)
326           this.user = user.replaceAll("<", "").replaceAll(">", "");
327         if (pass != null)
328           this.pass = pass.replaceAll("<", "").replaceAll(">", "");
329         this.mode = mode;
330         this.Used = Used;
331         this.Public = Public;
332         this.urlStr = urlString;
333       }
334
335       public void addPoint() {
336
337         pairs = new ArrayList<BasicNameValuePair>();
338         pairs.add(new BasicNameValuePair("userId", userId));
339         pairs.add(new BasicNameValuePair("Longitude", Longitude));
340         pairs.add(new BasicNameValuePair("Latitude", Latitude));
341         pairs.add(new BasicNameValuePair("Description", Description));
342         pairs.add(new BasicNameValuePair("user", user));
343         pairs.add(new BasicNameValuePair("pass", pass));
344         pairs.add(new BasicNameValuePair("Mode", Integer.toString(mode)));
345         pairs.add(new BasicNameValuePair("pointId", pointId));
346         pairs.add(new BasicNameValuePair("Used", Used));
347         pairs.add(new BasicNameValuePair("Public", Public));
348         if (urlStr != null)
349           if (!urlStr.equalsIgnoreCase(""))
350             if (urlStr.equalsIgnoreCase("null"))
351               pairs.add(new BasicNameValuePair("Url", "\"" + urlStr
352                   + "\""));
353             else
354               pairs.add(new BasicNameValuePair("Url", urlStr));
355
356         try {
357
358           String res = convertStreamToString(HttpConnector.this
359               .makeConnection(pairs, ADDPOINT_URL));
360           if (res.equalsIgnoreCase("OK")) {
361             connected = true;
362             return;
363           }
364         } catch (ClientProtocolException e) {
365           connected = false;
366           return;
367         } catch (IOException e) {
368           connected = false;
369           return;
370         }
371         connected = true;
372       }
373
374     }
375
376     /**
377      * Returns a ClosestLocations Object
378      *
379      * @param map
380      *            A vector with all the buddies ids
381      * @param longi
382      *            Longitude
383      * @param lati
384      *            Latitude
385      * @param radius
386      *            Radius
387      * @return A ClosestLocations Object
388      */
389     public ClosestLocations createClosestLocations(Vector<String> map,
390         String user, String pass, String longi, String lati, String radius) {
```

```
391       return new ClosestLocations(map, user, pass, longi, lati, radius);
392     }
393
394     /**
395      * This class helps finding the closest locations within a predefined radius
396      *
397      * @author Stavros
398      *
399      */
400     public class ClosestLocations {
401       Vector<String> ids;
402       String longi, lati, radius;
403       String user, pass;
404       public boolean connected;
405       public Vector<DescribedLocation> results;
406
407       ClosestLocations(Vector<String> ids, String user, String pass,
408           String longi, String lati, String radius) {
409         this.ids = ids;
410         this.longi = longi;
411         this.lati = lati;
412         this.radius = radius;
413         this.user = user;
414         this.pass = pass;
415
416       }
417
418       public void findClosestLocations() {
419         results = new Vector<DescribedLocation>();
420         pairs = new ArrayList<BasicNameValuePair>();
421         pairs.add(new BasicNameValuePair("user", user));
422         pairs.add(new BasicNameValuePair("pass", pass));
423
424         pairs.add(new BasicNameValuePair("Longitude", longi));
425         pairs.add(new BasicNameValuePair("Latitude", lati));
426         pairs.add(new BasicNameValuePair("Radius", radius));
427         Iterator<String> i = ids.iterator();
428         if (i.hasNext())
429           pairs.add(new BasicNameValuePair("myuserId", i.next()));
430         while (i.hasNext()) {
431           pairs.add(new BasicNameValuePair("userId", i.next()));
432         }
433
434         Document doc = null;
435         try {
436
437           DocumentBuilderFactory dbf = DocumentBuilderFactory
438               .newInstance();
439           DocumentBuilder dob = dbf.newDocumentBuilder();
440           doc = dob.parse(HttpConnector.this.makeConnection(pairs,
441               CLOSESTLOCATIONS_URL));
442           // check for a pointlist element if does not exist bad response
443           doc.getElementsByTagName("PointList").item(0).getNodeName();
444         } catch (Exception e) {
445           connected = false;
446           return;
447         }
448         NodeList points = doc.getElementsByTagName("Point");
449         NodeList children = null;
450         Node child = null;
451         int length = points.getLength();
452         for (int ind = 0; ind < length; ind++) {
453           DescribedLocation db = new DescribedLocation(
454               new Location("gps"), null, 0, null);
455           Node point = points.item(ind);
456           children = point.getChildNodes();
457           int cLength = children.getLength();
458           for (int j = 0; j < cLength; j++) {
459             child = children.item(j);
460             if (child.getNodeType() == Node.ELEMENT_NODE) {
461               String name = child.getNodeName();
462               String value = child.getFirstChild().getNodeValue();
463               if (name.equals("Longitude"))
464                 db.loc.setLongitude(Double.parseDouble(value));
465               else if (name.equals("Latitude"))
466                 db.loc.setLatitude(Double.parseDouble(value));
467               else if (name.equals("Description"))
468                 db.descr = value;
469               else if (name.equals("Distance"))
470                 db.distance = Float.parseFloat(value);
471               else if (name.equals("Url"))
472                 db.url = value;
473             }
```

```
474           }
475           results.add(db);
476         }
477         connected = true;
478
479       }
480     }
481
482     /**
483      * Creates a Closest Location Object
484      *
485      * @param map
486      * @param longi
487      * @param lati
488      * @param dl
489      * @return
490      */
491     public ClosestLocation createClosestLocation(Vector<String> map,
492         String user, String pass, String longi, String lati,
493         DescribedLocation dl) {
494       return new ClosestLocation(map, user, pass, longi, lati, dl);
495     }
496
497     /**
498      * <br>
499      * <b>Thread Free</b>
500      *
501      * @author Blim
502      *
503      */
504     public class ClosestLocation {
505       Vector<String> ids;
506       String longi, lati;
507       String user, pass;
508       public DescribedLocation dl;
509       public boolean correct;
510
511       ClosestLocation(Vector<String> ids, String user, String pass,
512           String longi, String lati, DescribedLocation dl) {
513         this.ids = ids;
514         this.longi = longi;
515         this.lati = lati;
516         this.dl = dl;
517         this.user = user;
518         this.pass = pass;
519       }
520
521       public void findClosestLocation() {
522
523         pairs = new ArrayList<BasicNameValuePair>();
524
525         pairs.add(new BasicNameValuePair("Longitude", longi));
526         pairs.add(new BasicNameValuePair("Latitude", lati));
527
528         pairs.add(new BasicNameValuePair("user", user));
529         pairs.add(new BasicNameValuePair("pass", pass));
530
531         Iterator<String> i = ids.iterator();
532         if (i.hasNext())
533           pairs.add(new BasicNameValuePair("myuserId", i.next()));
534         while (i.hasNext()) {
535           pairs.add(new BasicNameValuePair("userId", i.next()));
536         }
537
538         Document doc = null;
539         try {
540
541           DocumentBuilderFactory dbf = DocumentBuilderFactory
542               .newInstance();
543           DocumentBuilder db = dbf.newDocumentBuilder();
544           doc = db.parse(HttpConnector.this.makeConnection(pairs,
545               CLOSESTLOCATION_URL));
546           // check for a point element if does not exist bad response
547           doc.getElementsByTagName("Point").item(0).getNodeName();
548         } catch (Exception e) {
549           dl.loc = null;
550           dl.descr = "no␣Point␣of␣Interest";
551           correct = false;
552           return;
553         }
554
555         try {
556           NodeList list = doc.getElementsByTagName("Longitude");
```

```
557           Node in = list.item(0);
558           double longi = Double.parseDouble(in.getFirstChild()
559               .getNodeValue());
560           list = doc.getElementsByTagName("Latitude");
561           in = list.item(0);
562           double lati = Double.parseDouble(in.getFirstChild()
563               .getNodeValue());
564           list = doc.getElementsByTagName("Distance");
565           in = list.item(0);
566           float distance = Float.parseFloat(in.getFirstChild()
567               .getNodeValue());
568           list = doc.getElementsByTagName("Description");
569           in = list.item(0);
570           String description = in.getFirstChild().getNodeValue();
571
572           list = doc.getElementsByTagName("HasUrl");
573           in = list.item(0);
574           String hasUrl = in.getFirstChild().getNodeValue();
575           String url = null;
576           if (hasUrl.equalsIgnoreCase("Yes")) {
577             list = doc.getElementsByTagName("Url");
578             in = list.item(0);
579             url = in.getFirstChild().getNodeValue();
580           }
581
582           Location loc = new Location(LocationManager.GPS_PROVIDER);
583           loc.setLatitude(lati);
584           loc.setLongitude(longi);
585           dl.loc = loc;
586           dl.descr = description;
587           dl.distance = distance;
588           dl.url = url;
589           correct = true;
590           return;
591
592         } catch (Exception e) { // No data found
593           dl.loc = null;
594           dl.descr = "no␣Point␣of␣Interest";
595           correct = true;
596           return;
597         }
598
599       }
600
601     }
602
603     public BuddyGetter createBuddyGetter(int userId, String user, String pass) {
604       return new BuddyGetter(userId, user, pass);
605     }
606
607     /**
608      * Gets the list of the points for the specified userId <br>
609      * <b>Thread Free</b>
610      *
611      * @author Stavros
612      *
613      */
614     public class BuddyGetter {
615       private int userId;
616       private String user, pass;
617       public InputStream xmlBuddyStream;
618
619       BuddyGetter(int userId, String user, String pass) {
620         this.userId = userId;
621         this.user = user;
622         this.pass = pass;
623
624       }
625
626       public void getBuddies() {
627         xmlBuddyStream = null;
628         pairs = new ArrayList<BasicNameValuePair>();
629         pairs
630             .add(new BasicNameValuePair("userId", Integer
631                 .toString(userId)));
632         pairs.add(new BasicNameValuePair("user", user));
633         pairs.add(new BasicNameValuePair("pass", pass));
634         try {
635           xmlBuddyStream = HttpConnector.this.makeConnection(pairs,
636               GETBUDDY_URL);
637           return;
638         } catch (ClientProtocolException e) {
639         } catch (IOException e) {
```

