ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΜΕΣΟΛΟΓΓΙΟΥ
ΠΑΡΑΡΤΗΜΑ ΝΑΥΠΑΚΤΟΥ
ΤΜΗΜΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ &
ΔΙΚΤΥΩΝ

Πτυχιακή εργασία

Θέμα:

Ανάπτυξη κατανεμημένης εφαρμογής παρακολούθησης ασθενών με κινητές συσκευές

Κυριάκου Χρήστος Α.Μ. 449

ΕΠΙΒΛΕΠΩΝ: Παναγιώτης Αλεφραγκής – Καθηγητής Εφαρμογών

ΝΑΥΠΑΚΤΟΣ 2013

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Ναύπακτος **Ημερομηνία**

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

Ονοματεπώνυμο **Υπογραφή**

1.

2.

3.

Περίληψη Πτυχιακής

Ανάπτυξη κατανεμημένης εφαρμογής παρακολούθησης ασθενών με κινητές συσκευές

Η Πτυχιακή έχει σκοπό να παρουσιάσει και να αναλύσει την επίτευξη και τις προσδοκίες εφαρμογής και παρακολούθησης ασθενών μέσων κινητών συσκευών. Η εφαρμογή θα παρακολουθεί, καταγράφει, και αποδίδει μετρήσεις της κατάστασης του ασθενή με στόχο η σύγχρονη τεχνολογία να βοηθήσει την παρακολούθηση του ασθενούς από τον θεράποντα Ιατρό του, χωρίς να χρειάζεται ο ασθενής να επισκέπτεται τον Ιατρό του είτε λόγω απόστασης ή και λόγω κινητικών προβλημάτων ή και ακόμα προς αποφυγή άσκοπης μετάβασης του ασθενή στο νοσοκομείο.

Η κατασκευή της εφαρμογής βασίζεται σε τεχνολογίες ανοιχτού λογισμικού (open source). Για την δημιουργία της εφαρμογής στο κινητό του ασθενή καθώς και του γιατρού θα χρησιμοποιηθεί λογισμικό android SDK (Software Development kit), που χρησιμοποιεί Java ως γλώσσα προγραμματισμού. Για την επικοινωνία, συλλογή και καταγραφή δεδομένων και ιστορικού θα είναι υπεύθυνος ο server.

Ο server είναι υπεύθυνος για να αποθηκεύσει τα δεδομένα που παρέλαβε στην βάση του ασθενή και να ενημερώσει τον γιατρό για όποιες αλλαγές έχουν γίνει από την τελευταία φορά που έλεγξε τον ασθενή. Ο γιατρός έχει την δυνατότητα να ζητήσει το ιστορικό του ασθενή από τον server και να δημιουργήσει τα γεγονότα πού θα αποθηκευτούν στο server που με την σειρά του θα μεταβιβάσει στον ασθενή. Τέλος ο ασθενής έχει την δυνατότητα αναζήτησης της θέση του χωρίς την χρήση τεχνολογίας GPS άλλα με Location-Based Services ώστε να μπορεί να βρει την περιοχή που βρίσκεται και να εντοπίσει φαρμακεία στην περιοχή.

Οι υπηρεσίες που παρέχονται στον ασθενή:

- Ασύγχρονη επικοινωνία από τον ιατρό προς τον ασθενή.
- Καταγραφή μετρήσεων από ιατρικές συσκευές (π.χ. πιεσόμετρο). Το κινητό θα λαμβάνει δεδομένα από συσκευές μέσω Bluetooth και θα τις μεταβιβάζει στον server.
- Χρήση υπηρεσίας Location-Based Service για εύρεση θέσης του ασθενούς χωρίς την χρήση GPS και εύρεση φαρμακείων στην περιοχή.

- Δυνατότητα υπενθύμισης του ασθενούς για την φαρμακευτική αγωγή του.

Όλες αυτές οι υπηρεσίες έχουν δημιουργηθεί και εφαρμοστεί σε άλλες εφαρμογές μεμονωμένα, όπως το IHealth της Apple (καταγραφή μετρήσεων, αποθηκεύοντας και διαδίνοντας τις μετρήσεις σε τρίτους) και το MyLocation της Google (εύρεση της τοποθεσίας στο χάρτη με ή χωρίς την χρήση GPS).

Developing distributed application monitoring

patients with mobile devices

The Diploma aims to present and analyze the achievement and aspirations of application and patient, monitoring tools mobile devices. The application will monitor, record, and performs measurements of the patient's condition in order for the technology to help monitor the patient by the physician, without requiring the patient to visit a physician either because of distance or because of mobility problems or/ and to avoid unnecessary transition of patient in hospital.

The construction of the application based on open source technologies (open source). To create the application on his mobile patient and the physician will utilize software android SDK (Software Development kit), which uses Java as a programming language. For communication, collecting and recording data and history will be responsible the server.

The server is responsible for storing the data received from the patient and informs the doctor about any changes that have been made since the last time he has checked the patient. The doctor has the opportunity to retrieve the patient's history from the server and create the events which will be stored on the server which in turn will be send to the patient. Finally the patient has the possibility to search his position without using GPS technology with other Location-Based Services that can find the area to identify the location of pharmacies.

The services provided to the patient:

- Asynchronous communication by the physician to the patient.
- Record measurements from medical devices (sphygmomanometer). The mobile will receive data from devices via Bluetooth and will pass it on to the server.
- Use service Location-Based Service for finding a patient without the use of GPS and finding pharmacies in the area.
- Capability of patient reminders for medication.

All these services have been developed and applied to other applications individually as IHealth of the Apple (recording measurements and saving and also sending measurements

to third parties) and MyLocation of Google (for finding the location on the map with or without using GPS and union with Google Map).

Περιεχόμενα

ΕΥΧΑΡΙΣΤΙΕΣ

Είναι με ιδιαίτερη χαρά πού παρουσιάζω την πτυχιακή μου εργασία με θέμα την ανάπτυξη κατανεμημένης εφαρμογής παρακολούθησης ασθενών με κινητές συσκευές.

Με τις εξετάσεις της σχολής να αποτελούν παρελθόν, θέλω να εκφράσω τις ευχαριστίες μου στον καθηγητή κύριο Αλεφραγκή Παναγιώτη για την πολύτιμη βοήθεια του τόσο στην επιλογή του θέματος, όσο και για την βοήθεια στην διάρκεια της συγγραφής και διατύπωσης της πτυχιακής μου.

Επίσης θερμές ευχαριστίες στην οικογένεια μου, γονείς και αδέλφια, που έδειξαν κατανόηση στις στιγμές που η ένταση στην εξεύρεση των απαντήσεων στα πολλά ερωτήματα της πρακτικής εφαρμογής των παραμέτρων του συστήματος απαιτούσε και κατανόηση και ανοχή στα ξεσπάσματα δυσφορίας, και συμμετοχή στην χαρά των αποτελεσματικών εφαρμογών της πτυχιακής μου.

Ευχαριστίες εκφράζω και στους συμφοιτητές, τις φίλες και φίλους που με την θετική συμπεριφορά τους, και με την φυσική τους παρουσία ή την παρουσία τους μέσω διαδικτύου, συνέτειναν στην προσπάθεια μου να τελειώσω την πτυχιακή μου

1

1.1 Εισαγωγή

Η ανάπτυξη αυτής της εφαρμογής θα γίνει για το open source λειτουργικό Android για έξυπνα κινητά τηλέφωνα – Smart phones, το οποίο θα εξετάσουμε στην συνέχεια.

Τα Smart phones έχουν ήδη αντικαταστήσει τα παλιά κινητά, κάνοντας τα ένα μέρος της καθημερνής μας ζωής. Η μεγάλη διαφορά με τα παλιά κινητά είναι ότι έχουν ένα ολοκληρωμένο λειτουργικό σύστημα, σύνδεση στο διαδίκτυο μέσω τεχνολογίας κινητής τηλεφωνίας 3G (third generation mobile telecommunications) ή και ασύρματου δικτύου Wi-Fi, παγκόσμιο σύστημα εντοπισμού - GPS (Global Positioning System) και δυνατό επεξεργαστή. Το Android έχει κερδίσει ένα μεγάλο μέρος στην ελληνική αγορά των smart phones, και μας προσφέρει μια ανεκτή πλατφόρμα για ανάπτυξη εφαρμογών.

Το θέμα που μας απασχολεί είναι η επικοινωνία ανάμεσα σε ασθενή και γιατρό μέσω κατανεμημένης εφαρμογής σε κινητό τηλέφωνο. Αυτή η εφαρμογή έχει σκοπό να προσφέρει μια ασύγχρονη επικοινωνία ανάμεσα αυτών των δύο, με τον γιατρό να παρακολουθεί την κατάσταση του ασθενούς και να δίνει οδηγίες για την φαρμακευτική αγωγή και τις εξετάσεις του ασθενή, που θα κάνει στον χώρο του, μέσω συσκευών (π.χ. πιεσόμετρο).

Η εφαρμογή θα κρατά ιστορικό για την πορεία του ασθενούς την οποία ο γιατρός θα μπορεί ανά πάσα στιγμή να ελέγξει.

Τέλος ο ασθενής θα έχει την δυνατότητα να βρίσκει την θέση στην οποία βρίσκεται στον χάρτη, και να του παρέχεται δυνατότητα να βρει φαρμακεία και νοσοκομεία στην περιοχή που βρίσκεται, και την δήλωση της θέσης του σε περίπτωση επείγοντος περιστατικού.

1.2 Γιατί μια ασύγχρονη επικοινωνία;

Η σύγχρονη επικοινωνία, όπως η ομιλία, όλα τα άτομα πρέπει να είναι παρόν. Αντίθετα στην ασύγχρονη επικοινωνία, όπως το ηλεκτρονικό ταχυδρομείο, δεν χρειάζεται τα άτομα να είναι παρόν. Είναι βασικό ο γιατρός και ο ασθενής να μην χρειάζονται να είναι παρόν τη ίδια χρονική στιγμή και να μην μπορούν να επικοινωνήσουν λόγω διαφορετικού ωραρίου.

Θέλουμε ο γιατρός να είναι σε θέση να δίνει οδηγίες και ιατρικές συμβουλές τις οποίες ο ασθενής λαμβάνει άμεσα και να τις ελέγχει όποτε αυτός επιθυμεί. Για το σκοπό αυτό θέλουμε ένα είδος δεδομένων όπου ο γιατρός θα δημιουργεί και θα στέλνει στον ασθενή εύκολα και απλά, όλες τις απαραίτητες σημειώσεις που χρειάζεται ο ασθενής.

Επειδή στην ασύγχρονη επικοινωνία οι χρήστες μπορούν να μην είναι διαθέσιμοι την ίδια χρονική στιγμή, τα δεδομένα πρέπει να αποθηκεύονται κάπου αλλού όπου οι χρήστες μπορούν να έχουν πρόσβαση, με άλλα λόγια χρειαζόμαστε ένα σύστημα διαχείρισης βάσης δεδομένων (Database Management System - DBMS). Το DBMS μας επιτρέπει να δημιουργήσουμε και να διαχωρίσουμε μια βάση δεδομένων που θα περιέχει τα δεδομένα που θέλουμε. Η επικοινωνία στο DBMS γίνεται με ένα είδος ερωτημάτων, τα λεγόμενα Queries (Query Languages) όπου μας επιτρέπουν να αναζητήσουμε, να εισάγουμε, ή να διαγράψουμε δεδομένα. Όλα τα στοιχεία θα πρέπει να περνάνε μέσα από την βάση δεδομένων για την καταγραφή του ιστορικού.

Τα δεδομένα πρέπει να περιέχουν τις βασικές πληροφορίες και να είναι οργανωμένα κατάλληλα για την εύκολη πρόσβαση. Παρόλα αυτά το ποιό κρίσιμο σημείο είναι ο χρήστης να μην χρειάζεται να ασχοληθεί περισσότερο από μερικές πληκτρολογήσεις στη εφαρμογή.

Το κινητό είναι το μέσο που αναλαμβάνει να παίρνει τα δεδομένα από τον χρήστη και να τα προετοιμάζει για την αποστολή τους. Το κινητό πρέπει να ενημερώνεται για ότι αλλαγές αφορούν τον χρήστη.

Μια βασική δυνατότητα που πρέπει να δίνη η εφαρμογή είναι η εισαγωγή δεδομένων από ιατρικές συσκευές που ο ασθενής θα κατέχει μαζί του ή στον χώρο του,

χρησιμοποιώντας το Bluetooth της συσκευής, αυτά τα δεδομένα θα αποθηκεύονται στην συσκευή και θα αποστέλλονται στην κεντρική βάση δεδομένων.

Επίσης η εφαρμογή παρέχει την δυνατότητα ευρέσεως τις θέσης του ασθενούς μέσω GPS για παροχή Location-Based Services και την εμφάνιση χάρτη στον οποίον φαρμακεία και νοσοκομεία βρίσκονται κοντά. Ακόμα την γρήγορη εντόπιση τους σε περίπτωση έκτακτης ανάγκης.

2

2.1 Android

Το Android είναι ένα ενσωματωμένο λειτουργικό σύστημα για κινητές συσκευές που αποτελείτε από τέσσερα επίπεδα με πέντε στοιχεία όπως φαίνετε στο παρακάτω σχήμα. Οι εφαρμογές γράφονται σε Java Framework, άλλα όχι σε Java. Αυτό σημαίνει πως σε πολλές βιβλιοθήκες είτε δεν υπάρχουν ή έχουν αντικατασταθεί με βιβλιοθήκες του Android. Ένας "Πυρήνας Linux" (Linux Kernel) και μια συλλογή από βιβλιοθήκες C/C++ είναι διαθέσιμες μέσω του Application Framework .

Ποιό συγκεκριμένα:

2.2 Πυρήνας Linux (Linux Kernel):

Οι βασικές υπηρεσίες (επεξεργασία, διαχείριση μνήμης, ασφάλεια, δίκτυο, διαχείριση μπαταρίας) διαχειρίζονται μέσω ενός πυρήνα Linux 2.6. Ο πυρήνας είναι το επίπεδο ανάμεσα στο υλικό και τις υπόλοιπες στοίβες.

Βιβλιοθήκες (Libraries): Τρέχουν από πάνω από τον πυρήνα. Το Android περιέχει αρκετές κύριες βιβλιοθήκες C/C++ όπως libc και SSL, και άλλα όπως:

- Βιβλιοθήκη πολυμέσων για ήχο και video.
- Βιβλιοθήκη για διαχώριση οθόνης.
- Βιβλιοθήκη γραφικών περιλαμβανομένου SGL και OpenGL για δισδιάστατα και τρισδιάστατα γραφικά.
- SQLite για υποστήριξη βάσης δεδομένων.

- SSL και Web Kit για ενσωματωμένη web browser και Internet ασφάλεια.

2.3 Android Runtime

Περιλαμβάνει της Βασικές Βιβλιοθήκες (Core Libraries) και την εικονική μηχανή Dalvik (Dalvik Virtual Machine)

2.4 Core Libraries

Παρόλο που όλη η ανάπτυξη γίνεται σε Java, ο Dalvik δεν είναι μια εικονική μηχανή Java (Java Virtual Machine - JVM). Αυτές οι βιβλιοθήκες μας προσφέρουν τις περισσότερες λειτουργίες από τις βασικές βιβλιοθήκες της Java καθώς και βιβλιοθήκες ειδικά για το Android.

2.5 Εικονική μηχανή Dalvik

Είναι μια εικονική μηχανή register-based για να τρέχει πολλαπλές εικονικές μηχανές. Κάθε εφαρμογή τρέχει πολλαπλές διεργασίες αποδοτικά. Ο Dalvik VM εκτελεί αρχεία Dalvik Executable (.dex) που ελαχιστοποιεί την κατανάλωση μνήμης. Στηρίζεται στον πυρήνα Linux για την διαχείριση πολλαπλών διεργασιών και διαχειρίζεται μνήμη χαμηλού επιπέδου.

2.6 Πλαίσιο Εφαρμογής (Application Framework)

Προσφέρει τις κλάσεις που χρησιμοποιούν για την κατασκευή Android applications. Επίσης προσφέρει ένα γενικό abstraction για την πρόσβαση σε υλικό, την διαχείριση του user interface και τους πόρους της εφαρμογής.

Τα βασικά στοιχεία που περιλαμβάνει είναι:

- Views: Μια μεγάλη συλλογή από αντικείμενα που μπορούν να χρησιμοποιήσουν για την κατασκευή του user interface της εφαρμογής, όπως lists, grids, text boxes, buttons, και άλλα πολλά.

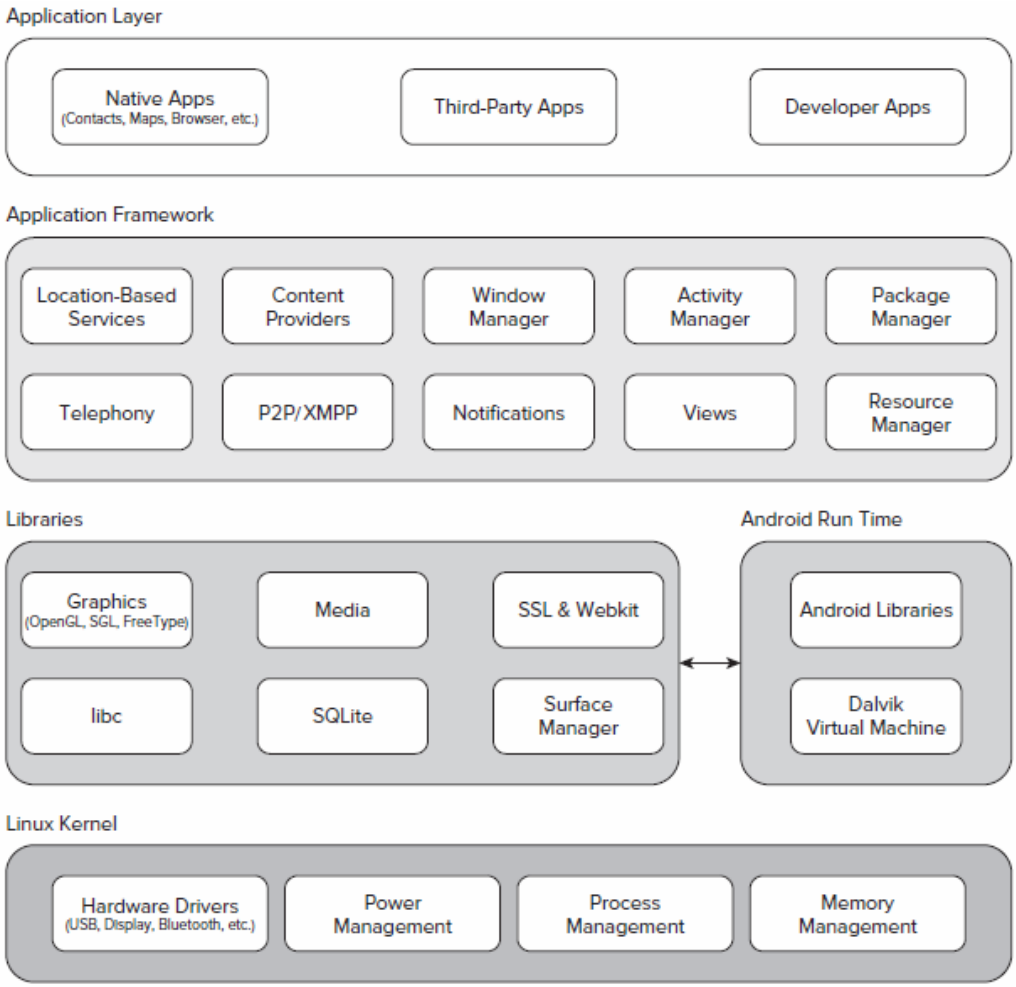
- Παροχέας Περιεχομένου - Content Providers: επιτρέπει την ανταλλαγή δεδομένων μεταξύ εφαρμογών.

- Διαχειριστής Πόρων - Resource Manager: παρέχει πρόσβαση σε πόρους που δεν είναι κώδικας, όπως σχεδιάγραμμα οθόνης σε XML.

- Διαχειριστής Κοινοποιήσεων - Notification Manager: παρέχει μηχανισμούς για την ειδοποίηση του χρήστη.

2.7 Application Layer

Κάθε εφαρμογή βρίσκετε σε αυτό το επίπεδο. Το Application Layer τρέχει μέσα στο Android run time χρησιμοποιώντας τις κλάσεις και τις υπηρεσίες που παρέχονται από το Application Framework.



Σχήμα 2 – 1: Στοίβα Λογισμικού του Android

3

3.1 Προγραμματίζοντας σε Android

Για να κατασκευάσουμε την εφαρμογή χρειαζόμαστε τα κατάλληλα εργαλεία, το λεγόμενο Android Software Development Toolkit (SDK). Το SDK είναι ένα σύνολο από εργαλεία που χρησιμοποιούμε για την κατασκευή εφαρμογών από λογισμικά πακέτα (software package), όπως android.Bluetooth package και android.location package, δηλαδή έτοιμοι κώδικες που μπορούμε να χρησιμοποιήσουμε. Μαζί με το Android SDK θα χρησιμοποιήσουμε και ένα ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse (Integrated Development Environments, IDE). Το Eclipse με τα ADT Plugins μας προσφέρει ένα

περιβάλλον για την κατασκευή και την δοκιμασία των εφαρμογών μέσω εικονικών συσκευών Android (Android Virtual Devices, AVD). Τα AVD μας επιτρέπουν να δοκιμάσουμε τις εφαρμογές σε μια συγκριμένη κινητή συσκευή, είτε Smartphone ή Tablet.

Κάθε εφαρμογή Android αποτελείτε από στοιχεία (components), τα οποία δένονται μέσω του Application Manifest. Το Manifest περιγράφει κάθε στοιχείο και πως αλληλεπιδρούν με την εφαρμογή.

Τα παρακάτω έξι στοιχεία αποτελούν τα δομικά components της εφαρμογής.

- **Activities** Η παρουσίαση της εφαρμογής. Κάθε οθόνη στην εφαρμογή θα είναι προέκταση (Extend) του στοιχείου Activity. Τα Activities χρησιμοποιούν Views για να κατασκευάσουν το Γραφικό περιβάλλον χρήστη (Graphical User Interface) που εμφανίζει τις πληροφορίες και ανταποκρίνεται στον χρήστη.

- **Services** Οι αόρατοι εργάτες της εφαρμογής. Τα Services τρέχουν στο background της εφαρμογής, ενημερώνουν τα data sources, τα ορατά Activities και ενεργοποιούν τα Notifications. Χρησιμοποιούνται κυρίως όταν θέλουμε ορισμένες διεργασίες να συνεχίσουν να εκτελούνται ακόμα και όταν τα Activities είναι ή δεν είναι ορατά ή ενεργά.

- **Content Providers** Κοινόχρηστος χώρος δεδομένων. Τα Content Providers χρησιμοποιούνται για την διαχώριση και την κοινοποίηση της βάσης δεδομένων της εφαρμογής. Ακόμα μπορούν να μοιράσουν τα δεδομένα όχι μόνο στην δικιά μας εφαρμογή άλλα και σε ξένες εφαρμογές.

- **Intents** Μια ενδιάμεση εφαρμογή που περνά μηνύματα στο Framework. Χρησιμοποιώντας τα Intents μπορούμε να στείλουμε μηνύματα στο σύστημα, να ξεκινήσουμε ένα Activity ή Service, δίνοντας την πρόθεση να ξεκινήσει μια εκτέλεση.

- **Broadcast Receivers** Χειρίζονται Broadcast Intents. Αν δημιουργήσουμε και καταχωρήσουμε ένα Broadcast Receiver, η εφαρμογή θα ακούει για αυτά τα Intents και θα ξεκινήσει αυτόματα για να ανταποκριθεί στα Intents που ταιριάζουν με το φίλτρο του.

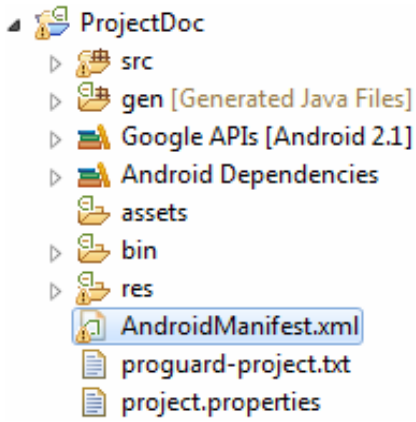
- **Widgets** Οπτικά components της εφαρμογής που μπορούν να προστεθούν στην αρχική οθόνη. Μια ειδική παραλλαγή του Broadcast Receiver, το Widget μας

επιτρέπει να δημιουργήσουμε δυναμικά, διαδραστικά components εφαρμογής για τον χρήστη για να ενσωματωθούν στην οθόνη του.

- **Notifications** Ένα Framework ενημέρωσης του χρήστη. Τα Notifications επιτρέπουν να επισημαίνεται στον χρήστη χωρίς να διακόψουμε την τρέχουσα δραστηριότητα. Προτείνονται για να αποσπούν την προσοχή του χρήστη μέσα από Service ή Broadcast Receiver. Για παράδειγμα όταν ο χρήστης λαμβάνει ένα μήνυμα ή κλήση ενεργοποιείτε κάποιο γεγονός.

3.2 Application Manifest

Κάθε Android project περιλαμβάνει ένα αρχείο με το όνομα AndroidManifest.xml, όπως φαίνεται στην παρακάτω σχήμα. Το Manifest μας επιτρέπει να καθορίσουμε την δομή και τα metadata της εφαρμογής, τα components και τις απαιτήσεις.



Σχήμα 3 – 1: Δομή μιας εφαρμογής

Κάθε component (Activities, Services, Content Providers, Broadcast) που βρίσκεται στην εφαρμογή περιγράφονται με Intent Filters και Premonitions, για να καθοριστεί πως θα αλληλοεπιδρά μεταξύ τους ή και με άλλες εφαρμογές . Το Manifest μας δίνει ορισμένα attributes (γνωρίσματα) για συγκεκριμένα metadata που μπορούν να χρησιμοποιηθούν για ρυθμίσεις ασφαλείας, ή και προσδιορισμό απαιτήσεων υλικού και πλατφόρμας.

Κάθε Manifest ξεκινά με <manifest> tag που περιέχει το χαρακτηριστικό package με το πακέτο της εφαρμογής, το android:version Code που δείχνει πια είναι η έκδοση του

κώδικα (η έκδοση αυτή είναι πάντα ακέραιος αριθμός και χρησιμοποιείται για να ελέγξει την έκδοση του προγράμματος) και το android:version Name που δείχνει την έκδοση του προγράμματος στον χρήστη.

```
<manifest xmlns:android=”http://schemas.android.com/apk/res/amdroid”
      package=”com.chkyriacou”
      android:verionCode=”1”
      android:versionName=”1.0” >
```

Μέσα στο σώμα του <manifest> περιέχονται άλλα tags με τα σημαντικότερα από αυτά να είναι τα εξής:

- <uses-sdk/> Με χαρακτηριστικά android:minSdkVersion, android:targetSdkVersion και android:maxSdkVersion. Το Sdk version δηλώνει το application programming interface level-API level (επίπεδο περιβάλλον εφαρμογής – Application Programming Interface Level). Κάθε API level μας δίνει διαφορετικά εργαλεία αλλά κάθε ανώτερο επίπεδο διαθέτει όλα τα εργαλεία των προηγούμενων επιπέδων. Για παράδειγμα η βιβλιοθήκη CalenderView βρίσκετε στο API Level 11 και απαιτεί android 3 που δεν μπορεί να χρησιμοποιηθεί από smart phones 2.3.5 που έχουν API Level 10. Έτσι εξασφαλίζουμε σε ποιο API Level θέλουμε να τρέχει η εφαρμογή μας ώστε να χρησιμοποιήσουμε τα συμβατά εργαλεία.

```
<uses-sdk android:miniSdkVersion=”7” />
```

- <uses-permission/> Για λόγους ασφαλείας, για να χρησιμοποιήσουμε υπηρεσίες όπως, Bluetooth, Ιντερνέτ, Location-based services ακόμα χρήση SMS θέλουμε “άδεια” από τον χρήστη. Για παράδειγμα μια εφαρμογή με ανοικτό συνέχεα το Bluetooth θα καταναλώσει την μπαταρία γρήγορα ή μπορεί ο χρήστης να μην θέλει να χρησιμοποιήσει το ιντερνέτ λόγω χρεώσεων. Σε περίπτωση που δεν δηλώσουμε permission η εφαρμογή θα βγάλει σφάλμα όταν ο χρήστης επιχειρήσει να τρέξει το κομμάτι που απαιτεί permission.

```
<uses-permission android:name = ”android.permission.INTERNET”/>
<uses-permission android:name = ”android.permission.ACCESS_FINE_LOCATION”/>
<uses-permission android:name = ”android.permission.BLUETOOTH”/>
```

- <application> Περιέχει χαρακτηριστικά για την εφαρμογή όπως το android:icon για το εικόνα της εφαρμογής και το android:label για το όνομα που θα εμφανίζεται στον χρήστη. Όλα τα components περιλαμβάνονται μέσα στο σώμα.

```
<application
        android:icon=”@drawable/ic_laucher”
        android:label=”@string/app_name” >
```

- <uses-library> Με το χαρακτηριστικό android:name μπορούμε να πούμε στο σύστημα να παραλάβει επιπρόσθετες βιβλιοθήκες που θα φορτώσει από το πακέτο.

```
<uses-livrary
        android:name=”com.google.android.maps”/>
```

- <activity> Δηλώνει ένα Activity class που είναι μέσα στο πακέτο της εφαρμογής, αυτή η δήλωση γίνεται μέσω του χαρακτηριστικού android:name που περιέχει το path μέσα στο οποίο βρίσκετε το συγκεκριμένο Activity, μπορούμε να γράψουμε τον τίτλο που θα εμφανίζεται πάνω στην οθόνη χρησιμοποιώντας το χαρακτηριστικό android:label.

- <intent-filter> Βρίσκεται μέσα στο σώμα του Activity. Τα δύο κύρια χαρακτηριστικά του είναι το <action android:name /> και το <category android:name />. Το <action android:name /> επιτρέπει να ορίσουμε μια αναφορά για να καλέσουμε το Activity μέσου της Intent. Το <category android:name /> χρησιμοποιείται για να ορίσουμε τι είδος κατηγορία ανήκει.

```
<activity
        android:name=”.activirties.Intro”
        android:name=”@string/app_name” >
        <intent-filter>
                <action android:name=”android.intent.action.MAIN”/>
```

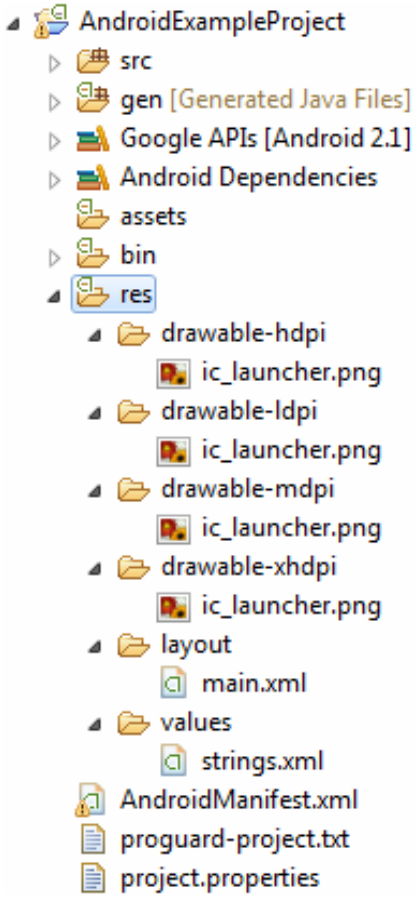
```
                <category android:name="android.intent.category.LAUNCHER"/>
            <Intent-filter/>
        </activity>
```

3.3 Εξωτερίκευση Πόρων

Μια καλή τεχνική ανάπτυξης εφαρμογής είναι να κρατάμε όλους τους non-code πόρους έξω από τον κώδικα μας. Το Android υποστηρίζει την εξωτερίκευση πόρων, από κείμενα, και εικόνες, μέχρι και layouts.

Εξωτερικεύοντας τους πόρους γίνεται ευκολότερη ενημέρωση. Φανταστείτε να έχουμε ένα τίτλο που πρέπει να δηλωθεί δέκα φορές σε διάφορα σημεία στον κώδικα μας. Θα έρεπε να βρούμε όλους του τίτλους και να τους αλλάζουμε. Αντιθέτως έχοντας τον τίτλο Εξωτερικευμένο το μόνο που χρειάζεται να κάνουμε θα είναι να αλλάξουμε τον τίτλο μόνο σε εκείνο το σημείο για να αλλαχτεί σε όλη την εφαρμογή.

Η κάθε εφαρμογή έχει τους πόρους οργανωμένα σε ένα φάκελο με το όνομα res (από το resources). Κάθε φορά που φτιάχνουμε μια εφαρμογή χρησιμοποιώντας ADT wizard (Android Developer Tools) μας φτιάχνει μια ιεραρχική δομή με φακέλους μέσα στο res. values, drawable-hdpi, drawable-mdpi, drawable-ldpi, layout είναι οι κύριοι φάκελοι που συναντάμε. Το values περιέχει μόνο μεταβλητές ενώ το layout περιέχει τις οθόνες που εμφανίζονται στην εφαρμογή. Τα drawable αντιστοιχούν σε εικόνες, τα High-Dots Per Inch, Medium-Dots Per Inch και Low-Dots Per Inch αντιστοιχούν σε διαφορετικές αναλύσεις οθόνης ώστε να χρησιμοποιείτε η κατάλληλη εικόνα ανά ανάλυση. Όλα αυτά συμπεριλαμβάνονται μέσα σε ένα φάκελο με τον όνομα res όπως φαίνεται στην παρακάτω σχήμα.



Σχήμα 3 – 2: Περιεχώμενα του φακέλου res

Περιεχόμενα αρχείου strings.xml

```
<?mxl version=”1.0” encoding=”utf-8”?>
<resources>
	<string name=”hello”>
		Hello World, AndroidExampleActivity!
	</string>
	<string name=”app_name”>
		AndroidExampleProject
	</string>
</resources>
```

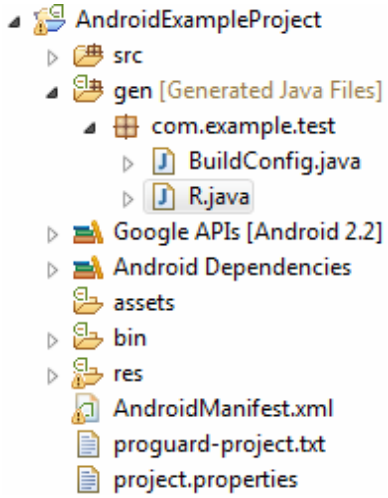
Όπως φαίνεται στο αρχείο string.xml περιέχονται δύο Strings, όπου στο name δηλώνουμε τη ID (hello, app_name) του String περιέχετε μέσα στο tag. Το ίδιο ισχύει και για τα άλλα tags που θα δούμε στην συνέχεια.

Περιεχόμενα αρχείου main.xml

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

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"    android:layout_height="match_parent"
android:orientation="vertical" >

          <TextView

              android:id="@+id/textView1"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:text="@string/hello" />

          <Button

              android:id="@+id/button1"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Button" />

</LinearLayout>
```

Το main.xml είναι ένα layout που μπορεί ένα Activity να χρησιμοποιήσει ως Content View που περιέχει δυο πόρους Views, ένα Text View με ID textView1 και ένα Button με ID button1. Και τα δύο περιέχουν τρία επιπλέον χαρακτηριστικά, layout_width, layout_height και text. Τα layout_width και layout_height καθορίζουν το πλάτος και το ύψος του στοιχείου, χρησιμοποιούν τις σταθερές wrap_content που θα επιτρέψει στο στοιχείο το μέγεθος που ταιριάζει με το περιεχόμενο του , match_parent (ή άλλος fill_parent για API Levels κάτω του 8) που θα μας πάρει τόσο χώρο όσο του επιτρέπεται. Μπορούμε επίσης να δηλώσουμε τα pixels γράφοντας των αριθμό των pixels που θέλουμε να πιάνει στην οθόνη, επειδή οι οθόνες διαφέρουν από συσκευή σε συσκευή είναι καλύτερο να χρησιμοποιούμε το dp (density-independent pixel units - ανεξάρτητη μονάδα πυκνότητας pixel). Τα IDs περιέχουν δυο σύμβολα το παπάκι “@” και το συν “+”. Το “@” ώστε το XML parser να αναγνωρίσει το υπόλοιπο string ως το ID του στοιχείου. Το “+” για να δηλώσουμε πως είναι ένα καινούριος

πόρος που πρέπει να δημιουργηθεί και να προστεθεί μέσα στη R class. Η R class δημιουργείτε αυτόματα από το Android Asset Packaging Tool διαβάζοντας τα XMLs με τους πόρους.



Σχήμα 3 – 3: Περιεχόμενα του φακέλου gen

Περιεχόμενα της R class

```
package com.example.test;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int button1=0x7f050001;
        public static final int textView1=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }

    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Όπως φαίνεται στην κλάση R μας έχει φτιάξει integers με τα ονόματα των IDs που δώσαμε. Αυτές οι integers είναι οι αναφορές που θα χρησιμοποιήσουμε για να αναφερθούμε στους πόρους όπως φαίνεται στο παρακάτω παράδειγμα.

```
package com.example.test;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;

public class AndroidExampleActivity extends Activity {

    /** Called when the activity is first created. */


@Override
public void onCreate(Bundle savedInstanceState) {

     super.onCreate(savedInstanceState);

     setContentView(R.layout.main);

       Button button1 = (Button) findViewById(R.id.button1); TextView textView1 =
                          (TextView) findViewById(R.id.textView1);

       String applicationText = getString(R.string.app_name);  String hello =
       getString(R.string.hello);

    }
}
```

Στο παράδειγμα μας δηλώνουμε το Layout του Activity μέσω της μεθόδου `setContentView` το Layout "main" (R.layout.main). Στην συνέχεια ορίζουμε δύο αντικείμενα τα οποία έχουμε δηλώσει μέσα στο XML του Layout main, το button1 και το textView1 μέσω της μεθόδου `findViewById` που μας βρίσκει τον ένα πόρο View που έχουμε ορίσει στο XML μας, βάση του ID. Μετά πρέπει να κάνουμε Casting στο αντικείμενο που επιστρέφεται από την `findViewById` στον κατάλληλο τύπο για να ορίσουμε τα αντικείμενα μας. Τέλος ορίζουμε τα δυο Strings που έχουμε στο string.xml μας μέσω της `getString` με το ID που θέλουμε.