```
640         }
641         xmlBuddyStream = null;
642       }
643
644     }
645
646     public BuddyPointGetter createBuddyPointGetter(int userId, String user,
647         String pass) {
648       return new BuddyPointGetter(userId, user, pass);
649     }
650
651     /**
652      * An Object that helps getting the points of a friend <br>
653      * <b>Thread Free</b>
654      *
655      * @author Blim
656      *
657      */
658     public class BuddyPointGetter {
659       private int userId;
660       private String user, pass;
661       public InputStream xmlPointStream;
662
663       BuddyPointGetter(int userId, String user, String pass) {
664         this.userId = userId;
665         this.user = user;
666         this.pass = pass;
667
668       }
669
670       public void getPoints() {
671         xmlPointStream = null;
672         pairs = new ArrayList<BasicNameValuePair>();
673         pairs
674             .add(new BasicNameValuePair("userId", Integer
675                 .toString(userId)));
676         pairs.add(new BasicNameValuePair("user", user));
677         pairs.add(new BasicNameValuePair("pass", pass));
678         try {
679           xmlPointStream = HttpConnector.this.makeConnection(pairs,
680               GETBUDDYPOINT_URL);
681           return;
682         } catch (ClientProtocolException e) {
683
684         } catch (IOException e) {
685
686         }
687
688         xmlPointStream = null;
689       }
690
691     }
692
693     /**
694      * Creates a BuddyAdder Object for manipulating the buddies
695      *
696      * @param userId
697      *            The users userId
698      * @param buddyUserId
699      *            The buddy userId
700      * @param mode
701      *            the mode of the buddy adder
702      * @return the BuddyAdder Object
703      */
704     public BuddyAdder createBuddyAdder(String userId, String user, String pass,
705         String buddyUserId, int mode) {
706       return new BuddyAdder(userId, user, pass, buddyUserId, mode);
707     }
708
709     /**
710      * A BuddyAdder is an object that helps adding,deleting and accepting
711      * buddies <br>
712      * <b>Thread Free</b>
713      *
714      * @author Stavros
715      *
716      */
717     public class BuddyAdder {
718       public final static int ADD_MODE = 1;
719       public final static int ACCEPT_MODE = 2;
720       public final static int DELETE_MODE = 3;
721       private int mode;
722
```

```
723       private String userId;
724       private String buddyUserId;
725       private String user, pass;
726       public boolean connected;
727
728       BuddyAdder(String userId, String user, String pass, String buddyUserId,
729           int mode) {
730         this.userId = userId;
731         this.buddyUserId = buddyUserId;
732         this.mode = mode;
733         this.user = user;
734         this.pass = pass;
735       }
736
737       public void addBuddy() {
738         pairs = new ArrayList<BasicNameValuePair>();
739         pairs.add(new BasicNameValuePair("userId", userId));
740         pairs.add(new BasicNameValuePair("buddyUserId", buddyUserId));
741         pairs.add(new BasicNameValuePair("Mode", Integer.toString(mode)));
742         pairs.add(new BasicNameValuePair("user", user));
743         pairs.add(new BasicNameValuePair("pass", pass));
744         try {
745           String res = convertStreamToString(HttpConnector.this
746               .makeConnection(pairs, ADDBUDDY_URL));
747           if (res.equalsIgnoreCase("OK")) {
748             connected = true;
749             return;
750           }
751         } catch (ClientProtocolException e) {
752           connected = false;
753           return;
754         } catch (IOException e) {
755           connected = false;
756           return;
757         }
758         connected = true;
759       }
760     }
761
762     public ImageSender createImageSender(String filepath, String userId,
763         String user, String pass, String pointId) {
764       return new ImageSender(filepath, userId, user, pass, pointId);
765     }
766
767     /**
768      * ImageSender helps sending images to the web server <br>
769      * <b>Thread Free</b>
770      *
771      * @author Stavros
772      *
773      */
774     public class ImageSender {
775       String filepath;
776       String userId;
777       String user, pass;
778       String pointId;
779       public boolean connected;
780
781       ImageSender(String filepath, String userId, String user, String pass,
782           String pointId) {
783         this.filepath = filepath;
784         this.userId = userId;
785         this.pointId = pointId;
786         this.user = user;
787         this.pass = pass;
788       }
789
790       public void sendImage() {
791         try {
792           MultipartEntity multipart = new MultipartEntity();
793           File file = new File(filepath);
794           FileInputStream fis = new FileInputStream(file);
795
796           Charset charset = Charset.forName("ISO-8859-1");
797           StringBody userIdBody = new StringBody(userId, charset);
798           StringBody pointIdBody = new StringBody(pointId, charset);
799           StringBody userBody = new StringBody(user,charset);
800           StringBody passBody = new StringBody(pass,charset);
801           InputStreamBody fileB = new InputStreamBody(fis, filepath);
802           multipart.addPart("userId", userIdBody);
803           multipart.addPart("pointId", pointIdBody);
804           multipart.addPart("user", userBody);
805           multipart.addPart("pass", passBody);
```

```
806           multipart.addPart("image", fileB);
807
808           httpclient = new DefaultHttpClient();
809           httpclient.getParams().setParameter("http.connection.timeout",
810               new Integer(HttpConnector.this.timeout));
811
812           httppost = new HttpPost(baseurl + UPLOADIMAGE_URL);
813           httppost.setEntity(multipart);
814         } catch (Exception e) {
815
816         }
817         try {
818           httpclient.execute(httppost);
819           connected = true;
820           return;
821         } catch (ClientProtocolException e) {
822           connected = false;
823           return;
824
825         } catch (IOException e) {
826           connected = false;
827           return;
828         }
829
830       }
831
832     }
833
834     /**
835      * Secondary function for debugging
836      *
837      * @param is:
838      * @return
839      */
840     public String convertStreamToString(InputStream is) {
841       BufferedReader reader = new BufferedReader(new InputStreamReader(is));
842       StringBuilder sb = new StringBuilder();
843
844       String line = null;
845       try {
846         while ((line = reader.readLine()) != null) {
847           sb.append(line + "\n");
848         }
849       } catch (IOException e) {
850         e.printStackTrace();
851       } finally {
852         try {
853           is.close();
854         } catch (IOException e) {
855           e.printStackTrace();
856         }
857       }
858
859       return sb.toString();
860     }
861
862     /**
863      * This function will be used to refactor this class code
864      *
865      * @param pairs
866      *            all the Parameter Pairs
867      * @param URL
868      *            The target url
869      * @return The response input stream
870      *
871      * @throws ClientProtocolException
872      * @throws IOException
873      *
874      */
875     public InputStream makeConnection(ArrayList<BasicNameValuePair> pairs,
876         String URL) throws ClientProtocolException, IOException {
877       httpclient = new DefaultHttpClient();
878       httpclient.getParams().setParameter("http.connection.timeout",
879           new Integer(HttpConnector.this.timeout));
880       httppost = new HttpPost(baseurl + URL);
881       UrlEncodedFormEntity p_entity;
882       if (pairs != null) {
883         p_entity = new UrlEncodedFormEntity(pairs, "ISO-8859-1");
884         httppost.setEntity(p_entity);
885       }
886       HttpResponse response = httpclient.execute(httppost);
887
888       return response.getEntity().getContent();
```

```
889
890     }
891 }
```

DescribedLocation.java

```
1   package gr.ntua.cn.GPS.Auxiliary;
2
3
4   import android.location.Location;
5
6   /**
7    * A simple class just to return both a string and a Location from one
8    * function
9    *
10   * @author Stavros
11   *
12   */
13  public class DescribedLocation {
14    public Location loc;
15    public String descr;
16    public float distance;
17    public String url;
18
19    /**
20     * Public Constructor
21     *
22     * @param loc:
23     *            The Location data
24     * @param descr:
25     *            The Discription data
26     * @param distance:
27     *            The Distance data
28     */
29    public DescribedLocation(Location loc, String descr, float distance,
30        String url) {
31      this.loc = loc;
32      this.descr = descr;
33      this.distance = distance;
34      this.url = url;
35    }
36
37    @Override
38    public boolean equals(Object o) {
39      if (o == null)
40        return false;
41      if (o instanceof DescribedLocation) {
42        DescribedLocation dl = (DescribedLocation) o;
43        return ((this.descr != null) && (this.descr.equals(dl.descr))
44            && (this.url != null) && (this.url.equals(dl.url)));
45      }
46      return super.equals(o);
47    }
48
49  }
```

DbAdapter.java

```
1
2   package gr.ntua.cn.GPS.Auxiliary;
3
4   import android.content.ContentValues;
5   import android.content.Context;
6   import android.database.Cursor;
7   import android.database.SQLException;
8   import android.database.sqlite.SQLiteDatabase;
9   import android.database.sqlite.SQLiteOpenHelper;
10  import android.util.Log;
11
12  /**
13   * The DbAdapter class is an intermediate class between the
14   * application and the database containing the application data
15   */
16  public class DbAdapter {
17
18    public static final String KEY_LAT = "lat";
19    public static final String KEY_LONG = "longt";
20    public static final String KEY_DESC = "descr";
```

```
21      public static final String KEY_ROWID = "_id";
22
23      private static final String TAG = "PointsDbAdapter";
24      private DatabaseHelper mDbHelper;
25      private SQLiteDatabase mDb;
26
27      /**
28       * Database creation SQL statement
29       */
30      private static final String DATABASE_CREATE = "create␣table␣pointsofinterest␣(_id␣integer␣primary␣key␣
            autoincrement,␣"
31          + "longt␣text␣not␣null,␣lat␣text␣not␣null,␣descr␣text␣not␣null);";
32
33      private static final String DATABASE_NAME = "locations";
34      private static final String DATABASE_TABLE = "pointsofinterest";
35      private static final int DATABASE_VERSION = 2;
36
37      private final Context mCtx;
38
39      /**
40       * DatabaseHelper is a private class that extends SQLiteOpenHelper
41       * in order to open the database correctly
42       */
43      private static class DatabaseHelper extends SQLiteOpenHelper {
44        /**
45         * Constructor
46         * @param context : The Context in which the helper is created
47         */
48        DatabaseHelper(Context context) {
49          super(context, DATABASE_NAME, null, DATABASE_VERSION);
50        }
51
52        @Override
53        public void onCreate(SQLiteDatabase db) {
54
55          db.execSQL(DATABASE_CREATE);
56        }
57
58        @Override
59        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
60          Log.w(TAG, "Upgrading␣database␣from␣version␣" + oldVersion + "␣to␣"
61              + newVersion + ",␣which␣will␣destroy␣all␣old␣data");
62          db.execSQL("DROP␣TABLE␣IF␣EXISTS␣pointsofinterest");
63          onCreate(db);
64        }
65      }
66      /**
67       * Constructor
68       * @param ctx : The Context in which the DbAdapter is created
69       */
70      public DbAdapter(Context ctx) {
71        this.mCtx = ctx;
72      }
73
74      /**
75       * Opens the database if it already exists or creates it if i does not exist
76       * @return A DbAdapter object for handling the database
77       * @throws SQLException
78       */
79      public DbAdapter open() throws SQLException {
80        mDbHelper = new DatabaseHelper(mCtx);
81        mDb = mDbHelper.getWritableDatabase();
82        return this;
83      }
84
85      /**
86       * Closes the database
87       */
88      public void close() {
89        mDbHelper.close();
90      }
91
92      /**
93       * Creates a new Point Entry in the database
94       * @param longt The Longitude of the point
95       * @param latt  The Latitude of the point
96       * @param desc  The Description of the point
97       * @return <li>The row id of the newly created point</li>
98       */
99      public long createPoint(String longt, String latt, String desc) {
100       ContentValues initialValues = new ContentValues();
101       initialValues.put(KEY_LONG, longt);
102       initialValues.put(KEY_LAT, latt);
```

```
103       initialValues.put(KEY_DESC, desc);
104
105       return mDb.insert(DATABASE_TABLE, null, initialValues);
106     }
107
108     /**
109      * Deletes the point with the specified rowId from the database
110      * @param rowId The specified rowId for the point to be deleted
111      * @return <li><b>true</b> if 1 or more lines were affected</li>
112      * <li><b>false</b> if 0 lines were affected
113      */
114     public boolean deletePoint(long rowId) {
115
116       return mDb.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
117     }
118
119     /**
120      * Fetches all the points in the database
121      * @return A Cursor which contains all the points of the database
122      */
123     public Cursor fetchAllPoints() {
124
125       return mDb.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_LONG,
126           KEY_LAT, KEY_DESC }, null, null, null, null, null);
127     }
128
129     /**
130      * fetches the point with the specified rowId
131      * @param rowId : The specified rowId
132      * @return A Cursor which contains the point with the specified rowId
133      * @throws SQLException
134      */
135     public Cursor fetchPoint(long rowId) throws SQLException {
136
137       Cursor mCursor =
138
139       mDb.query(true, DATABASE_TABLE, new String[] { KEY_ROWID, KEY_LONG,
140           KEY_LAT, KEY_DESC }, KEY_ROWID + "=" + rowId, null, null, null,
141           null, null);
142       if (mCursor != null) {
143         mCursor.moveToFirst();
144       }
145       return mCursor;
146
147     }
148
149     /**
150      * Updates a point with the specified rowId using the info that parameters provide
151      * @param rowId: The specified rowId
152      * @param longt: The Longitude of the point
153      * @param lat: The Latitude of the point
154      * @param desc: The Description of the point
155      * @return <li><b>true</b> if 1 or more lines were affected</li>
156      * <li><b>false</b> if 0 lines were affected
157      */
158     public boolean updatePoint(long rowId, String longt, String lat, String desc) {
159       ContentValues args = new ContentValues();
160       args.put(KEY_LONG, longt);
161       args.put(KEY_LAT, lat);
162       args.put(KEY_DESC, desc);
163
164       return mDb.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;
165     }
166   }
```

myOverlay.java

```
1   package gr.ntua.cn.GPS.Auxiliary;
2
3   import android.app.AlertDialog;
4   import android.content.Context;
5   import android.content.DialogInterface;
6   import android.content.Intent;
7   import android.graphics.Canvas;
8   import android.graphics.Paint;
9   import android.graphics.Point;
10  import android.graphics.Paint.Style;
11  import android.net.Uri;
12  import android.util.Log;
13  import android.view.MotionEvent;
14
```

```
15  import com.google.android.maps.GeoPoint;
16  import com.google.android.maps.MapView;
17  import com.google.android.maps.Overlay;
18  import com.google.android.maps.Projection;
19
20  /**
21   * This class extends Overlay it is used to display a circle at given
22   * Coordinates
23   *
24   * @author Stavros
25   *
26   */
27  public class myOverlay extends Overlay {
28    private Double curLong = null;
29    private Double curLat = null;
30    private Paint paint = null;
31    private float Rad;
32    private boolean absolute;
33    private String Description;
34    private float distance;
35    private String url;
36    private Context ctx;
37
38    public String getDescription() {
39      return Description;
40    }
41
42    public void setDescription(String description) {
43      Description = description;
44    }
45
46    public float getDistance() {
47      return distance;
48    }
49
50    public void setDistance(float distance) {
51      this.distance = distance;
52    }
53
54    public String getUrl() {
55      return url;
56    }
57
58    public void setUrl(String url) {
59      this.url = url;
60    }
61
62    /**
63     * The public constructor. It creates a default paint with ARGB
64     * (100,0,0,255) StrokeWidth 4 and FILL style
65     *
66     * @param Rad:
67     *            the Radius of the circle
68     * @param absolute:
69     *            <li><b>true</b> if we want a not affected radius by the zoom
70     *            level</li>
71     *            <li><b>false</b> if we want an affected radius by the zoom
72     *            level</li>
73     */
74    public myOverlay(float Rad, boolean absolute, Context ctx) {
75
76      curLong = null;
77      curLat = null;
78      this.ctx = ctx;
79
80      paint = new Paint();
81      paint.setStrokeWidth(4);
82      paint.setARGB(100, 0, 0, 255);
83      paint.setStyle(Style.FILL);
84      paint.setAntiAlias(true);
85
86      // pin = Bitmap.createScaledBitmap(BitmapFactory.decodeResource(res,
87      // R.drawable.curlocation), 30, 30, false);
88
89      this.Rad = Rad;
90      this.absolute = absolute;
91    }
92
93    /**
94     * Paint Getter
95     *
96     * @return the paint
97     */
```

```
98      public Paint getPaint() {
99        return paint;
100     }
101
102     /**
103      * Paint Setter
104      *
105      * @param paint:
106      *            the paint
107      */
108     public void setPaint(Paint paint) {
109       this.paint = paint;
110     }
111
112     /**
113      * Radius Getter
114      *
115      * @return the Radius
116      */
117     public float getRad() {
118       return Rad;
119     }
120
121     /**
122      * Radius Setter
123      *
124      * @param rad:
125      *            the Radius
126      */
127     public void setRad(float rad) {
128       Rad = rad;
129     }
130
131     /**
132      * Absolute Getter
133      *
134      * @return the Absolute
135      */
136     public boolean isAbsolute() {
137       return absolute;
138     }
139
140     /**
141      * Absolute Setter
142      *
143      * @param absolute:
144      *            The Absolute
145      */
146     public void setAbsolute(boolean absolute) {
147       this.absolute = absolute;
148     }
149
150     /**
151      * Coordinates Setter
152      *
153      * @param curLong:
154      *            The Longitude
155      * @param curLat:
156      *            The Latitude
157      */
158     public void setCoord(Double curLong, Double curLat) {
159       this.curLat = curLat;
160       this.curLong = curLong;
161
162     }
163
164     /**
165      * This functions is called its time the map is drawn. It draws a circle at
166      * the given coordinates (Longitude, Latitude) with the paint specified and
167      * specified Radius
168      */
169     @Override
170     public void draw(Canvas canvas, MapView mapView, boolean shadow) {
171       super.draw(canvas, mapView, shadow);
172       try {
173         if ((curLong == null) || (curLat == null))
174           return;
175         Projection proj = mapView.getProjection();
176         Point p = proj.toPixels(new GeoPoint((int) (curLat * 1E6),
177             (int) (curLong * 1E6)), null);
178         float Radt = Rad;
179         if (!absolute) {
180
```