3.4 Intents και Intent Filters

Τα βασικά στοιχεία μιας εφαρμογής (activities, services, broadcast receivers) ενεργοποιούνται μέσω μηνυμάτων με το όνομα intents. Ένα Intent αντικείμενο είναι μια αφηρημένη περιγραφή δεδομένων για την εργασία που πρέπει να εκτελεστεί ή στην περίπτωση των broadcast receivers, μια περιγραφή για κάτι που γίνεται ή ανακοινώνεται στην ίδια ή διαφορετική εφαρμογή. Υπάρχουν συγκεκριμένοι μηχανισμοί που περνούν τα Intents σε κάθε στοιχείο:

- Στα Activities ένα Intent περνούν στο `Context.startActivity` ή `Activity.startActivityForResult` για να ξεκινήσει ή να δώσει εντολή σε ένα ήδη υπάρχον activity να κάνει κάτι καινούριο.

- Στα Services το Intent περνούν στο `Context.startServices` για να εισάγει ένα service ή να παραδώσει νέες οδηγίες σε ένα τρέχον service. Παρομοίως, ένα Intent μπορεί να περάσει στο `Context.bindServices` για να δημιουργήσει μια σύνδεση μεταξύ του στοιχείου και κάποιου service.

- Στα Broadcasts περνούν από Broadcast μεθόδους (όπως `sendBroadcst`, `Context.sendOrderedBroadcast`, ή `Context.sendStickyBroadcast`) που παραδίδονται σε broadcast receivers που ακούνε για τα συγκεκριμένα Intents.

  Ένα Intent μπορεί να περιέχει τα εξής:

3.4.1 Component Name

Το όνομα του στοιχείου που θα πρέπει να χειριστεί το Intent. Αυτό το πεδίο περιέχει ένα Component Name αντικείμενο που με την σειρά του περιέχει δύο πληροφορίες, το πακέτο του στοιχείου και την κλάση του στοιχείου (και τα δύο με μορφή String).

```
Component Name(String package, String class);

Component Name(Context package, String class);
```

3.4.2 Action

Ένας τίτλος δηλώνει την ενέργεια που μπορεί να εκτελεστεί και στην περίπτωση των Broadcast Intents την ενέργεια που μπορεί να λάβει. Η κλάση Intent ορίζει ένα αριθμό ενεργειών όπως:

Για Activities:

- ACTION_CALL – Ξεκινά μια τηλεφωνική κλήση.
- ACTION_EDIT – Εμφανίζει δεδομένα του χρήστη για επεξεργασία.
- ACTION_MAIN – Το αρχικό Activity.
- ACTION_SYNC – Συγχρονίζει δεδομένα σε ένα server με τα δεδομένα της κινητής συσκευής

Για Broadcast Receiver:

- ACTION_BATTERY_LOW – Μια προειδοποίηση για χαμηλή μπαταρία.
- ACTION_HEADSET_PLUG – Τα ακουστικά έχουν συνδεθεί ή αποσυνδεθεί στην συσκευή.
- ACTION_SCREEN_ON – Η οθόνη έχει ενεργοποιηθεί.
- ACTION_TIMEZONE_CHANGED – Το time zone έχει αλλάξει.

Μπορούμε να ορίσουμε τα δικά μας actions για να ενεργοποιήσουμε τα στοιχεία μέσα στην εφαρμογή μας. Τα actions καθορίζουν πως τα Intents είναι δομημένα μέσα στην εφαρμογή μας.

3.4.3 Data

Αποτελείτε από το URI (Uniform Resource Identifier – Ενιαίος Εντοπιστής Πόρων) των δεδομένων που μπορεί να ενεργήσει και με το MIME (Multipurpose Internet Mail Extensions) τύπο των δεδομένων.

Διαφορετικά actions είναι συνδεμένα με διαφορετικά είδη δεδομένων. Για παράδειγμα, αν ένα action field είναι ACTION_EDIT το data field θα περιέχει το URI του εγγράφου που θα εμφανίσει για επεξεργασία. Για το action ACTION_CALL το data field θα είναι ένα URI tel: με το τηλέφωνο που θα καλέσει. Για το action ACTION_VIEW το data

field περιέχει ένα URI http: με μια διεύθυνση, το Activity θα κατεβάσει και θα εμφανίσει τα περιεχόμενα της διεύθυνσης.

Όταν ένα Intent ταιριάζει σε κάποιο στοιχείο που μπορεί να χειριστεί τα δεδομένα, είναι σημαντικό να ξέρουμε τον τύπο των δεδομένων (MIME) για κάθε URI, ώστε να μην προσπαθήσουμε να εμφανίσουμε για παράδειγμα έναν ήχο.

Σε πολλές περιπτώσεις τα δεδομένα μπορούν να περασθούν μέσα από το URI και κυρίως τα URIs content: , που δηλώνουν ότι τα δεδομένα είναι τοποθετημένα μέσα στην συσκευή και εκλέγονται από έναν content provider που είναι υπεύθυνος για την πρόσβαση των δεδομένων ακόμα και δεδομένα που είναι σε διαφορετικά process. Άλλος τρόπος μπορεί να δηλωθεί μέσω από τα Intent αντικείμενα. Η μέθοδος `setData` προσδιορίζει το URI ενώ η μέθοδος `setType` προσδιορίζει το MEMI. Φυσικά υπάρχει και το `setDataAndType` για να προσδιορίσουμε και τα δύο. Όσο για την ανάγνωση γίνεται μέσω των μεθόδων `getData` και `getType` αντίστοιχα.

3.4.4 Category

Είναι ένα String που περιέχει πρόσθετες πληροφορίες σχετικά με το είδος των στοιχείων που πρέπει να χειριστεί το Intent. Ένα αντικείμενο Intent μπορεί να περιέχει πολλές category περιγραφές, μερικές από αυτές είναι:

- CATEGORY_BROWSABLE Το Activity μπορεί να επικαλεσθεί από τον browser με ασφάλεια για να εμφανίσει τα δεδομένα που αναφέρονται από το link. Για παράδειγμα μια εικόνα η ένα e-mail.

- CATEGORY_GADGET Το Activity μπορεί να ενσωματώσει μέσα του άλλο Activity που φιλοξενεί gadgets.

- CATEGORY_HOME Το Activity εμφανίζει την home screen, την πρώτη οθόνη που βλέπει ο χρήστης όταν ανοίγει την συσκευή του ή όταν το Home Button πατηθεί.

- CATEGORY_LAUNCHER Το Activity μπορεί να είναι το αρχικό Activity ενός task και είναι στο ανώτερο επίπεδο εκκίνησης της εφαρμογής.

- CATEGORY_PREFERENCE Το Activity είναι ένα preference panel.

Μπορούμε να ορίσουμε το Category ενός αντικείμενου Intent με την μέθοδο `addCategory`. Για να πάρουμε τα Categories καλούμε την μέθοδο `getCategory`, επιστρέφοντας ένα αντικείμενο Set που περιέχει όλα τα Categories σε μορφή Strings, ενώ για να διαγράψαμε το Category του αντικειμένου Intent χρησιμοποιούμε την μέθοδο `removeCategory`.

3.4.5 Extras

Key-values ζευγάρια για επιπλέον πληροφορίες που πρέπει να μεταφερθούν στο στοιχείο που θα χειριστεί το Intent. Όπως κάποια Actions είναι συνδεδεμένα με URI δεδομένα, έτσι κάποια είναι συνδεδεμένα με Extras. Για παράδειγμα, το ACTION_TIMEZONE_CHANGE Intent περιέχει το "time-zone" Extra το οποίο προσδιορίζει το καινούριο time zone, και το ACTION_HEADSET_PLUG περιέχει το "state" που δηλώνει αν το headset είναι plugged ή unplugged, και επίσης το "name" για τον τύπο του headset. Για το Action Intent SHOW_COLOR το χρώμα θα οριστεί σε ένα Extra Key-value. Το αντικείμενο Intent περιέχει μεθόδους `put…` για να βάλουμε διαφόρους τύπους δεδομένων και μεθόδους `get…` για την ανάγνωση τους.

3.4.6 Flags

Τα Flags καθοδηγούν το Android πώς να εκκινήσει ένα Activity (για παράδειγμα, σε ποιο task το Activity ανήκει) και πως θα το χειριστεί (για παράδειγμα, αν ανήκει σε κάποια λίστα από πρόσφατα Activities). Όλα τα flags ορισμένα μέσα στην κλάση Intent.

Το Android και οι εφαρμογές που έρχονται μαζί με την πλατφόρμα χρησιμοποιούν Intent αντικείμενα για αποστολή System-Originated Broadcasts και για ενεργοποίηση System-Defined στοιχεία.

3.4.7 Intends

Τα Intents χωρίζονται σε δύο ομάδες:

- Explicit Intents που ορίζουν για ποιό στοιχείο απευθύνετε μέσω του πεδίου Component Name. Χρησιμοποιούνται κυρίως από τον developer της εφαρμογής μέσα από τις κλάσεις.

Παράδειγμα:

```
Intent myExplicitIntent = new
Intent(ActivityClassOne.this, ActivityClassTwo.class);
```

ή

```
Intent myExplicitIntent = new Intent();
final ComponentName cn = new ComponentName("com.example",
"com.example.activities.ActivityClassTwo");
myExplicitIntent.setComponent(com.example.ActivityClassTwo);
```

ή

```
Intent myExplicitIntent = new Intent();
myExplicitIntent.setClass(ActivityClassOne.this, ActivityClassTwo.class);
```

- Implicit Intents Δεν ορίζουν το πεδίο Component Name. Συχνά χρησιμοποιούνται για να ενεργοποιήσουν στοιχεία από άλλες εφαρμογές.

Παράδειγμα:

```
Intent myImplicitIntent = new Intent(Intent.ACTION_VIEW,
                    Uri.parser(“content://contects/people/”));
```

Το Android παραδίδει ένα Explicit Intent σε ένα στιγμιότυπο της ορισθείσης κλάσης. Τίποτα εκτός του Component Name χρησιμοποιείτε για να καθορίσει ποιο στοιχείο θα πάρει το Intent.

Στην περίπτωση των Implicit Intent τα πράγματα είναι διαφορετικά. Χάρη στην απουσία του Component Name το Android πρέπει να βρει το καλύτερο στοιχείο (ή στοιχεία) για να χειριστεί το Intent για activity ή service για να εκτελέσει κάποια ενέργεια ή broadcast receivers για να ανταποκριθεί σε κάποιο broadcast. Για να το κάνει αυτό συγκρίνει τα περιεχόμενα του αντικειμένου Intent με τους Intent Filters.

3.4.8 Intent Filters

Τα Filters είναι δομές που σχετίζονται με στοιχεία, διαφημίζοντας τις ικανότητες των στοιχείων και οριοθετούν τα Intents που μπορεί να χειριστεί. Έτσι τα στοιχεία μπορούν να χειριστούν τα Implicit Intents αν έχουν τα κατάλληλα Filters, στην περίπτωση που ένα στοιχείο δεν έχει κανένα Filter μπορεί να λάβει μόνο Explicit Intents

Μόνο τρία από τα περιεχόμενα του αντικείμενου Intent θα συγκριθούν με τα Intent Filters, το Action, το Data (και το URI και με το τύπο των δεδομένων) και το Category. Τα Extras και το flags δεν παίζουν κανένα ρόλο για την επιλογή του στοιχείου που θα αναλάβει να χειριστεί το Intent.

Ένα στοιχείο μπορεί να έχει πολλά Intent Filters, για παράδειγμα σε μια δικιά μας εφαρμογή με ένα Activity MyNotePad μπορεί να βάλουμε δυο Intent Filters, ένα για όταν θα εκκινήσουμε την εφαρμογή και ένα για άλλες περιπτώσεις όπως view, edit, pick. Όλα αυτά δηλώνονται μέσα στο Manifest.

```
<activity android:name=”.example.MyNotePad”
<intent-filter>
      <action android:name=”android.intent.action.MAIN” />
```

```
        <category android:name="android.intent.category.LAUNCHER" />
        <data android:mineType="vnd.andriod.cursor.dir/vnd.google.note" />
</intent-filter>

<intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <action android:name="android.intent.action.EDIT" />
        <action android:name="android.intent.action.PICK" />
        <category android:name="android.intent.category.DEFAULT"/>
        <data  android:mineType="vnd.andriod.cursor.dir/vnd.google.note"  />
</intent-filter>

</activity>
```

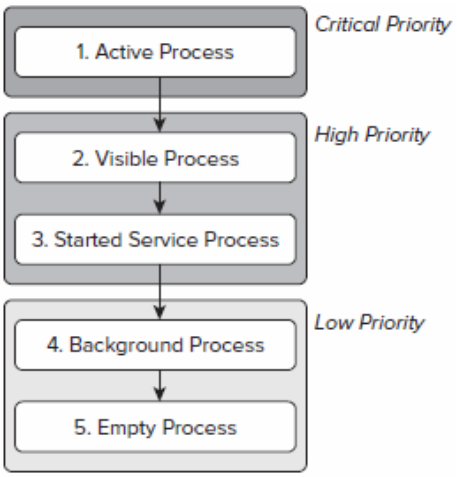
Το mineType προσδιορίζει το είδος των δεδομένων που παίρνει τα Actions. Το vnd.acdriod.cursor.dir δηλώνει πως το activity μπορεί να πάρει από κανένα έως πολλά αντικείμενα από ένα Content Provider που κρατά Note Pad δεδομένα (vnd.google.note).

3.5 Ο κύκλος της ζωής μιας εφαρμογής.

Μια Android εφαρμογή έχει περιορισμένο έλεγχο πάνω στην ζωή της. Υπάρχουν πολλοί λόγοι που μπορούν να τερματίσουν την εφαρμογή σε οποιαδήποτε κατάσταση. Για αυτό πρέπει να είναι προετοιμασμένη για πρόωρο τερματισμό.

Εξ ορισμού, κάθε εφαρμογή τρέχει στην δικιά της process, σε ξεχωριστό instance του Dalvik. Η μνήμη και η διαχείριση της process χειρίζονται αποκλειστικά από το Android Runtime. Το Android διατηρεί τους πόρους για να κρατήσει την ανταπόκριση της συσκευής προς τον χρήστη. Αυτό σημαίνει πως τα processes θα σκοτώνονται χωρίς προειδοποίηση, απελευθερώνοντας πόρους (όπως μνήμη) για να εκτελέσει εφαρμογές με μεγαλύτερη προτεραιότητα.

Η σειρά με την οποία οι processes σκοτώνονται εξαρτάται από την προτεραιότητα των εφαρμογών. Αν δυο εφαρμογές έχουν την ίδια προτεραιότητα, η process με την χαμηλότερη προτεραιότητα θα τερματιστεί. Στο παρακάτω σχεδιάγραμμα φαίνονται οι καταστάσεις της εφαρμογής και οι προτεραιότητες της κάθε process.



Σχήμα 3 – 4: Καταστάσεις μιας εφαρμογής

Active process (Προσκήνιο): Τα components της εφαρμογή αλληλεπιδρούν με τον χρήστη. Αυτές οι processes ανταποκρίνονται με τον χρήστη παίρνοντας πόρους. Μόνο σε λίγες περιπτώσεις αυτές οι processes σκοτώνονται ως τελική λύση.

Τα Active processes συμπεριλαβαίνουν:

- Activities σε "active" κατάσταση, δηλαδή αυτές που βρίσκονται στο προσκήνιο και πρέπει να ανταποκρίνονται στον χρήστη.
- Broadcast Receivers εκτελούνται στο γεγονός `onReceive`.
- Services που εκτελούνται στα γεγονότα, `onStart`, `onCreate`, ή `onDestroy`.
- Τρέχοντα Services που έχουν επισημανθεί να τρέξουν στο προσκήνιο.

Visible processes: Ορατά αλλά ανενεργά, δεν βρίσκονται στο προσκήνιο ή ανταποκρίνονται στον χρήστη. Αυτό συμβαίνει όταν ένα Activity δεν είναι σε πλήρη οθόνη ή είναι διάφανο (αυτό ρυθμίζεται από το category στο Manifest). Σκοτώνονται μόνο σε εξαιρετικές περιπτώσεις για να επιτρέψουν τα Active processes να συνεχίσουν.

Started Service processes: Τα services τρέχουν διεργασίες χωρίς ορατό interface. Επειδή τρέχουν στο background δεν αλληλεπιδρούν άμεσα με τον χρήστη, έχουν μειωμένη προτεραιότητα από τα visible processes και θα σκοτωθούν μόνο σε περίπτωση που χρειάζεται τους πόρους για κάποια active ή visible processes.

Background processes: Περιέχει Activities που δεν είναι ορατά. Υπάρχει ένας μεγάλος αριθμός από background processes το Android θα σκοτώσει με το πρότυπο last-seen-first-killed με σκοπό να πάρει πόρους για processes που τρέχουν στο προσκήνιο.

Empty processes: Το Android συχνά διατηρεί μια εφαρμογή στη μνήμη όταν φτάσει στο τέλος της ζωής του. Διατηρώντας μια προσωρινή μνήμη (cache) για να βελτιώσει την ταχύτητα επανεκτίμησης των εφαρμογών. Αυτές οι processes είναι ελεύθερες για σκότωμα.

3.6 Activity

Ένα Activity είναι ένα στοιχείο της εφαρμογής το οποίο προσφέρει ένα παράθυρο με την οποία ο χρήστης συνήθως αλληλεπιδρά με την συσκευή. Κάθε Activity περιέχει UI (User Interface – Διεπαφή Χρήστη) το οποίο δηλώνουμε μέσω της μεθόδου `setContentView`.

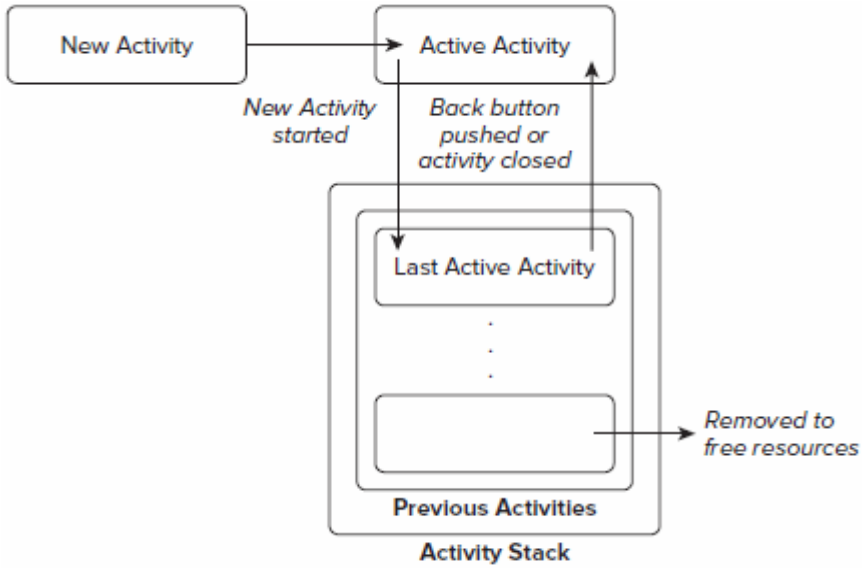
Κάθε εφαρμογή έχει έναν αριθμό από Activities, με ένα να έχει προσδιοριστεί ως "main" Activity. Το "main" Activity είναι το πρώτο Activity που θα εμφανιστεί στον χρήστη. Κάθε Activity μπορεί να καλέσει άλλα Activities για εκτέλεση κάποιας εργασίας.

Όπως αναφέραμε προηγούμενος μια Android εφαρμογή δεν ελέγχει την δικιά της ζωή. Υπεύθυνος είναι το Android Run Time. Παρόλα αυτά διατηρούμε κάποιον έλεγχο χρησιμοποιώντας τις καταστάσεις (states) των Activities.

Όταν ένα νέο Activity ξεκινά, το προηγούμενο Activity σταματά, το σύστημα όμως διατηρεί το σταματημένο Activity μέσα σε μια στοίβα LIFO (last in, first out). Αυτό συμβαίνει για κάθε καινούριο Activity αφήνοντας πάντα ένα στην κορυφή της στοίβας το

οποίο αλληλοεπιδρά με τον χρήστη. Όταν τελειώσει με το Activity και πατήσει το Back Button στην συσκευή επιστρέφει στο προηγούμενο Activity.

Οι καταστάσεις των Activities καθορίζονται από την θέση τους στην στοίβα, last-in-first-out. Όταν ένα καινούριο Activity ξεκινά, πηγαίνει στην κορυφή της στοίβας δηλαδή στο προσκήνιο. Αν ο χρήστης πατήσει το "Πίσω" στην συσκευή, ή το Activity κλείσει, τότε το επόμενο Activity στην στοίβα έρχεται στο προσκήνια όπως φαίνεται στο παρακάτω σχεδιάγραμμα.



Σχήμα 3 – 5: Η στοίβα των Activity

Ένα Activity έχει τέσσερεις κύριες καταστάσεις:

Active: Το Activity είναι στο προσκήνιο της οθόνης (στην κορυφή της στοίβας), είναι σε κατάσταση Active ή Running, κατά την οποία μπορεί να αλληλοεπιδρά με τον χρήστη. Το Android θα κρατήσει αυτό το Activity στην ζωή με κάθε κόστος, σκοτώνοντας τα Activities που βρίσκονται πιο κάτω στην στοίβα εξασφαλίζοντας τους πόρους που χρειάζεται. Όταν ένα άλλο Activity γίνει Active, αυτή περνά στο Paused.

Paused: Το Activity είναι ακόμα ορατό αλλά δεν είναι στο προσκήνιο (αυτό συμβαίνει όταν ένα νέο Activity εμφανιστεί άλλα δεν καλύπτει όλη την οθόνη, για παράδειγμα ένα Non-full-sized Activity ή ένα Transparent Activity) τότε είναι σε κατάσταση

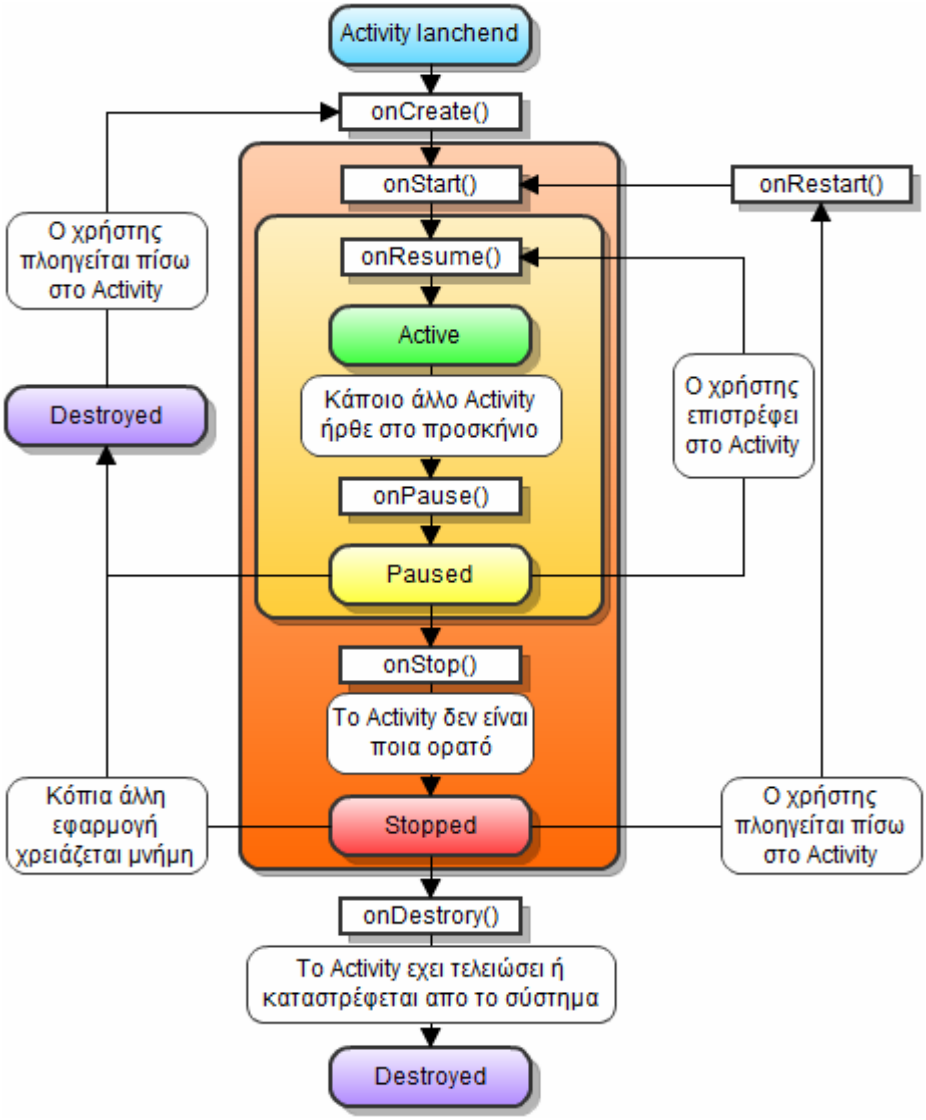
Pause. Το Pause σημαίνει πως διατηρεί όλες τις πληροφορίες και ότι εξακολουθεί να βρίσκετε στο Window Manager, αλλά δεν αλληλοεπιδρά με τον χρήστη. Σε εξαιρετικές περιπτώσεις το Android θα σκοτώσει ένα Paused Activity.

Stopped: Αν ένα Activity καλυφθεί ολοκληρωτικά από ένα άλλο, αυτό σημαίνει πως μπήκε σε κατάσταση Stopped. Ακόμα διατηρεί όλες της πληροφορίες του, αλλά ο χρήστης δεν έχει καμιά επαφή με αυτό. Το σύστημα θα το σκοτώσει για απελευθέρωση μνήμης. Σε αυτό το σημείο είναι καλό να σώσουμε όλα τα δεδομένα που έχουν σχέση με το User Interface. Το σύστημα θα τερματίσει το Activity αν χρειάζεται πόρους.

Inactive (Destroyed): Μετά που σκοτωθεί ένα Activity, και πριν να ξεκινήσει, είναι Inactive. Inactive σημαίνει πως το Activity έχει βγεί από την στοίβα και ότι χρειάζεται να ξαναρχίσει. Αυτό μπορεί να συμβεί αν ένα Activity είναι σε κατάσταση Pause ή Stopped, το σύστημα μπορεί να τερματίσει το Activity για να σώσει μνήμη, είτε καλώντας την Finish του Activity ή απλώς σκοτώνοντας την process του. Όταν ο χρήστης ζητήσει το Activity που τερματίστηκε, το Activity ξανά ξεκινά από την αρχή και επαναφέρεται στην προηγούμενη κατάσταση του.

Κάθε μετάβαση κατάστασης είναι διαχειρίσημη εξ ολοκλήρου από το Android Memory Manager. Το Android θα αρχίσει να κλίνει τις εφαρμογές που έχουν Inactive Activities, ακολουθώντας αυτές που είναι Stopped και σε εξαιρετικές περιπτώσεις αυτές που είναι Paused.

Στο παρακάτω σχήμα φαίνονται οι καταστάσεις που περνά το Activity κατά την διάρκεια της αιτίας που περιγράφτηκε παραπάνω.



Σχήμα 3 – 6: Ο κύκλος ζωής ενός Activity

Υπάρχουν τρείς βασικοί κύκλοι στην ζωή ενός Activity που συνήθως θέλουμε να παρακολουθούμε:

- Η ολόκληρη διάρκεια ζωής του Activity, ξεκινά πάντα με την μέθοδο `onCreate` και τελειώνει με την `onDestroy`. Στο `onCreate` δηλώνουμε όλους τους πόρους που θα χρησιμοποιήσει το Activity ενώ στο `onDestroy` θα τους απελευθερώσουν. Το `onCreate` χρησιμοποιείται για την αρχικοποίηση του Activity, όπως να θέσουμε το layout του Activity, δήλωση μεταβλητών, δημιουργία Services και Threads. Για παράδειγμα μπορούμε να βάλουμε ένα Thread να τρέχει για να κατεβάζει δεδομένα από το δίκτυο για όσο διάστημα το Activity ζει, δημιουργώντας το στο `onCreate`

και καταστρέφοντας το στο `onDestroy`, έτσι θα συνεχίσει να κατεβάζει ακόμα και όταν το Activity φύγει από την κορυφή της λίστας. Το `onDestroy` χρησιμοποιείτε για την απελευθέρωση πόρων ή και να κλείσουμε συνδέσεις με την βάση δεδομένων ή του δικτύου. Να σημειώσουμε πως το `onCreate` μας δίνει ένα αντικείμενο Bundle που περιέχει μια προηγούμενη κατάσταση του User Interface που μπορούμε να βάλουμε με την χρήση της μεθόδου `onSaveInstanceState` που καλείτε κάθε φορά πριν το σύστημα σταματήσει το Activity. Επίσης η μέθοδος `onRstoreInstanceState` μπορεί να χρησιμοποιηθεί για να πάρουμε τις καταστάσεις εκτός του `onCreate`. Η μέθοδος `onRstoreInstanceState` καλείτε μετά το `onStart`.

- Η διάρκεια ζωής του Activity κατά την οποία είναι ορατό στον χρήστη ξεκινά με την μέθοδο `onStart` και τελειώνει με την μέθοδο `onStop`. Σε όλη αυτή την περίοδο το Activity μπορεί να βρίσκετε στο προσκήνιο αλλά δεν είναι απαραίτητο και να αλληλοεπιδρά με τον χρήστη, άλλα διατηρεί όλους τους πόρους. Εδώ μπορούμε να ορίσουμε οτιδήποτε έχει σχέση με το τι κάνει ο χρήστης, για παράδειγμα έναν BroadcastReceiver για την ανανέωση της οθόνης όταν ο χρήστης αλλάζει κάτι. Στο `onStop` είναι καλό να σταματάμε διαδικασίες που τραβάνε την ισχύ του επεξεργαστή, όπως Animation, Services, τα οποία μπορούμε να τα ξανά ορίσουμε στο `onStart` ή στο `onRestart`.

- Η διάρκεια ζωής του Activity κατά την οποία βρίσκετε στο προσκήνιο , ξεκινά με τη μέθοδο `onResume`και τελειώνει με την μέθοδο `onPause`. Σε αυτή την περίοδο το Activity βρίσκετε στην κορυφή της στοίβας αλληλεπιδρώντας με τον χρήστη. Είναι ο μικρός από τους άλλους δύο κύκλους ζωής, δηλαδή το `onPause` και το `onResume` που καλούνται συχνά, για αυτό δεν πρέπει να είναι φορτωμένες με βάρη για το σύστημα. Για παράδειγμα όταν η συσκευή "κοιμηθεί" το Activity θα καλέσει το `onPause` για να χειριστεί την κατάσταση.

Πιο συγκεκριμένα οι κύριες μέθοδοι που παίρνουν μέρος στον κύκλο ζωής του Activity είναι οι ακόλουθες:

onCreate: Καλείτε όταν το Activity δημιουργείτε για πρώτη φορά. Στο σημείο αυτό θα πρέπει να ορίσουμε την στατική κατάσταση. Αυτή η μέθοδος μας δίνει ένα αντικείμενο Bundle που περιέχει την προηγούμενη κατάσταση του Activity. Αμέσως επομένη μέθοδος είναι η `onStart`. Το σύστημα δεν θα σκοτώσει το Activity.

onRestart: Καλείτε όταν το Activity έχει σταματήσει και πρόκειται να ξεκινήσει ξανά. Αμέσως μετά η μέθοδος `onStart` θα καλεστεί. Το σύστημα δεν θα σκοτώσει το Activity.

onStart: Καλείτε όταν το Activity γίνεται ορατό στον χρήστη. Οι μέθοδοι που ακλουθεί μετά είναι η `onResume` αν το Activity περάσει στο προσκήνιο και `onStop` αν είναι κρυμμένο πίσω από κάποιο άλλο Activity. Το σύστημα δεν θα σκοτώσει το Activity.

onResume: Καλείτε όταν το Activity περάσει στο προσκήνιο, σε αυτό το σημείο το Activity είναι στην κορυφή της λίστας και αλληλεπιδρά μα τον χρήστη. Ακολουθείται από την μέθοδο `onPause`. Το σύστημα δεν θα σκοτώσει το Activity.

onPause: Καλείτε όταν το σύστημα είναι να ξεκινήσει ένα καινούριο ή επιστρέψει σε ένα άλλο Activity. Χρησιμοποιείτε για να αποθηκεύσει αλλαγές των δεδομένων, να σταματήσει Animations και άλλα πράγματα που μειώνουν την ισχύ του επεξεργαστή. Αυτή η μέθοδος πρέπει να είναι φορτωμένη ελαφρά γιατί το επόμενο Activity δεν πρόκειται να ξεκινήσει αν δεν τελειώσει πρώτα το `onPause`. Ακολουθούν οι μέθοδοι `onResume` αν ξανά έρθει το Activity στο προσκήνιο ή `onStop` αν φύγει τελείως από την οθόνη. Το σύστημα θα σκοτώσει το Activity μονό αν το API Level της εφαρμογής είναι χαμηλότερο από το 11.

onStop: Καλείτε όταν το Activity δεν είναι ποια ορατό στην χρήστη, είτε επειδή ένα άλλο Activity ξεκίνησε και το κάλυψε, είτε επειδή το σύστημα αποφάσισε να το σκοτώσει. Στην περίπτωση που το Activity επιστρέψει στο προσκήνιο η μέθοδος `onRestart` θα

καλεστεί. Στην περίπτωση που το σύστημα αποφασίσει να καταστρέψει το Activity θα καλεστεί η μέθοδος `onDestroy`.

`onDestroy`: Καλείται πριν το Activity καταστρέψει. Μπορεί να καταστραφεί είτε καλώντας την μέθοδο `finish` μέσα στον κώδικα του Activity, ή από το ίδιο το σύστημα για να αποδεσμεύσει πόρους όπως μνήμη. Μπορούμε να ελέγξουμε αν ένα Activity έχει μπει στην διαδικασία καταστροφής καλώντας την μέθοδο `isFinishing` που μας επιστρέφει True ή False, η οποία χρησιμοποιείτε πιο συχνά μέσα στην μέθοδο `onPause` για να ξέρουμε αν το Activity απλώς θα πάει σε κατάσταση Pause είτε θα καταστραφεί.

3.7 Ξεκινώντας ένα Activity και Παίρνοντας Αποτελέσματα

Όταν ξεκινάμε μια εφαρμογή, ξεκινά με κάποιο Activity με Intent Filter action.MAIN και category.LAUNCHER. Μέσα μπορούμε να ξεκινήσουμε άλλο Activity. Για να ξεκινήσουμε ένα Activity δημιουργούμε πρώτα ένα Intent είτε Explicit, ή Implicit. Οι δύο κύριοι τρόποι για να ξεκινήσουμε ένα καινούριο Activity μέσα από ένα άλλο Activity είναι `startActivity` και `startActivityForResult`.Το Activity που χρησιμοποιεί αυτές τις μεθόδους λέγεται Parent (γονέας) ενώ το Activity που ξεκινά ονομάζεται Child (παιδί).

Με την `startActivity` ξεκινά ένα καινούριο Child Activity, το οποίο τοποθετείτε πάνω στην κορυφή της λίστας. Στην περίπτωση που όμως θέλουμε να πάρουμε αποτελέσματα από το Child Activity θα χρησιμοποιούμε το `startActivityForResult`.

Με την `startActivityForResult` ξεκινά ένα καινούριο Child Activity όπως η `startActivity`, με την διαφορά ότι περνάμε και ένα `int` που αντιπροσωπεύει το Request Code. Αυτό το Request Code επιστρέφεται όταν το τερματίσει το Child Activity ώστε το Parent Activity να αναγνωρίσει εκτέλεση το `onActivityResult`.

Το `onActivityResult` εκτελείτε όταν το Parent Activity ξαναέρθει στο προσκήνιο επιστρέφοντας το `requestCode` που είχαμε δώσει, ένα `resultCode`, που θα δούμε στην συνέχεια, και ένα Intent για να πάρουμε δεδομένα.

Το `resultCode` ορίζεται στο Child Activity μέσω της `setResult` ή setResult με τρείς κύριες τιμές RESULT_CANCELED που δηλώνει ακύρωση του Activity,

RESULT_FIRST_USER για περιπτώσεις αποτελέσματος του χρήστη, στο συγκεκριμένο μπορούμε να χρησιμοποιήσουμε πολλαπλά RESULT_FIRST_USER (δηλαδή να έχουμε RESULT_FIRST_USER+1 για κάποιο άλλο αποτέλεσμα του χρήστη, για παράδειγμα το resultCode RESULT_FIRST_USER σημαίνει σφάλμα σε πράξη και το RESULT_FIRST_USER+1 για σφάλμα στην σύνδεση με τον server). Τέλος το RESULT_OK που δηλώνει πως όλα πήγαν καλά.

Στο παρακάτω παράδειγμα έχουμε ένα Parent Activity που καλεί ένα Child Activity για να κάνει μια διαίρεση, να πάρει το αποτέλεσμα και να το εμφανίσει πίσω στο Parent Activity. Δηλώνουμε δύο αντικείμενα για τα View που θα πάρουμε από το XML μέσω της `findViewbyId` και ένα `static final int` για να δηλώσουμε το requestCode.

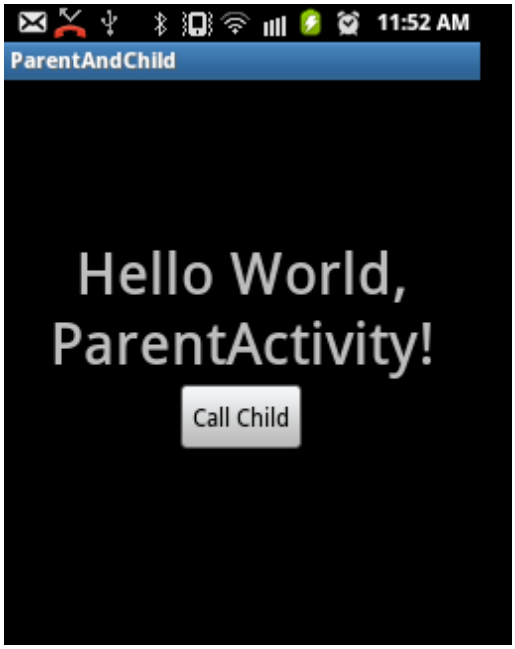
Στο `onCreate` δηλώνουμε το `ContentView` και τα αντικείμενα καθώς και ένα listener στο button μέσω της μεθόδου `setOnClickListener` παίρνει ένα αντικείμενο `OnClickListener`, αφού όμως έχουμε κάνει implements το `OnClickListener` χρησιμοποιούμε μόνο το `this` ως όρισμα στο `setOnClickListener`.