```
181           Radt = proj.metersToEquatorPixels(Rad)*(1/(float) Math.cos(Math.toRadians(curLat)));
182         }
183         // canvas.drawBitmap(pin, p.x-pin.getWidth()/2,
184         // p.y-pin.getHeight()/2, null);
185         canvas.drawCircle(p.x, p.y, Radt, paint);
186
187       } catch (Exception e) {
188         Log.e("Error", "Drawing␣Error");
189       }
190
191     }
192
193     @Override
194     public boolean onTouchEvent(MotionEvent e, MapView mapView) {
195       if (e.getAction() != MotionEvent.ACTION_UP)
196
197         return false;
198       if (!this.absolute)
199         return false;
200       float x = e.getX();
201       float y = e.getY();
202       Projection proj = mapView.getProjection();
203       Point p = proj.toPixels(new GeoPoint((int) (curLat * 1E6),
204           (int) (curLong * 1E6)), null);
205       float dist = (float) Math.sqrt((x - p.x) * (x - p.x) + (y - p.y)
206           * (y - p.y));
207       if ((dist <= Rad) && (this.Description != null)) {
208         AlertDialog ad = new AlertDialog.Builder(ctx).setMessage(
209             this.Description+ "\nDistance␣from␣here:␣"+this.distance+"␣meters").setPositiveButton("OK",
210             new DialogInterface.OnClickListener() {
211               @Override
212               public void onClick(DialogInterface arg0, int arg1) {
213
214                 arg0.dismiss();
215               }
216             }).setTitle("Point").create();
217         if (this.url != null)
218           ad.setButton2("WebSite", new DialogInterface.OnClickListener() {
219             @Override
220             public void onClick(DialogInterface arg0, int arg1) {
221               if (!url.startsWith("http://"))
222                 url = "http://" + url;
223               Intent i =new Intent(
224                   Intent.ACTION_VIEW, Uri.parse(url));
225               ctx.startActivity(i);
226             }
227           });
228         ad.show();
229         return true;
230
231       }
232       return false;
233     }
234
235   }
```

SettingManager.java

```
1   package gr.ntua.cn.GPS.Auxiliary;
2
3   import java.io.FileInputStream;
4   import java.io.FileNotFoundException;
5   import java.util.StringTokenizer;
6
7   import android.content.Context;
8   import android.os.Parcel;
9   import android.os.Parcelable;
10
11  public class SettingManager implements Parcelable{
12    private final static String FILENAME = "webserver.ini";
13    private Context ctx = null;
14    private String webserver = null;
15    private int timeout = -1;
16    private String username = null;
17    private String password = null;
18    private int userId = -1;
19
20
21    public int getUserId() {
22      return userId;
23    }
```

```
24
25      public void setUserId(int userId) {
26        this.userId = userId;
27      }
28
29      public String getWebserver() {
30        return webserver;
31      }
32
33      public int getTimeout() {
34        return timeout;
35      }
36
37      public String getUsername() {
38        return username;
39      }
40
41      public String getPassword() {
42        return password;
43      }
44
45      public SettingManager(Context ctx) {
46        this.ctx = ctx;
47      }
48
49      public SettingManager(Parcel source) {
50
51        this.webserver = source.readString();
52        this.username = source.readString();
53        this.password = source.readString();
54        this.timeout = source.readInt();
55        this.userId = source.readInt();
56        this.ctx = null;
57
58      }
59
60      public boolean retrieveSettings()  {
61        FileInputStream webserverini = null;
62        try {
63          webserverini = ctx.openFileInput(FILENAME);
64        }
65        catch (FileNotFoundException e) {
66          ctx.deleteFile(FILENAME);
67          return false;
68        }
69        try {
70        StringBuffer sb = new StringBuffer("");
71        int ch;
72        while ((ch = webserverini.read()) != -1)
73          sb.append((char) ch);
74        StringTokenizer st = new StringTokenizer(sb.toString(), "\n");
75        this.webserver = st.nextToken().split("=")[1];
76        this.timeout = Integer.parseInt(st.nextToken().split("=")[1]);
77        this.username = st.nextToken().split("=")[1];
78        this.password = st.nextToken().split("=")[1];
79        }
80        catch (Exception e) {
81          ctx.deleteFile(FILENAME);
82          return false;
83        }
84        return true;
85
86      }
87
88      // Parcelable implementation
89      @Override
90      public void writeToParcel(Parcel out, int flags) {
91             out.writeString(this.webserver);
92             out.writeString(this.username);
93             out.writeString(this.password);
94             out.writeInt(this.timeout);
95             out.writeInt(this.userId);
96         }
97
98      @Override
99      public int describeContents() {
100
101       return 0;
102     }
103
104     public static final Parcelable.Creator<SettingManager> CREATOR
105       = new Parcelable.Creator<SettingManager>() {
106
```

```
107       @Override
108       public SettingManager createFromParcel(Parcel source) {
109
110         return new SettingManager(source);
111       }
112
113       @Override
114       public SettingManager[] newArray(int arg0) {
115
116         return new SettingManager[arg0];
117       }
118
119     };
120   }
```

Πακέτο gr.ntua.cn.GPS.Services

LocationService.java-LocationTripService.java

Παραθέτουμε το LocationService.java, το LocationTripService.java είναι παρόμοιο:

```
1   package gr.ntua.cn.GPS.Services;
2
3   import gr.ntua.cn.GPS.GPS;
4   import gr.ntua.cn.GPS.R;
5   import gr.ntua.cn.GPS.Auxiliary.DescribedLocation;
6   import gr.ntua.cn.GPS.Auxiliary.HttpConnector;
7   import gr.ntua.cn.GPS.Auxiliary.SettingManager;
8   import gr.ntua.cn.GPS.Auxiliary.HttpConnector.BuddyGetter;
9   import gr.ntua.cn.GPS.Auxiliary.HttpConnector.ClosestLocation;
10
11  import java.io.IOException;
12  import java.util.Iterator;
13  import java.util.Vector;
14
15  import javax.xml.parsers.DocumentBuilder;
16  import javax.xml.parsers.DocumentBuilderFactory;
17  import javax.xml.parsers.ParserConfigurationException;
18
19  import org.w3c.dom.Document;
20  import org.w3c.dom.Node;
21  import org.w3c.dom.NodeList;
22  import org.xml.sax.SAXException;
23
24  import android.app.Notification;
25  import android.app.NotificationManager;
26  import android.app.PendingIntent;
27  import android.app.Service;
28  import android.content.ComponentName;
29  import android.content.Context;
30  import android.content.Intent;
31  import android.content.ServiceConnection;
32  import android.location.Location;
33  import android.location.LocationListener;
34  import android.location.LocationManager;
35  import android.net.Uri;
36  import android.os.Bundle;
37  import android.os.IBinder;
38
39  public class LocationService extends Service {
40    private NotificationManager nm;
41    private LocationManager lm;
42    private int NOTIFICATION_ID = 1;
43    private MyLocationListener myLocListener;
44    private SettingManager settingManager;
45    private HttpConnector remoteConnector;
46    private boolean includebuddies;
47    private Vector<String> buddies;
48
49    @Override
50    public void onCreate() {
51
52      nm = (NotificationManager) this
53          .getSystemService(Context.NOTIFICATION_SERVICE);
54      lm = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
55      myLocListener = new MyLocationListener();
56      remoteConnector = new HttpConnector(0);
57      // Bind to Notification DeleteService
```

```
58        this.bindService(new Intent(LocationService.this,
59            NotificationDeleteService.class), this.mConnection,
60            Context.BIND_AUTO_CREATE);
61
62        super.onCreate();
63      }
64
65      @Override
66      public void onStart(Intent intent, int startId) {
67        String actionString = intent.getAction();
68        if (actionString.equals("gr.ntua.cn.GPS.ServiceNotificationDeleted"))
69          reshowNotification();
70        else if (actionString.equals("gr.ntua.cn.GPS.StartService")) {
71          settingManager = (SettingManager) intent
72              .getParcelableExtra("settingmanager");
73          this.includebuddies = intent.getBooleanExtra("includebuddies",
74              false);
75          remoteConnector.setBaseUrl(settingManager.getWebserver());
76          remoteConnector.setTimeout(settingManager.getTimeout());
77          if (this.includebuddies)
78            this.retrieveBuddies();
79          else
80            buddies = null;
81          lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
82              this.myLocListener);
83
84        }
85        super.onStart(intent, startId);
86      }
87
88      @Override
89      public void onDestroy() {
90        lm.removeUpdates(this.myLocListener);
91        nm.cancel(this.NOTIFICATION_ID + 1);
92        super.onDestroy();
93      }
94
95      private void reshowNotification() {
96        // Create a notification that can restart the application
97        Notification appnot = new Notification(R.drawable.map,
98            "GPS␣Location␣Minimized", System.currentTimeMillis());
99
100       PendingIntent pi = PendingIntent.getActivity(this, 1, new Intent(
101           LocationService.this, GPS.class).putExtra("pointmode",
102           GPS.REMOTE_MODE).putExtra("includebuddies",
103           LocationService.this.includebuddies), 0);
104
105       appnot.setLatestEventInfo(this, "GPS␣Location", "Restart", pi);
106
107       appnot.deleteIntent = PendingIntent.getService(this, 1, (new Intent(
108           LocationService.this, LocationService.class))
109           .setAction("gr.ntua.cn.GPS.ServiceNotificationDeleted"), 1);
110       nm.notify(this.NOTIFICATION_ID + 1, appnot);
111
112     }
113
114     @Override
115     public IBinder onBind(Intent intent) {
116       // TODO Auto-generated method stub
117       return null;
118     }
119
120     /**
121      * A Extension of the LocationListener class that takes action on location
122      * changes
123      *
124      * @author Stavros
125      *
126      */
127     private class MyLocationListener implements LocationListener {
128       /**
129        * Called when new location data is available
130        */
131       DescribedLocation lastLoc = null;
132
133       @Override
134       public void onLocationChanged(Location loc) {
135         DescribedLocation dl = findClosestRemotePointOfInterest(loc);
136         if (dl.loc == null)
137           dl.loc = loc; // If no point of interest found
138         // If we have a new Location or notifications have been deleted
139         if (!dl.equals(lastLoc)
140             || ((notificationBinder != null) && (notificationBinder
```

```
141                 .hasBeenDeleted()))) {
142           LocationService.this.nm
143               .cancel(LocationService.this.NOTIFICATION_ID);
144           if (dl.url != null) {
145             LocationService.this.showNotification(dl.url, dl.descr);
146           }
147         }
148         lastLoc = dl;
149       }
150
151       @Override
152       public void onProviderDisabled(String provider) {
153
154       }
155
156       @Override
157       public void onProviderEnabled(String provider) {
158
159       }
160
161       @Override
162       public void onStatusChanged(String provider, int status, Bundle extras) {
163
164       }
165     }
166
167     private void showNotification(String url, String msg) {
168       Notification notification = new Notification(R.drawable.webpage,
169           "Location␣Web␣Info", System.currentTimeMillis());
170
171       if (!url.startsWith("http://"))
172         url = "http://" + url;
173
174       PendingIntent pi = PendingIntent.getActivity(this, 1, new Intent(
175           Intent.ACTION_VIEW, Uri.parse(url)), 0);
176
177       notification.setLatestEventInfo(this, "Location␣Web␣Data", msg, pi);
178
179       notification.deleteIntent = PendingIntent.getService(this, 1,
180           (new Intent(LocationService.this,
181               NotificationDeleteService.class))
182               .setAction("gr.ntua.cn.GPS.NotificationDeleted"), 1);
183       nm.notify(this.NOTIFICATION_ID, notification);
184     }
185
186     // Code for binding to the NotificationDeleteService
187     private NotificationDeleteService.customBinder notificationBinder = null;
188     private ServiceConnection mConnection = new ServiceConnection() {
189
190       @Override
191       public void onServiceConnected(ComponentName name, IBinder service) {
192         LocationService.this.notificationBinder = (NotificationDeleteService.customBinder) service;
193       }
194
195       @Override
196       public void onServiceDisconnected(ComponentName arg0) {
197         LocationService.this.notificationBinder = null;
198
199       }
200     };
201
202     private DescribedLocation findClosestRemotePointOfInterest(Location loc) {
203       // Some Communication takes place here
204       Vector<String> params = new Vector<String>();
205       DescribedLocation dl = new DescribedLocation(null, null, 0, null);
206
207       params.add(Integer.toString(settingManager.getUserId()));
208       if ((this.includebuddies) && (this.buddies != null)) {
209         Iterator<String> it = buddies.iterator();
210         while (it.hasNext()) {
211           params.add(it.next());
212         }
213       }
214
215       ClosestLocation cl = remoteConnector.createClosestLocation(params,
216           settingManager.getUsername(), settingManager.getPassword(),
217           Double.toString(loc.getLongitude()), Double.toString(loc
218               .getLatitude()), dl);
219
220       cl.findClosestLocation();
221
222       return cl.dl;
223     }
```

```
224
225     private boolean retrieveBuddies() {
226       this.buddies = null;
227       BuddyGetter bg = remoteConnector.createBuddyGetter(settingManager
228           .getUserId(), settingManager.getUsername(), settingManager
229           .getPassword());
230       bg.getBuddies();
231
232       if (bg.xmlBuddyStream != null) {
233
234         try {
235           DocumentBuilderFactory dbf = DocumentBuilderFactory
236               .newInstance();
237           DocumentBuilder db = dbf.newDocumentBuilder();
238           Document doc = db.parse(bg.xmlBuddyStream);
239           if (!doc.getElementsByTagName("Status").item(0).getFirstChild()
240               .getNodeValue().equals("OK")) {
241
242             return false;
243           }
244
245           NodeList nodelist = doc.getElementsByTagName("Buddy");
246           this.buddies = new Vector<String>();
247           Node iNode, icNode;
248           int length = nodelist.getLength();
249           for (int i = 0; i < length; i++) {
250
251             iNode = nodelist.item(i);
252             NodeList children = iNode.getChildNodes();
253
254             int cLength = children.getLength();
255             for (int j = 0; j < cLength; j++) {
256               icNode = children.item(j);
257               if (icNode.getNodeType() == Node.ELEMENT_NODE) {
258                 NodeList texts = icNode.getChildNodes();
259                 for (int k = 0; k < texts.getLength(); k++) {
260                   String val = texts.item(k).getNodeValue();
261                   String name = icNode.getNodeName();
262                   if (name.equals("buddyId"))
263                     this.buddies.add(val);
264                 }
265               }
266             }
267
268           }
269         } catch (ParserConfigurationException e) {
270
271           return false;
272         } catch (IOException e) {
273
274           return false;
275         } catch (SAXException e) {
276
277           return false;
278         } catch (Exception e) {
279
280           return false;
281         }
282         return true;
283       } else {
284
285         return false;
286       }
287
288     }
289   }
```

NotificationDeleteService.java

```
1   package gr.ntua.cn.GPS.Services;
2
3   import android.app.Service;
4   import android.content.Intent;
5   import android.os.Binder;
6   import android.os.IBinder;
7   /**
8    * This Service is called when notifications are deleted
9    * and returns the current status of notifications via the customBinder object
10   * @author Blim
```

```
 *
 */
public class NotificationDeleteService extends Service {
  private boolean mDeleted;
  private IBinder mBinder = new customBinder();

  @Override
  public void onCreate() {
    mDeleted = false;
    super.onCreate();
  }

  @Override
  public IBinder onBind(Intent arg0) {
    // TODO Auto-generated method stub
    return mBinder;
  }

  @Override
  public void onStart(Intent intent, int startId) {
    String intentAction = intent.getAction();
    if (intentAction.equals("gr.ntua.cn.GPS.NotificationDeleted"))
      mDeleted = true;
    super.onStart(intent, startId);
  }

  /**
   *
   * @author Stavros
   *
   */
  public class customBinder extends Binder {
    /**
     *
     * @return if notifications have been deleted and then resets the
     *         deleted variable
     */
    public boolean hasBeenDeleted() {
      boolean ret = mDeleted;
      mDeleted = false;
      return ret;
    }
  }
}
```

Πακέτο gr.ntua.cn.GPS.ImageManagement

CameraPreview.java

```
package gr.ntua.cn.GPS.ImageManagement;

import gr.ntua.cn.GPS.ImageManagement.Preview;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Window;

public class CameraPreview extends Activity {
  private Preview mPreview;
  private String filename;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    filename = getIntent().getStringExtra("filename");
    // Hide the window title.
    requestWindowFeature(Window.FEATURE_NO_TITLE);

    // Create our Preview view and set it as the content of our activity.
    mPreview = new Preview(this);
    mPreview.setFilename(filename);

    mPreview.setFilename(filename);

    setContentView(mPreview);
  }

  @Override
```

```
30    public boolean onKeyDown(int keyCode, KeyEvent event) {
31
32      mPreview.takePic(keyCode, event);
33
34      this.setResult(Activity.RESULT_OK);
35      this.finish();
36      return super.onKeyDown(keyCode, event);
37    }
38
39  }
```

picGallery.java

```
1   package gr.ntua.cn.GPS.ImageManagement;
2
3   import gr.ntua.cn.GPS.R;
4
5   import java.io.File;
6   import java.io.FileFilter;
7   import java.util.HashMap;
8   import java.util.List;
9   import java.util.Map;
10  import java.util.Vector;
11
12  import android.app.Activity;
13  import android.app.AlertDialog;
14  import android.content.DialogInterface;
15  import android.content.Intent;
16  import android.graphics.BitmapFactory;
17  import android.os.Bundle;
18  import android.view.Menu;
19  import android.view.MenuItem;
20  import android.view.View;
21  import android.widget.AdapterView;
22  import android.widget.EditText;
23  import android.widget.FrameLayout;
24  import android.widget.Gallery;
25  import android.widget.ImageView;
26  import android.widget.SimpleAdapter;
27  import android.widget.TextView;
28  import android.widget.AdapterView.OnItemSelectedListener;
29  import android.widget.SimpleAdapter.ViewBinder;
30
31  public class picGallery extends Activity {
32    Gallery gal;
33    TextView caption;
34    private String filename;
35    private FrameLayout fl;
36    private EditText input;
37    private AlertDialog ad;
38
39    @Override
40    protected void onCreate(Bundle savedInstanceState) {
41      // TODO Auto-generated method stub
42
43      input = new EditText(this);
44
45      fl = new FrameLayout(this);
46      fl.addView(input, new FrameLayout.LayoutParams(
47          FrameLayout.LayoutParams.FILL_PARENT,
48          FrameLayout.LayoutParams.WRAP_CONTENT));
49
50      ad = new AlertDialog.Builder(this).setView(fl).setTitle(
51          "Input␣Filename␣finishing␣with␣.jpg").setCancelable(false)
52          .setPositiveButton("OK", new DialogInterface.OnClickListener() {
53
54            @Override
55            public void onClick(DialogInterface arg0, int arg1) {
56              // TODO Auto-generated method stub
57              filename = input.getText().toString();
58              if ((filename != null) && (filename.endsWith(".jpg"))) {
59                Intent i = new Intent(picGallery.this,
60                    CameraPreview.class);
61                i.putExtra("filename", filename);
62                startActivityForResult(i, 1);
63                arg0.dismiss();
64              } else
65                arg0.cancel();
66
67            }
68
```