Κάνοντας Implements την `OnClickListener` πρέπει να ορίσουμε την μέθοδο `onClick` που θα καλεστεί κάθε φορά που πατιέται το button. Μέσα στην `onClick` ορίζουμε ένα Implicit Intent για το Child Activity και καλούμε την `startActivityForResult` που περνάμε ως ορίσματα το intent και το requestCode CALCULATIONCODE που θα χρησιμοποιήσουμε και στο `onActivityForResult`. Το `onClick` μας δίνει ένα αντικείμενο View που είναι το αντικείμενο View που χρησιμοποιεί τον Listener. Για την περίπτωση που είχαμε περισσότερα αντικείμενα View χρησιμοποιούμε μια switch περνώντας το Id από το View και ελέγχοντας το με τις case με το R.id όπως φαίνεται στο παράδειγμα.

Τέλος έχουμε την `onActivityResult`. Η `onActivityResult` μας δίνει τρείς μεταβλητές το `requestCode` που με switch και case βρίσκουμε ποιο result ήρθε από που, το `resultCode` που θα μας πει πως πήγε η δραστηριότητα, και τέλος ένα Intent για να περάσουμε τα δεδομένα από το Child Activity.

```
public class ParentActivity extends Activity implements OnClickListener{

private TextView textView;
private Button button;
private static final int CALCULATIONCODE = 1234;


/** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        textView = (TextView) findViewById(R.id.tv_parent);
        button = (Button) findViewById(R.id.bt_parent);

        button.setOnClickListener(this);
    }


       @Override
       public void onClick(View v) {
              // TODO Auto-generated method stub
              switch(v.getId()){

              case R.id.bt_parent:
              Intent intent = new Intent ("com.parentandchild.ChildActivity");
              startActivityForResult(intent, CALCULATIONCODE);

              }
       }


       @Override
       protected void onActivityResult(int requestCode, int resultCode, Intent
       data) {

       // TODO Auto-generated method stub
              super.onActivityResult(requestCode, resultCode, data);

              switch(requestCode){
              case CALCULATIONCODE:

                     if(resultCode==RESULT_OK){

                     textView.setText(
       String.valueOf(data.getDoubleExtra("calculation", 0)));

                     }
                     if(resultCode==RESULT_FIRST_USER){
                            textView.setText("error");
                     }
```

```
            }
        }
}
```



Εικόνα 3 – 1: Το Parent Activity

Παρακάτω φαίνεται το Child Activity. Δηλώνουμε τρία αντικείμενα View, δύο `EditText` και ένα `Button`. Όπως σε κάθε Activity στο `onCreate` ορίζουμε το `Content View`, τα `View` και τέλος βάζουμε ένα Listener στο `Button` όπως στον Parent Activity.

Στην `onClick` θέλουμε να πάρουμε τα δεδομένα από τα `EditText`, τον αριθμητή numerator και τον παρανομαστή denominator. Για να σιγουρέψουμε πως δεν θα κάνουμε διαίρεση με κενές μεταβλητές, φροντίζουμε να ελέγξουμε πρώτα αν υπάρχουν περιεχόμενα μέσα στα `EditText` ελέγχοντας το `length` τους, αν είναι κάποιο από τα δύο κενά θα κάνει διαίρεση με παρανομαστή και αριθμητή μηδέν, αν όχι θα πάρει τις τιμές από τα `EditText`. Για να πάρουμε το double που θέλουμε πρέπει πρώτα να πάρουμε το `charSequensce` μέσω της `getText`, στην συνέχεα το μετατρέπουμε σε `String` μέσω της `toString`, και τέλος με την χρήση της κλάσης Double χρησιμοποιούμε την μέθοδο `parseDouble` για να μετατρέψουμε το `String` σε `Double`. Μετά αφού πάρουμε τις

τιμές μας κάνουμε την διαίρεση και προχωράμε στο τελικό κομμάτι του Child Activity για να περάσει τα αποτελέσματα πίσω στο Parent.

Στην τελευταία `if` ελέγχουμε αν πρόκειται να κάνουμε διαίρεση με το μηδέν, όπου σε αυτή την περίπτωση κάνουμε `setResult`(RESULT_FIRST_USER) ώστε να πούμε στο Parent πως κάτι πήγε στραβά. Στην `else` έχουμε την περίπτωση όπου όλα έχουν πάει καλά και έχουμε κάνει την διαίρεση. Σε αυτό το σημείο περνούμε το Intent που ξεκίνησε το Child Activity με όνομα data, βάζοντας το αποτέλεσμα της πράξης μέσω της `putExtra` με ορίσματα ένα όνομα και την μεταβλητή που θέλουμε να περάσουμε. Μετά κάνουμε `setResult`(RESULT_OK) για να πούμε στο Parent πως όλα πήγαν καλά. Τέλος με την `finish` τερματίζουμε το Child Activity κάνοντας το Parent Activity να επιστρέψει στο προσκήνιο και να τρέξει το `onActivityResult`.

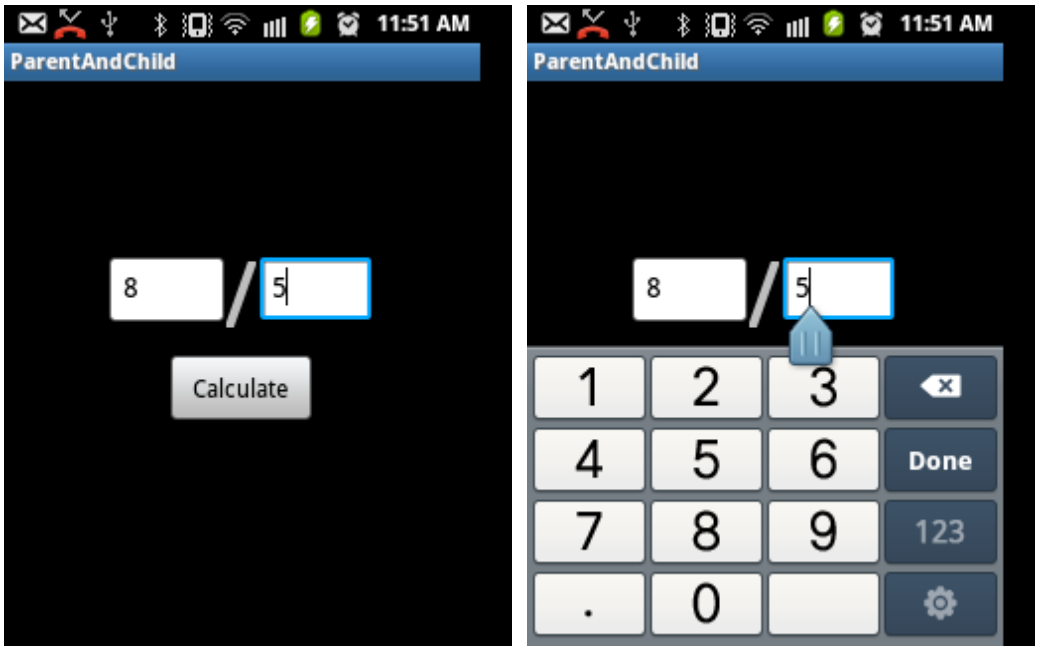
```
public class ChildActivity extends Activity implements OnClickListener {

	private EditText tvNumerator;
	private EditText tvDenominator;
	private Button button;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.calculation);

		tvNumerator = (EditText) findViewById(R.id.et_numerator);

		tvDenominator = (EditText) findViewById(R.id.et_denominator);

		button = (Button) findViewById(R.id.bt_child);

		button.setOnClickListener(this);
	}


	@Override
	public void onClick(View v) {
		// TODO Auto-generated method stub

		switch(v.getId()){

		case R.id.bt_child:

		double num;
		double den;

		if(tvNumerator.getText().length()==0||
		tvDenominator.getText().length()==0){
			num=0;
```

```
                den=0;

            }else{

            num = Double.parseDouble(
            tvNumerator.getText().toString());

            den = Double.parseDouble(
            tvDenominator.getText().toString());
            }


            if(den==0&&num!=0){
                setResult(RESULT_FIRST_USER);

            }else{
                double cal=num/den;
                Intent data = this.getIntent();

                data.putExtra("calculation", cal);
                setResult(RESULT_OK, data);
            }

            finish();
            }
        }
}
```

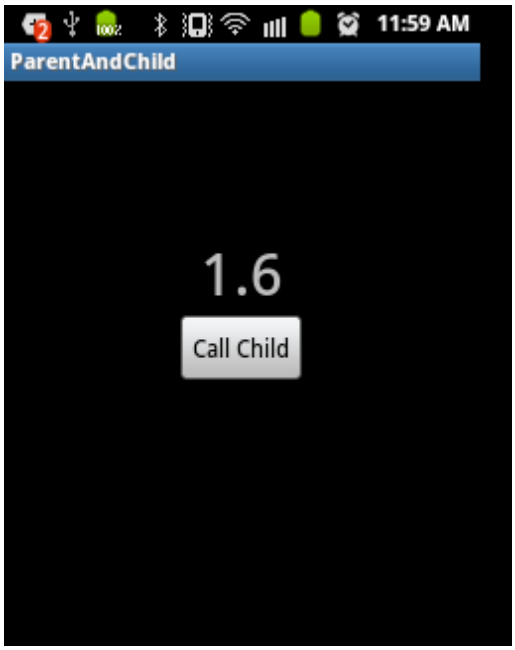
Ας ρίξουμε μια ματιά στο XML του Content View του Child Activity, το calculation.xml. Με την χρήση δύο `LinearLayout`, το ένα με vertical orientation (κατακόρυφο προσανατολισμό), που ταξινομεί τα στοιχεία το ένα κάτω από το άλλο, και το άλλο με το default – horizontal orientation (οριζόντιο προσανατολισμό), για να βάλουμε τα δύο `EditText` και το ένα `TextView` στην ιδία σειρά. Ένα κύριο κομμάτι για την εφαρμογή βρίσκετε στα `EditText`, το inputType numberDecimal το οποίο θα αφήσει τον χρήστη να εισάγει μονό δεκαδικούς αριθμούς βγάζοντας μας από την θέση να ελέγξουμε αν ο χρήστης πάει να εισάγει γράμματα αντί για νούμερα.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="top|center"
    android:paddingTop="100dp">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:gravity="center" >


        <EditText
            android:id="@+id/et_numerator"
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="numberDecimal"/>

        <TextView
            android:layout_width="20dp"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="/"
            android:textSize="60sp" />

        <EditText
            android:id="@+id/et_denominator"
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="numberDecimal"/>

    </LinearLayout>

    <Button
        android:id="@+id/bt_child"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="Calculate" />

</LinearLayout>
```



Εικόνα 3 – 2: To Child Activity



Εικόνα 3 – 3: Το Parent Activity μετά το onActivityResult

Στη παραπάνω εικόνα φαίνεται το Child Activity όταν πατάμε σε κάποιο από τα δύο EditText. Το πληκτρολόγιο του εξομοιωτή περιέχει και σύμβολα παρόλα αυτά δεν εισήγαγε τίποτα άλλο έκτος από αριθμούς και την τελεία.

3.8 Services

Αντίθετα με τα Activities, τα Services τρέχουν στο background της εφαρμογής χωρίς User Interface και ανεξάρτητα από τα Activities. Τα Services μπορούν να ξεκινήσουν, τερματιστούν και να ελέγχουν από την ιδία ή άλλες εφαρμογές ακόμα και από άλλα Services, Activities, Content Providers ακόμα και από Broadcast Receivers.

Όπως προαναφέραμε στο κεφάλαιο για τον κύκλο ζωής των εφαρμογών, τα Services έχουν υψηλότερη προτεραιότητα από τα μη ορατά Activities. Το Android θα σταματήσει ένα Service για να πάρει πόρους για κάποιο στοιχείο που είναι στο προσκήνιο, συνήθως ένα Activity. Παρόλα αυτά το Service θα επανέλθει σε λειτουργία όταν υπάρξουν ξανά διαθέσιμοι πόροι της συσκευής.

Τα Services πρέπει να αντιστοιχούν σε δήλωση μέσα στο AndroidManifest.xml με tag <service> που περιέχει χαρακτηριστικά όπως, android:name με το όνομα της κλάσης που θα εφαρμόσει το Service, android:enabled για να πούμε αν το service θα είναι διαθέσιμο ή όχι. Για να ξεκινήσουμε ένα Service χρησιμοποιούμε την μέθοδο `Context.startService` ή `Context.bindService`.

Το `Context.startService` ξεκινά το Service και τρέχει μέχρι κάποιος να το σταματήσει με την `Context.stopService` ή σταματήσει μόνο του με την `Service.stopSelf` ή την `Service.stopSelfResult`, για παράδειγμα αν ένα Activity καλέσει ένα Service, το Service μπορεί να συνεχίσει να εκτελείτε ακόμα και αν το Activity τερματιστεί. Ακόμα και στην περίπτωση που διάφορα Activities καλέσουν ένα Service, με το που καλεστεί μια από της παραπάνω μεθόδους για να σταματήσει το Service αυτό σημαίνει πως θα σταματήσει για όλους.

Το `Context.bindService` είναι ένας μηχανισμός ώστε μια εφαρμογή εκθέτει ένα μέρος των λειτουργιών της σε άλλες. Δημιουργεί μια σύνδεση μεταξύ μιας εφαρμογής και ένα Service και καλώντας την μέθοδο `Context.unbindService` αποσυνδέεται. Το Service θα συνεχίσει να λειτουργεί μέχρι όλοι όσοι έχουν συνδεθεί με αυτό αποσυνδεθούν. Μόνο Activities, Services, Content Providers μπορούν να κάνουν “Bind” με ένα Service.

Το `Context.startService(Intent service)` παίρνει σαν παράμετρο ένα Intent με το Service που θέλουμε. Αμέσως μετά περνά αυτό το Intent στο `onStartCommand(Intent service, int flags, int started)` με το οποίο λέμε στο σύστημα πώς να χειριστεί το Service όταν τερματιστεί από το Android Runtime και ξανά δημιουργείται προτού τελεσθεί η μέθοδος `stopService` ή `stoSelf`. Αυτό γίνεται μέσω της return μέσα στην μέθοδο `onStartCommand` επιστρέφοντας μια σταθερά όπως φαίνεται στο παρακάτω παράδειγμα.

```
@Override
public int onStartCommand(Intent intent, int flags,
int startId){
	// TODO Launch a background thread to do processing.

	return Service.START_STICKY;
}
```

Οι σταθερές είναι:

START_STICKY Το `onStartCommand` θα καλεστεί κάθε φορά όταν το Service τερματιστεί από το Android Runtime. Προσοχή, σε κάθε επανεκκίνηση η παράμετρος Intent θα είναι null. Χρησιμοποιείτε κυρίως για Services που χειρίζονται τις δικές τους καταστάσεις, και ξεκινούν και τερματίζονται όπως προβλέπεται (δηλαδή μέσα από την `startService` και `stopService`). Παραδείγματος χάρη Services που παίζουν μουσική.

START_NOT_STINK Χρησιμοποιείτε στα Services που θα χρησιμοποιήσουν το `stopSelf` για να τερματιστούν. Αν τερματιστεί από το Android Runtime το Service θα ξεκινήσει ξανά όταν το ξανακαλέσει κάποιος με την `startService`. Χρησιμοποιείτε συνήθως για updates ή network polling.

START_REDELIVER_INTENT Αυτό είναι συνδυασμός των δύο παραπάνω. Αν το Service τερματιστεί από το Android Runtime θα ξαναξεκινήσει μόνο αν εκκρεμεί κάποια κλήση για ξεκινήσει το Service ή αν έχει τερματιστεί πριν καλεστεί η `stopSelf`. Στην δεύτερη περίπτωση καλείται η `onStartCommand` με το αρχικό Intent. Χρησιμοποιείτε όταν θέλουμε να είμαστε σίγουροι πως το Service θα ολοκληρωθεί.

Στο `onStartCommand` περιέχεται και μια παράμετρος flag. Τα flags χρησιμοποιείτε για να μας δώσει επιπλέον πληροφορίες σχετικά με το Intent του Service για το πως θα ξεκινήσει. Ποιο συγκεκριμένα χρησιμοποιούνται δύο τιμές η START_FLAG_REDELIVERY και η START_FLAG_RETRY.

- Η START_FLAG_REDELIVERY μας δηλώνει πως το Intent έχει ξανά παραδοθεί επειδή δεν τερματίστηκε από την μέθοδο `stopSelf`.

- Η STAR_FLAG_RETRY μας δηλώνει πως το Service έχει ξανά ξεκινήσει μετά από μη φυσιολογικό τερματισμό όταν το `onStartCommand` έχει επιστρέψει την σταθερά START_STICKY.

3.9 Binding Client σε Services

Όταν ένα Activity δένεται με ένα Service, διατηρώντας μια αναφορά στο Service επιτρέποντας μας να καλέσουμε τις μεθόδους του. Για να γίνει "Bind" ένα Service πρέπει να κάνουμε Override την μέθοδο `onBind`, δημιουργώντας ένα interface αντικείμενο IBinder (Interface Binder) για να καλέσουμε τις μεθόδους του Service όπως φαίνεται στο παρακάτω παράδειγμα.

```
public class MyService extends Service {
	private final IBinder myBinder = new MyBinder();

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

	public class MyBinder extends Binder {
		MyService getService(){
			return MyService.this;
		}

	}
}
```

Η σύνδεση μεταξύ ενός Activity και ενός Service εκπροσωπείτε μέσω ενός `ServiceConnection` Interface. Το Activity πρέπει να κατασκευάσει ένα στιγμιότυπο της `ServiceConnection` και να το περάσει μέσα στην `bindService()`, κάνοντας impliments τις μεθόδους `onServiceConnected()` και `onServiveDisconnected()`. Η `onServiceConnected()` καλείτε από το σύστημα για να παραδώσει το IBinder που επιστρέφεται από το Service `onBind()`. Η `onServiceDisconnected()` καλείτε από το σύστημα όταν η σύνδεση με το Service χαθεί ξαφνικά, όπως αν το Service "κρασάρει" ή τερματιστεί, όχι όταν το Service αποσυνδεθεί.

```
	private MyService serviceBinder;

	private ServiceConnection myServiceConnection =
			new ServiceConnection() {
```

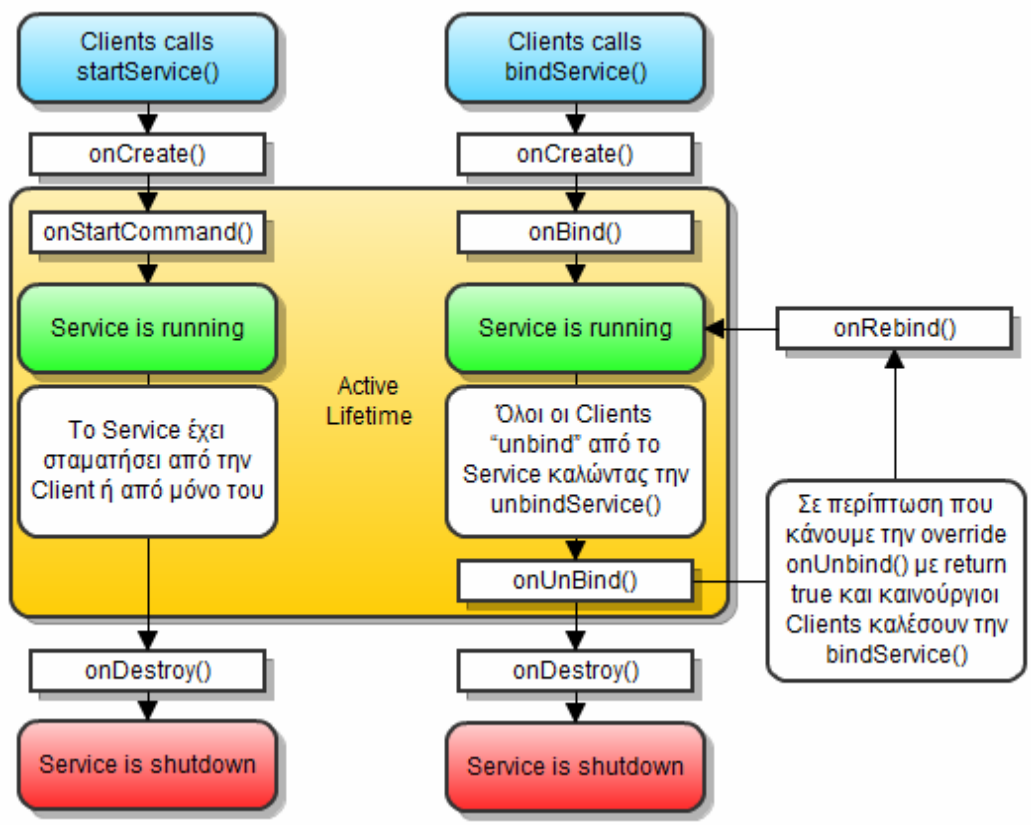
```
public void onServiceConnected(
        ComponentName className, IBinder service) {
                serviceBinder = ((MyService.MyBinder) service).getService();
        }


        public void onServiceDisconnected(
                                ComponentName className) {

                serviceBinder = null;
        }
};
```

Το επόμενο βήμα για συνδέσουμε το Activity με το Services είναι να καλέσουμε την `BindService`, επιστρέφοντας μια `Boolean` για το αν έγινε η σύνδεση ή όχι. Τα γνωρίσματα που παίρνει είναι `BindService(Intent` service, `ServiceConnection` myServiceConnection, `int` flags). Το Intent με το service που θέλουμε να συνδεθούμε, το ServiceConnection αντικείμενο θα το ορίσουμε και θα το χρησιμοποιήσουμε όταν γίνει η σύνδεση, και τέλος flags για να δηλώσει τον τρόπο λειτουργίας του "Bind".

Τα flags είναι τα ακόλουθα:

- BIND_ABOVE_CLIENT Δηλώνει πως ο Client "Binds" με το Service.
- BIND_AJUST_WITH_ACTIVITY Αν το Service γίνει "Bind" με ένα Activity, το Service αυξάνει την προτεραιότητα του ανάλογα με το αν το Activity είναι ορατό στον χρήστη.
- BIND_ALLOW_OOM_MANAGEMENT Επιτρέπει στην process που κρατά την σύνδεση με το Service να περάσει μέσα από το memory management.
- BIND_AUTO_CREATE Αυτόματα δημιουργεί το Service όσο το "Bind" υπάρχει.
- BIND_DEBUG_UNBIND Περιλαμβάνει βοήθεια για εντοπισμό σφαλμάτων για mismatched calls για να αποσυνδέσει το Service.
- BIND_IMPORTANT Δηλώνει πως το Service είναι πολύ σημαντικό για τον χρήστη, φέρνοντας το Service σε foreground process level.
- BIND_NOT_FOREGROUND Δεν επιτρέπει στο Service να φτάσει σε foreground process level.
- BIND_WAIVE_PRIORITY Δεν επηρεάζει ούτε το process level, ούτε την memory management προτεραιότητα της process του Service.

Όπως τα Activities έτσι και τα Services έχουν τον κύκλο ζωής και τις μεθόδους τους για τον έλεγχο των καταστάσεων.



Σχήμα 3 – 7: Ο κύκλος ζωής ενός Service

3.10 Broadcast Receiver

Ένα Broadcast Receiver ακούει για Broadcast Intents που αποστέλλονται μέσω της `sendBroadcast`. Για να χρησιμοποιήσουμε ένα Broadcast Receiver πρέπει είτε να καταχωρηθεί μέσα από τον κώδικα μας, μέσω της μεθόδου `registerReceiver`, ή μέσα στο Manifest. Η βασική διαφορά αυτών των δύο είναι πως όταν θέλουμε να χρησιμοποιήσουμε ένα Broadcast Receiver μέσα σε κάποιο στοιχείο (Activity/Service) το οποίο πρέπει να διαγράψουμε από το σύστημα μέσω της μεθόδου `unregister` πριν τερματιστεί το στοιχείο.

```
registerReceiver(broadcastSMSReceiver, new
IntentFilter(android.provider.Telephony.SMS_RECEIVED));
```

Αν καταχωρίσουμε το BroadcastReceiver στο Manifest θα εκτελεστεί όταν λάβει το Intent.

```
<receiver android:name=".SMSReceiver">
   <intent-filter>
      <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    <intent-filter>
</receiver>
```

Υπάρχουν δύο κύριες κατηγορίες των Broadcasts που μπορούμε να λάβουμε τα Normal και τα Order.

- Τα Normal Broadcasts λαμβάνονται από όλους τους Receivers χωρίς κάποια σειρά την ίδια χρονική στιγμή (αν υπάρξει κίνδυνος υπερφόρτωσης του συστήματος το Android θα παραδώσει μόνο σε ένα Receiver την φορά) . Δεν διαδίδουν τα αποτελέσματα και ούτε μπορούν να απορριφθούν από τα Broadcast Receivers. Η αποστολή Normal Broadcast γίνετε μέσω της `Context.sendBroadcast`.

    - Τα Order Broadcasts παραλαμβάνονται από μόνο έναν receiver την φορά. Έτσι το κάθε Receiver μπορεί να διαδώσει τα αποτελέσματα του στο επόμενο Receiver ή να απορρίψει το Broadcast ώστε να μην περάσει στα επόμενα Receivers. Η σειρά με την οποία τα Receivers παίρνουν ένα Broadcast καθορίζεται από την προτεραιότητα τους στο Manifest μέσω του android:priority στο intent-filter (αν δύο Receivers έχουν την ίδια προτεραιότητα η σειρά τους θα καθοριστεί τυχαία). Η αποστολή Normal Broadcast γίνετε μέσω της `Context.sendOrderedBroadcast`.

Η ζωή ενός Broadcast Receiver διαρκεί όσο διαρκεί η εκτέλεση της μεθόδου `onReceive` που τρέχει σε επίπεδο process προσκηνίου.

4

4.1 Location-based Service

Location-based Service (LBS) είναι μια υπηρεσία από την κινητή συσκευή προς τον χρήστη σχετικά με την θέση και την τοποθεσία που βρίσκεται την δεδομένη στιγμή παρέχοντας του πληροφορίες όπως σε ποια οδό βρίσκεται, που βρίσκονται τα κοντινότερα καταστήματα, ή άλλες πληροφορίες σχετικά πάντα με την τοποθεσία που βρίσκεται.

Για να βρούμε την τοποθεσία του χρήστη το Android μας προσφέρει τρείς Location Provider, GPS, network και passive.

Το GPS χρησιμοποιεί τους δορυφόρους γύρω από την Γη για να βρει την τοποθεσία του χρήστη. Η συσκευή λαμβάνει τα σήματα από τους δορυφόρους χωρίς να μεταδίδει κάτι πίσω. Το GPS απαιτεί την οπτική επαφή με τους δορυφόρους, δηλαδή δεν θα δουλέψει μέσα σε κτίρια και σε πολύ συννεφιασμένες μέρες, δάση και κοντά σε ψηλά κτίρια. Κάθε δορυφόρος στέλνει δεδομένα που δηλώνουν την πολυθεσία τους και την ώρα. Όλοι οι δορυφόροι είναι συγχρονισμένοι και μεταδίδουν την ιδία χρονική στιγμή. Τα σήματα αυτά φτάνουν στον δέκτη με μια μικρή διάφορα το ένα με το άλλο. Υπολογίζοντας την απόσταση του από τέσσερεις τουλάχιστον δορυφόρους για να υπολογίσει την τοποθεσία του και στις τρείς διαστάσεις. Το GPS μπορεί να δουλέψει και μέσα σε κτίρια, αλλά συνήθως αν είμαστε στον τελευταίο όροφο ενός κτιρίου, επίσης μπορεί να επηρεαστεί από ψηλά κτίρια και κακές καιρικές συνθήκες.

Το Network είναι ένας συνδυασμός με Cell-ID και Wi-Fi:

- Το Cell-ID χρησιμοποιεί την σύνδεση της συσκευής για να πάρει το Cell-ID (CID) και το Location Area Code (LAC), από εκεί λαμβάνει τα σήματα από τους σταθμούς βάσεις (Base Transreceiver Station – BTS) που μοιράζονται τον ίδιο βασικό σταθμό ελέγχου (Base Station Controller – BSC). Οι θέσεις των κεραιών είναι γνωστές, ώστε με αυτό βγαίνει μια προσέγγιση της θέσης του χρήστη, όσο ποιό πυκνή είναι μια περιοχή με κεραίες τόσο ποιό ακριβές είναι το αποτέλεσμα. Το Cell-ID πρέπει να

αναζητηθεί μέσα σε μια βάση δεδομένων για να βρεθεί το γεωγραφικό μήκος και ύψος. Το Cell-ID έχει χαμηλή κατανάλωση ενέργειας.

- Το Wi-Fi παρομοίως χρησιμοποιεί τις θέσεις των Wi-Fi access points χρησιμοποιώντας τα SSIDs (Service Set Identifier) που λαμβάνει.

Αυτά τα δεδομένα αποστέλλοντα στην Google που γνωρίζουν που βρίσκονται αυτά τα σημεία. Για να λειτουργήσει το Network Provider πρέπει να έχουμε επιλέξει Use wireless networks μέσα από το μενού Settings – Location and security – Use wireless networks που θα μας επιτρέψει να βρούμε την θέση μας μέσω Cell-Ids και Wi-Fi, και Use packet data για να στείλουμε δεδομένα στην Google από το μενού Settings – Wireless and network settings – Mobile network settings – Use packet data

Τέλος, το Passive είναι ένας ειδικός παροχέας που παθητικά λαμβάνει ανανεώσεις από άλλες εφαρμογές ή υπηρεσίες.

Για να αποκτήσουμε πρόσβαση στο Location Services χρησιμοποιούμε την κλάση `LocationManager`. Για να πάρουμε έναν `LocationManager` χρησιμοποιούμε την μέθοδο `getSystemService` της Context που επιστρέφει ένα αντικείμενο από την υπηρεσία που ζητάμε, σε αυτήν την περίπτωση LOCATION_SERVICE.

```
LocationManager locationManager;

locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
```

Και φυσικά θέλουμε και τα κατάλληλα Permission στο Manifest. ACCESS_FINE_LOCATION για GPS και ACCESS_COARSE_LOCATION για Cell-ID και Wi-Fi.

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

Οι πάροχοι αυτοί μας προσφέρονται μέσα από την κλάση `LocationManager`. Ο `LocationManager` περιέχει δύο static String που `LocationManager`.GPS_PROVIDER και NETWORK_PROVIDER που αναθέτουν στους Providers. Μπορούμε φυσικά να πάρουμε όλους τους Providers που είναι διαθέσιμοι όπως φαίνεται στο παρακάτω παράδειγμα.

```
List<String> providers = myLocationManager.getProviders(true);
```

Μπορούμε να ζητήσουμε τον κατάλληλο Provider που ταιριάζει στις ανάγκες μας χρησιμοποιώντας την κλάση `Criteria`. Με την Criteria μπορούμε να βρούμε τον κατάλληλο provider βάση του επιθυμητού, ακρίβεια, κατανάλωση ενέργειας, κόστος, και δυνατότητα να έχουμε αναφορά σε υψόμετρο και ταχύτητα.

```
Criteria criteria = new  Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setPowerRequirement(Criteria.POWER_HIGH);
criteria.setAltitudeRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);
```

Αφού ορίσουμε το criteria μας μπορούμε να πάρουμε τον κατάλληλο Provider μέσω της μεθόδου `getBestProvider` από το `locationManager` με παραμέτρους το criteria και μια `Boolean` για να επιστρέψει μόνο ένα Provider που είναι ήδη ενεργοποιημένος.

```
String bestProvider =
            locationManager.getBestProvider(criteria, true);
```

Αν δηλώσουμε false σαν ορισμό θα μας επιστρέψει μια λίστα με του κατάλληλους Providers

```
List<String> bestProviders =
            locationManager.getBestProvider(criteria, false);
```

Αν βρεθούν περισσότεροι Providers θα μας δοθεί αυτός που καταναλίσκει την λιγότερη ενέργεια, μετά αυτόν που έχει την καλύτερη ακρίβεια και τέλος αυτόν που μας προσφέρει το υψόμετρο ή/και την ταχύτητα. Αν δεν βρεθεί κανένας τότε θα μας επιστρέψει null.

Αφού πάρουμε τον Provider μπορούμε είτε να πάρουμε την τελευταία γνωστή τοποθεσία μέσω της `getLastKnownLocation` από το locationManager μας με παράμετρο τον Provider μας.

```
Location = locationManager.getLastKnownLocation(bestProvider);
```

Αν δεν υπάρχει καμιά διεύθυνση ακόμα από τον Provider θα μας επιστρέψει null.

Έχουμε την δυνατότητα να ακούμε για κάθε ειδοποίηση από τον `LocationManager` όταν αλλάζει το `Location` μας χρησιμοποιώντας το `LocationListener`. Το Location παρέχει τέσσερεις μεθόδους που καλούνται όταν συμβαίνουν τα αντίστοιχα γεγονότα. `onLocationChanged`, `onProviderDisabled`, `onProviderEnabled`, `onStatusChanged`.

- `onLocationChanged(Location` location) μας δίνει το καινούριο `Location` όταν αλλάζει η τοποθεσία.
- `onProviderDisabled(String` provider) καλείτε όταν ο Provider έχει απενεργοποιηθεί από τον χρήστη.
- `onProviderEnabled(String` provider) καλείτε όταν ο Provider ενεργοποιηθεί από τον χρήστη.
- `onStatusChanged (String` provider, `int` status, `Bundle` extras) καλείτε όταν αλλάξει κατάσταση ο Provider. Το status μπορεί να επιστρέψει OUT_OF_SERVICE, TEMPORALITY_UNAVAILABLE και AVAILABLE, ενώ τα extras περιέχει ειδικές τιμές σύμφωνα με το status, για παράδειγμα `extras.getInt("satellites");` θα μας επιστρέφει τον αριθμό των δορυφόρων που μπορεί να δει.

Οι παραπάνω μέθοδοι θα καλεστούν αν το `LocationListener` έχει γίνει register με το location manager service μέσω της `requestLocationUpdates` μέσω της `LocationManager`. Το `requestLocationUpdates` παίρνει τέσσερεις παραμέτρους, `requestLocationUpdates(String` provider, `long` minTime, `float` minDistance, `LocationListener` listener). Το minTime είναι ο ελάχιστος χρόνος ανάμεσα σε δύο ενημερώσεις σε microseconds. Το minDistance είναι ελάχιστη απόσταση των δύο ενημερώσεων σε μέτρα.

Σε ένα Activity το ποιό συνηθισμένο σημείο να καλέσουμε την `requestLocationUpdates` είναι στο `onResume` και αντίστοιχα στο `onPause` καλούμε την `removeUpdates` με παράμετρο το `LoctionListener`.

Το `Location` που μας δίνεται είτε από το `onLocationChanged`() ή από το `getLastKnownLocation` περιέχει πληροφορίες για την τοποθεσία μας μέσα από τις ακόλουθες μεθόδους. Μερικές από αυτές είναι:

- `getLatitude` επιστρέφει το γεωγραφικό πλάτος σε double με έξι δεκαδικά στοιχεία.
- `getLongitude` επιστρέφει το γεωγραφικό μήκος σε double με έξι δεκαδικά στοιχεία.
- `setLatitude` για να δηλώσουμε το δικό μας γεωγραφικό πλάτος.

- `setLongitude` για να δηλώσουμε το δικό μας γεωγραφικό μήκος.
- `getProvider` το όνομα του Provider που δημιούργησε αυτό το Location.
- `setProvider` δηλώνουμε το όνομα του Provider που το δημιούργησε.
- `getAccuracy` επιστρέφει πόσο ακριβές είναι το location μας σε μέτρα.
- `setAccuracy` δηλώνουμε πόσο ακριβές είναι το location μας.

Στο παρακάτω παράδειγμα εμφανίζουμε τις γεωγραφικές συντεταγμένες αφού ορίσουμε ένα Criteria και πάρουμε τον Provider.