```
69          }).create();
70
71      this.setContentView(R.layout.galery);
72      gal = (Gallery) this.findViewById(R.id.gallery);
73
74      gal.setOnItemSelectedListener(new OnItemSelectedListener() {
75
76        @SuppressWarnings("unchecked")
77        @Override
78        public void onItemSelected(AdapterView<?> arg0, View arg1,
79            int arg2, long arg3) {
80          caption
81              .setText(((File) ((HashMap<String, File>) arg0
82                  .getAdapter().getItem(arg2)).get("filename"))
83                  .getName());
84
85
86        }
87
88        @Override
89        public void onNothingSelected(AdapterView<?> arg0) {
90          // TODO Auto-generated method stub
91          caption.setText("");
92        }
93
94      });
95
96      caption = (TextView) findViewById(R.id.caption);
97      this.setTitle("Picture␣Gallery");
98      super.onCreate(savedInstanceState);
99    }
100
101   @Override
102   public boolean onCreateOptionsMenu(Menu menu) {
103     menu.add(1, 1, 1, "Take␣Photo").setIcon(R.drawable.camera);
104     menu.add(2, 2, 1, "Delete␣Photo").setIcon(R.drawable.removepoi);
105     menu.add(3,3,1,"Refresh").setIcon(R.drawable.refresh);
106     if (getIntent().getBooleanExtra("returnvalue", false))
107       menu.add(4,4,1,"Select").setIcon(R.drawable.circle_blue);
108     return super.onCreateOptionsMenu(menu);
109   }
110
111   @SuppressWarnings("unchecked")
112   @Override
113   public boolean onOptionsItemSelected(MenuItem item) {
114     switch (item.getItemId()) {
115     case 4:
116       if (gal.getSelectedItemPosition() <0)
117         return false;
118       Intent i = new Intent();
119       i.putExtra("filename", (((HashMap<String,File>)gal.getSelectedItem()).get("filename")).getAbsolutePath
              ());
120       setResult(Activity.RESULT_OK,i);
121       this.finish();
122       return true;
123     case 3:
124       populateGallery();
125       break;
126     case 2:
127       if (gal.getSelectedItemPosition() < 0)
128         return false;
129       HashMap<String, File> map = (HashMap<String, File>) gal
130           .getSelectedItem();
131       map.get("filename").delete();
132       this.caption.setText("");
133       this.populateGallery();
134       return true;
135     case 1:
136
137       ad.show();
138
139       return true;
140     }
141
142     return super.onOptionsItemSelected(item);
143   }
144
145   @Override
146   protected void onResume() {
147     this.populateGallery();
148     super.onResume();
149   }
150
```

```
151     private void populateGallery() {
152
153       File file = new File("/sdcard/android/");
154       if (!file.exists()) {
155         this.finish();
156         return;
157       }
158       List<Map<String, File>> data = new Vector<Map<String, File>>();
159       File fileList[] = file.listFiles(new FileFilter() {
160
161         @Override
162         public boolean accept(File arg0) {
163           return arg0.getName().endsWith(".jpg")
164               || arg0.getName().endsWith(".JPG");
165
166         }
167
168       });
169       for (File findex : fileList) {
170         Map<String, File> map = new HashMap<String, File>();
171         map.put("filename", findex);
172         data.add(map);
173       }
174       String from[] = { "filename" };
175       int to[] = { R.id.galimage };
176       SimpleAdapter sa = new SimpleAdapter(this, data, R.layout.image, from,
177           to);
178       sa.setViewBinder(new ViewBinder() {
179
180         @Override
181         public boolean setViewValue(View arg0, Object arg1, String arg2) {
182           switch (arg0.getId()) {
183           case R.id.galimage:
184
185             ((ImageView) arg0).setImageBitmap(BitmapFactory
186                 .decodeFile(((File) arg1).getAbsolutePath()));
187             return true;
188           }
189           return false;
190         }
191
192       });
193       this.gal.setAdapter(sa);
194
195     }
196
197     @Override
198     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
199       if ((requestCode == 1) && (resultCode == Activity.RESULT_OK)) {
200         populateGallery();
201       }
202       super.onActivityResult(requestCode, resultCode, data);
203     }
204
205
206 }
```

Preview.java

```
1   package gr.ntua.cn.GPS.ImageManagement;
2
3   import java.io.File;
4   import java.io.FileOutputStream;
5   import java.io.IOException;
6
7   import android.content.Context;
8   import android.graphics.Bitmap;
9   import android.graphics.BitmapFactory;
10  import android.graphics.Bitmap.CompressFormat;
11  import android.hardware.Camera;
12  import android.hardware.Camera.PictureCallback;
13  import android.hardware.Camera.ShutterCallback;
14  import android.view.KeyEvent;
15  import android.view.SurfaceHolder;
16  import android.view.SurfaceView;
17
18  public class Preview extends SurfaceView implements SurfaceHolder.Callback {
19    SurfaceHolder mHolder;
20    Camera mCamera;
21    String filename;
22
```

```
23      public void setFilename(String filename) {
24        this.filename = filename;
25      }
26
27      public Preview(Context context) {
28        super(context);
29
30        filename = "";
31        // Install a SurfaceHolder.Callback so we get notified when the
32        // underlying surface is created and destroyed.
33        mHolder = getHolder();
34        mHolder.addCallback(this);
35        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
36      }
37
38      public void surfaceCreated(SurfaceHolder holder) {
39        // The Surface has been created, acquire the camera and tell it where
40        // to draw.
41        mCamera = Camera.open();
42        mCamera.setPreviewDisplay(holder);
43      }
44
45      public void surfaceDestroyed(SurfaceHolder holder) {
46        // Surface will be destroyed when we return, so stop the preview.
47        // Because the CameraDevice object is not a shared resource, it's very
48        // important to release it when the activity is paused.
49        mCamera.stopPreview();
50        mCamera = null;
51      }
52
53      public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
54        // Now that the size is known, set up the camera parameters and begin
55        // the preview.
56        Camera.Parameters parameters = mCamera.getParameters();
57        parameters.setPreviewSize(w, h);
58        mCamera.setParameters(parameters);
59        mCamera.startPreview();
60      }
61
62      public void takePic(int keyCode, KeyEvent event) {
63        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
64          final Object synchObj = new Object();
65          mCamera.takePicture(new ShutterCallback() {
66
67            @Override
68            public void onShutter() {
69              // TODO Auto-generated method stub
70
71            }
72          }, null, new PictureCallback() {
73            public void onPictureTaken(byte[] data, Camera camera) {
74
75              File f = new File("/sdcard/android");
76              if (!f.exists())
77                return;
78              File f2 = new File("/sdcard/android/" + filename);
79              if (f2.exists())
80                f2.delete();
81              FileOutputStream fos = null;
82              try {
83
84                Bitmap bm = BitmapFactory.decodeByteArray(data, 0,
85                    data.length);
86
87                f2.createNewFile();
88                fos = new FileOutputStream(f2);
89
90                bm.compress(CompressFormat.JPEG, 100, fos);
91                fos.write(data);
92                fos.flush();
93                fos.close();
94
95
96              } catch (IOException e) {
97
98                e.printStackTrace();
99              }
100
101           }
102
103         });
104         try {
105           synchronized (synchObj) {
```

```
106             synchObj.wait(50);
107           }
108         } catch (InterruptedException e) {
109           // TODO Auto-generated catch block
110           e.printStackTrace();
111         }
112       }
113
114     }
115
116   }
```

Παράρτημα Β'

Κώδικας Web Εφαρμογής

verifyuser.jsp

```
<%@page import="java.sql.*" %><%@include file = "connectioninfo.jsp" %><%
        int USERID = 0;
        String andUser = request.getParameter("user");
        String andPass = request.getParameter("pass");


        Class.forName(driver).newInstance();
        Connection CON = DriverManager.getConnection(conURL);
        PreparedStatement STATEMENT = CON.prepareStatement("SELECT␣*␣FROM␣Users␣WHERE␣username=?␣AND␣
              password=?");
    STATEMENT.setString(1,andUser);
    STATEMENT.setString(2,andPass);

        ResultSet RS = null;
        try {
            RS = STATEMENT.executeQuery();
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (RS.next()) {
            USERID = RS.getInt(1);
    } else {
    USERID = -1;
    }
    RS.close();
    STATEMENT.close();
    CON.close();

%>
```

verifybuddy.jsp

```
<%@page import="java.sql.*" %><%@include file = "verifyuser.jsp" %><%

    String buddyUserId = request.getParameter("userId");
    int BUDDYUSERID = Integer.parseInt(buddyUserId);

        Connection CON2 = DriverManager.getConnection(conURL);
        PreparedStatement STATEMENT2 = CON2.prepareStatement("SELECT␣*␣FROM␣buddies␣WHERE␣((budy1Id=?␣AND␣
              budy2Id=?)␣OR␣(budy2Id=?␣AND␣budy1Id=?))␣AND␣Status1='Friend'");
        STATEMENT2.setInt(1,BUDDYUSERID);
    STATEMENT2.setInt(2,USERID);
    STATEMENT2.setInt(3,BUDDYUSERID);
    STATEMENT2.setInt(4,USERID);


        ResultSet RS2 = null;
        try {
            RS2 = STATEMENT2.executeQuery();
```

```
17          } catch (Exception e) {
18              e.printStackTrace();
19          }
20          if (RS2.next()) {
21
22      } else {
23      BUDDYUSERID = -1;
24      }
25      RS2.close();
26      STATEMENT2.close();
27      CON2.close();
28
29  %>
```

addbuddy.jsp

```
1   <%@page contentType="text/html"%><%@page import="java.sql.*" %><%@include file = "verifyuser.jsp" %><%
2
3           String userId = request.getParameter("userId");
4           String buddyUserId = request.getParameter("buddyUserId");
5           String Mode = request.getParameter("Mode");
6       Connection con = null;
7       PreparedStatement statement = null;
8
9           if (USERID == Integer.parseInt(userId)) { //BIG IF STARTS
10    try { // TRY STARTS
11          con = DriverManager.getConnection(conURL);
12
13          if (userId.equals(buddyUserId)) {
14        %>FAIL<%
15      }
16      else if (Mode.equals("1")) {
17            statement = con.prepareStatement("SELECT␣*␣FROM␣Buddies␣WHERE␣(budy1Id=?␣AND␣budy2Id=?)␣OR␣(
                  budy1Id=?␣AND␣budy2Id=?)");
18        statement.clearParameters();
19        statement.setInt(1, Integer.parseInt(userId));
20        statement.setInt(2, Integer.parseInt(buddyUserId));
21        statement.setInt(3, Integer.parseInt(buddyUserId));
22        statement.setInt(4, Integer.parseInt(userId));
23              ResultSet rs = statement.executeQuery();
24              if (!rs.next()) {
25                  statement = con.prepareStatement("INSERT␣INTO␣Buddies(budy1Id,budy2Id,Status1,Status2)␣
                        VALUES␣(?,?,'Request␣Made','Requested')");
26          statement.clearParameters();
27          statement.setInt(1, Integer.parseInt(userId));
28                  statement.setInt(2, Integer.parseInt(buddyUserId));
29          statement.executeUpdate();
30                  %>OK<%
31              }
32              else {
33                  %>FAIL<%
34              }
35          } else if (Mode.equals("2")) {
36              statement = con.prepareStatement("UPDATE␣Buddies␣SET␣Status1='Friend',␣Status2='Friend'␣WHERE␣(
                      budy1Id=?␣AND␣budy2Id=?)␣OR␣(budy1Id=?␣AND␣budy2Id=?)");
37              statement.clearParameters();
38        statement.setInt(1, Integer.parseInt(userId));
39        statement.setInt(2, Integer.parseInt(buddyUserId));
40        statement.setInt(3, Integer.parseInt(buddyUserId));
41        statement.setInt(4, Integer.parseInt(userId));
42            statement.executeUpdate();
43              %>OK<%
44          } else if (Mode.equals("3")) {
45              statement = con.prepareStatement("DELETE␣FROM␣Buddies␣WHERE␣(budy1Id=?␣AND␣budy2Id=?)␣OR␣(
                      budy1Id=?␣AND␣budy2Id=?)");
46        statement.clearParameters();
47        statement.setInt(1, Integer.parseInt(userId));
48        statement.setInt(2, Integer.parseInt(buddyUserId));
49        statement.setInt(3, Integer.parseInt(buddyUserId));
50        statement.setInt(4, Integer.parseInt(userId));
51              statement.executeUpdate();
52
53              %>OK<%
54          } else {
55         %>FAIL<%
56      }
57  } // TRY END
58  catch (Exception e) {
59  %>FAIL<%
60  }
```

```
finally {
  if (statement!=null)
    statement.close();
  if (con!=null)
    con.close();
}
} //BIG IF ENDS

%>
```

addpoint.jsp

```
<%@page contentType="text/html"%><%@page import="java.sql.*" %><%@include file = "verifyuser.jsp" %><%
        String userId = request.getParameter("userId");
        String Longitude = request.getParameter("Longitude");
        String Latitude = request.getParameter("Latitude");
        String Description = request.getParameter("Description");
        String Mode = request.getParameter("Mode");
        String pointId = request.getParameter("pointId");
        boolean Used = Boolean.parseBoolean(request.getParameter("Used"));
        boolean Public = Boolean.parseBoolean(request.getParameter("Public"));
      String urlStr = request.getParameter("Url");
      PreparedStatement statement = null;
    Connection con = null;
        if (USERID == Integer.parseInt(userId)) { // BIG IF STARTS
    try { //Big try



        //Class.forName(driver).newInstance();
        con = DriverManager.getConnection(conURL);

        String querystr = null;
        if (Mode.equals("1")) {
      statement = con.prepareStatement("INSERT␣INTO␣Points␣(userId,Longitude,Latitude,Description,Used,
          Public,Url)␣"+
        "VALUES␣(?,?,?,?,?,?,?)");
      statement.clearParameters();
      statement.setInt(1,Integer.parseInt(userId));
      statement.setDouble(2,Double.parseDouble(Longitude));
      statement.setDouble(3,Double.parseDouble(Latitude));
      statement.setString(4,Description);
      statement.setBoolean(5,Used);
      statement.setBoolean(6,Public);
      statement.setString(7,urlStr);
        }
        else if (Mode.equals("2")) {
      statement = con.prepareStatement("UPDATE␣Points␣SET␣"+
      "userId=?,␣Longitude=?,␣Latitude=?,Description=?,Used=?,Public=?,Url=?␣"+
      "WHERE␣pointId=?␣AND␣userId=?");
      statement.clearParameters();
      statement.setInt(1,Integer.parseInt(userId));
      statement.setDouble(2,Double.parseDouble(Longitude));
      statement.setDouble(3,Double.parseDouble(Latitude));
      statement.setString(4,Description);
      statement.setBoolean(5,Used);
      statement.setBoolean(6,Public);
      statement.setString(7,urlStr);
      statement.setInt(8,Integer.parseInt(pointId));
      statement.setInt(9,USERID);
        }
    else if (Mode.equals("3")) {
            statement= con.prepareStatement("DELETE␣FROM␣Points␣WHERE␣pointId=?␣AND␣userId=?");
      statement.clearParameters();
      statement.setInt(1,Integer.parseInt(pointId));
      statement.setInt(2,USERID);
    }
            statement.executeUpdate();
            %>OK<%
    }
    catch (Exception e) {
     %>FAIL<%
    }
    finally {
      if (statement != null)
        statement.close();
      if (con != null)
        con.close();
    }
 } // BIG IF STOPS
```

```
68      %>
```

getpoint.jsp

```
1   <%@page contentType="text/xml"%>
2   <%@page import="java.sql.*" %>
3   <%@include file = "verifyuser.jsp" %>
4   <Message>
5   <Status>OK</Status>
6   <PointList>
7       <%
8       String userId = request.getParameter("userId");
9       if (USERID == Integer.parseInt(userId)) { // START BIG IF
10          Connection con = DriverManager.getConnection(conURL);
11          PreparedStatement statement = con.prepareStatement("SELECT␣*␣FROM␣Points␣WHERE␣userId=?");
12      statement.setInt(1,USERID);
13
14          ResultSet rs = null;
15
16          try {
17              rs = statement.executeQuery();
18          } catch (Exception e) {
19              e.printStackTrace();
20          }
21          while (rs.next()) {
22              String pointId = rs.getString(1);
23              String Longitude = rs.getString(3);
24              String Latitude = rs.getString(4);
25              String Description = rs.getString(5);
26      boolean Used = rs.getBoolean(6);
27      boolean Public = rs.getBoolean(7);
28      String urlStr = rs.getString(8);
29              %>
30              <Point>
31              <pointId><%=pointId%></pointId>
32              <Longitude><%=Longitude%></Longitude>
33              <Latitude><%=Latitude%></Latitude>
34              <Description><%=Description%></Description>
35      <Used><%=Used%></Used>
36      <Public><%=Public%></Public>
37      <Url><%=urlStr%></Url>
38              </Point>
39              <%
40          }
41      rs.close();
42      statement.close();
43      con.close();
44    } // END BIG IF
45      %>
46  </PointList>
47  </Message>
```

closestlocations.jsp

```
1   <%@page contentType="text/xml"%>
2   <%@page import="java.sql.*" %>
3   <%@include file = "verifyuser.jsp" %>
4   <PointList>
5       <%
6           String myuserId = request.getParameter("myuserId");
7           String userIds[] = request.getParameterValues("userId");
8           String longi = request.getParameter("Longitude");
9           String lati = request.getParameter("Latitude");
10
11  if (USERID == Integer.parseInt(myuserId)) { // BEGIN BIG IF
12    float radius = Float.parseFloat(request.getParameter("Radius"));
13          System.out.println("Simio␣" +longi+"␣," + lati + "␣Radius:"+radius);
14          String querystr = null;
15          querystr = "SELECT␣*␣FROM␣Points␣WHERE␣";
16          if (myuserId != null) {
17              querystr = querystr + "(userId=?␣AND␣Used=1)"; //" + myuserId + " AND Used=1)";
18          }
19
20          if (userIds != null) {
21              int length = userIds.length;
22              // Create the querystr
```