```
public class LocationTestActivity extends Activity  implements LocationListener  {

 private TextView latituteField;
 private TextView longitudeField;
 private TextView providerField;
 private LocationManager locationManager;
 private String provider;

 @Override
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   latituteField = (TextView) findViewById(R.id.TextView02);
   longitudeField = (TextView) findViewById(R.id.TextView04);

   locationManager = (LocationManager)
                   getSystemService(Context.LOCATION_SERVICE);

   Criteria criteria = new Criteria();
   criteria.setAltitudeRequired(false);
   criteria.setCostAllowed(false);

   provider = locationManager.getBestProvider(criteria, true);

   Location location = locationManager.getLastKnownLocation(
                               LocationManager.GPS_PROVIDER);

   if (location != null) {
      onLocationChanged(location);
   } else {
      latituteField.setText("Location not available");
      longitudeField.setText("Location not available");
   }
  }
```

```
@Override
protected void onResume() {
 super.onResume();
 locationManager.requestLocationUpdates(
                                      provider, 2000, 10f, this);

 providerField = (TextView) findViewById(R.id.TextView11);
 providerField.setText(provider);
}


@Override
protected void onPause() {
 super.onPause();
 locationManager.removeUpdates(this);
}

@Override
public void onLocationChanged(Location location) {
 double lat = location.getLatitude();
 double lng = location.getLongitude();
 latituteField.setText(String.valueOf(lat));
 longitudeField.setText(String.valueOf(lng));
}

@Override
public void onStatusChanged(String provider, int status,
                                                  Bundle extras) {
 // TODO Auto-generated method stub

 switch (status) {
  case LocationProvider.AVAILABLE:
              Toast.makeText(this, "AVAILABLE" ,
                     Toast.LENGTH_SHORT).show();
              break;
  case LocationProvider.OUT_OF_SERVICE:
              Toast.makeText(this, "OUT OF SERVICE" ,
                     Toast.LENGTH_SHORT).show();
              break;
  case LocationProvider.TEMPORARILY_UNAVAILABLE:
              Toast.makeText(this, "TEMPORAILY UNAVAILABLE" ,
                     Toast.LENGTH_SHORT).show();
              break;
  }
 }

 @Override
 public void onProviderEnabled(String provider) {
   Toast.makeText(this, "Enabled new provider " + provider,
                                      Toast.LENGTH_SHORT).show();

  }

  @Override
  public void onProviderDisabled(String provider) {
    Toast.makeText(this, "Disabled provider " + provider,
                                 Toast.LENGTH_SHORT).show();
  }
 }
```

5

5.1 Bluetooth

Το Android υποστηρίζει Bluetooth APIs από Level 5 και πάνω, που επιτρέπει στη συσκευή να επικοινωνεί ασύρματα με συσκευές Bluetooth. Χρησιμοποιώντας το Bluetooth APIs η εφαρμογή μπορεί να εκτελέσει τα ακόλουθα:

- Εύρεση άλλων συσκευών Bluetooth.
- Query το Local Bluetooth Adapter για "Paired" συσκευές Bluetooth
- Εγκατάσταση καναλιών RFCOMM/Sockets
- Σύνδεση με συγκεκριμένα Sockets σε άλλες συσκευές
- Μετάδοση δεδομένων από και προς άλλες συσκευές
- Διαχείριση πολλαπλών συνδέσεων.

Οι συσκευές Bluetooth και οι συνδέσεις διαχωρίζοντα από τα παρακάτω αντικείμενα:

- **`BluetoothAdapter`** Ο `BluetoothAdapter` αντιπροσωπεύει το τοπικό Bluetooth, δηλαδή το Bluetooth της συσκευής που τρέχει η εφαρμογή.

- **`BluetoothDevice`** Κάθε απομακρυσμένη συσκευή που επιθυμούμε να συνδεθούμε αντιπροσωπεύεται από το ένα `BluetoothDevice` αντικείμενο.

- **`BluetoothSocket`**
  Καλώντας το `createRfcommSocketToServiceRecord` σε ένα `BluetoothDevice` αντικείμενο δημιουργεί ένα Bluetooth Socket που μας επιτρέπει να κάνουμε μια αίτηση σύνδεσης με την απομακρυσμένη συσκευή για να ξεκινήσει η επικοινωνία.

- **`BluetoothServerSocket`** Δημιουργώντας ένα Bluetooth Server Socket χρησιμοποιώντας την μέθοδο `listenUsingRfcommWithServiceRecord` στο

τοπικό `BluetoothAdapter`, μπορούμε να ακούσουμε για εισερχόμενες αιτήσεις για σύνδεση από `BluetoothSockets` των απομακρυσμένων συσκευών.

Το τοπικό Bluetooth της συσκευής ελέγχεται από το `BluetoothAdapter`. Για να πάρουμε το Adapter καλούμε την μέθοδο `getDefaultAdapter`. Να σημειωθεί πως είναι δυνατόν μια συσκευή να έχει πολλαπλούς Bluetooth Adapters.

```
BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

Για να διαβάσουμε τα properties του τοπικού Bluetooth Adapter πρέπει να συμπεριλάβουμε στο Manifest το BLUETOOTH permission και για να τροποποιήσουμε τα properties ιδιότητες της τοπικής συσκευής να συμπεριλάβουμε το BLUETOOTH_ADMIN.

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

Το `BluetoothAdapter` μας παρέχει μεθόδους για να διαβάζουμε και να δηλώσουμε τα properties του τοπικού Bluetooth Hardware. Αν είναι ανεχτό και έχουμε BLUETOOTH permission στο Manifest μας, έχουμε πρόσβαση στο όνομα της συσκευής που έχει δηλώσει ο χρήστης με την μέθοδο `getName` και στην MAC address με την μέθοδο `getAddress`. Μπορούμε να ελέγξουμε αν το Bluetooth είναι ενεργοποιημένο με την `isEnable`, αν δεν είναι ενεργοποιημένο δεν μπορούμε να πάρουμε τα properties και φυσικά ούτε να τα αλλάξουμε. Για να τα αλλάξουμε πρέπει να έχουμε δηλώσει το BLUETOOTH_ADMIN permission στο Manifest, ώστε να μπορέσουμε να δηλώσουμε το όνομα μέσω της `setName`.

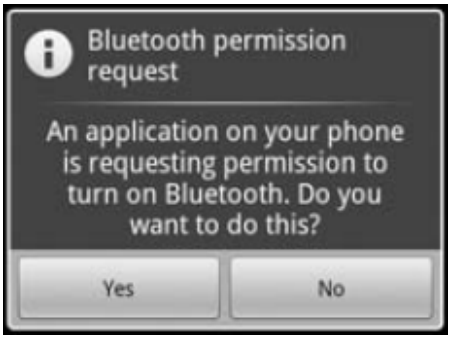
Μπορούμε να βρούμε πληροφορίες για την κατάσταση του Bluetooth Adapter με την μέθοδο `getState` που επιστρέφει μια από τις παρακάτω τιμές:

- STATE_TURNING_ON
- STATE_ON
- STATE_TURNING_OFF
- STATE_OFF

Το `BluetoothAdapter` είναι σε κατάσταση STATE_OFF όταν ξεκινά η εφαρμογή ώστε να μη καταναλώνει την μπαταρία. Για να ενεργοποιήσουμε το Adapter

πρέπει μέσω της `startActivityForResult` να ξεκινήσουμε ένα Activity χρησιμοποιώντας ένα Implicit Intent την `BluetoothAdapter`.ACTION_REQUEST_ENABLE που θα ρωτάει τον χρήστη αν θέλει να ενεργοποιήσει το Bluetooth της συσκευής. Αν ο χρήστης έχει επιλέξει "Yes" θα επιστρέψει στο Parent Activity όταν το Bluetooth έχει ενεργοποιηθεί τελείως.

```
Intent bluetoothRequest = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
stringActivityForResult(bluetoothRequest,0);
```



Εικόνα 5 – 1: Ζητώντας άδεια από τον χρηστή για να ανοίξουμε το Bluetooth.

Όταν όμως έχουμε ορίσει το BLUETOOTH_ADMIN permission στο Manifest μπορούμε να ενεργοποιήσουμε και να απενεργοποιήσουμε καλώντας τις μεθόδους `enable` και `disable`. Επειδή ο χρήστης δεν ρωτάτε για ενεργοποίηση ή την απενεργοποίηση του Bluetooth δεν πρέπει να χρησιμοποιηθεί μόνο σε Explicit Actions, γιατί σε περίπτωση ενός Implicit Action (παραδείγματος χάρη, Intent(com.example.bluetoothActivity)) οποιοσδήποτε μπορεί να καλέσει το Implicit Action και να ενεργοποιήσει ή απενεργοποιήσει το Bluetooth χωρίς την άδεια του χρήστη.

Για να παρακολουθήσουμε πότε το Bluetooth μας αλλάζει καταστάσεις, μπορούμε να χρησιμοποιήσουμε ένα `BroadcastReceiver`. Η ενεργοποίηση και η απενεργοποίηση είναι χρονοβόρες διαδικασίες και είναι καλό να ενημερώνουμε τον χρήστη για αυτές τις αλλαγές. Ο `BroadcastReceiver` ακούει για ACTION_STATE_CHANGED με το Intent να περνά μέσα δύο Extras, το EXTRA_STATE που περιέχει την κατάσταση που βρίσκετε το Bluetooth και το EXTRA_PREVIOUS_STATE που περιέχει την κατάσταση που βρισκόταν το Bluetooth.

```
BroadcastReceiver bluetoothState = new BroadcastReceiver() {
```

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

        int state =
        intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, -1);

        String text = “”;


        switch(state){
                case(BluetoothAdapter.STATE_TURNING_ON):{
                text =“ STATE_TURNING_ON”;
                break;
                }

                case(BluetoothAdapter.STATE_ON):{
                text =“ STATE _ON”;
                unregisterReceiver(this);
                break;
                }
                case(BluetoothAdapter.STATE_TURNING_OFF):{
                text =“ STATE_TURNING_OFF”;
                break;
                }

                case(BluetoothAdapter.STATE_OFF):{
                text =“ STATE_OFF”;
                break;
                }

                default:{ break;
                }
                }//end of case
                Toast.makeText(getApplicationContext(),
                                text, Toast.LENGTH_LONG).show();
        }
};
```

Για να κάνουμε register το `bluetoothState` πρέπει να καλέσουμε την `registerReceiver` όπως φαίνεται παρακάτω.

```
registerReceiver(bluetoothState,
     new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED));
```

Για να δεθούν οι δύο Bluetooth συσκευές πρέπει πρώτα να βρει η μια την άλλη. Η διαδικασία αυτή ονομάζεται Discovery. Για να βρει μια απομακρυσμένη Android συσκευή το τοπικό `BluetoothAdapter` πρέπει να είναι “discoverable”. Το `BluetoothAdapter` έχει τρία modes:

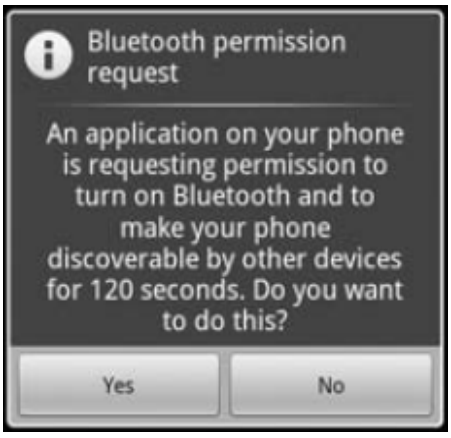
- SCAN_MODE_CONNECTABLE_DISCOVERABLE : Η συσκευή είναι “discoverable” από οποιαδήποτε Bluetooth συσκευή.

- SCAN_MODE_CONNECTEBLE : Η συσκευή είναι “discoverable” σε όσες συσκευές είχαν συνδεθεί και δεθεί με την συσκευή μπορούν να την βρουν κατά την διάρκεια του “discovery” , άλλα οι καινούριες συσκευές δεν μπορούν.

- SCAN_MODE_NONE : Η συσκευή δεν είναι “discoverable” από καμιά συσκευή κατά την διάρκεια του discovery.

Το “discoverability”της συσκευής είναι απενεργοποιημένη και χρειάζεται έγκριση του χρήστη. Αυτό γίνεται ξεκινώντας ένα νέο Activity χρησιμοποιώντας το ACTION_REQUEST_DESCOVERABLE.

```
Intent requestDiscoverable =
    new Intent( BluetoothAdapter.ACTION_REQUEST_DESCOVERABLE);

startActivityForResult(requestDiscoverable,DISCOVERY_REQUEST);
```



Εικόνα 5 – 2: Ζητώντας από το χρηστή να κάνει την συσκευή του “discoverable”

Αυτό θα ενεργοποιήσει το “discovery” για δύο λεπτά. Αν ο χρήστης πατήσει “Yes” θα μας επιστρέψει τον αριθμό τον δευτερολέπτων της διάρκεια του “discovery” και θα ζητήσει από τον χρήστη να ανοίξει το Bluetooth αν δεν είναι ενεργοποιημένο. Αν επιλέξει “No” θα επιστρέψει έναν αρνητικό αριθμό. Την επιλογή του χρήστη θα την χειριστούμε στο `onActionResult`, όπως φαίνεται παρακάτω.

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {

      if (requestCode == DISCOVERY_REQUEST) {
            boolean isDiscoverable = resultCode > 0;

      if (isDiscoverable)
            int discoveravleDuration = resultCode;
      }    }
```

Στην περίπτωση που θέλουμε να αλλάξουμε το χρονικό διάστημα κατά το οποίο η συσκευή θα είναι "discoverable" μπορούμε να περάσουμε μια Extra τιμή στο Intent για το ACTION_REQUEST_DISCOVERABLE μας, όπως φαίνεται στο παρακάτω παράδειγμα χρησιμοποιώντας το EXTRA_DISCOVERABLE_DURATION με μεγίστη τιμή 300 δευτερόλεπτα.

```
      Intent requestDiscoverable =
            new Intent(BluetoothAdapter.ACTION_REQUEST_DESCOVERABLE);

      requestDiscoverable.putExtra(
                  BluetoothAdapter. EXTRA_DISCOVERABLE_DURATION, 300);

      startActivityForResult(requestDiscoverable,DISCOVERY_REQUEST);
```

Φυσικά μπορούμε να παρακολουθήσουμε τις αλλαγές του "scan mode" μέσω ενός BroadcastReceiver(), με Intent Filter το ACTION_SCAN_MODE_CHANGED.

```
      BroadcastReciver bluetoothScanMode = new BroadcastReceiver() {

      @Override
      public void onReceive(Context context, Intent intent) {
            int scanmode = intent.getIntExtra(
            BluetoothAdapter.EXTRA _SCAN_MODE, -1);
            int prevMode = intent.getIntExtra(
            BluetoothAdapter.EXTRA_PREVIOUS_SCAN_MODE, -1);
            }
      };

registerReceiver( bluetoothScanMode, new IntentFilter(
                        BluetoothAdapter.ACTION_SCANMODE_CHANGED));
```

Για να ξεκινήσουμε και να ακυρώσουμε το "discovery" καλούμε τις μεθόδους από το αντικείμενο που φτιάξαμε από το `BluetoothAdapter startDiscovery` και `stopDiscovery`.

```
      bluetoothAdapter.startDiscovery();

      bluetoothAdapter.stopDiscovery();
```

Κατά την διάρκεια του “discovery” το Android μας ειδοποιεί μέσα από το Broadcast Intents όταν ξεκινά μέσω της `BluetoothAdapter`.ACTION_DISCOVERY_STARTED και όταν τελειώνει μέσω της `BluetoothAdapter`.ACTION_DISCOVERY_FINISHED. Κάθε φορά που βρίσκουμε μια συσκευή λαμβάνουμε ένα Broadcast Intent `BluetoothDevice`.ACTION_FOUND που περιέχει Extras EXTRA_DEVICE και EXTRA_CLASS, κι αν είναι διαθέσιμα EXTRA_NAME και EXTRA_RSSI.

```
BroadcastReceiver bluetoothDiscovery =
                                              new BroadcastReceiver(){
    @Override
    public void onReceive(Context context, Intent intent){

    if (BluetoothAdapter.ACTION_DISCOVERY_STARTED.equals(
                                                  intent.getAction()){

            Toast.makeText(getApplicationContext(),
                                              “Discovery Started”,
                                    Toast.LENGTH_SHORT).show();
     }

      if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(
                                                  intent.getAction()){

            Toast.makeText(getApplicationContext(),
                                              “Discovery Finished”,
                                    Toast.LENGTH_SHORT).show();
     }


      if (BluetoothAdapter.ACTION_FOUND.equals(
                                              intent.getAction()){
     BluetoothDevice remoteDevice;
     remoteDevice = intent.getParcelableExtra(
                               BluetoothDevice.EXTRA_DEVICE);

       Toast.makeText(getApplicationContext(),
                   “Found:” +   intent.getStringExtra(
                                    BluetoothDevice.EXTRA_NAME,
                                    Toast.LENGTH_SHORT).show();
     }
```

Η συσκευή Bluetooth που βρέθηκε αντιπροσωπεύεται από το αντικείμενο remoteDevice της κλάσης `BluetoothDevice` που παίρνουμε μέσω της intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE). Ένα Parcelable είναι ένα Interface που μας επιτρέπει να περνάμε αντικείμενα όπως το Serializable στην Java, μόνο πιο γρήγορα.

Η επικοινωνία του Bluetooth στηρίζεται γύρω από το RFCOMM (Radio Frequency Communications protocol). Για να εδραιώσουμε μια RFCOMM σύνδεση χρησιμοποιούμε τις ακόλουθες κλάσεις:

`BluetoothServerSocket` Χρησιμοποιείτε στην μεριά του Server για να φτιάξει ένα Socket που ακούει για εισερχόμενες αιτήσεις σύνδεσης.

`BluetoothSocket` Χρησιμοποιείτε για να φτιάξει έναν Client Socket για να συνδεθεί στον Server Socket.

Για μια σύνδεση Bluetooth Peer-to-Peer χρειαζόμαστε και το Bluetooth Server Socket για να ακούμε για συνδέσεις και το Bluetooth Socket για να ξεκινήσουμε ένα νέο κανάλι και να χειριστούμε τις συνδέσεις. Όταν γίνει η σύνδεση το Socket Server επιστρέφει ένα Bluetooth Socket που χρησιμοποιείται από τον Server για να αποστείλει και να λάβει δεδομένα όπως ο Client, με άλλα λόγια το Client-Server ισχύει μόνο για να δημιουργηθεί η σύνδεση.

Για να ακούσουμε για εισερχόμενες αιτήσεις καλούμε την μέθοδο `listenUsingRfcommWithServiceRecord` από το `BluetoothAdapter`, περνώντας ως ορίσματα ένα String με ένα όνομα του Server και ένα UUID (Universally Unique Identifier) που αποτελείτε από 128 bits. Αυτό θα μας επιστρέψει ένα `BluetoothServerSocket` αντικείμενο. Για να συνδεθεί ο Client πρέπει να γνωρίζει το UUID του Server. Το `BluetoothServerSocket` ακούει για αιτήσεις σύνδεσης από τα απομακρυσμένα Bluetooth.

Για ξεκινήσουμε να ακούμε για αιτήσεις σύνδεσης καλούμε την μέθοδο `accept` του `ServerSocket`, επιστρέφοντας ένα `BluetoothSocket` συνδεδεμένο με την συσκευή του Client. Όταν καλέσουμε το accept η εφαρμογή θα περιμένει μέχρι να έρθει κάποια αίτηση σύνδεσης, για αυτό τον λόγο είναι καλό το `accept` να είναι σε ξεχωριστό `Thread` από ότι το Activity μας.

Αφού βεβαιωθούμε πως το `BluetoothAdaptor` είναι “discoverable” μπορούμε να αρχίσουμε να ακούμε για αίτηση σύνδεσης όπως φαίνεται στο παρακάτω παράδειγμα

```
UUID uuid = UUID.randomUUID();
```

```
Sting name = “Bluetooth Server”;

final BluetoothServerSocket btServer =
       bluetoothAdapter.listenUsingRfcommWithServiceRecord(name, uuid);
Thread acceptThread = new Thread(new Runnable() {
       public void run(){
       try{
              BluetoothSocket serverSocket = btServer.accept();

       } catch (IOException e) {
              Toast.makeText(getApplicationContext(), e.getMessage(),
                     Toast.LENGTH_LONG).show();
       }
       }
}); // End of Thread
acceptThread.start();
```

Από την άλλη μεριά ο Client καλεί την μέθοδο `createRfcommSocketToServiceRecord` στο `BluetoothDevice` που αντιπροσωπεύει τον Server. Υπάρχουν διάφοροι δρόμοι να πάρουμε μια αναφορά στη απομακρυσμένη συσκευή Bluetooth και για να ιδρύσουμε μια σύνδεση πρέπει η απομακρυσμένη συσκευή να είναι “discoverable”και να χρησιμοποιεί `BluetoothServerSocket`, καθώς η τοπική συσκευή και η απομακρυσμένη πρέπει να είναι “paired” ή “bonded. Αν οι συσκευές δεν είναι “paired” ο χρήστης θα ρωτηθεί αν θέλει να τις κάνει “paired” για να ξεκινήσει η αίτηση σύνδεσης.

Έχουμε την δυνατότητα να βρούμε μια απομακρυσμένη συσκευή Bluetooth και μέσω της `getRemoteDevice` στο Bluetooth Adapter αν γνωρίζουμε την MAC Address της συσκευής που θέλουμε.

```
BluetoothDevice wantedDevice =
         bluetoothAdapter.getRemoteDevice(“00:11:22:33:AA:BB”);
```

Επίσης μπορούμε να βρούμε όλες τις συσκευές με τις οποίες έχουμε “paired” καλώντας την μέθοδο `getBondedDevices` στο `BluetoothAdapter`.

```
Set <BluetoothDevice> bondedDevices =
                              bluetoothAdapter.getBondedDevieces();
```

Μπορούμε να συνδυάσουμε τα δύο για να βρούμε μια συσκευή που είναι “paired” με συγκεκριμένο Hardware Address.

```
BluetoothDevice wantedDevice =
      bluetoothAdapter.getRemoteDevice("00:11:22:33:AA:BB");

finel set<BluetoothDevice> bondedDevices =
      Bluetooth.getBondedDevices();

BroadcastReceiver bluetoothDiscovery =
      new BroadcastReceiver() {


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

      BluetoothDevice remoteDevice =
      Intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

      if  ((remoteDevice.equals(wantedDevice) &&
            (remoteDevice.contais(remoteDevice)){

            Toast.makeText(getApplicationContext(),
                                    remoteDevice.getName(),
                                    Toast.LENGTH_LONG).show();
      }
} //End of onReceive
}; //End of BroadcastReceiver

registerReceiver(bluetoothDiscovery,
      new IntentFilter(BluetoothDevice.ACTION_FOUND));

if (!bluetoothAdapter.isDiscovering())
      bluetoothAdapter.startDiscovery();
```

Για να ξεκινήσει η επικοινωνία πρέπει να πάρουμε το Bluetooth Socket από το BluetoothDevice αντικείμενο καλώντας την μέθοδο `createRfcommSocketToServiceRecord` που μας επιστρέφει ένα `BluetoothSocket` αντικείμενο. Το `createRfcommSocketToServiceRecord` είναι από την μεριά του Client, ενώ ο Server καλεί την μέθοδο `listenUsingRfcommWithServiceRecord` από τον `BluetoothAdapter` του. Το `createRfcommSocketToServiceRecord` παίρνει παράμετρο το UUID, το ίδιο UUID που δήλωσε ο Server στο `listenUsingRfcommWithServiceRecord`.

Στην περίπτωση που θέλουμε να συνδεθούμε σε μια συσκευή Bluetooth που δεν είναι "paired" ο χρήστης πρέπει να δεχτεί να κάνει "paired" με τον Server. Το επιστρεφόμενο `BluetoothSocket` αντικείμενο μπορεί να χρησιμοποιηθεί για να συνδεθεί.

```
try{

BluetoothDevice wantedDevice =
      bluetoothAdapter.getRemoteDevice("00:11:22:33:AA:BB");

BluetoothSocket clientSocket =
      wantedDevice.createRfcommSocketToServiceRecord(uuid);

clientSocket.connect();
```

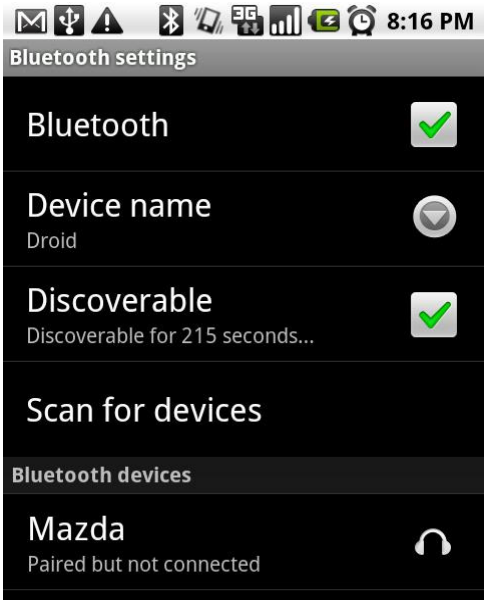
```
} catch(IOException e){
}
```

Όταν η σύνδεση έχει ολοκληρωθεί, ο Server και ο Client μπορούν να επικοινωνήσουν μέσα από τα Bluetooth Socket τους επιτρέποντας μας να στείλουμε και να λάβουμε δεδομένα.

```
OutputStream os;
try {
      os=socket.getOutputStream();
      byte[] bytes = ("hello").getBytes();
      os.write(bytes);
}catch (IOException e) {
}

InputStream is;
      try {
      is=socket.getInputStream();

      if(is !=  -1){
            byte[] bytes = is.read();
            String text =  new String(bytes);
      }
      }catch(IOException e){
      }
```

Παρότι μπορούμε να βρούμε τις συσκευές που μόλις έγιναν "discovered", τις συσκευές που έχουν γίνε "paired" με το κινητό μας, ή να συνδεθούμε με την τεχνική του Client-Server χρησιμοποιώντας το UUID, δεν μπορούμε να κάνουμε "paired" με μια συσκευή που δεν γνωρίζουμε (π.χ. ένα Bluetooth Headset) έτσι θα πρέπει να πάμε μέσα από τα Settings τις συσκευής μας. Μέσα από το μενού Settings – Wireless and network settings – Bluetooth settings μπορούμε να βρούμε και να κάνουμε "paired" την συσκευή μας με κάποια άλλη συσκευή η οποία θα μας εμφανιστεί στο Bluetooth.getBondedDevices. Για να εμφανίσουμε το μενού αυτό μέσα από την εφαρμογή μας θα πρέπει να την καλέσουμε μέσα από ένα Implicit Intent όπως φαίνεται στο παρακάτω παράδειγμα.

```
Intent bluetoothSettingsIntent =  new
      Intent(android.provider.Settings.ACTION_BLUETOOTH_SETTINGS);


startActiviry(bluetoothSettingsIntent);
```



Εικόνα 5 – 3: Ρηθμηζεις του Bluetooth της συστευης.

6

6.1 Βάση Δεδομένων

Μια Βάση Δεδομένων είναι ένα ολοκληρωμένο σύστημα που αποτελείται από λογισμικό και υλικό για την οργανωμένη αποθήκευση, διαχείριση και εύρεση των δεδομένων.

Μια Σχεσιακή Βάση Δεδομένων είναι μια συλλογή από δεδομένα οργανωμένα σε εγγραφές μέσα σε πίνακες με λογική σημασία για εμάς (π.χ. Μια Βάση Δεδομένων ενός σχολείου που περιέχει ένα πίνακα με εγγραφές των μαθητών). Κάθε πίνακας περιέχει ένα σταθερό αριθμό από πεδία (Όνομα, Επώνυμο, Κωδικός Μαθητή) με ένα από αυτά να αποτελεί το Primary Key (Πρωτεύον Κλειδί) της εγγραφής. Το Primary Key είναι μοναδικό για κάθε εγγραφή και χρησιμοποιείται για την ταυτοποίηση της εγγραφής μέσα στον πίνακα.

- Οι πίνακες αυτοί συσχετίζονται μεταξύ τους χρησιμοποιώντας σχέσεις ένα προς ένα, ένα προς πολλά, ή πολλά προς πολλά.
- Για σχέσεις ένα προς ένα το ίδιο Primary Key χρησιμοποιείται σε εγγραφές διαφορετικών πινάκων.
- Για σχέσεις ένα προς πολλά μια εγγραφή έχει ένα επιπλέον πεδίο που περιέχει το Key μιας εγγραφής ενός άλλου πίνακα. Ένα Primary Key σε ξένο πίνακα ονομάζετε Foreign Key (Ξένο Κλειδί).
- Τέλος η σχέση πολλά προς πολλά γίνεται με την χρήση ενός τρίτου πίνακα που περιέχει τα ξένα κλειδιά άλλων πινάκων.

Η Σχεσιακή Βάση Δεδομένων είναι εύκολη στην κατανόηση καθώς εξασφαλίζει ότι οι εγγραφές δεν επαναλαμβάνονται στην ίδια βάση. Τα μειονεκτήματα της είναι ότι έχει περιορισμένους τύπους που μπορούν να πάρουν τα πεδία της, καθώς και τα ερωτήματα μπορούν να γίνουν περίπλοκα. Παρόλα αυτά η Σχεσιακή Βάση Δεδομένων καλύπτει τις ανάγκες της εφαρμογής.

6.2 SQLite DataBase

Το Android μας παρέχει μια SQL βάση δεδομένων, την SQLite. Το SQLite διαβάζει και γράφει απευθείας μέσα στην μνήμη, αντιθέτως με τις περισσότερες άλλες SQL βάσεις. Η βιβλιοθήκη του δεν είναι μεγαλύτερη από 350KiB και προσφέρεται από το Android για κάθε εφαρμογή. Κάθε εφαρμογή μπορεί να φτιάξει ανεξάρτητες σχεσιακές βάσεις δεδομένων. Κάθε βάση αποθηκεύεται στο data/data/<package_name>/databases φάκελο της συσκευής και εξ ορισμού είναι "private", δηλαδή εφαρμογή μόνο για την εφαρμογή.

Το SQLite μας παρέχει πέντε κλάσεις, TEXT, INTEGER, REAL, BLOB και NULL. Το TEXT μας επιτρέπει να βάλουμε Strings με κωδικοποίηση UTF-8, UTF-16BE ή 16LE). Το INTEGER για ακεραίους αριθμούς, 1 – 8 bytes (έκτο από 7 bytes) ανάλογα με το μέγεθος της μεταβλητής. Το REAL είναι ένα 8 bytes float, δηλαδή μπορεί να πάρει αριθμούς με υποδιαστολή. Το NULL για κενές τιμές. Και τέλος BLOB που είναι μια άμορφη μάζα δεδομένων.

Για την απλοποίηση του χειρισμού της βάσης κατασκευάζουμε έναν Database Adapter που θα μας προσφέρει μεθόδους για να εισάγουμε, ενημερώσουμε και να διαγράψουμε εγγραφές μέσα στα Tables μας καθώς διαχειρίζεται τα ερωτήματα και μεθόδους για την δημιουργία, άνοιγμα και κλείσιμο της βάσης.

Αυτή η Database Adapter περιέχει μια εσωτερική κλάση που επεκτείνει την `SQLiteOpenHelper` που θα μας βοηθήσει για την κατασκευή αν δεν υπάρχει, αν υπάρχει και αν χρειάζεται αναβαθμίσεις της βάσης με τις μεθόδους `onCreate`, `onUpgrade`, `onOpen`.

Για να εισάγουμε εγγραφές στα Tables χρησιμοποιούμε αντικείμενα τις κλάσεις `ContentValues`. Το κάθε αντικείμενο αντιστοιχεί σε μια εγγραφή. Τα ερωτήματα στο Android επιστρέφονται ως `Cursor` αντικείμενα. Τα `Cursors` είναι δείκτες που περιέχουν της εγγραφές από τα ερωτήματα που ζητήσαμε από την βάση. Για να κινήσουμε τον δείκτη του Cursor μας παρέχονται οι ακόλουθες μέθοδοι:

- `moveToFirst` Μεταφέρει τον δείχτη στην πρώτη εγγραφή.
- `moveToNext` Μεταφέρει τον δείχτη στην επομένη εγγραφή.

- `moveToPrevious` Μεταφέρει τον δείχτη στην προηγούμενη εγγραφή.
- `moveToPosition` Μεταφέρει τον δείχτη στην εγγραφή στην θέση που του δίνουμε.
- `getPosition` Επιστρέφει την θέση του δείχτη.
- `getCount` Επιστρέφει τον αριθμό των εγγραφών που περιέχει.
- `getColumnIndexOfThrow` επιστρέφει την θέση της στήλης με το όνομα που του δίνουμε (ένα `IllegalAgrumentException` μπορεί να συμβεί)
- `getColumnName` Μας επιστρέφει το όνομα της στήλης με βάση του αριθμού που του δίνουμε.
- `getColumnNames` Μας επιστρέφει ένα πίνακα με τα ονόματα των στηλών των εγγραφών.

Το Android μας παρέχει έναν μηχανισμό για να διαχειριστούμε τους Cursors μέσα από τα Activities μας. Η `startManagingCursor` ενσωματώνει τον κύκλο ζωής ενός Cursor στο Activity που το καλεί. Αφού τελειώσουμε με το Cursor καλούμε την `stopManagingCursor`, αυτό σημαίνει πως το Activity δεν θα κλίσει αυτόματα το Cursor και θα πρέπει να το κλείσουμε εμείς.

Παρακάτω ακολουθεί η δομή μιας κλάσης Adapter

```
public class MyDatabaseAdapter{
	// το Όνομα της βάσης μας
	private static final String DATABASE_NAME =  “myDatabase.db”;
	//τα Tables μας
	private static final String DATBLE_TABLE = “myNamesTable”;
	private static final String DATBLE_TABLE = “myNicknamesTable”;
	//η έκδοση της βάσης μας
	private static final String DATABASE_VESION = 1;
	//ο πίνακας names
	private static final String DATABASE_TABLE_NAMES = “names”;
	//το κλειδί του πίνακα
	private static final String KEY_ID_NAME = “_id_name”;
	// τα υπόλοιπα πεδία μας
	private static final String KEY_FIRST_NAME = “first_name”;
	private static final String KEY_LAST_NAME = “last_name”;
	//ο πίνακας nicknames
	private static final String DATABASE_TABLE_NICKNAMES = “nicknames”;
	//το κλειδί του πίνακα
	private static final String KEY_ID_NICK = “_id_nick”;
	// τα υπόλοιπα πεδία μας
	private static final String KEY_NICKNAME = “nickname”;
	private static final String KEY_FK_NAME = “foreign_key_name”;
```

```
//οι μεταβλητές που θα χρησιμοποιήσουμε
private SQLiteDatabase db;
private final Context context;
private DbHelper myHelper;

//κατασκευαστής
public MyDatabaseAdapter (Context context){
	this.context=context;
}

//άνοιγμα της βάσης
public MyDatabaseAdapter open(){
	myHelper = new DbHelper(context, DATABASE_NAME, null,
				DATABASE_VESION);
	db = myHelper.getWritableDatabase();
	return this;
}

//κλείσιμο της βάσης
public void close(){
	db.close();
}

//εσωτερική κλάση Dbhelper
private static class DbHelper extends SQLiteOpenHelper{

//κατασκευαστής
public DbHelper(Context context){
	super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

//κατά την δημιουργία της βάσης
@Override
public void onCreate(SQLiteDatabase db){
	db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_NAMES + " INTEGER PRIMARY KEY, " +
KEY_FIRST_NAME + " TEXT, " +
KEY_LAST_NAME+ " TEXT);"
);

	db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_NICKNAMES + " INTEGER PRIMARY KEY, " +
KEY_NICKNAME + " TEXT, " +
KEY_FK_NAME + " INTEGER);"
);
}

//κατά την αναβάθμιση της βάσης
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
// καταστροφή των παλαιών πινάκων
db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE_NAMES);
db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE_NICKNAMES);

//δημιουργία από την αρχή
onCreate(db);
}}
```

7

7.1 Εφαρμογή ProjectDoc

Είναι η βασική εφαρμογή που χρησιμοποιεί ο ασθενής στο κινητό του. Σκοπός της είναι να παίρνει τις πληροφορίες από τον Server μας και να τις παρουσιάζει στον ασθενή, να του παρέχει υπηρεσίες Location-Based Services και δυνατότητα σύνδεσης με ιατρικές συσκευές Bluetooth.

Ποιο συγκεκριμένα η εφαρμογή αποτελείται από εννέα Activities, εννέα βοηθητικές κλάσεις και δύο Services. Τα Activities είναι Intro, Login, TabSelection, MainScreen, EventScreen, AlarmSetup, Schedule, MyFirstGoogleMap, BluetoothScreen. Οι βοηθητικές κλάσεις είναι οι POJOPatient, POJODoctor, POJOEvent, POJOMedication, POJOMeasure, POJOAlarm, AlarmList, PatientDBAdapter, JSONHelper. Τέλος τα δύο Service είναι AlarmService και BluetoothHDPService. Όλα τα παραπάνω θα εξηγηθούν αναλυτικά στην συνέχεια.

7.2 POJOClasses

Το POJO είναι τα αρχικά από το Plain Old Java Object. Ένα POJO είναι απλό αντικείμενο χωρίς κανένα χαρακτηριστικό όπως Extend, Implements.

Όλες οι κλάσεις POJO περιέχουν κατασκευαστές και getters setters. Τα POJO αντικείμενα χρησιμοποιούνται για να αντιπροσωπεύουν τα δεδομένα της βάσης που πρόκειται να εισάγουμε, ενημερώσουμε, διαγράψουμε και τα δεδομένα που λαμβάνουμε ή στέλνουμε από και προς τον Server.