```
23
24              for (int i = 0; i < length; i++) {
25                  if ((i == 0) && (myuserId == null)) {
26                      querystr = querystr + "(userId=? AND Public=1)"; // + userIds[i];
27                  } else {
28                      querystr = querystr + " OR (userId=? AND Public=1)"; // + userIds[i]+" AND Public=1)";
29                  }
30              }
31          }
32          // The querystr is ready
33          //Class.forName(driver).newInstance();
34          Connection con = DriverManager.getConnection(conURL);
35          PreparedStatement statement = con.prepareStatement(querystr);
36      int offset = 0;
37      if (myuserId !=null) {
38        statement.setInt(1,Integer.parseInt(myuserId));
39        offset = 1;
40      }
41      if (userIds != null) {
42        int length = userIds.length;
43        for (int i=0;i<length;i++) {
44          statement.setInt(i+1+offset,Integer.parseInt(userIds[i]));
45        }
46      }
47          ResultSet rs = statement.executeQuery();
48
49
50          try {
51
52              double loclati = Double.parseDouble(lati);
53              double loclongi = Double.parseDouble(longi);
54              String retDesc = "no Point of Interest";
55              String retUrl = null;
56      double retlati = 0, retlongi = 0;
57
58              double distance = -1, tempDist;
59              double longt, lat;
60              double a, c;
61              String descr, UrlStr;
62              double dLat, dLon;
63
64              while (rs.next()) {
65
66                  longt = rs.getDouble("Longitude");
67                  lat = rs.getDouble("Latitude");
68                  descr = rs.getString("Description");
69          UrlStr = rs.getString("Url");
70
71                  dLat = Math.toRadians(lat - loclati);
72                  dLon = Math.toRadians(longt - loclongi);
73
74                  a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
75                          Math.cos(Math.toRadians(loclati)) * Math.cos(Math.toRadians(lat)) *
76                          Math.sin(dLon / 2) * Math.sin(dLon / 2);
77
78                  c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
79                  tempDist = 6371009 * c;
80
81          if ((tempDist < distance) || (distance == -1)) {
82                      distance = tempDist;
83                      retDesc = descr;
84                      retlati = lat;
85                      retlongi = longt;
86          retUrl = UrlStr;
87
88                  }
89
90
91                  if (tempDist <= radius) {
92
93
94          // write the data to the xml file
95          System.out.println("Simio!\n");
96           %>
97  <Point>
98  <Longitude><%=longt%></Longitude>
99  <Latitude><%=lat%></Latitude>
100 <Description><%=descr%></Description>
101 <Distance><%=tempDist%></Distance>
102 <%
103   if (UrlStr != null) {
104    %><HasUrl>Yes</HasUrl>
105         <Url><%=UrlStr%></Url><%
```

```
106       } else {
107          %><HasUrl>No</HasUrl><%
108       }
109   %>
110   </Point><%
111
112                   }
113               }
114
115      if (distance > radius) {
116     System.out.println("There was no distance inside radius");
117   %><Point>
118       <Longitude><%=retlongi%></Longitude>
119       <Latitude><%=retlati%></Latitude>
120       <Description><%=retDesc%></Description>
121       <Distance><%=distance%></Distance>
122       <%
123     if (retUrl != null) {
124      %><HasUrl>Yes</HasUrl>
125          <Url><%=retUrl%></Url><%
126       } else {
127          %><HasUrl>No</HasUrl><%
128       }
129
130   %></Point><%
131   }
132
133
134           } catch (Exception e) {
135               e.printStackTrace();
136           }
137
138
139     if (statement != null)
140       statement.close();
141     if (con != null)
142       con.close();
143
144   } // CLOSE BIG IF
145
146       %>
147   </PointList>
```

login.jsp

```
1   <%@page contentType="text/html"%><%@page import="java.sql.*" %><%@include file = "connectioninfo.jsp" %><%
2           int userId = 0;
3           String andUser = request.getParameter("user");
4           String andPass = request.getParameter("pass");
5           Class.forName(driver).newInstance();
6           Connection con = DriverManager.getConnection(conURL);
7           PreparedStatement statement = con.prepareStatement("SELECT * FROM Users WHERE username=? AND
                password=?");
8           statement.setString(1,andUser);
9       statement.setString(2,andPass);
10
11          ResultSet rs = null;
12          try {
13              rs = statement.executeQuery();
14          } catch (Exception e) {
15              e.printStackTrace();
16          }
17          if (rs.next()) {
18              userId = rs.getInt(1);
19  %>AUTH OK <%=userId%><%
20  } else {
21  %>AUTH FAILED<%         }
22          rs.close();
23      statement.close();
24      con.close();
25  %>
```

uploadimage.jsp

```
1   <%@ page contentType="text/html;charset=windows-1252"%>
2   <%@ page import="org.apache.commons.fileupload.DiskFileUpload"%>
```

```
<%@ page import="org.apache.commons.fileupload.FileItem"%>
<%@ page import="java.util.List"%>
<%@ page import="java.util.Iterator"%>
<%@ page import="java.io.File"%>
<%@page import="java.sql.*" %>
<%@include file = "connectioninfo.jsp" %>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Process File Upload</title>
</head>
<%
        System.out.println("Content Type ="+request.getContentType());

        DiskFileUpload fu = new DiskFileUpload();
        // If file size exceeds, a FileUploadException will be thrown
        fu.setSizeMax(1000000);

        List fileItems = fu.parseRequest(request);
        Iterator itr = fileItems.iterator();

    FileItem uploadedFile = null;
    String userId = null,pointId = null, user = null, pass = null;
        while(itr.hasNext()) {
          FileItem fi = (FileItem)itr.next();

          //Check if not form field so as to only handle the file inputs
          //else condition handles the submit button input
          if(!fi.isFormField()) {
            uploadedFile = fi;
        System.out.println("\nNAME: "+fi.getName());
            System.out.println("SIZE: "+fi.getSize());
          }
          else {
    String name = fi.getFieldName();
    String value = fi.getString();
    System.out.println("\nField Name: "+ name);
    System.out.println("Value: " +value);
    if (name.equals("userId"))
      userId = value;
    if (name.equals("pointId"))
      pointId = value;
    if (name.equals("user"))
      user = value;
    if (name.equals("pass"))
      pass = value;
         }
        }

    int USERID;
        Class.forName(driver).newInstance();
        Connection CON = DriverManager.getConnection(conURL);
        Statement STATEMENT = CON.createStatement();
        String QUERYSTR = "SELECT * FROM Users WHERE username='" + user + "' AND password='" + pass + "'";
        ResultSet RS = null;
        try {
            RS = STATEMENT.executeQuery(QUERYSTR);
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (RS.next()) {
            USERID = RS.getInt(1);
    } else {
    USERID = -1;
    }


  if (USERID != -1) {

  File directory = new File(application.getRealPath("/")+"userdata/"+userId);
  if (!directory.exists())
    directory.mkdir();
  File file = new File(application.getRealPath("/")+"userdata/"+userId+"/"+pointId+".jpg");
  uploadedFile.write(file);


%>
<body>
Upload Successful!!
<% } %>
</body>
</html>
```

Παράρτημα Γ’

Create Statements Βάσης Δεδομένων

```
1  CREATE DATABASE `android` /*!40100 DEFAULT CHARACTER SET utf8 */;
2
3  DROP TABLE IF EXISTS `android`.`users`;
4  CREATE TABLE `android`.`users` (
5    `userId` int(10) unsigned NOT NULL auto_increment,
6    `username` varchar(45) NOT NULL,
7    `password` varchar(45) NOT NULL,
8    PRIMARY KEY (`userId`)
9  ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
10
11 DROP TABLE IF EXISTS `android`.`points`;
12 CREATE TABLE `android`.`points` (
13   `pointId` int(10) unsigned NOT NULL auto_increment,
14   `userId` int(10) unsigned NOT NULL,
15   `Longitude` double NOT NULL,
16   `Latitude` double NOT NULL,
17   `Description` varchar(100) NOT NULL,
18   `Used` tinyint(1) NOT NULL,
19   `Public` tinyint(1) NOT NULL,
20   `Url` varchar(1024) default NULL,
21   PRIMARY KEY (`pointId`),
22   KEY `FK_points_1` (`userId`),
23   CONSTRAINT `FK_points_1` FOREIGN KEY (`userId`) REFERENCES `users` (`userId`)
24 ) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8;
25
26 DROP TABLE IF EXISTS `android`.`buddies`;
27 CREATE TABLE `android`.`buddies` (
28   `budy1Id` int(10) unsigned NOT NULL,
29   `budy2Id` int(10) unsigned NOT NULL,
30   `Status1` varchar(45) NOT NULL,
31   `Status2` varchar(45) NOT NULL,
32   PRIMARY KEY (`budy1Id`,`budy2Id`),
33   KEY `FK_buddies_2` (`budy2Id`),
34   CONSTRAINT `FK_buddies_1` FOREIGN KEY (`budy1Id`) REFERENCES `users` (`userId`),
35   CONSTRAINT `FK_buddies_2` FOREIGN KEY (`budy2Id`) REFERENCES `users` (`userId`)
36 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```