Το POJOPatient αντιπροσωπεύει έναν ασθενή με το ΑΜΚΑ ως ID (για αυτό χρειαζόμαστε long για να παίρνει και τους 13 χαρακτήρες), το ονοματεπώνυμο του, τον ID του γιατρού που τον παρακολουθεί και την τελευταία γνωστή θέση του ασθενή από την τελευταία ενημέρωση της τοποθεσίας του μέσου του LocationListener.

```
public class POJOPatient
	private long patientID;
	private String firstName;
	private String lastName;
	private String lastKnownLocation;
	private long doctorID;
```

Το POJODoctor αντιπροσωπεύει ένα γιατρό, με το ID του και το ονοματεπώνυμο του. Δεν περιέχει το ID του ασθενή, η σχέση γιατρού με ασθενή είναι ένα προς πολλά (δηλαδή ένας γιατρός μπορεί να έχει πολλούς ασθενείς).

```
public class POJODoctor
	private long doctorID;
	private String firstName;
	private String lastName;
```

Το POJODEvent αντιπροσωπεύει μια δομή δεδομένων που θα αποσταλεί από και προς την βάση του Server. Το Event έχει την έννοια ενός γεγονότος που ο γιατρός θέλει να στείλει στον ασθενή. Τα γεγονότα αυτά διακρίνονται σε τρεις κατηγορίες. Note, το οποίο είναι το default με καμία επιπλέον πληροφορία, Medication για φαρμακευτική αγωγή και Measure που θα περιέχει τα δεδομένα από τα Data που θα παίρνει από το Bluetooth. Ένα Event έχει τα χαρακτηριστικά που εμφανίζονται από κάτω. Τα μόνα χαρακτηριστικά που δεν είναι ακριβώς αυτό που λένε είναι το date και το type. Το date αποτελείται από ένα οκταψήφιο αριθμό για να χωρέσει μια πλήρης ημερομηνία. Η ημερομηνία αυτή γράφεται ως πρώτα τέσσερα ψηφία να είναι η χρονιά, ακολουθεί ο μήνας και τέλος η ημέρα (δηλαδή 13/8/2012 σε 20120813). Ο λόγος είναι να μπορούμε να κάνουμε αναζήτηση στην βάση και να μας εμφανίζει κατά φθίνω αριθμό. Έτσι όταν ζητάμε το κατά φθίνουσα η αναζήτηση θα μας δώσει την εγγραφή με το μεγαλύτερο date. Το type περιέχει τον τύπο του Event. Το type περιέχει τον πίνακα που αντιστοιχεί ο τύπος. Για παράδειγμα εάν είναι Medication το type θα περιέχει το όνομα του πίνακα medications, εάν είναι Measure θα περιέχει το όνομα του πίνακα measures και εάν είναι Note τότε απλώς θα έχει το όνομα του πίνακα events μιας και το Note δεν προσθέτει κανένα επιπλέον χαρακτηριστικό. Χρησιμοποιώντας αυτήν την τεχνική ξέρουμε από πού θα πρέπει να πάρουμε τα επιπλέον χαρακτηριστικά. Αυτό γίνεται για να αντικαταστήσει την κληρονομικότητα στα αντικείμενα μας, αφού η κληρονομικότητα δεν υπάρχει στις σχεσιακές βάσεις. Αντί αυτού χρησιμοποιείται η σχέση ένα προς ένα όπου ένα Event μπορεί να έχει σχέση με ένα πεδίο του πίνακα medications ή pressures ή με κανένα (δηλαδή είναι Note). Για την αντιστοίχιση ένα προς ένα γίνεται χρησιμοποιώντας το type και το ID, αφού για σχέσεις ένα προς δύο οι εγγραφές των πινάκων πρέπει να μοιράζονται το ίδιο

κλειδί (ID). Έτσι ξέροντας το ID και τον πίνακα που θέλουμε να ψάξουμε μπορούμε να βρούμε την εγγραφή που ψάχνουμε.

```
public class POJOEvent
	private long eventID;
	private long patientID;
	private long doctorID;
	private String title;
	private int date;
	private String note;
	private String type;
```

Το POJODMedication αντιπροσωπεύει ένα φάρμακο με επιπλέον πληροφορίες για τη λήξη της δοσολογίας και την δοσολογία (που παίρνει έως 6).

```
public class POJODoctor
	private long doctorID;
	private String firstName;
	private String lastName;
```

Το POJODMeasure αντιπροσωπεύει την μέτρηση που θα κάνει ο ασθενής μέσα από την ιατρική συσκευή Bluetooth και τα δεδομένα που θα λάβει.

```
public class POJOMeasure
	private long measureID;
	private int type;
	private byte[] data;
```

Το POJOAlarm και το AlarmList είναι μόνο για την συσκευή του ασθενή με σκοπό να βοηθήσουν στην κατασκευή μια υπηρεσίας η οποία θα δώσει στον ασθενή να ορίσει Alarms για τις υπενθυμίσεις για την φαρμακευτική του αγωγή. Στο κεφάλαιο Alarms παρακάτω εμπεριέχονται περισσότερες πληροφορίες

7.3 PatientDBAdapter

Το PatientDBAdapter αποτελεί τον μεσάζοντα μεταξύ της εφαρμογής μας και της SQLite database. Περιέχει μια εσωτερική στατική κλάση για την δημιουργία, και την αναβάθμιση της βάσης.

```
static class DbHelper extends SQLiteOpenHelper{

	public DbHelper(Context context) {
		super(context, DATABASE_NAME , null, DATABASE_VERSION);
	}

	@Override
	public void onCreate(SQLiteDatabase db) {
	  try{
	  db.execSQL("CREATE TABLE IF NOT EXISTS " +  DATABASE_TABLE_PATIENTS +
	  		" (" + KEY_PATIENT_ID + " INTEGER PRIMARY KEY, " +
	  		KEY_PATIENT_FIRST_NAME + " TEXT, " +
	  		KEY_PATIENT_LAST_NAME + " TEXT, " +
	  		KEY_PATIENT_LAST_KNOWN_ADDRESS + " TEXT, "+
	  		KEY_PATIENT_FK_DOCTOR_ID + " INTEGER);"
	  		);

	   db.execSQL("CREATE TABLE IF NOT EXISTS " + DATABASE_TABLE_DOCTORS +
	  		" (" + KEY_DOCTOR_ID + " INTEGER PRIMARY KEY, " +
	  		KEY_DOCTOR_FIRST_NAME + " TEXT, " +
	  		KEY_DOCTOR_LAST_NAME + " TEXT);"
	  		);

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

	@Override
	public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
		db.execSQL("DROP TABLE IF EXISTS "+DATABASE_TABLE_DOCTORS);
		db.execSQL("DROP TABLE IF EXISTS "+DATABASE_TABLE_EVENTS);

		onCreate(db);
		}
	}
```

Χρησιμοποιώντας αυτή τη στατική εσωτερική κλάση, το PatientDBAdapter κατασκευάζει ένα SQLiteDatabase που θα χρησιμοποιηθεί για την ανάγνωση, διαγράφη και ανανέωση πεδίο μέσα στην βάση.

```
	private DbHelper myHelper;
	private SQLiteDatabase myDatabase;

	public PatientDBAdapter(Context c){
		myHelper = new DbHelper(c);
	}

	public void open()throws SQLException{
		myDatabase = myHelper.getWritableDatabase();
	}

	public void close(){
		myHelper.close();
	}
```

7.4 JSONHelper

Το JSONHelper είναι η κλάση με την όποια μετατρέπουμε τα αντικείμενα σε JSON για να αποσταλούν στο διαδίκτυο. Η ονομασία JSON προέρχεται από τα αρχικά JavaScript Object Notation και είναι μια ελαφριά μορφή ανταλλαγής δεδομένων. Η μορφή του είναι text όπως ένα Xml αλλά ποιο ελαφρύ. Οι βιβλιοθήκες για JSON υπάρχουν σε πολλές γλώσσες όπως C, C++, C#, Java, JavaScript, PHP και άλλες. Αυτό σωμένη πως μπορούμε να μετατρέψουμε της μεταβλητές ενός αντικείμενου Java σε JSON, να το στείλουμε και να το παραλάβουμε από την άλλη μεριά μέσω μιας εφαρμογής σε PHP. Οι τιμές που μας επιτρέπει το JSON να περάσουμε μέσα είναι αριθμοί, strings, πινάκες, true, false και null. Η δομή του είναι η ακόλουθη για δύο Events, ένα Medication και ένα Note:

```
[{"events":{
        "doctor_fk_id": 101,
        "patient_fk_id": 012,
        "_event_id": 321,
        "title":"Medication",
        "type":"medicatiions",
        "date":"20130701",
        "note":"take your pills"
        },
        {
        "doctor_fk_id": 101,
        "patient_fk_id": 012,
        "_event_id": 301,
        "title":"Just a Note",
        "type":"events",
        "date":"20120301",
        "note":"Why you never call?"
        }
]
```

Ποιο συγκεκριμένα, ένα αντικείμενο είναι ένα JSONObject ενώ ένας πινάκας είναι ένα JSONArray, δηλαδή αχούμε με βάλει δύο POJOEvents, ένα type medications και ένα και ένα type events σε έναν πίνακα [{“events”:{.

Αυτό που θέλουμε να περιχούμε είναι να αχούμε όλα τα αντικείμενα – στοιχεία του πίνακα μέσα σε ένα JSONObject που θα είναι οργανωμένα σε πινάκες JSONArray που θα περιέχουν μέσα τους τα αντικείμενα που τους αντιστοιχούν.

Στο παρακάτω παράδειγμα φαίνεται πως πέρανε ένα ένα κάθε event μέσα σε ένα καινούριο JSONObject (αν χρησιμοποιηθεί ένα κοινό αντικείμενο JSONObject θα μας μετατρέψει ότι έχουμε βάλει στο JSONArray όπως το τελευταίο event) και τα περνάνε μέσα σε ένα JSONArray. Κάθε τιμή μπαίνει και ένα “Κλειδί” με το όποιο θα μπορέσουμε να εξάγουμε τα POJOEvent. Για τα στοιχεία χρησιμοποιούμε τα ονόματα των στηλών του

πίνακα Events καθώς και για το JSONArray χρησιμοποιούμε το όνομα του ίδιου του πίνακα για να το εισάγαμε στο τελικό JSONObject.

```
        JSONObject jsonEvent = new JSONObject();
        JSONArray jsonEvents = new JSONArray();
        JSONObject jObj = new JSONObject();


for(POJOEvent event : events){
        jsonEvent = new JSONObject();
        jsonEvent.put(PatientDBAdapter.KEY_EVENT_ID, event.getEventID());
        jsonEvent.put(PatientDBAdapter.KEY_EVENT_FK_PATIENT_ID,
                                                    event.getPatientID());
        jsonEvent.put(PatientDBAdapter.KEY_EVENT_FK_DOCTOR_ID,
                                                    event.getDoctorID());
        jsonEvent.put(PatientDBAdapter.KEY_EVENT_TITLE, event.getTitle());
        jsonEvent.put(PatientDBAdapter.KEY_EVENT_DATE, event.getDate());
        jsonEvent.put(PatientDBAdapter.KEY_EVENT_NOTE, event.getNote());
        jsonEvent.put(PatientDBAdapter.KEY_EVENT_TYPE, event.getType());

        jsonEvents.put(jsonEvent);
    }

    jObj.put(PatientDBAdapter.DATABASE_TABLE_EVENTS, jsonEvents);

    return jObj;
```

Για να πάρουμε τα event απλώς αντιστρέφουμε την διαδικασία όπως φαίνεται στο παρακάτω παράδειγμα.

```
   ArrayList<POJOEvent> events = new ArrayList<POJOEvent>();
   JSONArray contacts = jObj.getJSONArray(PatientDBAdapter.DATABASE_TABLE_EVENTS);

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

              JSONObject c = contacts.getJSONObject(i);
              events.add( new POJOEvent(c.getLong(PatientDBAdapter.KEY_EVENT_ID),
                           c.getLong(PatientDBAdapter.KEY_EVENT_FK_PATIENT_ID),
                           c.getLong(PatientDBAdapter.KEY_EVENT_FK_DOCTOR_ID),
                           c.getString(PatientDBAdapter.KEY_EVENT_TITLE),
                           c.getInt(PatientDBAdapter.KEY_EVENT_DATE),
                           c.getString(PatientDBAdapter.KEY_EVENT_NOTE),
                           c.getString(PatientDBAdapter.KEY_EVENT_TYPE)));
              }

       return events;
```

Η χρήση των JSON πρέπει να γίνεται μέσα σε try catch JSONException είτε μέσα στην μέθοδο που υλοποιεί την “ρουτίνα” ή όπου καλείτε η μέθοδος αυτή.

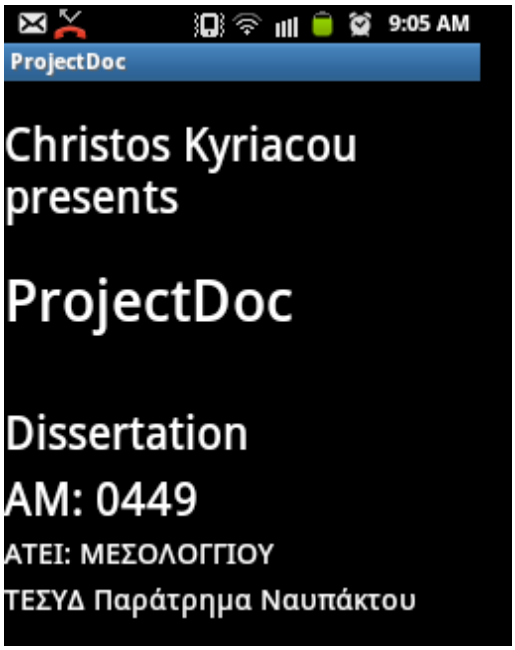
Η κλάση περιέχει μεθόδους για την αποστολή και την λήψη των δεδομένων. Χρησιμοποιώντας ένα DefaultHttpClient και περνώντας του παραμέτρους, για timeout για την περίπτωση που δεν συνδέεται και την περιποίηση που δεν περνούμε δεδομένα πίσω, το οποίο θα εκτέλεση μια αίτηση Post.To JSON περνά μέσα στο entity σε μορφή UTF-8 String μαζί με τα απαραίτητα Tags για τον τύπο και το είδος του format. Τέλος κάνει την αίτηση και παίρνει την απάντηση. Η απάντηση περνά από μια δεύτερη μέθοδο που θα μας δώσει το JSON που μας έστειλε που ήταν μέσα στην απάντηση.

```
public static HttpResponse doPost(String url, JSONObject c) throws
ClientProtocolException, IOException

    {
       int TIMEOUT_MILLISEC = 10000;
       HttpParams httpParams = new BasicHttpParams();
       HttpConnectionParams.setConnectionTimeout(httpParams, TIMEOUT_MILLISEC);
       HttpConnectionParams.setSoTimeout(httpParams, TIMEOUT_MILLISEC);.
       DefaultHttpClient httpclient = new DefaultHttpClient(httpParams);

       HttpPost request = new HttpPost(url);
      StringEntity entity = new StringEntity(c.toString(),"UTF-8");
      entity.setContentType("application/json;charset=UTF-8");
      entity.setContentEncoding((Header) new BasicHeader(HTTP.CONTENT_TYPE,
                                                          "application/json"));
       request.setEntity(entity);
       HttpResponse response;
       response = httpclient.execute(request);
      return response;
    }

public static JSONObject responseToJson(HttpResponse httpResponse) throws
                          IllegalStateException, IOException, JSONException{

       JSONObject jObj = null;
       BufferedReader reader;
       StringBuilder sb;
       String line;
       String jsonSting = "";
       InputStream is = null;

       HttpEntity httpEntity = httpResponse.getEntity();
       is = httpEntity.getContent();
       reader = new BufferedReader(new InputStreamReader(
                                              is, "UTF-8"),10);
       sb = new StringBuilder();
       line = null;
       while ((line = reader.readLine()) != null) {
              sb.append(line + "\n");
       }
       is.close();
       jsonSting = sb.toString();
       jObj = new JSONObject(jsonSting);
       return jObj;
    }
```

7.5 Intro Activity

Το Intro έχει ως στόχο να εμφανίσει για λίγα δευτερόλεπτα μια οθόνη και αμέσως μετά να καλέσει την Login. Για να γίνει αυτό χρησιμοποιούμε ένα `Thread` το οποίο τρέχει για λίγα δευτερόλεπτα και μόλις τελειώσει καλεί την `startActivity` για να ξεκινήσει το Login. Αμέσως μετά καλείτε η `finish` ώστε να τερματίσει το συγκεκριμένο Activity και στην περίπτωση που ο χρήστης πατήσει το Back να μην του ξαναπαίξει το Intro.

```
Thread introTimer = new Thread (){
	public void run(){
		try{
		short countTimer = 0;
		while(countTimer<1000){
			sleep(100);
			countTimer = (short) (countTimer +100);
			}
		startActivity(new Intent(Intro.this,Login.class));
		}catch(InterruptedException e){
		}finally{
		finish();
		}
	}
};

introTimer.start();
```



Εικόνα 7 – 1: Activity Intro

7.6 Login Activity

Στο Login πρέπει να αποστείλουμε τον κωδικό πρόσβασης του χρήστη και να πάρουμε τα δεδομένα αν υπάρχουν. Αν όλα έχουν πάει καλά, τα δεδομένα θα περαστούν στην βάση και ο χρήστης θα περάσει στο Activity MainScreen. Πρώτο πράγμα που πρέπει να κάνει, αφού ο χρήστης έχει γράψει το ΑΜΚΑ του, είναι να δει αν η συσκευή έχει σύνδεση στο Internet. Αν όχι, ένα μήνυμα θα εμφανιστεί στον χρήστη και δεν θα τον αφήσει να στείλει αίτηση στον Server. Αν δεν είναι συνδεδεμένος στο διαδίκτυο μπορεί αν προχωρήσει στο επόμενο Activity χωρείς ανανέωση εάν υπάρχουν τα στοιχεία του στην βάση. Είτε με ή χωρίς Update το ID του χρήστη περνά σε ένα SharedPreferences για όλη την εφαρμογή. Σε κάθε Update ολόκληρη η βάση ανανεώνεται με νέες εγγραφές, αυτό σημαίνει πώς αν ένας άλλος χρήστης κάνει Login τότε θα διαγράψει τα στοιχεία του αλλού χρήστη. Αυτό δεν μας απασχολεί ιδιαίτερα αφού αυτή η περίπτωση είναι σπάνια. Στην εφαρμογή υπάρχει και η επιλογή Demo με την όποια γίνεται μια αυτόματη ανανέωση της βάσεις και αφήνει να πέραση στην επομένη φάση με User ID 109876543210.

Για να κατασκευάσουμε ένα μήνυμα πρώτα φτιάχνουμε έναν “builder” και τον διαμορφώσουμε όπως θέλουμε.

```
AlertDialog.Builder adBuilder = new AlertDialog.Builder(Login.this);

adBuilder.setTitle("No Internet Connection");
adBuilder.setMessage("Please Check if you are Connected to the Internet")
             .setCancelable(false)
             .setPositiveButton("OK", new DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog, int which) {
             dialog.cancel();
             }})
             .setNegativeButton("Off-line Mode",
                    new DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog, int which) {
                    if(db.getPatient(id)!=null){
                           String text = etID.getText().toString();
                           Intent intent = new Intent(Login.this,
                                                  TabSelection.class);
                           intent.putExtra(PatientDBAdapter.KEY_PATIENT_ID, text);
                           startActivity(intent);
                    }else{
                    Toast.makeText(getApplicationContext(),
                            "Wrong ID", Toast.LENGTH_LONG).show();
                    }
             }})
             .setNeutralButton("Demo", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                           Intent intent = new Intent(Login.this,
                                                  TabSelection.class);
                           demo();
```

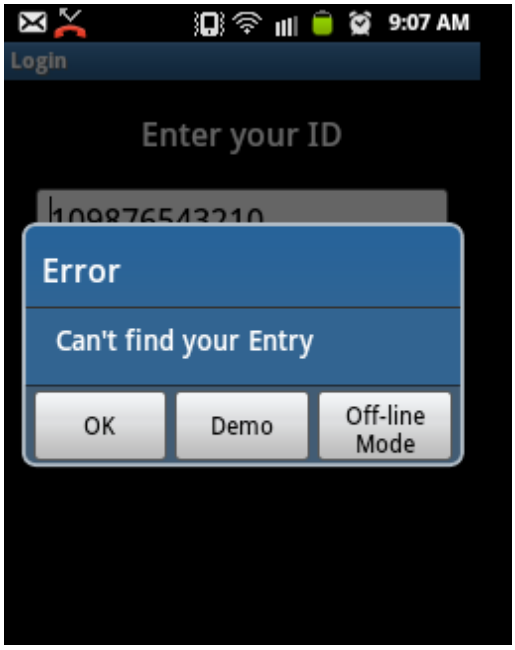
```
                id = 109876543210L;
                Toast.makeText(getApplicationContext(),"DEMO MODE ",
                                                Toast.LENGTH_LONG).show();
                startActivity(intent);
        }});
```

Επειδή μια αποστολή και ανάγνωση δεδομένων από το διαδίκτυο είναι βαριά και χρονοβόρα (αν και διαρκεί μονό λίγα δευτερόλεπτα) διαδικασία είναι καλό να μπει σε ξεχωριστό process. Το Android μας προσφέρει ένα ειδικό τύπο Thread το AsyncTask που μας εξασφαλίζει ότι θα πάρουμε τα δεδομένα και άμεσος μετά μπορούμε πράξουμε αναλόγως. Ποιο συγκεκριμένα, το AsyncTask μας προσφέρει τέσσερεις σημαντικές μεθόδους. Τις doInBackground, onProgressUpdate, onPreExecute και onPostExecute. Τα δυο τελευταία χρησιμοποιούνται για να ενημερώσουν το User Interface του χρήστη πριν και μετά της doInBackground, ενώ το doInBackground είναι το κύριο κομμάτι του Thread που βάζουμε την βαριά ρουτίνα που θέλουμε. Το AsycTask δέχεται τρις μεταβλητές που της ορίζουμε εμείς AsycTask <Params, Progress, Result> με ό,τι δήποτε θέλουμε εμείς. Το Params είναι οι παράμετροι που θα δέχεται το doInBackground είτε Integer, είτε String ότι άλλη κλάση θέλουμε (όχι όμως τύποι int, long αλλά Integer και Long). Το Params είναι ένας πινάκες με τιμές. Το Progress χρησιμοποιείται για την ανανέωση του onProgressUpdate, ενώ το Result είναι το return του doInBackground που θα πέραση στο onPostExecute. Το onProgressUpdate καλείτε μέσα από το doInBackground μέσω της publishProgress για την ανανέωση του User Interface κατά την διάρκεια εκτέλεσης του Thread.

Το LoadingTask αναλαμβάνει να κάνει την βαριά διαδικασία αποστολείς αίτησης (Post) στην σελίδα που αναλαμβάνει να μας στήλη τα δεδομένα (JSON) και να πέραση τα δεδομένα στην βάση. Αν όλα έχουν πάει καλά το errorFlag θα παραμείνει false, αν όχι τότε θα εμφάνιση το μήνυμα (alDialog) αντί να περάσει στην επόμενη φάση.

```
private class LoadingTask extends AsyncTask<Void, Void, Boolean>{
        @Override
        protected Boolean doInBackground(Void... params) {
                String url="http://www.sitehere.com";
                HttpResponse httpResponse;
                JSONObject jObj;
                JSONObject c;
                Boolean errorFlag = false;

                try {
                        c = JSONHelper.idJson(id);
                        httpResponse = JSONHelper.doPost(url, c);
                        jObj = JSONHelper.responseToJson(httpResponse);
```

```
                if(jObj.length()>0 ){
                        JSONHelper.updateDB(jObj, db);
                }else{
                        errorFlag = true;
                }
                } catch (JSONException e) {
                        e.printStackTrace();
                        loadingDialog.dismiss();
                        errorFlag = true;
                } catch (ClientProtocolException e) {
                        e.printStackTrace();
                        loadingDialog.dismiss();
                        errorFlag = true;
                } catch (IOException e) {
                        e.printStackTrace();
                        loadingDialog.dismiss();
                        errorFlag = true;
                }
                return errorFlag;
        }
        @Override
        protected void onPreExecute() {
                super.onPreExecute();
               loadingDialog = new ProgressDialog(Login.this);
                loadingDialog.setMessage("Logging In Please Wait . . .");
                loadingDialog.show();

        }

        @Override
        protected void onPostExecute(Boolean result) {
                super.onPostExecute(result);
                loadingDialog.dismiss();

                if(!result){
                        Intent intent = new Intent(Login.this,
                                                        TabSelection.class);
                        startActivity(intent);
                }else{
                        alDialog.setTitle("Error");
                        alDialog.setMessage("Can't find your Entry");
                        alDialog.show();
                }
        }
}
```



Εικόνα 7 – 2: Login Activity όταν δεν βρίσκει τα στοιχεία.

Εικόνα 7 – 3: Login Activity όταν δεν είναι συνδεδεμένο στον Internet.

7.7 TabSelection Activity

Το TabSelection είναι ένα Activity που περιέχει άλλα Activities σε κάθε Tab. Τα Activities που είναι μέσα σε αυτό περνάνε από τα σταδία onStop και onStart αλλά το TabSelection όχι, μέχρι να φύγει εντελώς από την οθόνη. Εκτός από τα Tabs, αυτό το

Activity μας δίνει ένα κοινό Menu για όλα τα Tabs και όλα ένα κοινό στιγμιότυπο της βάσης ώστε να μην κλείνουμε και ανοίγουμε κάθε φορά που αλλάζουμε Tab. Για την κατασκευή των Tabs θα χρησιμοποιούμε ένα TabHost και του περνάμε TabSpec. Μέσα στο TabSpec περνάμε το κείμενο και την εικόνα που θέλουμε να εμφανίζουμε καθώς και το Intent που θέλουμε να περιέχει.

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

startService(new Intent(this, AlarmService.class ));

db = new PatientDBAdapter(this);
db.open();

mySharedPreferences =
getSharedPreferences("com.chkyriacoy.projectdoc",Context.MODE_PRIVATE);

TabHost tabHost = getTabHost();

TabSpec allSpec = tabHost.newTabSpec("All");
allSpec.setIndicator("All",getResources().getDrawable(R.drawable.star));
Intent allIntent = new Intent(this, MainScreen.class);
allSpec.setContent(allIntent);

TabSpec notesSpec = tabHost.newTabSpec("Notes");
notesSpec.setIndicator("Notes",
            getResources().getDrawable(R.drawable.note));
Intent notesIntent = new Intent(this, MainScreen.class);
notesSpec.setContent(notesIntent);

TabSpec medicationsSpec = tabHost.newTabSpec("Medications");
medicationsSpec.setIndicator("Medications",
            getResources().getDrawable(R.drawable.drug));
Intent medicationsIntent = new Intent(this, MainScreen.class);
medicationsSpec.setContent(medicationsIntent);

TabSpec measuresSpesc = tabHost.newTabSpec("Measures");
measuresSpesc.setIndicator("Measures",
            getResources().getDrawable(R.drawable.blutooth));
Intent measuresIntent = new Intent(this, MainScreen.class);
measuresSpesc.setContent(measuresIntent);

tabHost.addTab(allSpec);
tabHost.addTab(notesSpec);
tabHost.addTab(medicationsSpec);
tabHost.addTab(measuresSpesc);

}
```

Και φυσικά, μια μέθοδος που θα μας επιτρέψει να πάρουμε το Database από τα Tabs.

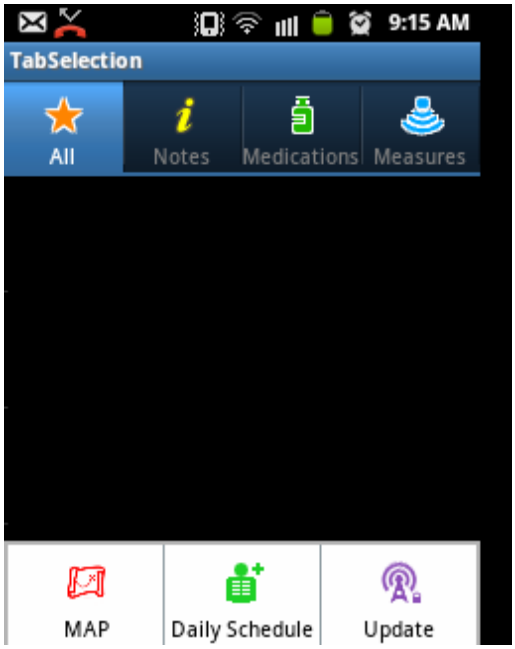
```
protected PatientDBAdapter getDB(){
	return this.db;
}
```

Τέλος το Menu περιέχει τρις επιλογές. Μια για να ξεκινήσει το Activity για το Google Map, μια για να ξεκινήσει το Ημερήσιο Πρόγραμμα Φαρμακευτικής Αγωγής, το Schedule Activity και φυσικά το μια επιλογή για Update.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

	switch (item.getItemId()){
	case R.id.menu_map:
		Intent mapIntent = new Intent(this, MyFirstGoogleMap.class);
		startActivity(mapIntent);
		return true;

	case R.id.menu_schedule:
		Intent scheduleIntent = new Intent(this, Schedule.class);
		startActivity(scheduleIntent);
		return true;


	case R.id.menu_update:
		ConnectivityManager connMgr = (ConnectivityManager)
		getSystemService(Context.CONNECTIVITY_SERVICE);
		NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

		if (networkInfo != null && networkInfo.isConnected()) {
		long id = mySharedPreferences.getLong(
				PatientDBAdapter.KEY_PATIENT_ID, -1);
			if (id>-1){
				new LoadingTask().execute(id);
			}else{
			Toast.makeText(getApplicationContext(),
					"Error 37. . .",
					Toast.LENGTH_LONG).show();
			}
		}else{
		Toast.makeText(getApplicationContext(),
				"Your device is not connected to the internet",
				Toast.LENGTH_LONG).show();
		}
		return true;
	}
	return false;
}
```

Το LoadingTask δεν έχει κάποια διαφορά με το προηγούμενο παρά μονό το ότι απλός δείχνει ένα μήνυμα χωρίς κουμπιά.

```
@Override
protected void onPostExecute(Boolean result) {
	super.onPostExecute(result);
	loadingDialog.dismiss();
	if(result){
		AlertDialog.Builder adBuilder =
			new AlertDialog.Builder(TabSelection.this);
		adBuilder.setTitle("ERROR");

		adBuilder.setMessage("Can't find Server")
			.setCancelable(false)
			.setPositiveButton("OK",
			 new DialogInterface.OnClickListener() {
			  public void onClick(DialogInterface dialog,
								int which) {
				dialog.cancel();
			}
		});

		AlertDialog alDialog = adBuilder.create();
		alDialog.show();
	}
}
```



Εικόνα 7 – 4: TabSelection Activity με το menu.

7.8 MainScreen Activity

Το MainScreen περιέχει μια λίστα που απεικονίζει τα Events που αντιστοιχούν στο Tab που διάλεξε ο χρήστης. Ο χρήστης έχει τέσσερα Tags να επιλέξει: All, Notes, Medications και Measures. Πρώτου προχωρήσουμε στον εμπλουτισμό της λίστας, πρέπει να πάρουμε την βάση και το επιλεγμένο Tab από τον "πατερά" και του Tab που έχει επιλεχτεί στο onCreate.

```
TabSelection parent = (TabSelection) getParent();
db = parent.getDB();
tabHost = parent.getTabHost();
```

Ενώ στο onResume κοιτάμε ποιο Tab είναι επιλεγμένο και ζητάμε από την βάση να πάρουμε τα events που αντιστοιχούν.

```
ArrayList<POJOEvent> events = null;

if(tabHost.getCurrentTabTag().equals("All")){
	events = db.getEvents();
} else if(tabHost.getCurrentTabTag().equals("Notes")){
	events = db.getEventsNotes();
} else if(tabHost.getCurrentTabTag().equals("Medications")){
	events = db.getEventsMedications();
} else if(tabHost.getCurrentTabTag().equals("Measures")){
	events = db.getEventsMeasures();
} else {
	events = null;
}
```

Έχοντας τα αντικείμενα που θέλουμε να εμφανίσουμε στην λίστα αυτό που μας μένει είναι η φτιάξουμε ένα ArrayAdapter που θα πάρει δεδομένα από τα events. Κάθε σειρά στο λίστα αποτελείτε από Views, για αυτό τον λόγο χριζόμαστε μια εσωτερική κλάση που τα αντικείμενα της θα κρατούν τις τιμές των Views. Ο ViewHolder αποτελείται από τρία Views και δύο μεθόδους, μια για τον κατασκευαστή και μια για ενημέρωση των Views.

```
class ViewHolder{
	public ImageView icon = null;
	public TextView title = null;
	public TextView brief = null;


ViewHolder(View row) {
	icon = (ImageView)row.findViewById(R.id.iv_list_item);
	title = (TextView)row.findViewById(R.id.tv_list_item_title);
	brief = (TextView)row.findViewById(R.id.tv_list_item_brief);
}
```

```
        void populateFrom(POJOEvent event){
                if(event.getType().equals(PatientDBAdapter.DATABASE_TABLE_EVENTS)){
                        icon.setImageDrawable(getResources()
                                        .getDrawable(R.drawable.note));
                }else if(event.getType()
                        .equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){

                        icon.setImageDrawable(getResources()
                        .getDrawable(R.drawable.drug));
                }else if(event.getType()
                        .equals(PatientDBAdapter.DATABASE_TABLE_MEASURES)){
                        icon.setImageDrawable(getResources()
                        .getDrawable(R.drawable.blutooth));
                }else {
                        icon.setImageDrawable(getResources()
                        .getDrawable(R.drawable.me));
                }

                title.setText(event.getTitle());
                brief.setText(event.getFormedDate());
        }
}
```

Το title και το brief παίρνονται από τα event άλλα το ImageView αποφασίζεται με βάση συγκρίνοντας το type του event με το όνομα του πινάκων τις PatientDBAdapter. Ο ArrayAdapter είναι μας προσφέρει την getView που μας προσφέρει τρις μεταβλητές position, convertView που θα γίνει Tag με τον ViewHolder μας και τέλος το parent που δεν θα χρειαστούμε. Το getView χρησιμοποιείτε από τον ArrayAdapter για να πάρει το View που θα χρησιμοποίηση. Για αν φτιάξουμε τον δικό μας ArrayAdapter απλός επεκτείνουμε ένα ArrayAdapter και δηλώνουμε το είδος των αντικειμένων που θέλουμε να περνά.

```
class EventAdapter extends ArrayAdapter<POJOEvent>{

        EventAdapter(){
                super(MainScreen.this,
                 android.R.layout.simple_list_item_1, arrayOfEvents);
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
                ViewHolder holder;
                if(convertView == null){
                        LayoutInflater inflater = getLayoutInflater();
                        convertView=inflater
                        .inflate(R.layout.listview_item_event, null);
                        holder = new ViewHolder(convertView);
                        convertView.setTag(holder);
                }else{
                        holder=(ViewHolder) convertView.getTag();
                }
                holder.populateFrom(arrayOfEvents.get(position));
                return(convertView);
                }
        }
```

Ο κατασκευαστής περιέχει ένα default list item απλός για την αρχικοποίηση, το οποίο θα αλλάξουμε στο getView. Το getView περιέχει ένα ViewHolder το οποίο γίνεται tag με το convertView. Ποιο συγκεκριμένα περνούμε κάνουμε inflater το convertView με το xml που έχουμε φτιάξει. Αυτό το xml περιέχει τα ιδία Views που περιέχει και το ViewHolder. Στο τέλος έχουμε ένα ViewHolder που είναι tag με το convertView και όταν αλλάζουμε το holder καλώντας την populateFrom θα αλλάξει και το convertView με τις ίδιες αλλαγές.

Για τον εμπλουτισμό του ListView χιαζόμαστε τρία αντικείμενα. Ένα ListView για το View της λίστας, ένα ArrayList<POJOEevent>() που θα κρατά τα Events που θέλουμε και το EventAdapter που φτιάξαμε.

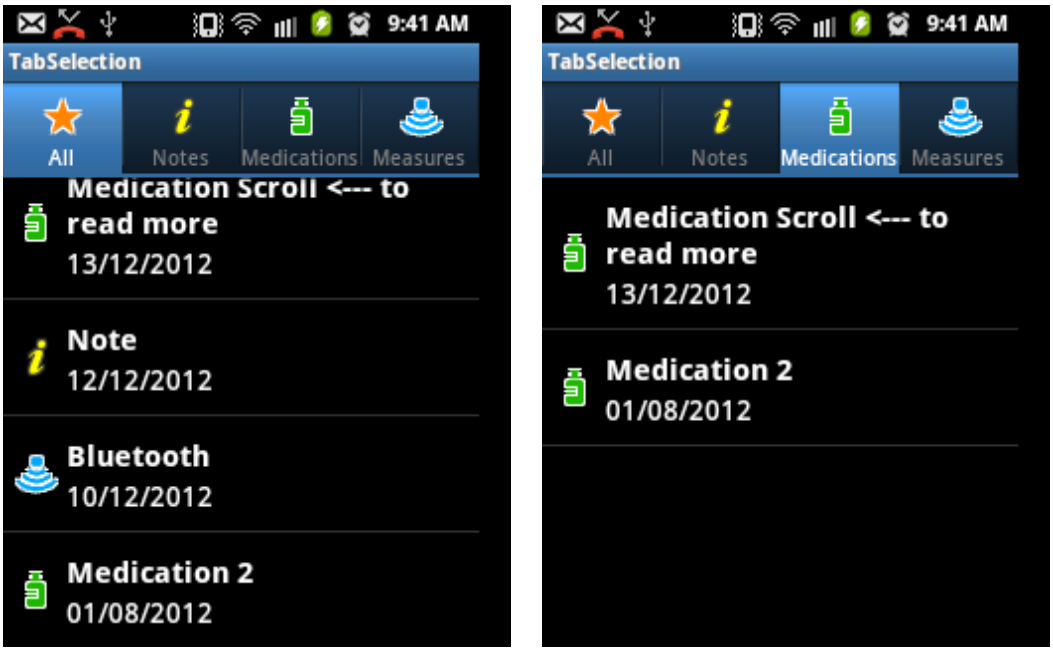
```
listView = (ListView)findViewById(R.id.lv_events);
arrayOfEvents = new ArrayList<POJOEvent>();
eventAdapter = new EventAdapter();
listView.setAdapter(eventAdapter);
```

και για το onResume που θα τρέχει κάθε φορά θέλουμε να καθαρίζουμε την λίστα arrayOfEvents, να βρίσκουμε τα events που θέλουμε, να τα περνάμε ένα ένα μέσα στην λίστα arrayOfEvents και τέλος να ανανεώσουμε τον adapter.

```
if(!arrayOfEvents.isEmpty()){
	arrayOfEvents.clear();
}

ArrayList<POJOEvent> events = null;
if(tabHost.getCurrentTabTag().equals("All")){
	events = db.getEvents();
} else if(tabHost.getCurrentTabTag().equals("Notes")){
	events = db.getEventsNotes();
} else if(tabHost.getCurrentTabTag().equals("Medications")){
	events = db.getEventsMedications();
} else if(tabHost.getCurrentTabTag().equals("Measures")){
	events = db.getEventsMeasures();
} else {
	events = null;
}


for(POJOEvent event: events){
	arrayOfEvents.add(event);
}
eventAdapter.notifyDataSetChanged();
```

Τέλος θέλουμε να πατάμε πάνω στα αντικείμενα της λίστας και να πηγαίνουμε στο αντίστοιχο event. Για αυτό χρησιμοποιούμε έναν OnItemClickListener για αυτό τον σκοπό που θα μας δόση την θέση του event του arrayOfEvents και περνώντας το ID ως extra στο Intent πριν καλέσουμε την startActivity.

```
listView.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> parent,
                        View view, int position, long id) {

      POJOEvent event = arrayOfEvents.get(position);
      Intent data = new Intent(MainScreen.this,EventScreen.class);
      data.putExtra(PatientDBAdapter.KEY_EVENT_ID, event.getEventID());
      startActivity(data);
      }});
```



Εικόνα 7 – 5: MainScreen Activity μεσα στο TabSelection για All και Medications Tabs.

7.9 EventScreen Activity

Το EventScreen αναλαμβάνει να παρουσιάσει όλα τα περιεχόμενα ενός Event. Το Activity μέσα από το Intent που το κάλεσε (getIntent) παίρνει το ID του Event που πατήθηκε από την λίστα στο MainScreen Activity. Χρησιμοποιώντας αυτό το ID μπορούμε να πάρουμε το Event που αντιστοιχεί και να ελέγξουμε τι type έχει. Ανάλογα το type το layout θα εμφανιστεί διαφορετικά. Με άλλα λόγια αντί να έχουμε τρία διαφορετικά layouts για το κάθε type έχουμε ένα layout που ανάλογα το type αλλάζει. Επίσης η συμπεριφορά του αλλάζει με βάση το type.

Ποιο συγκεκριμένα όταν το type είναι “events” τότε πρέπει να αποκρύψει οτιδήποτε περισσεύει, δηλαδή Buttons και TextViews. Αν είναι “measures” τότε πρέπει να μας

εμφανίσει μια επιλογή να καλέσουμε το BluetoothScreen Activity και αν έχει δεδομένα να εμφανίσει ένα επιπλέον Button για να στείλει τα δεδομένα μέσα από Socket. Και τέλος αν είναι "medication" να μας δώσει την δυνατότητα να ορίσουμε Alarms για να ορίσουμε υπενθυμίσεις που θα καλούνται και θα μας ανοίγουν το Schedule με τα medications που ορίσαμε.

Για τον καλύτερο χειρισμό του User Interface έχουμε βάλει όλες τις αλλαγές που πρέπει να γίνουν μέσα στην μέθοδο setupUI. Πριν ασχοληθούμε με το type πρώτα πρέπει να ορίσουμε όλα τα Views που θα χρησιμοποιήσουμε και να πάρουμε το Event από την βάση, βάση του ID που πήραμε από το Intent που άρχισε αυτό το Activity.

```
private void setupIU(){
	tvTitle = (TextView) findViewById(R.id.tv_event_title);
	tvDate = (TextView) findViewById(R.id.tv_event_date);
	tvEndDate = (TextView) findViewById(R.id.tv_event_enddate);
	tvDosageLabel = (TextView) findViewById(R.id.tv_event_dosage_label);
	tvDosageVolume = (TextView) findViewById(R.id.tv_event_dosage_volume);
	tvNote = (TextView) findViewById(R.id.tv_event_note);
	btButton = (Button) findViewById(R.id.bt_event_button);
	btSend = (Button) findViewById(R.id.bt_event_send);
	ivIcon = (ImageView) findViewById(R.id.iv_event_icon);

	Intent intent = this.getIntent();
	eventID = intent.getLongExtra(PatientDBAdapter.KEY_EVENT_ID, -1);
	event = db.getEvent(eventID);

	String type = event.getType();

	tvTitle.setText(event.getTitle());
	tvTitle.setMovementMethod(new ScrollingMovementMethod());
	tvDate.setText(event.getFormedDate());
	tvNote.setText(event.getNote());
	tvNote.setMovementMethod(new ScrollingMovementMethod());
```

Στην περίπτωση που το type είναι "events" τότε θέλουμε να αποκρύψουμε τα Buttons κάνοντας τα "GONE", και αδειάζουμε τα περιεχόμενα από τα TextView tvEndDate, tvDosageLabel, tyDosageVolume. Τέλος αλλάζουμε το ImageView ivIcon σε "note".

```
	String type = event.getType();

	if(type.equals(PatientDBAdapter.DATABASE_TABLE_EVENTS)){
		tvEndDate.setText("");
		tvDosageLabel.setText("");
		tvDosageVolume.setText("");
		btButton.setVisibility(Button.GONE);
		btSend.setVisibility(Button.GONE);
		ivIcon.setImageDrawable(getResources()
				.getDrawable(R.drawable.note));
```

Εάν το type είναι “medications” παίρνουμε από την βάση το medication μέσω του eventID που αντιστοιχεί. Στην συνέχεα ορίζουμε τα TextView και το ImageView όπως περιγράφονται παρακάτω.

```
}else if(type.equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){
	medication = db.getMedication(eventID);
	tvEndDate.setText(medication.getFormedEndDate());
	tvDosageLabel.setText("Dosage: ");
	tvDosageVolume.setText(String.valueOf(medication.getDosage()));
	btButton.setVisibility(Button.VISIBLE);
	btSend.setVisibility(Button.GONE);
	btButton.setText("Setup Alarms");
	ivIcon.setImageDrawable(getResources()
				.getDrawable(R.drawable.drug));
```

Ορίζουμε ένα onClickListener το όποιο θα καλέσει το Activity AlarmSetup.

```
btButton.setOnClickListener( new OnClickListener() {
	public void onClick(View v) {
	Intent data = new Intent(EventScreen.this, AlarmSetup.class);
	data.putExtra(PatientDBAdapter.KEY_MEDICATION_DOSAGE,
					medication.getDosage());
	startActivityForResult(data, ALARM_SETUP_CODE);

	}
});
```

Την ίδια πορεία ακλουθεί το Activity όταν το type είναι “measures”. Η πιο βασική διαφορά είναι πως το Button btSend θα εμφανιστεί μόνο εάν υπάρχουν δεδομένα μέσα στο pressure που πήραμε από την βάση.

```
}else if(type.equals(PatientDBAdapter.DATABASE_TABLE_MEASURES)){
	measure = db.getMeasure(eventID);
	tvEndDate.setText("");
	tvDosageLabel.setText("");
	tvDosageVolume.setText("");
	btButton.setVisibility(Button.VISIBLE);
	btSend.setVisibility(Button.VISIBLE);
	btButton.setText("Get Pressure");
	ivIcon.setImageDrawable(getResources()
				.getDrawable(R.drawable.blutooth));

	if(measure.getData()==null){
		btSend.setVisibility(Button.GONE);
	}else{
		btSend.setVisibility(Button.VISIBLE);
	}
```

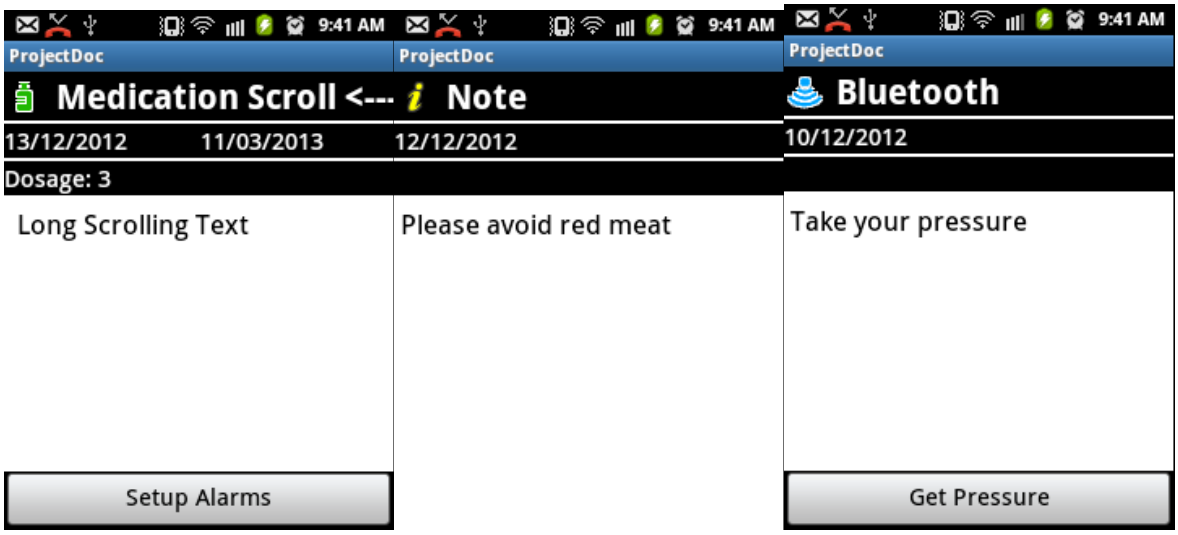
Τα Buttons μας έχουν και αυτά παρόμοια λειτουργία, με το btButton να καλεί το Activity BluetoothScreen.

```
btButton.setOnClickListener( new OnClickListener() {
	public void onClick(View v) {
	Intent intent = new Intent(EventScreen.this,
					BluetoothScreen.class);
	intent.putExtra(PatientDBAdapter.KEY_MEASURE_ID,
					measure.getMeasureID());
	startActivityForResult(intent, BLUETOOTH_CODE);
	}
});
```

Για το btSend εφόσον υπάρχουν δεδομένα στο pressure για να αποσταλούν θέλουμε να ανοίξουμε ένα Socket με τον Server. Αφού έχουμε ελέγξει πως υπάρχει σύνδεση στο Internet, εμπλουτίζουμε την αίτηση μας με το ID του Event (αφού το Event περιέχει τα κλειδιά του γιατρού και του ασθενή καθώς και το type δεν χρειάζεται να στείλουμε καμία άλλη πληροφορία.

```
btSend.setOnClickListener( new OnClickListener() {
	public void onClick(View v) {
	ConnectivityManager connMgr = (ConnectivityManager)
	getSystemService(Context.CONNECTIVITY_SERVICE);
	NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
	if (networkInfo != null){
		if(networkInfo.isConnected()) {
			new LoadingTask().execute();
		}else{
		Toast.makeText(getApplicationContext(),
			"Your device is not connected to the internet",
				Toast.LENGTH_LONG).show();
		}
	}else{
		Toast.makeText(getApplicationContext(),
		"Your device is not connected to the internet",
		Toast.LENGTH_LONG).show();
		}
	}
});
```

Το EventScreen, όπως αναφέρθηκε παραπάνω, καλεί δύο άλλα Activities ανάλογα με το type. Για αυτό τον λόγο χρειαζόμαστε το onActivityResult. Για την καλύτερη κατανόηση θα χωρίσουμε την υπόλοιπη ανάλυση του EventScreen σύμφωνα με τις λειτουργίες AlarmSetup και BluetoothScreen.



Εικόνα 7 – 6: EventScreen Activity στις τρις διαφορετικώς περιπτώσεις

7.10 Alarm

7.10.1 AlarmSetup Activity

Το AlarmSetup έχει σκοπό να δώσει την δυνατότητα στον ασθενή να ορίσει ώρες για την λήψη της ιατρικής αγωγής του. Το layout του εμφανίζει το πολύ έξι TimePicker, ανάλογα με ποσό dosage αναφέρεται μέσα στο medication, για να επιλέξει τι ώρα θέλει να τον υπενθυμίσει. Το tvs περιέχει όλα τα TextView και το tps όλα τα TimePicker.

```
for(int i=0;i<dosage;i++){
	tvs.get(i).setVisibility(View.VISIBLE);
	tps.get(i).setVisibility(View.VISIBLE);
	tps.get(i).setIs24HourView(true);
	}
```

Το layout περιλαμβάνει έξι TimePicker για τον ασθενή να επιλέξει τις ώρες που θέλει, δύο πίνακες int για να κρατάμε στην μνήμη τις ώρες και τα λεπτά που έχει επιλέξει ώστε να μην χρειάζεται να ξαναβάζει τις ώρες από την αρχή όταν ξανά ανοίξει το Activity.

```
mySharedPreferences = getSharedPreferences("com.chkyriacoy.projectdoc",
                                                    Context.MODE_PRIVATE);
editor = mySharedPreferences.edit();

private void save(){

	if(mins[0]!=-1&&hours[0]!=-1){
		editor.putInt("tp_min_one", mins[0]);
		editor.putInt("tp_hour_one", hours[0]);
```

```
        }
        if(mins[1]!=-1&&hours[1]!=-1){
              editor.putInt("tp_min_two", mins[1]);
              editor.putInt("tp_hour_two", hours[1]);
        }
        if(mins[2]!=-1&&hours[2]!=-1){
              editor.putInt("tp_min_three", mins[2]);
              editor.putInt("tp_hour_three", hours[2]);
        }
        if(mins[3]!=-1&&hours[3]!=-1){
              editor.putInt("tp_min_four", mins[3]);
              editor.putInt("tp_hour_four", hours[3]);
        }
        if(mins[4]!=-1&&hours[4]!=-1){
              editor.putInt("tp_min_five", mins[4]);
              editor.putInt("tp_hour_five", hours[4]);
        }
        if(mins[5]!=-1&&hours[5]!=-1){
              editor.putInt("tp_min_six", mins[5]);
              editor.putInt("tp_hour_six", hours[5]);
        }


        editor.commit();

        Toast.makeText(getApplicationContext(),
                              "The Alarms has been set",
                              Toast.LENGTH_LONG).show();
}
```

Το Activity θα πάρει αυτές της τιμές στο onResume.

```
@Override
protected void onResume() {
        super.onResume();

        mins[0] = mySharedPreferences.getInt("tp_min_one", 0);
        mins[1] = mySharedPreferences.getInt("tp_min_two", 0);
        mins[2] = mySharedPreferences.getInt("tp_min_three", 0);
        mins[3] = mySharedPreferences.getInt("tp_min_four", 0);
        mins[4] = mySharedPreferences.getInt("tp_min_five", 0);
        mins[5] = mySharedPreferences.getInt("tp_min_six", 0);


        hours[0] = mySharedPreferences.getInt("tp_hour_one", 0);
        hours[1] = mySharedPreferences.getInt("tp_hour_two", 0);
        hours[2] = mySharedPreferences.getInt("tp_hour_three", 0);
        hours[3] = mySharedPreferences.getInt("tp_hour_four", 0);
        hours[4] = mySharedPreferences.getInt("tp_hour_five", 0);
        hours[5] = mySharedPreferences.getInt("tp_hour_six", 0);

        for(int i=0;i<dosage;i++){
              if(mins[i]!=-1&&hours[i]!=-1){
              tps.get(i).setCurrentMinute(mins[i]);
              tps.get(i).setCurrentHour(hours[i]);
              }
        }}
```

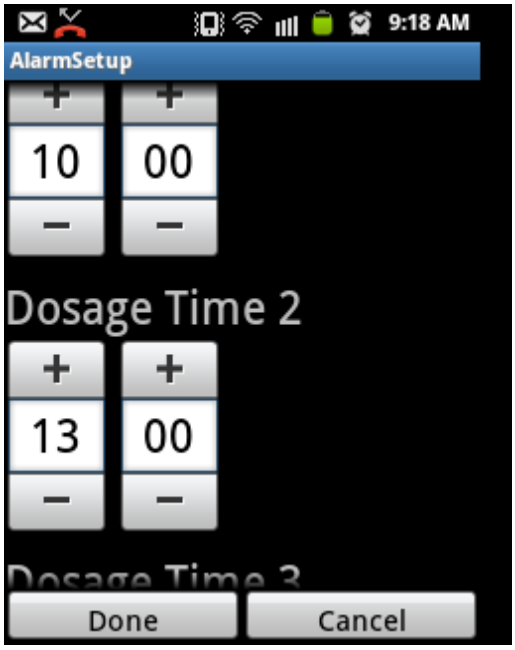
Το AlarmSetup προσφέρει δύο επιλογές μέσα από τα Button btDone και btCancel. Το btDone θα ενημερώσει τους πίνακες hours και mins, θα καλέσει την save(), θα τις περάσει στο Intent και θα της στείλει μαζί με το RESULT_OK πίσω στο EventScreen. Τέλος καλώντας την finish() τερματίζει το AlarmSetup. Για όσα TimesPicker είναι εκτός dosage οι πίνακες θα πάρουν τομές "-1" ώστε να ξέρουμε ποιες τιμές είναι σωστές.

```
btDone = (Button) findViewById(R.id.bt_alarm_setup_done);
btDone.setOnClickListener(new OnClickListener() {
	public void onClick(View v) {
		for(int i=0;i<6;i++){
			if(i<dosage){
				mins[i] = tps.get(i).getCurrentMinute();
				hours[i] = tps.get(i).getCurrentHour();
			}else{
				mins[i] = -1;
				hours[i] = -1;
			}
		}
		Intent data = getIntent();
		save();
		data.putExtra("tp_mins", mins);
		data.putExtra("tp_hours", hours);
		setResult(RESULT_OK, data);
		finish();
	}
});
```

Το btCancel απλώς επιστρέφει RESUST_CANCELED στο EventScreen.

```
btCancel = (Button) findViewById(R.id.bt_alarm_setup_cancel);
btCancel.setOnClickListener(new OnClickListener() {
	public void onClick(View v) {
		setResult(RESULT_CANCELED);
		Toast.makeText(getApplicationContext(),
			"All Alarms have been canceled",
				Toast.LENGTH_LONG).show();
		finish();
	}
});

mySharedPreferences = getSharedPreferences("com.chkyriacoy.projectdoc",
                                                    Context.MODE_PRIVATE);
editor = mySharedPreferences.edit();

}
```

Το EventScreen αναλαμβάνει να χειριστεί αυτά τα Results μέσα από `onActivityResult`. Στο `onActivityResult` θα χρησιμοποιήσουμε της κλάσεις POJOAlarms και AlarmList.



Εικόνα 7 – 7: AlarmSetup Activity

7.10.2 POJOAlarm

Το POJOAlarm περιέχει τρία χαρακτηριστικά που θα χρησιμοποιήσουν στο AlarmService.

```
public class POJOAlarm
	private long alarmID;
	private int time;
	private int endDate;
```

Το χαρακτηριστικό alarmID κρατά το eventID μαζί με έναν επιπλέον αριθμό που δηλώνει σε ποιο TimePicker αντιστοιχεί. Για παράδειγμα για eventID 9887 και TimePicker 2 το alarmID θα είναι 98872. Για να γίνει αυτό πολλαπλασιάζουμε το eventID επί δέκα και προσθέτουμε το TimePicker που αντιστοιχεί στο τέλος.

Το time ακολουθεί την ίδια λογική με ώρες επί δέκα χιλιάδες, τα λεπτά επί εκατό και τα δευτερόλεπτα απλώς να προστίθενται στο τέλος. Ενώ το endDate αντιπροσωπεύει το endDate του medication.

7.10.3 AlarmList

Είναι μια κλάση η οποία επεκτείνει την ArrayList<POJOAlarm> και υλοποιεί όλες τις μεθόδους. Ο λόγος που την χρειαζόμαστε είναι για να αποφύγουμε τα σφάλματα που συμβαίνουν εάν χρησιμοποιήσουμε το ArrayList<POJOAlarm>.

Πίσω στο EventScreen, αναλαμβάνει να χειριστεί το RESULT από το AlarmSetup μέσα από το onActivityResult. Στην περίπτωση που του επιστραφεί RESULT_OK, μέσα από το resultCode, παίρνουμε τα mins και τα hours και φτιάχνουμε ένα AlarmList alarms με POJOAlarms όπως φαίνεται παρακάτω.

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {

      super.onActivityResult(requestCode, resultCode, data);
      switch(requestCode){
            case ALARM_SETUP_CODE:
                  if(resultCode==RESULT_OK){
                        int[]  mins = data.getIntArrayExtra("tp_mins");
                        int[]  hours = data.getIntArrayExtra("tp_hours");

                        AlarmList alarms = new AlarmList();
                        int sec=0;

                  for(int i=0;i<6;i++){
                        if(hours[i]>-1&&mins[i]>-1){
                              alarms.add( new POJOAlarm((
                              event.getEventID()*10)+1+i,
                              (hours[i]*10000)+(mins[i]*100)+sec,
                              medication.getEndDate()));
                        }
                  }
```

Στην συνέχεα στέλνεται το alarms στο AlarmService. Το AlarmService θα παρουσιαστεί στην παρακάτω ενότητα.

```
            try {
                  alarmMessenger.send(Message.obtain(null,
                        AlarmService.MSG_ADD_ALARMS, alarms));
            } catch (RemoteException e) {
                  e.printStackTrace();
            }
      }
```

Εάν το resultCode είναι RESULT_FIRST_USER τότε θα ακυρώσει τα alarms και θα βάλει τα POJOAlarm όπως φαίνεται παρακάτω. Δεν μας ενδιαφέρει τα χαρακτηριστικά time και endDate αλλά το alarmID ώστε να το αφαιρέσουμε από το AlarmService.

```
if(resultCode==RESULT_FIRST_USER){
	AlarmList alarms = new AlarmList();
	for(int i=0;i<6;i++){
		alarms.add( new POJOAlarm((event.getEventID()*10)+1+i,
					0,medication.getEndDate()));
	}
```

Και τέλος στέλνουμε ένα μήνυμα στο AlarmService με τα alarms.

```
	try {
		alarmMessenger.send( Message.obtain(null,
			AlarmService.MSG_REMOVE_ALARMS, alarms));
	} catch (RemoteException e) {
			e.printStackTrace();
	}
}
break;
```

7.10.4 AlarmService

Το AlarmService αναλαμβάνει να θέση ένα alarm από το AlarmManager της συσκευής. Το alarm αυτό μπορεί να στείλει ένα Intent στο Schedule Activity την ώρα που θα του δηλώσουμε με ένα Extra "ring" το οποίο θα ενεργοποίηση το Ringtone και την δόνηση της συσκευής. Όλα τα alarms θα μπουν σε ένα πίνακα κρατώντας στο ID από ποιο medication ήρθαν και σε πιο TimePicker αντιστοιχούν. Αυτό γίνεται πολλαπλασιάζονται επί δέκα του ID του medication και προσθέτοντας το αριθμό του timePicker που αντίστοιχη.

Κατά την εκκίνηση του Service ανοίγουμε την βάση, παίρνουμε το AlarmManager, παίρνουμε την σημερινή ημερομηνία, διαγράφουμε όλες τις εγγραφές που έχουν λήξη βάση του endDate.

```
@Override
public void onCreate() {
	super.onCreate();
	db = new PatientDBAdapter(this);
	db.open();
	myAlarmManager = (AlarmManager)getSystemService(
					Context.ALARM_SERVICE);
	int date = 0;
	Calendar cal = Calendar.getInstance();
	date = cal.get(Calendar.DAY_OF_MONTH);
	date = date + ((cal.get(Calendar.MONTH)+1)*100);
	date = date + (cal.get(Calendar.YEAR)*10000);

	db.deleteOutdatedAlarms(date);
	alarms=null;
}
```

Για να πάρουμε την σημερινή ημερομηνία καλούμε ένα στιγμιότυπο της Calendar και ζητάμε το DAY_OF_MONTH, MONTH+1 (για τι μας επιστρέφει από 0 ως 11) και YEAR, ώστε να φτιάξουμε το date που θέλουμε, για παράδειγμα το 12/8/2012 αντιστοιχεί στο "20120812".

Οι μέθοδοι για να λάβουμε τα alarms με βάση την ώρα και την ημερομηνία καθώς και η διαγραφή βάσει της ημερομηνίας συντάσσονται στο PatientDBAdapter ως εξής.

```
public ArrayList<POJOAlarm> getAlarms(int time, int correntDate){
	String[] columns = new String[]{
		KEY_ALARM_ID,KEY_ALARM_TIME, KEY_ALARM_END_DATE};

	Cursor c = myDatabase.query(DATABASE_TABLE_ALARMS,
		columns, KEY_ALARM_TIME+" >= ?
		AND "+ KEY_ALARM_END_DATE+" >= ?",
		new String[]{Integer.toString(time),
		Integer.toString(correntDate)},
		null, null, KEY_ALARM_TIME);

	if(c.moveToFirst()){
		ArrayList<POJOAlarm> alarms =  new ArrayList<POJOAlarm>();
		int iAlarmID = c.getColumnIndex(KEY_ALARM_ID);
		int iAlarmTime = c.getColumnIndex(KEY_ALARM_TIME);
		int iAlarmEndDate = c.getColumnIndex(KEY_ALARM_END_DATE);

		for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){

		alarms.add(new POJOAlarm(c.getLong(iAlarmID),
				c.getInt(iAlarmTime),c.getInt(iAlarmEndDate)));
		}
	return alarms;
	}else{return null;}
}

public long insertAlarm(POJOAlarm alarm){
	ContentValues cv = new ContentValues();
	cv.put(KEY_ALARM_ID, alarm.getAlarmID());
	cv.put(KEY_ALARM_TIME, alarm.getTime());
	cv.put(KEY_ALARM_END_DATE, alarm.getEndDate());

	return myDatabase.insert(DATABASE_TABLE_ALARMS, null, cv);
}

public int updateAlarm(POJOAlarm alarm){
	ContentValues cv = new ContentValues();
	cv.put(KEY_ALARM_ID, alarm.getAlarmID());
	cv.put(KEY_ALARM_TIME, alarm.getTime());
	cv.put(KEY_ALARM_END_DATE, alarm.getEndDate());

	return myDatabase.update(DATABASE_TABLE_ALARMS, cv, KEY_ALARM_ID+" = ?",
			new String[]{ String.valueOf(alarm.getAlarmID()) });
}
```

```
public int deleteAlarm(POJOAlarm alarm){
	return myDatabase.delete(DATABASE_TABLE_ALARMS,KEY_ALARM_ID+" = ?",
			new String[]{ String.valueOf(alarm.getAlarmID()) });
}

public int deleteOutdatedAlarms(int date){
	return myDatabase.delete(DATABASE_TABLE_ALARMS,KEY_ALARM_END_DATE+" < ?",
				new String[]{ String.valueOf(date) });
}

public int emptyAlarmTable(){
	return myDatabase.delete(DATABASE_TABLE_ALARMS, null, null);
}
```

Η μέθοδος setNextAlarm αναλαμβάνει να θέσει στο AlarmManager το επόμενο alarm. Παίρνουμε τα alarms βάση της ώρας με αυξητική σειρά (OrderBy KEY_ALARM_TIME). Με την Βοήθιο της Calendar παίρνουμε την ημερομηνία και την ώρα (ένας έλεγχος χιάζεται για την περίπτωση που η ώρα είναι 24:00 ώστε να το κάνει 00:00). Στην συνέχεια βρίσκουμε την διαφορά μεταξύ της ώρας του κινητού και της ώρας του alarm. Τέλος παίρνουμε το υψηλότερο στην λίστα alarm και βγάζουμε το eventID, φτιάχνουμε ένα Intent με extras το eventID και ένα Boolean. Στην συνέχεια θα περάσουμε το Intent σε ένα PendingIntent. Δίνοντας το Intent στο PendingIntent επιτρέπουμε στον AlarmManager να χειριστεί αυτό το Intent. Τέλος θέτουμε στο AlarmManager να εκτελέσει το Intent σε τόσα milliseconds και να ξυπνήσει την συσκευή σε περιπτοση που κιματε.

```
	private void setNextAlarm(){
		ArrayList<POJOAlarm> alarms = null;
		int correntCal = 0;

		Calendar cal = Calendar.getInstance();
		correntCal = cal.get(Calendar.DAY_OF_MONTH);
		correntCal = correntCal + ((cal.get(Calendar.MONTH)+1)*100);
		correntCal = correntCal + (cal.get(Calendar.YEAR)*10000);

		int calMin = cal.get(Calendar.MINUTE);
		int calHour = cal.get(Calendar.HOUR_OF_DAY);
		if(calHour==24)calHour=0;
		int calSec = cal.get(Calendar.SECOND);
		int time= (calHour*10000)+(calMin*100)+calSec;

		alarms = db.getAlarms(time, correntCal);

		if(alarms!=null){

		int hour = alarms.get(0).getTime()/10000;
		int min = (alarms.get(0).getTime()/100)%100;
		int sec = alarms.get(0).getTime()%100;
```

```
            min=min-calMin;
            sec=sec-calSec;
            hour=hour-calHour;

            if(sec<0){
                  sec=60+sec;
                  min= min - 1;
            }

            if(min<0){
                  min=60+min;
                  hour=hour-1;
            }

            if(hour<0){
                  hour=24+hour;
            }

            Intent data = new Intent(this, Schedule.class);
            data.putExtra("ring", true);
            PendingIntent pi = PendingIntent.getActivity(this, 0, data, 0);

            time = 3600000*hour +60000*min+ 1000*sec + 1000;

            myAlarmManager.set(AlarmManager.ELAPSED_REALTIME_WAKEUP,
                              SystemClock.elapsedRealtime() + time, pi);
            }else{

                  PendingIntent pi = PendingIntent.getActivity(this, 0,
                                    new Intent(this, Schedule.class), 0);
                  myAlarmManager.cancel(pi);
            }
      }
```

Για να βεβαιωθούμε πως δεν θα χτυπήσω πολλαπλές φορές προσθέτουμε ένα επιπλέον δευτερόλεπτο (+1000) στον χρόνο (time) που θα χτυπούσε κανονικά το alarm. Και τέλος στην περίπτωση που η λίστα είναι άδεια θέλουμε να ακυρώσουμε κάποιο τυχόν alarm που έχει τεθεί.

Για να μπορέσουμε να στείλουμε μηνύματα από το EventScreen στο AlarmService χρειαζόμαστε ένα Handler. Το EventScreen στέλνει τα μηνύματα ανάλογα με το resultCode που παρέλαβε από το AlarmSetup. Με resultCode RESULT_OK το EventScreen στέλνει ένα μήνυμα MSG_ADD_ALARMS που θα ανανεώσει ή θα εισάγει τα alarms που έλαβε. Με resultCode RESULT_CANCELED θα διαγράψει από την βάση τα συγκεκριμένα alarms.

```
public static final int MSG_ADD_ALARMS = 101;
public static final int MSG_REMOVE_ALARMS = 102;

private static class IncomingHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
```

```
                case MSG_ADD_ALARMS:
                alarms = (AlarmList) msg.obj;
                for(POJOAlarm alarm : alarms){
                      if( db.updateAlarm(alarm) == 0){
                            db.insertAlarm(alarm);

                      }
                }
                break;

               case MSG_REMOVE_ALARMS:
                alarms = (AlarmList) msg.obj;
                for(POJOAlarm alarm : alarms){
                      db.deleteAlarm(alarm);
                }
                break;

                default:
                   super.handleMessage(msg);
          }
     }
}
```

Ο `Handler` περνά σε ένα Messenger που θα χρησιμοποιηθεί για να περάσει τα μηνύματα από το EventScreen στο AlarmService. Θα αποσταλεί στο EventScreen μέσα από την `onBind`.

```
final Messenger mMessenger = new Messenger(new IncomingHandler());

@Override
public IBinder onBind(Intent intent) {
       return mMessenger.getBinder();
}
```

Στο onRebind θα καλέσουμε την setNextAlarm() και θα επιστρέψει true ώστε να μπορέσουμε να ξανακαλέσουμε την `onBind`.

```
@Override
public void onRebind(Intent intent) {
      super.onRebind(intent);

      setNextAlarm();
}
```

Στην μεριά του EventScreen για να ενωθούμε με το Service χιαζόμαστε ένα αντικείμενο `ServerConnection` και ένα `Messenger`. Το `mConnection` στο `onServiceConnected` θα μας δώσει το `Messenger` του AlarmService.

```
private Messenger alarmMessenger;

private ServiceConnection mConnection = new ServiceConnection() {
```

```
public void onServiceDisconnected(ComponentName name) {
		alarmMessenger = null;

	}

public void onServiceConnected(ComponentName name, IBinder service) {
		alarmMessenger = new Messenger(service);
	}
};
```

Υπάρχουν δυο σημεία που θέλουμε να ξεκινήσουμε το AlarmService με startService και ένα σημείο που θέλουμε να κάνουμε Rebind. Όταν ο χρήστης κάνει επιτυχώς login δηλαδή στο TabSelection και Schedule όταν τρέχει ένα alarm.

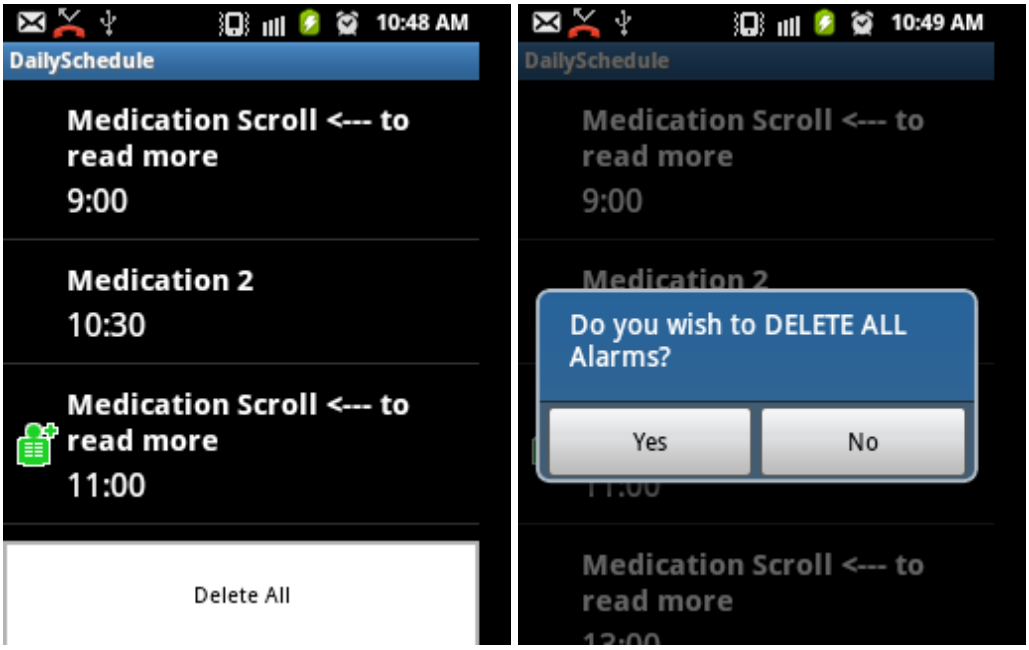
```
startService(new Intent(this,AlarmService.class));
```

Και όταν κάνουμε Rebind στο EventScreen. Η διαφορά με το Bind είναι πως για να γίνει Rebind πρέπει να γίνε Bind για δεύτερη φορά. Βάζοντας το setNextAlarm στο Rebind του AlarmService και κάνοντας Bind στο onResume του EventScreen εξασφαλίζουμε πως θα τρέξει την δεύτερη φορά που το EventScreen περάσει από το onResume. Αυτό μας βολεύει για τι το OnActivityResualt θα τρέξει πριν το onResume, θέτοντας έτσι πρώτα της νέες τιμές των alarm και ύστερα καλώντας την setNextAlarm στο AlarmService.

```
@Override
protected void onResume() {
super.onResume();

if(event.getType().equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){
	bindService(new Intent(this, AlarmService.class),
		mConnection, Context.BIND_AUTO_CREATE);
	}
}
```

Σε κάθε `onPause` θέλουμε να καλέσουμε την `unbindService` και στο `onResume` καλούμε την `bindService` καθώς και ελέγχουμε ένα `Boolean` αν είναι true ή false.

```
@Override
protected void onPause() {
	super.onPause();
	if(event.getType()
		.equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){
		unbindService(mConnection);
	}
}
```



Εικόνα 7 – 8: Schedule Activity με το menu. Το εικονίδιο δείχνει πως μέσα στην επόμενη μια ωρα θα υπενθήμηση τον χρήστη για το Medication 2.

7.10.5 Schedule Activity

Το Schedule Activity έχει σκοπό να εμφάνιση όλα τα alarms που έχουν ορισθεί, παρουσιάζοντας με ένα πράσινο εικονίδιο τα alarms που θα χτυπήσουν μέσα σε μια ώρα. Όταν ο AlarmManager καλέσει αυτό το Activity, η συσκευή ξύπνα αν κάματε και αρχίζει να δονείται και να παίζει μια μελώδια μέχρι ο χρήστης αλλάξει Activity. Το Schedule περιέχει τα γνωστά ListView, ArrayList<POJOAlarm> και AlarmAdapter extends ArrayAdapter<POJOAlarm>, καθώς και OnItemClickListener που θα καλέσει το σωστό EventScreen έχοντας το ID και βάζοντας το στο Intent. Για να φτιάξουμε το “Ringtone και Vibrator” χιαζόμαστε μερικά αντικείμενα. Ένα Ringtone, ένα Uri, ένα Vibrator, ένα long[] για το pattern και ένα flag για όταν καλή ο AlarmManager.

```
private Vibrator vib = null;
private Ringtone r;
private Uri uri;
private boolean flag = false;
private long[] pattern = {500,1000};
```

Το uri θα πάρει ένα default Ringtone τύπου Alarm και χρησιμοποιούνται για να πάρουμε το Ringtone. Το Vibrator το περνούμε από το Σύστημα της συσκευής εάν υπαρχή,

αν όχι παίρνει null. Το flag χρησιμοποιείται για να γνωρίζουμε πότε έχουν παίξει το Ringtone και το Vibrator και να τα κλείσουμε στο onPause.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
uri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);
r = RingtoneManager.getRingtone(getApplicationContext(), uri);
vib = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
}

@Override
protected void onResume() {
// TODO Auto-generated method stub
super.onResume();
Intent intent = this.getIntent();
Flag = intent.getBooleanExtra("ring", false);
       if(flag){
              r.play();
              if(vib!=null){
                     vib.vibrate(pattern,0);
              }
              startService(new Intent(Schedule.this,AlarmService.class));
       }
    }

@Override
protected void onPause() {
// TODO Auto-generated method stub
super.onPause();

if(flag){
       getIntent().putExtra("ring", false);

       r.stop();
              if(vib!=null){
                     vib.cancel();
              }
       }
}
```

7.11 GoogleMap Activity

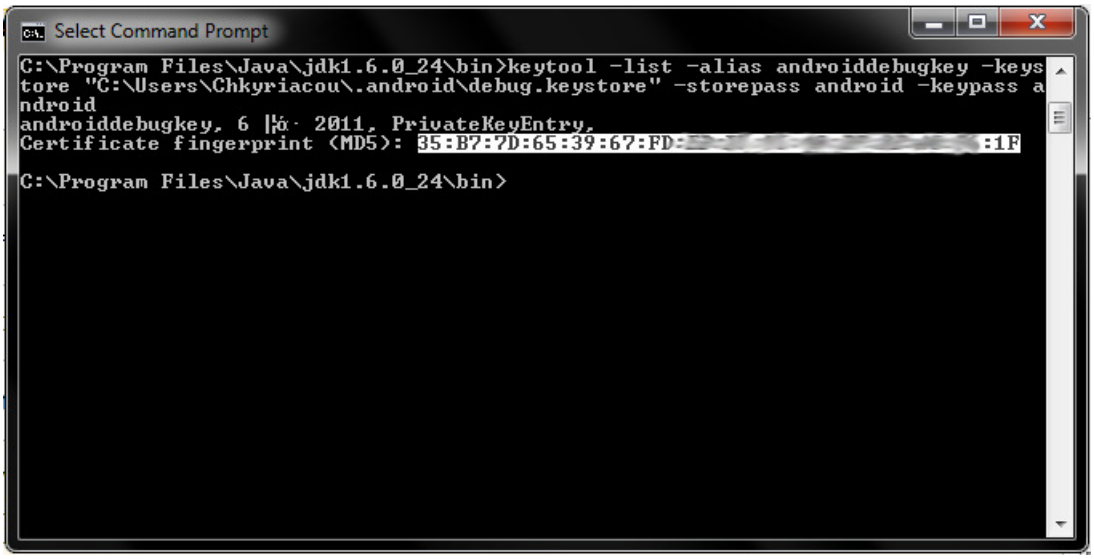
Ο σκοπός αυτού το Activity είναι να εμφανίσει στον ασθενή την θέση του πάνω στο Google Map και με βάση την θέση του να εμφανίσουμε φαρμακεία και νοσοκομεία. Αρχικά χρειαζόμαστε το Google Map να εμφανιστεί στην οθόνη, φτιάχνοντας το xml layout όπως φαίνεται στο παρακάτω.

```
<?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" >

    <com.google.android.maps.MapView
        android:id="@+id/mapView"
        android:enabled="true"
        android:clickable="true"

        android:layout_width="fill_parent"
        android:layout_height="fill_parent"

        android:apiKey="0fh_I1kIJVRIwQQfuo5wwe64B3ashsNOPEevA"
     />

</RelativeLayout>
```

Όπως φαίνεται χρησιμοποιούμε ένα RelativeLayout (η τοποθεσία των περιεχόμενων μπορεί να περιγραφεί με σχέση μεταξύ τους) περιέχει μέσα στο "body" του μόνο ένα στοιχείο, το com.google.android.maps.MapView το οποίο μας παρέχεται από την βιβλιοθήκη της Google το οποίο λειτουργεί όπως ακριβώς το γνωστό Google Map στο Internet. Το MapView παίρνει τα δεδομένα από το Google Map Service.Πέρα από τα γνωστά attributes έχει ένα που δεν το έχουμε ξαναδεί, το apiKey.

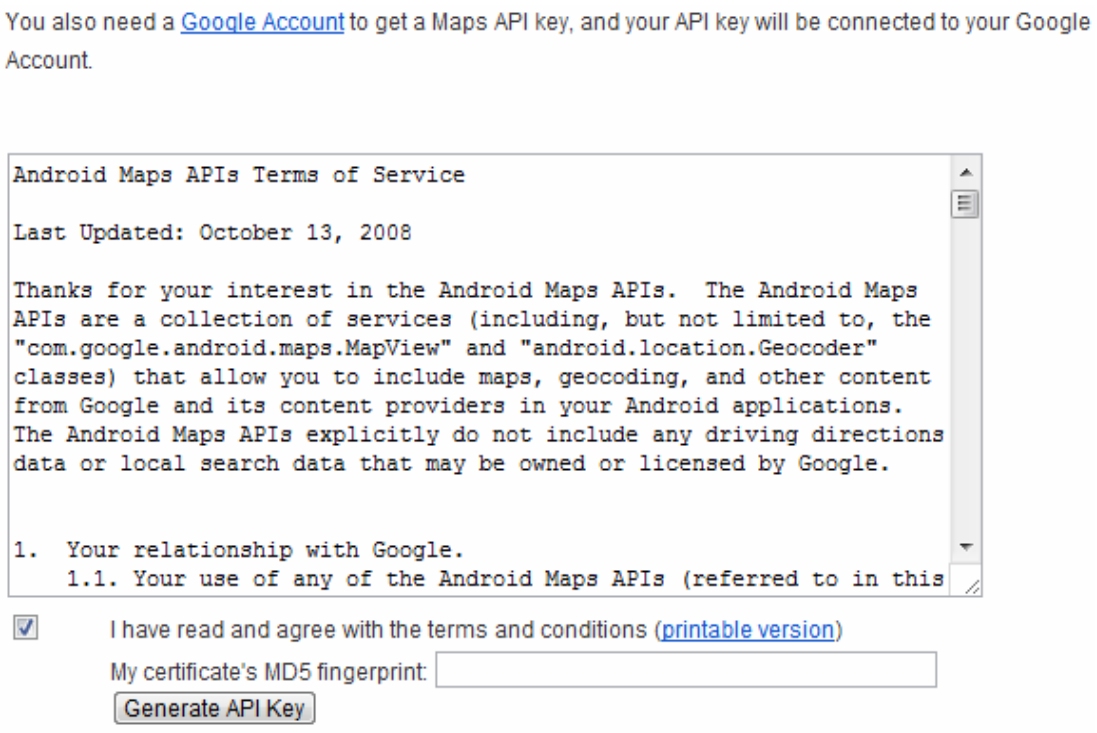
Το apiKey επιτρέπει στην εφαρμογή μας να εγγραφεί στο Server και να λάβει δεδομένα. Για να πάρουμε αυτό το κλειδί πρέπει να έχουμε εγκατεστημένο το Java SE 6 JDK. Αφού κατεβάσουμε και εγκαταστήσουμε από την σελίδα της Oracle, ανοίγουμε το Command Prompt των Windows και μπαίνουμε στο αρχεία bin του jdk1.6 και θα χρησιμοποιήσουμε τις παρακάτω εντολές

keytool –list –alias androiddebugkey –keystore "το path για το debug.keystore του android" –storepass android –keypass android

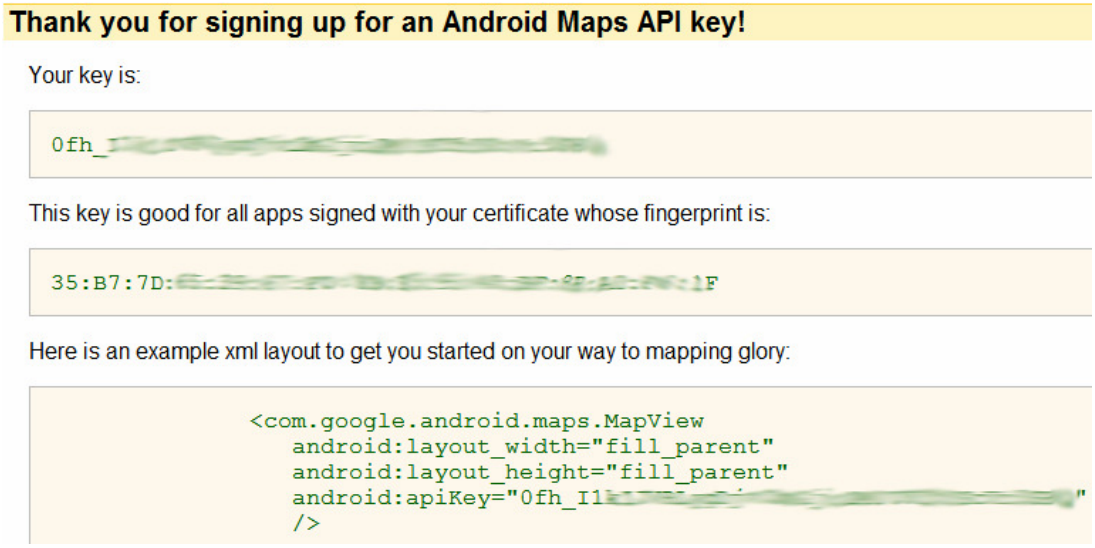


Εικόνα 7 – 9: παίρνοντας το fingerprint σε MD5

Αυτό μας επιστρέφει ένα "fingerprint" σε MD5 κρυπτογράφηση το οποίο το χρησιμοποιούμε στην σελίδα developers.google.com/android/maps-api-signup για να πάρουμε το κλειδί. Αυτό το κλειδί είναι συνδεδεμένο με το Google Account μας.



Εικόνα 7 – 10: Ζητώντας το API Key



Εικόνα 7 – 11: Τα τελικα στοιχεία και παράδιγμα για το MapView

Για να φτιάξουμε ένα Google Map πρέπει πρώτα να επεκτείνουμε την MapActivity και να υλοποιήσουμε την LocationListener. Το MapActivity διαχειρίζεται τον χάρτη μας (MapView) και το LocationListener για να ακούμε τις αλλαγές της τοποθεσίας μας.

```
public class MyFirstGoogleMap extends MapActivity implements LocationListener
```

Οι μεταβλητές που χρησιμοποιούμε είναι οι ακόλουθες:

- `MapController controller;` Διαχειρίζεται την μετατόπιση και το zoom του χάρτη.
- `LocationManager manager;` Πρόσβαση στο Location Services μας επιτρέπει να λάβουμε τις γεωγραφικές συντεταγμένες.
- `GeoPoint myGeoPoint;` Αντικείμενο για την εκπροσώπηση των γεωγραφικών συντεταγμένων.
- `Geocoder myGeocoder;` Για την μετατροπή των γεωγραφικών συντεταγμένων και αντίστροφα.
- `List<Overlay> listOfOverlays;` Μια λίστα που περιέχει αντικείμενα Overlay τα οποία θα εμφανιστούν πάνω στον χάρτη.
- `PinsOverlay pinsOverlay, mePin;` Μια κλάση που φτιάξαμε για να βάζουμε pins μέσα στο `listOfOverlays`.
- AccuracyOverlay `accuracyOverlay;` Ένας κύκλους που μπαίνει κάτω από το mePin για να δήλωση το accuracy της τοποθεσίας.

- `Boolean locationFlag;` Δηλώνει αν έχει βρεθεί η τοποθεσία του χρήστη.
- `Boolean locationUpdating;` Δηλώνει αν τρέχει το onLocationChange αυτήν την στιγμή.
- `Boolean rangeFlag;` Ένα flag για το myGeocoder, αν είναι true θα ψάξουμε να βρούμε τις γεωγραφικές συντεταγμένες μέσα σε κάποιο γεωγραφικό όριο.
- `String accuracy;` Περιέχει την εμβέλεια της πιθανής τοποθεσίας του χρήστη σε μέτρα.
- `Criteria criteria;` Για να μας δώσει τον απαραίτητο Provider.
- `String provider;` Κρατά τον Provider από τον criteria.
- `MapView myMap;` Το View που αντιπροσωπεύει τον χάρτη.

Στο `onCreate` δηλώνουμε το `ContentView` του Activity, παίρνουμε το `MapView`, τον Controller και τον `LocationManager` ως πρώτα βήματα.

```
@Override
protected void onCreate(Bundle icicle) {
	super.onCreate(icicle);
	setContentView(R.layout.map_view);

	locationFlag = false;
	locationUpdating = false;

	ConnectivityManager connMgr = (ConnectivityManager)
				getSystemService(Context.CONNECTIVITY_SERVICE);
	NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

	if (networkInfo != null && networkInfo.isConnected()) {

		myMap = (MapView) findViewById(R.id.mapView);
		myMap.setBuiltInZoomControls(true);
```

Το MapView μας παρέχει μια συγχρονισμένη λίστα (Collections.synchonizedList (java.util.List)) στην οποία ότι Overlay αντικείμενα περιέχει θα εμφανιστούν στον χάρτη μας.

```
		listOfOverlays = myMap.getOverlays();
```

Παίρνουμε τον controller από τον χάρτη και ρυθμίζουμε το ζουμ.

```
		controller = myMap.getController();
		controller.setZoom(15);
```

Στην συνέχεια ελέγχουμε αν τουλάχιστον ένας από τους δύο Providers (GPS, NETWORK) είναι ενεργοποιημένος, αν όχι θα δημιουργήσει ένα Implicit Intent για να εμφανίσει το "Location and security" από τα Settings της συσκευής.

```
manager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

if( !(manager.isProviderEnabled(android.location.LocationManager.GPS_PROVIDER)||
manager.isProviderEnabled(android.location.LocationManager.NETWORK_PROVIDER))){
      Intent myIntent = new Intent( Settings.ACTION_SECURITY_SETTINGS );
      startActivity(myIntent);
}
```

Αφού ενεργοποιηθεί τουλάχιστον έναν Provider δηλώνουμε το"criteria" μας για να πάρουμε τον καλύτερο δυνατό provider . Αμέσως μετά προσπαθούμε να πάρουμε την τελευταία γνωστή τοποθεσία από τον `LocationManager`, αν υπάρχει θα καλέσει το `onLactionChanged` για να ανανεώσει την θέση μας, αν όχι θα μας εμφανίσει ένα μήνυμα.

```
      Criteria criteria = new Criteria();
      criteria.setAltitudeRequired(false);
      criteria.setCostAllowed(false);
      criteria.setAccuracy(Criteria.ACCURACY_COARSE);
      provider = manager.getBestProvider(criteria, true);

      Location location = manager.getLastKnownLocation(provider);

      if (location != null) {
          onLocationChanged(location);
      } else {
            Toast.makeText(getApplicationContext(),
                        "No last known locations has been found",
                                          Toast.LENGTH_LONG).show();
      }
} else {
      Toast.makeText(getApplicationContext(),
            "Your Device is not connected to the internet",
                                    Toast.LENGTH_LONG).show();
      finish();
}
```

Τέλος κατασκευάζουμε το myGeocoder χρησιμοποιώντας το `getBaseContext` που επιστρέφει το Context του MapActivity της google και το Locale που αντιπροσωπεύει την γλώσσα της χώρας που βρισκόμαστε δίνοντας το ISO-code που χρειαζόμαστε για να κάνουμε τα ερωτήματα (στη περίπτωση της Ελλάδος χρησιμοποιείτε το ISO 639-1 για Αγγλικά). Έπεισες περνούμε την database και το SharedPreferences που θα χρειαστούμε να βάλουμε τα δεδομένα στο location του ασθενή στην βάση.

```
        myGeocoder = new Geocoder(getBaseContext(), Locale.getDefault());
        db = new PatientDBAdapter(this);
        mySharedPreferences = getSharedPreferences("com.chkyriacoy.projectdoc",
                                                        Context.MODE_PRIVATE);
}
```

Στο onStart ενημερώνουμε τον χρηστό για τον provider που χρησιμοποίει.

```
        @Override
        protected void onStart() {
                super.onStart();
                Toast.makeText(getApplicationContext(),
                        "You use "+provider+" provider", Toast.LENGTH_LONG).show();
                db.open();
        }
```

Αμέσως μετά καλείτε το `onResume` που θα καλέσει την `requestLactionUpdates` για ανανέωση κάθε οκτώ δευτερόλεπτα ή δέκα μέτρα απόστασης από την παλιά θέση καθώς θα μας πει και ποιον Provider χρησιμοποιούμε.

```
@Override
protected void onResume() {
        super.onResume();
        manager.requestLocationUpdates(provider, 8000, 50f, this);
        locationUpdating = true;
}
```

Ότι "ανοίγουμε" στο `onResume` πρέπει να το "κλείνουμε" στο `onPause`, έτσι κάνουμε `removeUpdates` για να σταματήσουμε τις ανανεώσεις.

```
        @Override
        protected void onPause() {
                // TODO Auto-generated method stub
                super.onPause();
                manager.removeUpdates(this);
                locationUpdating = false;
        }
```

Στο onStop καταχωρούμε την το Latitude και Longitude καθώς και το accuracy πριν κλείσουμε την database.

```
        @Override
        protected void onStop() {
                super.onStop();

                if(locationFlag){
                String lastKnownLocation =
                        String.valueOf(myGeoPoint.getLatitudeE6())+
                        "@"+String.valueOf(myGeoPoint.getLongitudeE6()+
                                                "#"+accuracy);
```

```
                long id = mySharedPreferences.getLong(
                            PatientDBAdapter.KEY_PATIENT_ID, -1L);

                if(id!=-1L){
                        POJOPatient patient = db.getPatient(id);
                        patient.setLastKnownLocation(lastKnownLocation);
                        db.updatePatient(patient);
                }
        }

        db.close();
    }
```

Το `onLocationChanged` καλείτε κάθε φορά που ανανεώνετε η θέση μας επιστρέφοντας μας ένα αντικείμενο Location που αρχικά το περνάμε στο myGeoPoint, αφού μετατρέψουμε τις double γεωγραφικές συντεταγμένες σε int πολλαπλασιάζοντας το με το 1E6 (10^6). Στην συνέχεα περνάμε το myGeoPoint και στον controller μας για να μας κεντράρει τον χάρτη. Για ασφάλεια χρησιμοποιούμε το locationUpdating ώστε σε περίπτωση που ξαναγίνει update πριν τελείωση το τρέχον να μην "κρασάρει" η εφαρμογή, στο τέλος θα περάσει ξαναγίνει true.

```
public void onLocationChanged(Location location) {
    if(locationUpdating){
        locationUpdating = false;
        myGeoPoint = new GeoPoint((int)(location.getLatitude()*1E6),
                                  (int)(location.getLongitude()*1E6));
        controller.setCenter(myGeoPoint);
```

Σε αυτό το σημείο θέλουμε να ανανεώσουμε το pin και το accuracyOverlay (το σχέδιο που μας δείχνει πάνω στον χάρτη). Το listOfOverlays περιέχει όλα τα pins που θα χρησιμοποιήσουμε και στην θέση ένα έχουμε το δικό μας pin το mePin, ενώ στην θέση μηδέν βάζουμε το accuracyOverlay. Πρώτα κοιτάζουμε αν το mePin είναι null αν δεν είναι σημαίνει πως πρέπει να αφαιρεθεί προτού προσθέσουμε στο listOfOverlays.

```
    if(mePin!=null){
        listOfOverlays.remove(0);
        listOfOverlays.remove(1);
    }else{
        mePin = new PinsOverlay(getResources()
                        .getDrawable(R.drawable.me));
    }
```

Περνούμε το accuracy που μας δίνεται πάντα από το location (αν δεν υπαρχή accuracy τότε το accuracy είναι μηδέν). Το PinsOverlay επεκτείνει ItemizedOverlay και

περιέχει OverlayItem, για αυτό πρώτα καλούμε τον κατασκευαστή της PinsOverlay με παραμέτρους ένα Drawable και μετά εισάγουμε ένα OverlayItem με παραμέτρους ένα GeoPoint, ένα String τίτλο και ένα String κείμενο με τη accuracy ώστε να εμφανίζεται όταν πατάμε επάνω του. Το AccuracyOverlay επεκτείνει την Overlay και περνάμε απευθείας τις μεταβλητές στον κατασκέπαστη. Τέλος τα κάνουμε add στην θέση μηδέν και ένα και ανανεώνουμε τον χάρτη μας και κάνουμε τα flag μας true. Το locationFlag για αν δηλώσουμε πως έχουμε την θέση του χρήστη, και το locationUpdating ώστε να μπορεί να ξανά τρέξε ο κώδικας μέσα στην if.

```
accuracy = String.valueOf(location.getAccuracy());
mePin.insertPin(new OverlayItem(myGeoPoint, "Me",
             "You are here"+"\n Accuracy : "+accuracy+" meters"));
accuracyOverlay = new AccuracyOverlay(myGeoPoint, location.getAccuracy());

listOfOverlays.add(0, accuracyOverlay);
listOfOverlays.add(1, mePin);

myMap.invalidate();

locationFlag = true;

locationUpdating = true;
```

Η εσωτερική κλάση PinsOverlay επεκτείνει την `ItemizedOverlay` που παίρνει αντικείμενα της κλάσης `OverlayItem`, έχει σκοπό να μας φτιάξει όλα τα Pins που θα βάλουμε μέσα στον χάρτη. Περιέχει μια `ArrayList` με αντικείμενα της `OverlayItem`.

```
class PinsOverlay extends ItemizedOverlay<OverlayItem>{
      private ArrayList<OverlayItem> pins;
```

Στο κατασκευαστή περνάμε το `Drawable` μέσα στην `super` (δηλαδή μέσα στον κατασκευαστή της `ItemzedOverlay`). Παρατηρούμε πως έχουμε καλέσει την μέθοδο `boundCenterBottom` μέσα στην `super`. Αυτή η μέθοδος θα μας στοιχίσει το Drawable ώστε να είναι κεντραρισμένη και πάνω από το σημείο που θέλουμε.

```
public PinsOverlay(Drawable defaultMarker) {
      super(boundCenterBottom(defaultMarker));
            this.pins = new ArrayList<OverlayItem>();
      }
```

Για να πάρουμε και να πάρει το σύστημα τα αντικείμενα από την `ArrayList` κάνουμε Override την μέθοδο `createItem`.

```
@Override
protected OverlayItem createItem(int i) {
	return pins.get(i);
}
```

Προσθέτουμε τα δικά μας OverlayItem στην λίστα μας.

```
public void insertPin(OverlayItem pin){
	pins.add(pin);
	this.populate();
}
```

Όπως είπαμε το σύστημα διαβάζει τα αντικείμενα από την ArrayList μας όταν καλούμε την populate(), για αυτό πρέπει να ξέρει πόσα αντικείμενα περιέχει μέσα, για αυτό επιστρέφει το μέγεθος της pins.

```
@Override
public int size() {
	return pins.size();
}
```

Η `onTap` καλείτε κάθε φορά που πατάμε σε κάποιο pin και μας δίνει την διεύθυνση που βρίσκεται στο pins μας. Παίρνοντας την θέση αυτή καλούμε την `getSnippet` που μας δίνει το κείμενο που βάλαμε, δηλαδή αν πατήσουμε στο pin αυτό θα μας εμφανίσει το "you are here" στο Toast.

```
@Override
protected boolean onTap(int index) {
	Toast.makeText(getApplicationContext(),
				pins.get(index).getSnippet(),
				Toast.LENGTH_LONG).show();
		return super.onTap(index) ;
	}
}
```

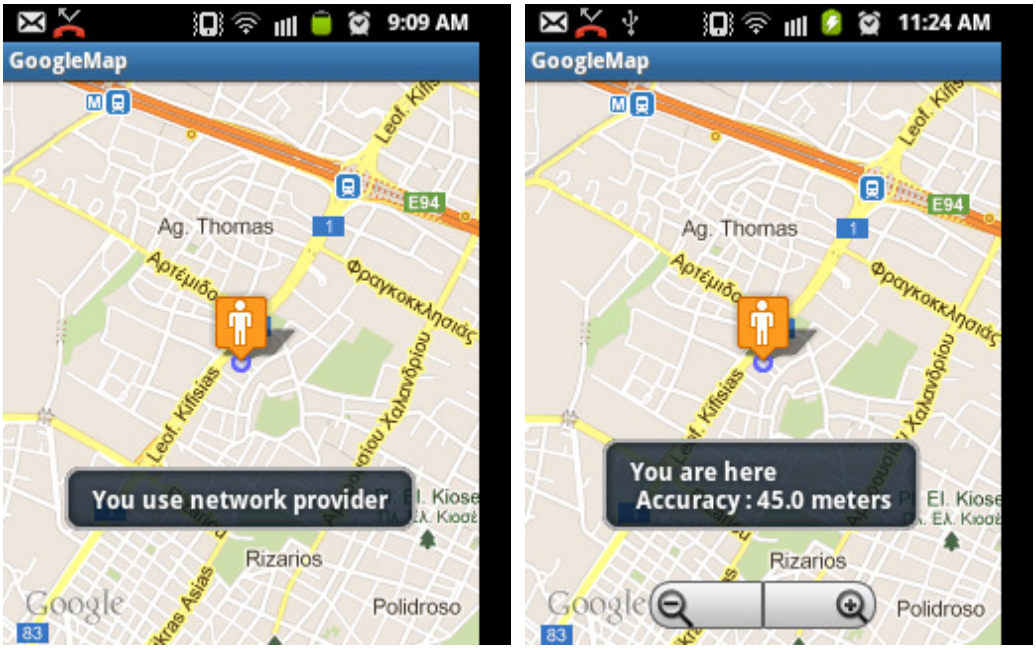
Για να εμφανίσουμε το Accuracy σαν ένα κύκλο κάτω από το pin του χρήστη πρέπει να κατασκευάσουμε μια κλάση που πηκτίνη την Overlay με δυο μεταβλητές. Ένα GeoPoint για την θέση του σημείου σε γεωγραφικές συντεταγμένες ένα float για τον αριθμό του accuracy σε μετρά.

```
class AccuracyOverlay extends Overlay {
```

```
	private GeoPoint sourcePoint;
	private float accuracy;

	public AccuracyOverlay(GeoPoint geoPoint, float accuracy) {
		super();
		sourcePoint = geoPoint;
		this.accuracy = accuracy;
	}

	public void setSource(GeoPoint geoPoint, float accuracy) {
		sourcePoint = geoPoint;
		this.accuracy = accuracy;
	}
```

Στο draw είναι η μέθοδος που θα χρησιμοποιηθεί για να ζωγραφίσει τους δυο ομοκέντρους κύκλους. Ο πρώτος είναι ένα "δαχτυλίδι" ενώ ο δεύτερος είναι το γέμισμα. Για να ζωγραφίσουμε πάνω στο mapView χρειαζόμαστε ένα Projection. Χρησιμοποιώντας το projection.metersToEquatorPixels με το accuracy

```
	@Override
	public void draw(Canvas canvas, MapView mapView, boolean shadow) {
		super.draw(canvas, mapView, false);
		Projection projection = mapView.getProjection();
		Point center = new Point();
		int radius = (int) (projection.metersToEquatorPixels(accuracy));
		projection.toPixels(sourcePoint, center);
		Paint accuracyPaint = new Paint();

		accuracyPaint.setAntiAlias(true);
		accuracyPaint.setStrokeWidth(2.0f);
		accuracyPaint.setColor(0xff6666ff);
		accuracyPaint.setStyle(Style.STROKE);
		canvas.drawCircle(center.x, center.y, radius, accuracyPaint);

		accuracyPaint.setColor(0x186666ff);
		accuracyPaint.setStyle(Style.FILL);
		canvas.drawCircle(center.x, center.y, radius, accuracyPaint);
	}
}
```

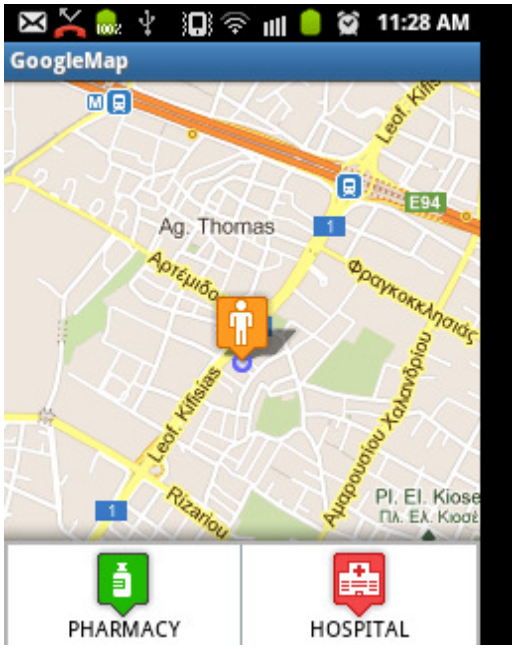


Εικόνα 7 – 12: GoogleMap με την τοποθεσία του χρήστη

Ο χρήστης έχει την δυνατότητα να βρίσκει φαρμακεία και νοσοκομεία πατώντας το κουμπί Menu της συσκευής του. Αφού πάρουμε το id από το item που πάτησε ο χρήστης, η ρουτίνα που ακολουθείτε είναι παρόμοια. Αν επέλεγε το Pharmacy παίρνουμε μια λίστα από τα φαρμακεία βάση της τοποθεσίας μας και μετά παίρναμε την λίστα αυτήν μέσα στο μέθοδο addressesToPoints μαζί με το `Drawable` που θέλουμε και το rangeFlag. Η ίδια ρουτίνα ακολουθείτε και στην επιλογή Hospitals. Αυτή η ρουτίνα γίνεται μέσα σε ένα AsyncTask Πριν ξεκινήσουμε το AsyncTask ελέγχουμε αν το locationFlag είναι true, ώστε να είμαστε σίγουροι πως έχει βρεθεί η τοποθεσία του χρήστη, διαφορετικά εμφανίζει ένα μήνυμα.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
	super.onCreateOptionsMenu(menu);
	MenuInflater menuInflater = getMenuInflater();
	menuInflater.inflate(R.menu.map_menu, menu);
	return true;
}


@Override
public boolean onOptionsItemSelected(MenuItem item) {
	switch (item.getItemId()){
	case R.id.menu_pharmacy:

	 if (locationFlag) {
		new LoadingTaskAddress().execute("drogstores");
	} else {
	Toast.makeText(getApplicationContext(),
				"You'r location is not found",
				Toast.LENGTH_LONG).show();
```

```
            }
            return true;

            case R.id.menu_hospital:

            if (locationFlag) {
            new LoadingTaskAddress().execute("hospitals");
            } else {
            Toast.makeText(getApplicationContext(),
                        "You'r location is not found",
                        Toast.LENGTH_LONG).show();
            }
            return true;
            }

      return false;
      }
```



Εικόνα 7 – 13: GoogleMap Menu

Το AsyncTask LoadingTaskAddress έχει σκοπό να βρει και να φτιάξει μια λίστα από Strings και περνώντας σαν παράμετρο ένα String για το τη ψάχνουμε. Με την χρήση μιας Boolean ελέγχουμε αν η παράμετρος είναι hospitals για το result.

```
private class LoadingTaskAddress extends AsyncTask<String, Void, Boolean>{

      @Override
      protected void onPreExecute() {
            super.onPreExecute();
            loadingDialog = new ProgressDialog(MyFirstGoogleMap.this);
            loadingDialog.show();
            loadingDialog.setMessage("Searching, Please Wait . . .");
      }
```

```
@Override
protected Boolean doInBackground(String... params) {

        DefaultHttpClient client = new DefaultHttpClient();
        HttpGet request = null;
        HttpResponse response = null;
        InputStream in = null;
        BufferedReader reader = null;
        String url = "";
        String line = "";
        int endPointer = 0;
        myAddresses = new ArrayList<String> ();
        List<String> lines = new ArrayList<String> ();

        Boolean hospitalsCase = (params[0].equals("hospitals"));
```

Για να πάρουμε την διεύθυνση που βρισκόμαστε χρησιμοποιούμε το `myGeocoder` που θα μας επιστρέφει μια λίστα από διευθύνσεις που έχουν σχέση με τις γεωγραφικές συντεταγμένες. Το "1" στο τέλος της μεθόδου δηλώνει πως θέλουμε μόνο μια διεύθυνση. Η μέθοδος `getFromLaction` μπορεί να προκαλέσει IOException για αυτό πρέπει να το περικλείσουμε με try catch.

```
try {
List<Address> myAddressList = myGeocoder.getFromLocation(
                                    (double)(myGeoPoint.getLatitudeE6()/1E6),
                                    (double)(myGeoPoint.getLongitudeE6()/1E6),
                                                                        1);
```

Για να βρούμε τα φαρμακεία που βρίσκονται στην περιοχή μας κάνουμε αίτηση στο Χρυσό Οδηγό. www.xo.gr. Για να κάνουμε μια αίτηση GET θα χρησιμοποιήσουμε το url "http://www.xo.gr/search/?what=pharmacy&where= την διεύθυνση του χρήστη &lang=en". Για αυτήν την αίτηση χρειαζόμαστε ένα HttpClient που διαχειρίζεται τα cookies, πιστοποίηση και την σύνδεση. Για την αποστολή της αίτησης GET και για πάρουμε το αποτέλεσμα χρησιμοποιούμε το HttpGet και HttpResponse αντίστοιχα.

Παίρνοντας το πρώτο στοιχείο από την λίστα (myAddressList.get(0)) παίρνουμε την πόλη (getLocality()) που θα προσθέσουμε στο url μας. Στην συνέχεια εξετάζουμε αν υπάρχει ο Ταχυδρομικός Κώδικας για την περιοχή που βρισκόμαστε. Αν ναι τον προσθέτουμε στο url και κάνουμε το rangeFlag true. Αν όχι απλώς θα θέσουμε το rangeFlag false. Τέλος προσθέτουμε το &lang=en για αγγλικά.

```
if(myAddressList.size()>0){
```

```
url+="http://www.xo.gr/search/?what="+params[0]+"&where=";
url+=myAddressList.get(0).getLocality();

if(myAddressList.get(0).getPostalCode()!=null){
	url+="%20"+myAddressList.get(0).getPostalCode()+"&lang=en";
	rangeFlag = true;
}else{
	url+="&lang=en";
	rangeFlag=false;
}
```

Αφού πάρουμε το url περνούμε το Http για Get με το url μας και το εκτελούμε μέσω της client.execute(request) παίρνοντας το αποτέλεσμα της αίτησης. Αμέσως μετά παίρνουμε την οντότητα που θα μας δώσει το InputStream (response.getEntity().getContent()). Και τέλος περνούμε το Stream στο BufferedReader.

```
request = new HttpGet(url);
response = client.execute(request);
in = response.getEntity().getContent();
reader = new BufferedReader(new InputStreamReader(in),16384);
```

Αμέσως μετά περνάμε κάθε γραμμή σε μια άλλη λίστα την lines για να την χρησιμοποιήσουμε εκτός try catch. Ο λόγος είναι πως ότι γίνεται μέσα στο try catch έχει παραπάνω επιβάρυνση και το σύστημα της συσκευής μπορεί να μην το αντέξει.

```
while((line=reader.readLine())!= null){
		lines.add(line);
	}
}
} catch (IOException e) {
	e.printStackTrace();
	this.cancel(true);
}
```

Αφού τελειώσουμε με το try catch θέλουμε να περάσουμε τα Addresses από τα lines. Για να γίνει αυτό πρέπει πρώτα να τα βρούμε. Αν διαβάσουμε το source που μας έδωσε η σελίδα του Χρυσού οδηγού, θα παρατηρήσουμε πως το κομμάτι που ψάχνουμε έχει την δομή παρατηρούμε πως όλες οι διευθύνσεις τελειώνουν σε “,</”. Αυτά τα σύμβολα θα χρησιμοποιηθούν για να βρούμε αν η γραμμή αυτή περιέχει κάποια διεύθυνση. Αν όχι, προχωράμε στην επόμενη γραμμή. Αν ναι θα πάρουμε σαράντα χαρακτήρες και θα “κόψουμε” το κομμάτι που θέλουμε. Το “nowrap” συμβολίζουν την αρχή της διευθύνσεως. Αυτή η ιστορία μας αφήνει, με μόνο το κομμάτι που θέλουμε στο partOfLine που το περνάμε στο myAddress. Δηλαδή από: <p>Moraïti 2 Athina, σε Moraïti 2 Athina

```
        String partOfLine = "";
        for(String l : lines){
                endPointer=l.lastIndexOf(",</");
                if(endPointer!=-1){
                partOfLine = l.substring(endPointer-40, endPointer);
                partOfLine = partOfLine.substring(
                                partOfLine.lastIndexOf("nowrap")+8);
                myAddresses.add(partOfLine);
                }
        }
        return hospitalsCase;
}
```

Μετά την doInBackground καλείτε η onPostExecute στην όποια περνά το Boolean hospitalsCase για να δούμε τη Drawable θα δώσουμε στην μέθοδο addressToPoints.

Η μέθοδος addressToPoints έχει σκοπό να πάρουμε τα αποτελέσματα της από τα Addresses που βρήκαμε και να τα περάσει μέσα στο listOfOverlays, ένα Drawable με αυτό που θα βαλουμε για pins και το range για το αν εχει βρεθη το Postal Code.

```
        protected void addressesToPoints(List<String> stringAddresses,
                                                Drawable draw ,boolean range ){
```

Για να μην μας βγάλει σφάλμα παίρνουμε το Geopoint της δεδομένης στιγμής μήπως και αλλάξει κατά την διάρκεια που τρέχει η μέθοδος.

```
        GeoPoint myGeoPoint = this.myGeoPoint;
```

Πρώτο πράγμα που κάνουμε είναι καθαρίσουμε την λίστα με τα Overlays. Αμέσως μετά προσθέτουμε ξανά το mePin και το accuracyOverlay.

```
                //clear the old Pins
                //and re-enter the accuracy and user's Pin
                listOfOverlays.clear();
                listOfOverlays.add(0, accuracyOverlay);
                listOfOverlays.add(1, mePin);

                pinsOverlay = new PinsOverlay(draw);
```

Ελέγχουμε αν η λίστα μας είναι άδεια και αν ναι ειδοποιούμε τον χρήστη με ένα Toast.

```
                if(stringAddresses.size()==0){
                Toast.makeText(this,
                                "No Address have been found in your location",
                                Toast.LENGTH_LONG).show();
                }else{
```

Στην περίπτωση που το range είναι true θέλουμε να πάρουμε τέσσερα σημεία που θα εξετάσουμε αν τα αποτελέσματα του Geocoder βρίσκονται εντός αυτών των γεωγραφικών συντεταγμένων.

```
try {
	if(range){
		final int DISTANCE = (int)(00.27*1E6);
		double lowerLeftLatitude = (myGeoPoint.getLatitudeE6()
								-DISTANCE)/1E6;
		double lowerLeftLongitude = (myGeoPoint.getLongitudeE6()
								-DISTANCE)/1E6;
		double upperRightLatitude = (myGeoPoint.getLatitudeE6()
								+DISTANCE)/1E6;
		double upperRightLongitude = (myGeoPoint.getLongitudeE6()
								+DISTANCE)/1E6;
```

Για κάθε διεύθυνση που του δίνουμε στο Geocoder θα μας επιστρέψει μια λίστα με τις διευθύνσεις που βρίσκονται. Ποιο συγκεκριμένα αν βρει το Address από τις διευθύνσεις που του δίνουμε να μας επιστρέψει μια List<Address> με μόνο "1" Address μέσα. Το Address αυτό πρέπει να είναι εντός των σημείων που του θέσαμε, διαφορετικά θα επιστρέψει μια άδεια λίστα.

```
	for (int i=0;i<stringAddresses.size();i++){
		List<Address> myAddressList =
			myGeocoder.getFromLocationName(stringAddresses.get(i), 1,
					lowerLeftLatitude, lowerLeftLongitude,
					upperRightLatitude, upperRightLongitude);
```

Αν η λίστα μας έχει κάποιο Address παίρνουμε το Latitude, το Longitude και την διεύθυνση του και φτιάχνουμε ένα OverlayItem και το βάζουμε μέσα στο pinsOverlay.

```
	if(myAddressList.size()>0){
		OverlayItem overlayItem = new OverlayItem(
			new GeoPoint((int)(myAddressList.get(0).getLatitude() *1E6),
			(int)(myAddressList.get(0).getLongitude()*1E6)),
								"", stringAddresses.get(i) );

		pinsOverlay.insertPin(overlayItem);
		}
	}
	}else{
```

Το else αναφέρεται αν τελικά δεν έχουμε βρει τον Ταχυδρομικό Κώδικα που θα μας επιτρέπει να κάνουμε μια πιο ακριβή αίτηση στον Χρυσό Οδηγό. Με άλλα λόγια ο Χρυσός Οδηγός θα μας δώσει την πρώτη σελίδα για φαρμακεία σε όλη την Αθήνα αντί για την

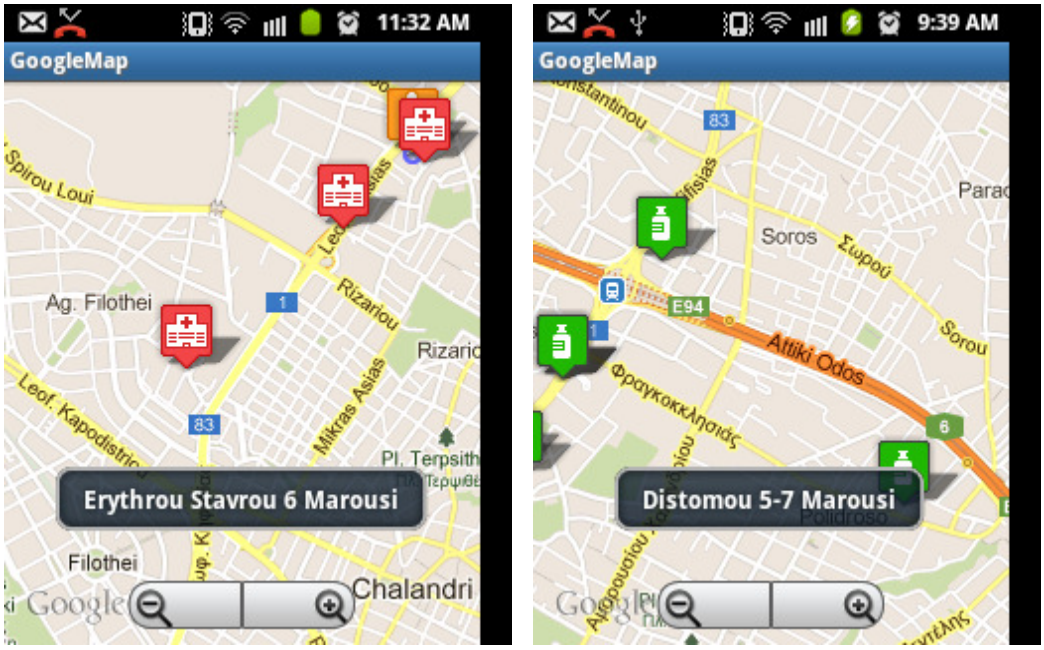
περιοχή που βρισκόμαστε. Ο λόγος που εφαρμόζουμε αυτή την τεχνική είναι ότι δεν είναι σίγουρο πως θα μας δοθεί ο Ταχυδρομικός Κώδικας.

```
for (int i=0;i<stringAddresses.size();i++){

List<Address> myAddressList = myGeocoder.getFromLocationName(
                                        stringAddresses.get(i), 1);

if(myAddressList.size()>0){
      OverlayItem overlayItem = new OverlayItem( new GeoPoint(
                  (int)(myAddressList.get(0).getLatitude() *1E6),

                  (int)(myAddressList.get(0).getLongitude()*1E6)),
                                        "", stringAddresses.get(i));
pinsOverlay.insertPin(overlayItem);
      }
}
}
```

Και τα δύο καταλήγουν στο pinsOverlay να γίνεται add στο listOfOverlays και να ανανεώνετε ο χάρτης.

```
listOfOverlays.add(pinsOverlay);
myMap.invalidate();

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

Δυστυχώς δεν είναι δυνατόν να κάνουμε πολλές αιτήσεις στον Χρυσό Οδηγό λόγω ότι θα μας βάλει σε block list για κάποιο χρονικό διάστημα.



Εικόνα 7 – 14: Φαρμακεία και νοσοκομεία στον χάρτη

7.12 BluetoothScreen

Το Activity BluetoothScreen είναι υπεύθυνο για να παρουσιάσει όλες τις συσκευές που έχουν γίνει paired με την συσκευή του χρήστη. Το Activity ζητά από την χρήστη να κάνει paired την συσκευή που επιθυμεί να συνδεθεί δίνοντας του οδηγίες μέσω μηνυμάτων. Ο λόγος είναι επειδή η εφαρμογή δεν μπορεί να κάνει pair συσκευές χωρίς τη βοήθεια του χρήστη.

Πατώντας πάνω στο αντικείμενο της λίστας παίρνουμε το BluetoothDevice το οποίο μπορούμε να το χρησιμοποιήσουμε για να το συνδέσουμε με το BluetoothHPDService. Αυτό όμως είναι αδύνατον για συσκευές με API Level χαμηλότερου του 11 λόγο συμβατότητας. Αντί αυτού για να δήξουμε πως πήραμε την συσκευή που επιθυμούμε να συνδεθούμε εμφανίζουμε το όνομα της.

Πρώτη του ενέργεια είναι να βρει αν υπάρχουν ήδη συσκευές που έχουν γίνει paired για να ενημερώσει το ViewList. Η διαδικασία ενημέρωσης είναι η ίδια όπως έχουμε δει σε προηγούμενη ανάλυση, όπου φτιάχνουμε μια εσωτερική κλάση που επεκτείνει την ArrayAdapter<BluetoothDevice> με ένα ViewHolder.

Εάν δεν έχουμε καμιά συσκευή pair ή ακόμα αν το Bluetooth είναι κλειστό δίνουμε την δυνατότητα μέσα από ένα Button να μπούμε στο Bluetooth Settings της συσκευής όπου ο χρήστης μπορεί να ανοίξει, αναζητήσει και να κάνει pair τις συσκευές.

Κατά το ξεκίνημα δηλαδή στο `onStart` ελέγχεται αν η συσκευή του χρήστη υποστηρίζει Bluetooth. Αν όχι θα τερματίζει το Activity.

```
    @Override
    protected void onStart() {
    super.onStart();
    bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    if (bluetoothAdapter == null) {
      Toast.makeText(this,
          "Bluetooth is not supported", Toast.LENGTH_LONG);

    finish();
    return;
   }
}
```

Εφόσον είναι όλα εντάξει ζητάμε από το bluetoothAdapter να μας επιστρέψει όλες της συσκευές που είναι paired. Αφού έχουμε καθαρίσει την λίστα από ότι παλιά paired έχει παίρνει ένα ένα τα καινούρια pairs τα οποία θα εμφανιστούν μέσα στην λίστα ListView.

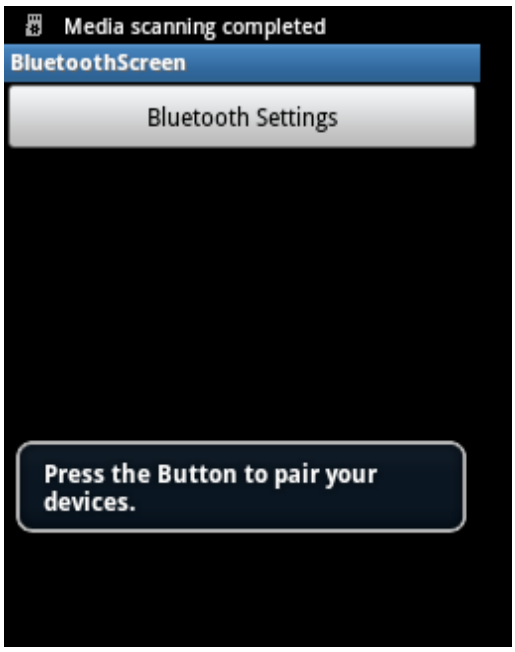
```
    @Override
    protected void onResume() {
        super.onResume();

        if(!arrayOfDevices.isEmpty()){
            arrayOfDevices.clear();
        }

        Set<BluetoothDevice> pairedDevices =
                            bluetoothAdapter.getBondedDevices();

        for (BluetoothDevice pairedDevice : pairedDevices){
            arrayOfDevices.add(pairedDevice);
        }

        devicesAdapter.notifyDataSetChanged();
    }
```

Ο χρήστης μπορεί να πατήσει πάνω στα αντικείμενα της λίστας ListView για να πάρει μια αναφορά στο BluetoothDevice που επιθυμεί. Για να γίνει αυτό το Activity χρησιμοποιεί το onItemClick το οποίο θα μας δώσει ένα μήνυμα με το όνομα της συσκευής που επιλέκτικε να κρατήσει μια αναφορά σε αυτό.

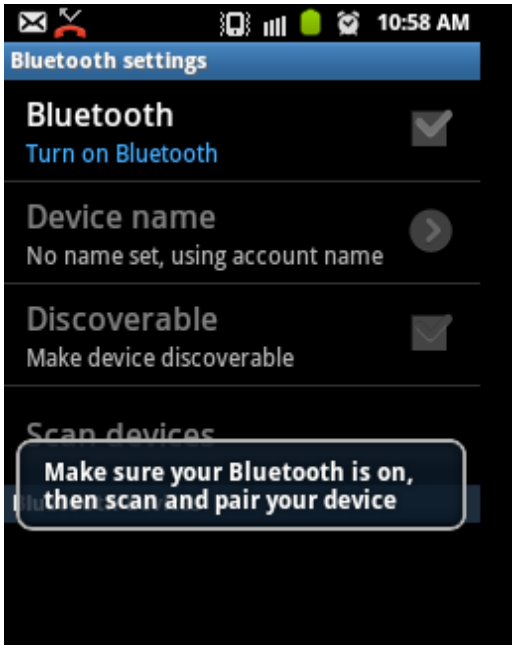
```
listView.setOnItemClickListener(new OnItemClickListener() {
	@Override
	public void onItemClick(AdapterView<?> parent, View view,
						int position, long id) {

	wantedPairedDevice = arrayOfDevices.get(position);
		Toast.makeText(getApplicationContext(),
				wantedPairedDevice.getName(),
					Toast.LENGTH_LONG).show();
		}
	});
```

Τέλος ο χρήστης μπορεί να πάει στο Bluetooth Settings της συσκευής του για να ενεργοποιήσει το Bluetooth ή και να κάνει περισσότερες συσκευές paired.

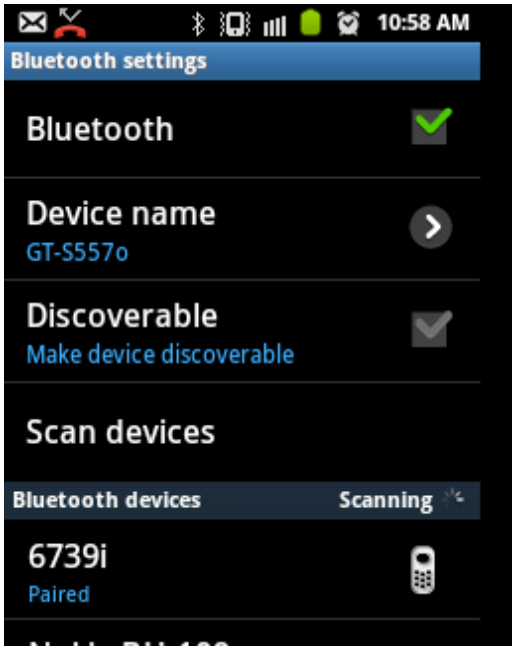
```
btbutton.setOnClickListener(new OnClickListener() {

	@Override
	public void onClick(View v) {
		// TODO Auto-generated method stub
		Toast.makeText(getApplicationContext(),
				"Make sure your Bluetooth is on,
					then scan and pair your device",
						Toast.LENGTH_LONG).show();

	Intent intentBluetooth = new Intent();

	intentBluetooth.setAction(
android.provider.Settings.ACTION_BLUETOOTH_SETTINGS);
			startActivity(intentBluetooth);
		}
});
```



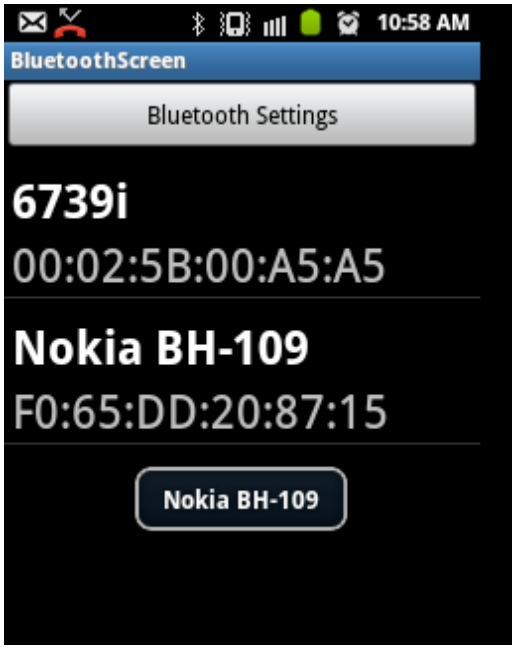
Εικόνα 7 – 15: Η οθονη BluetoothScreen με απενοργοποιημένο Bluetooth ή χωρίς pair devices



Εικόνα 7 – 16: Η οθόνη Bluetooth settings της συσκευής



Εικόνα 7 – 17: Ενεγροποίηση του Bluetooth της συσκευής



Εικόνα 7 – 18: Εμφάνιση όλων των pair devices και πάτημα στο Nokia για να τρέξι το Toast

7.13 BluetoothHDPService

Το Service αυτό έχει σκοπό να συνδέσει το Activity μας με την Bluetooth HDP (Health Device Profile) συσκευή καθώς και να αναλάβει την διαχείριση και την αποσύνδεση. Ως πρώτη δουλειά που πρέπει να κάνει το Service είναι να πάρει μια αναφορά του Proxy αντικειμένου `BluetoothHealth` μέσω του BluetoothProfile.ServiceListener.

Όταν πάρουμε το Proxy αντικείμενο `BluetoothHealth` μέσω του `BluetoothProfile.ServiceListener` στέλνουμε μέσω από Massage του Client για να καταχωρίσουμε την εφαρμογή ως SINK (δηλαδή να λαμβάνει δεδομένα από κάποια άλλη συσκευή) μέσα από την μέθοδο `registerSinkAppConfiguration` του αντικειμένου `BluetoothHealth` που πήραμε. Η `registerSinkAppConfiguration` λαμβάνει τρία χαρακτηριστικά, ένα όνομα, το είδος των δεδομένων και ένα Callback που θα μας πει αν η καταχώριση πέτυχε ή όχι, καθώς και όλες τις εργασίες που έχουν γίνει σε αυτό το configuration εφαρμογής.

Για το Callback χρησιμοποιούμε ένα αντικείμενο από την κλάση `BluetoothHealthCallback` που μέσω της μεθόδου `onHealthAppConfigurationStatusChange` μας δίνει ένα αντικείμενο `BluetoothHealthAppConfiguration` και ένα int status. Εφόσον το

`BluetoothHealth` έχει καταχωρηθεί με επιτυχία (status== `BluetoothHealth`.APP_CONFIG_REGISTRATION_SUCCESS) παίρνουμε το αντικείμενο `BluetoothHealthAppConfiguation` που θα χρησιμοποιήσουμε για να εγκαταστήσουμε την σύνδεση με την Health συσκευή. Μετά την σύνδεση μπορούμε να διαβάσουμε ή να γράψουμε δεδομένα μέσα από ένα ParcelFileDescriptor. Όταν τελειώσουμε κλίνουμε το κανάλι και καταργούμε την καταχωρημένη εφαρμογή.

Στην κατασκευή του Service ελέγχουμε αν είναι διαθέσιμο το Bluetooth και το Health Profile στην συσκευή και καλούμε την `getProfileProxy` για να πάρουμε το Proxy αντικείμενο `BluetoothAdapter` στο `onCreate`. Σε περίπτωση που ένα από τα δύο δεν είναι διαθέσιμα καλούμε την `stopSelf` για να τερματίσει το Service.

```
@Override
public void onCreate() {
  super.onCreate();
   mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
   if (mBluetoothAdapter == null ||
       !mBluetoothAdapter.isEnabled()) {
            stopSelf();
            return;
        }
   if (!mBluetoothAdapter.getProfileProxy(this,
             mBluetoothServiceListener,
             BluetoothProfile.HEALTH)) {
            stopSelf();
            return;
        }
    }
```

START_STICKY για να παραμένει το Service στην ζωή.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
        return START_STICKY;
    }
```

Επιστρέφει το Messenger στον Client

```
@Override
public IBinder onBind(Intent intent) {
        return mMessenger.getBinder();
};
```

Για να εξασφαλίσουμε την επικοινωνία μεταξύ του Service και του Client χρησιμοποιούμε τα εξής μηνύματα από το Activity προς το Service:

```
public static final int MSG_REG_CLIENT = 200;
public static final int MSG_UNREG_CLIENT = 201;
public static final int MSG_REG_HEALTH_APP = 300;
```

```
public static final int MSG_UNREG_HEALTH_APP = 301;
public static final int MSG_CONNECT_CHANNEL = 400;
public static final int MSG_DISCONNECT_CHANNEL = 401;
```

και από το Service προς το Activity:

```
public static final int STATUS_HEALTH_APP_REG = 100;
public static final int STATUS_HEALTH_APP_UNREG = 101;
public static final int STATUS_CREATE_CHANNEL = 102;
public static final int STATUS_DESTROY_CHANNEL = 103;
public static final int STATUS_READ_DATA = 104;
public static final int STATUS_READ_DATA_DONE = 105;
```

Επιπλέον έχουμε δύο ακόμα μεταβλητές για να δηλώσουμε πως όλα πήγαν καλά ή όχι.

```
public static final int RESULT_OK = 0;
public static final int RESULT_FAIL = -1;
```

Για την διαχείριση των μηνυμάτων που στέλνει ο Client κατασκευάζουμε ένα Messenger όπως φαίνεται παρακάτω. Στην περίπτωση που λάβουμε ένα MSG_REG_CLIENT περνούμε το ένα Messenger, μέσω της msg.replyTo που θα χρησιμοποιήσουμε για να στείλουμε μηνύματα στον Client μας. Ο Client στην άλλη μεριά στέλνει το Messenger του κάνοντας

Message msg = Message.*obtain*(null, BluetoothHDPService.*MSG_REG_CLIENT*);

msg.replyTo = mMessenger;

mHealthService = new Messenger(service); και mHealthService.send(msg);

```
private Messenger mClient;

private class IncomingHandler extends Handler {
      @Override
      public void handleMessage(Message msg) {
            switch (msg.what) {
            case MSG_REG_CLIENT:
                  Log.d(TAG, "Activity client registered");
                  mClient = msg.replyTo;
                  break;
                 case MSG_UNREG_CLIENT:
                  mClient = null;
                  break;
               case MSG_REG_HEALTH_APP:
                  registerApp(msg.arg1);
                  break;
               case MSG_UNREG_HEALTH_APP:
                  unregisterApp();
                  break;
               case MSG_CONNECT_CHANNEL:
                  mDevice = (BluetoothDevice) msg.obj;
                  connectChannel();
```

```
                    break;
                case MSG_DISCONNECT_CHANNEL:
                    mDevice = (BluetoothDevice) msg.obj;
                    disconnectChannel();
                    break;



                default:
                    super.handleMessage(msg);
            }
        }
    }

final Messenger mMessenger =
                            new Messenger(new IncomingHandler());
```

Όπως φαίνεται παραπάνω, ανάλογα με το μήνυμα καλούνται και μια από τις τέσσερεις μεθόδους που υλοποιούνται ως εξής.

```
private static final String TAG = "BluetoothHDPService";
```

Καταχώρηση εφαρμογής

```
private void registerApp(int dataType) {
      mBluetoothHealth.registerSinkAppConfiguration(TAG,
                                        dataType, mHealthCallback);
       }
```

Το dataType δηλώνει τον τύπο των δεδομένων της συσκευής. Για παράδειγμα, για μετρήσεις πίεσης αίματος χρησιμοποιούμε την τιμή 0x1007, για θερμοκρασία σώματος 0x1008, και άλλοι τύποι βάση του IEEE 11073.

Κατάργηση της καταχωρημένης εφαρμογής.

```
private void unregisterApp() {
             mBluetoothHealth.unregisterAppConfiguration(
                                                mHealthAppConfig);
    }
```

Σύνδεση στο κανάλι.

```
private void connectChannel() {
        mBluetoothHealth.connectChannelToSource(mDevice,
                                                mHealthAppConfig);
    }
```

Αποσύνδεση από το κανάλι.

```
private void disconnectChannel() {
        mBluetoothHealth.disconnectChannel(mDevice,
                           mHealthAppConfig, mChannelId);
    }
```

Κατασκευή αντικειμένου mBluetoothServiceListener για πάρουμε το mBluetoothHealth.

```
private final BluetoothProfile.ServiceListener
mBluetoothServiceListener =
            new BluetoothProfile.ServiceListener() {

       public void onServiceConnected(int profile,
                                      BluetoothProfile    proxy) {

              if (profile == BluetoothProfile.HEALTH) {
            mBluetoothHealth = (BluetoothHealth) proxy;
            }
        }

      public void onServiceDisconnected(int profile) {
            if (profile == BluetoothProfile.HEALTH) {
                mBluetoothHealth = null;
            }
        }
    };
```

Για την απλοποίηση της αποστολής μηνυμάτων στον Activity κατασκευάζουμε μια κλάση η οποία ελέγχει εάν το mClient είναι null (δηλαδή ο χρήστης δεν επιθυμεί να λάβει άλλα μηνύματα) αλλιώς θα αποσταλεί ένα μήνυμα και μια τιμή.

```
    // Sends an update message to registered UI client.
    private void sendMessage(int what, int value) {
        if (mClient == null) {
            Log.d(TAG, "No clients registered.");
            return;
        }

        try {
            mClient.send(Message.obtain(null, what, value, 0));
        } catch (RemoteException e) {
            // Unable to reach client.
            e.printStackTrace();
        }
    }
```

Κατασκευή του Callback.

```
private final BluetoothHealthCallback mHealthCallback =
 new   BluetoothHealthCallback() {

       public void onHealthAppConfigurationStatusChange
       (BluetoothHealthAppConfiguration config,int status) {
              if (status ==
              BluetoothHealth.APP_CONFIG_REGISTRATION_FAILURE) {
                    mHealthAppConfig = null;
```

```
                sendMessage(STATUS_HEALTH_APP_REG, RESULT_FAIL );
        } else if (status ==
              BluetoothHealth.APP_CONFIG_REGISTRATION_SUCCESS) {
                    mHealthAppConfig = config;
                    sendMessage(STATUS_HEALTH_APP_REG, RESULT_OK);
       } else if (status ==
              BluetoothHealth.APP_CONFIG_UNREGISTRATION_FAILURE ||
              status ==
              BluetoothHealth.APP_CONFIG_UNREGISTRATION_SUCCESS) {

                    sendMessage(STATUS_HEALTH_APP_UNREG,
                    status ==
             BluetoothHealth.APP_CONFIG_UNREGISTRATION_SUCCESS ?
                         RESULT_OK : RESULT_FAIL);
       }
}
```

Διαχείριση των καταστάσεων του καναλιού.

```
public void onHealthChannelStateChange(
                          BluetoothHealthAppConfiguration config,
                          BluetoothDevice device, int prevState,
                          int newState, ParcelFileDescriptor fd,
                          int channelId) {

       if (prevState ==
                    BluetoothHealth.STATE_CHANNEL_DISCONNECTED &&
           newState ==
                    BluetoothHealth.STATE_CHANNEL_CONNECTED) {

             if (config.equals(mHealthAppConfig)) {

                    mChannelId = channelId;
                    sendMessage(STATUS_CREATE_CHANNEL, RESULT_OK);
                    (new ReadThread(fd)).start();
             } else {

                 sendMessage(STATUS_CREATE_CHANNEL, RESULT_FAIL);
             }
       } else if (prevState ==
                    BluetoothHealth.STATE_CHANNEL_CONNECTING &&
                    newState ==
                    BluetoothHealth.STATE_CHANNEL_DISCONNECTED) {

             sendMessage(STATUS_CREATE_CHANNEL, RESULT_FAIL);
       } else if (
                    newState ==
                    BluetoothHealth.STATE_CHANNEL_DISCONNECTED) {

       if (config.equals(mHealthAppConfig)) {

                 sendMessage(STATUS_DESTROY_CHANNEL, RESULT_OK);
       } else {

                sendMessage(STATUS_DESTROY_CHANNEL, RESULT_FAIL);
    }
   }
  }};
```

Thread για την ανάγνωση δεδομένων.

```
private class ReadThread extends Thread {

private ParcelFileDescriptor mFd;

public ReadThread(ParcelFileDescriptor fd) {
        super();
        mFd = fd;
        }

@Override
public void run() {
       FileInputStream fis = new
       FileInputStream(mFd.getFileDescriptor());
      final byte data[] = new byte[8192];
      try {
           while(fis.read(data) > -1) {
           sendMessage(STATUS_READ_DATA, 0);
           }
      } catch(IOException ioe) {}
           if (mFd != null) {
                try {
                    mFd.close();
                } catch (IOException e) {
             }
          }
          sendMessage(STATUS_READ_DATA_DONE, 0);
       }
    }
```

ΣΥΜΠΕΡΑΣΜΑΤΑ - ΕΠΙΛΟΓΟΣ

Η πτυχιακή μου με θέμα την "ανάπτυξη κατανεμημένης εφαρμογής παρακολούθησης ασθενών με κινητές συσκευές" μου έδωσε την ευχέρεια να εμβαθύνω στην όλη αποκτηθείσα γνώση κατά την διάρκεια των σπουδών μου.

Ο βασικός στόχος, ήταν η παρουσίαση και η ανάλυση των προσδοκιών εφαρμογής και παρακολούθησης ασθενών μέσων κινητών συσκευών, σίγουρα μπορεί να έχει πρακτική εφαρμογή και έτσι η μέχρι στιγμής χρησιμοποιούμενη σύγχρονη τεχνολογία μπορεί να εφαρμοστεί στην παρακολούθηση του ασθενούς από τον θεράποντα Ιατρό του, χωρίς να χρειάζεται ο ασθενής να επισκέπτεται τον Ιατρό του είτε λόγω απόστασης ή και λόγω κινητικών προβλημάτων ή και ακόμα προς αποφυγή άσκοπης μετάβασης του ασθενή στο νοσοκομείο.

Η κατασκευή της εφαρμογής βασίζεται σε τεχνολογίες ανοιχτού λογισμικού (open source). Για την δημιουργία της εφαρμογής στο κινητό του ασθενή καθώς και του γιατρού χρησιμοποιήθηκε λογισμικό android SDK (Software Development kit), που χρησιμοποιεί Java ως γλώσσα προγραμματισμού.

Λόγω της απαίτησης για Android 4 για να συνδεθεί η συσκευή με το BluetoothHPD και να επιτρέψει στον ασθενή να πάρει μετρήσεις η εφαρμογή δεν έχει ολοκληρωθεί.

Οι συσκευές με Android 4 δεν είναι ευρέως διαδεδομένες ακόμα λόγω κόστους, αν και με την εξέλιξη της τεχνολογίας θα διατεθούν περισσότερες συσκευές στο μέλλον.

Παρόλα αυτά η συσκευή είναι έτοιμη για να συνδεθεί με Server που θα υποστηρίζει HTTP requests με JSON, επεξεργασία κειμένου, διαχείριση Sockets και διαχείριση βάσης δεδομένων.

Απαραίτητη προϋπόθεση για την εύχρηστη και πρακτικότητα της εφαρμογής είναι η ύπαρξη ασύρματης σύνδεσης στο Internet είτε μέσω 3G ή Wi-Fi, που είναι απαραίτητες για την επικοινωνία με τον Server και την λειτουργία του Location-Base Service.

Και τέλικος η εφαρμογή πρέπει να αντιμετωπίσει το χρηματικό κόστος της όλης προσπάθειας μέσω ειδικών οικονομικών προγραμμάτων για πρόσβαση στο διαδίκτυο, καθώς και έναν εναλλακτικο τροπο για εύρεσης φαρμακείων και νοσοκομείων χωρίς "bugs".

ΠΑΡΑΡΤΗΜΑ

ProjectDoc src com.shkyriacou.items

AlarmList.java

```
package com.chkyriacou.items;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;

public class AlarmList extends ArrayList<POJOAlarm> {

	/**
	 *AlarmList: Used to pass the an ArrayList of POJOAlarm Ojbect to
	 *from EventScreen (Medication Case) to AlarmService.
	 *
	 *
	 * serialVersionUID for the Serialization Runtime to verify that
	 * the sender and receiver of the serialized object have the same
	 * Ojbect. Otherwise each Machine will generate it own serialVersionUID.
	 * So when we going to take back the same Object will have different UID.
	 * Example: Write and Read the same Object, we will get an Object with
	 * different  UID.
	 *
	 *
	 */
	private static final long serialVersionUID = 691718718247014332L;

	public AlarmList() {
		super();
		// TODO Auto-generated constructor stub
	}

	public AlarmList(Collection<? extends POJOAlarm> collection) {
		super(collection);
		// TODO Auto-generated constructor stub
	}

	public AlarmList(int capacity) {
		super(capacity);
		// TODO Auto-generated constructor stub
	}

	@Override
	public boolean add(POJOAlarm object) {
		// TODO Auto-generated method stub
		return super.add(object);
	}

	@Override
	public void add(int index, POJOAlarm object) {
		// TODO Auto-generated method stub
		super.add(index, object);
	}
```

```
	@Override
	public boolean addAll(Collection<? extends POJOAlarm> collection) {
		// TODO Auto-generated method stub
		return super.addAll(collection);
	}

	@Override
	public boolean addAll(int index, Collection<? extends POJOAlarm>
collection) {
		// TODO Auto-generated method stub
		return super.addAll(index, collection);
	}

	@Override
	public void clear() {
		// TODO Auto-generated method stub
		super.clear();
	}

	@Override
	public Object clone() {
		// TODO Auto-generated method stub
		return super.clone();
	}

	@Override
	public boolean contains(Object object) {
		// TODO Auto-generated method stub
		return super.contains(object);
	}

	@Override
	public void ensureCapacity(int minimumCapacity) {
		// TODO Auto-generated method stub
		super.ensureCapacity(minimumCapacity);
	}

	@Override
	public boolean equals(Object o) {
		// TODO Auto-generated method stub
		return super.equals(o);
	}

	@Override
	public POJOAlarm get(int index) {
		// TODO Auto-generated method stub
		return super.get(index);
	}

	@Override
	public int hashCode() {
		// TODO Auto-generated method stub
		return super.hashCode();
	}

	@Override
	public int indexOf(Object object) {
		// TODO Auto-generated method stub
```

```
			return super.indexOf(object);
		}

		@Override
		public boolean isEmpty() {
			// TODO Auto-generated method stub
			return super.isEmpty();
		}

		@Override
		public Iterator<POJOAlarm> iterator() {
			// TODO Auto-generated method stub
			return super.iterator();
		}

		@Override
		public int lastIndexOf(Object object) {
			// TODO Auto-generated method stub
			return super.lastIndexOf(object);
		}

		@Override
		public POJOAlarm remove(int index) {
			// TODO Auto-generated method stub
			return super.remove(index);
		}

		@Override
		public boolean remove(Object object) {
			// TODO Auto-generated method stub
			return super.remove(object);
		}

		@Override
		protected void removeRange(int fromIndex, int toIndex) {
			// TODO Auto-generated method stub
			super.removeRange(fromIndex, toIndex);
		}

		@Override
		public POJOAlarm set(int index, POJOAlarm object) {
			// TODO Auto-generated method stub
			return super.set(index, object);
		}

		@Override
		public int size() {
			// TODO Auto-generated method stub
			return super.size();
		}

		@Override
		public Object[] toArray() {
			// TODO Auto-generated method stub
			return super.toArray();
		}

		@Override
		public <T> T[] toArray(T[] contents) {
```

```
		// TODO Auto-generated method stub
		return super.toArray(contents);
	}

	@Override
	public void trimToSize() {
		// TODO Auto-generated method stub
		super.trimToSize();
	}


}
```

JSONHelper.java

```
package com.chkyriacou.items;

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

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

/**
 * Handles all our needs of JSON. Every method is public static to be called
 * without constructing JSONHelper objects.
 *
 * Every POJO Object is pass in a JSONObject with "key-names" the Static
 * Strings from PatientDBAdapter. Those JSONObjects are pass in a JSONArray
 * who holds Objects from the same class. Then finally all JSONArrays are
 * passed in one more JSONObject with "key-names" the Table Names from
 * PatientDBAdapter Static Strings.
 *
 * In order to get our POJO Objects back we will use the Table Names to get
 * from our Final JSONObect the JSONArry we want. Then get every JSONObect
 * from JSONArry and use the "key-names" take the Values to create new POJO
 * Objects.
 *
 * All methods DOES NOT handle Exceptions, they will be handle from those who
calls them.
```

```
 *
 * JSONHelper will not be used to send Bytes (Patient's Measures from Bluetooth
Devices)
 */

public class JSONHelper {


    public JSONHelper(){

    }


    public static ArrayList<POJODoctor> getPOJODoctors(JSONObject jObj) throws
JSONException{
       ArrayList<POJODoctor> doctors = new ArrayList<POJODoctor>();


                     JSONArray jsonArray =
jObj.getJSONArray(PatientDBAdapter.DATABASE_TABLE_DOCTORS);

                     for(int i = 0; i < jsonArray.length(); i++){
                            JSONObject c = jsonArray.getJSONObject(i);

                            doctors.add( new
POJODoctor(c.getLong(PatientDBAdapter.KEY_DOCTOR_ID),

       c.getString(PatientDBAdapter.KEY_DOCTOR_FIRST_NAME),

       c.getString(PatientDBAdapter.KEY_DOCTOR_LAST_NAME)));
                     }

       return doctors;
    }

    public static ArrayList<POJOEvent> getPOJOEvents(JSONObject jObj) throws
JSONException{
       ArrayList<POJOEvent> events = new ArrayList<POJOEvent>();


                     JSONArray jsonArray =
jObj.getJSONArray(PatientDBAdapter.DATABASE_TABLE_EVENTS);

                     for(int i = 0; i < jsonArray.length(); i++){
                            JSONObject c = jsonArray.getJSONObject(i);

                            events.add( new
POJOEvent(c.getLong(PatientDBAdapter.KEY_EVENT_ID),

       c.getLong(PatientDBAdapter.KEY_EVENT_FK_PATIENT_ID),

       c.getLong(PatientDBAdapter.KEY_EVENT_FK_DOCTOR_ID),

       c.getString(PatientDBAdapter.KEY_EVENT_TITLE),
                                          c.getInt(PatientDBAdapter.KEY_EVENT_DATE),

       c.getString(PatientDBAdapter.KEY_EVENT_NOTE),

       c.getString(PatientDBAdapter.KEY_EVENT_TYPE)));
```

```
                    }

        return events;
    }

    public static ArrayList<POJOMedication> getPOJOMedications(JSONObject jObj)
throws JSONException{
        ArrayList<POJOMedication> medications = new ArrayList<POJOMedication>();


                    JSONArray jsonArray =
jObj.getJSONArray(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS);

                    for(int i = 0; i < jsonArray.length(); i++){
                            JSONObject c = jsonArray.getJSONObject(i);

                            medications.add( new
POJOMedication(c.getLong(PatientDBAdapter.KEY_MEDICATION_ID),

        c.getInt(PatientDBAdapter.KEY_MEDICATION_END_DATE),

        c.getInt(PatientDBAdapter.KEY_MEDICATION_DOSAGE))
                            );
                    }

        return medications;
    }

    public static ArrayList<POJOPatient> getPOJOPatients(JSONObject jObj) throws
JSONException{
        ArrayList<POJOPatient> patients = new ArrayList<POJOPatient>();


                    JSONArray jsonArray =
jObj.getJSONArray(PatientDBAdapter.DATABASE_TABLE_PATIENTS);

                    for(int i = 0; i < jsonArray.length(); i++){
                            JSONObject c = jsonArray.getJSONObject(i);

                            patients.add( new
POJOPatient(c.getLong(PatientDBAdapter.KEY_PATIENT_ID),

        c.getString(PatientDBAdapter.KEY_PATIENT_FIRST_NAME),

        c.getString(PatientDBAdapter.KEY_PATIENT_LAST_NAME),

        c.getString(PatientDBAdapter.KEY_PATIENT_LAST_KNOWN_ADDRESS),

        c.getLong(PatientDBAdapter.KEY_PATIENT_FK_DOCTOR_ID)) );
                    }

        return patients;
    }

    public static ArrayList<POJOMeasure> getPOJOPressures(JSONObject jObj) throws
JSONException{
        ArrayList<POJOMeasure> measures = new ArrayList<POJOMeasure>();
```

```
                    JSONArray jsonArray =
jObj.getJSONArray(PatientDBAdapter.DATABASE_TABLE_PATIENTS);

                    for(int i = 0; i < jsonArray.length(); i++){
                          JSONObject c = jsonArray.getJSONObject(i);

                          measures.add( new
POJOMeasure(c.getLong(PatientDBAdapter.KEY_MEASURE_ID),

      c.getInt(PatientDBAdapter.KEY_MEASURE_TYPE)) );
                    }

      return measures;
   }

   //Used to make a JSON holding the ID of the Patient.
   public static JSONObject idJson(long id) throws JSONException{
      JSONObject jObj = new JSONObject();
                    jObj.put(PatientDBAdapter.KEY_PATIENT_ID, id);

      return jObj;
   }

   //Returns a Final JSONObject holding the whole Database.
   public static JSONObject dBtoJSON(PatientDBAdapter db) throws JSONException{
      JSONObject jObj = new JSONObject();

      ArrayList<POJODoctor> doctors = db.getDoctors();
      ArrayList<POJOEvent> events = db.getEvents();
      ArrayList<POJOMedication> medications = db.getMedications();
      ArrayList<POJOPatient> patients = db.getPatients();
      ArrayList<POJOMeasure> measures = db.getMeasures();

      JSONObject jsonDoctor = new JSONObject();
      JSONArray jsonDoctors = new JSONArray();

      JSONObject jsonEvent = new JSONObject();
      JSONArray jsonEvents = new JSONArray();

      JSONObject jsonMedication = new JSONObject();
      JSONArray jsonMedications = new JSONArray();

      JSONObject jsonPatient = new JSONObject();
      JSONArray jsonPatients = new JSONArray();

      JSONObject jsonMeasure = new JSONObject();
      JSONArray jsonMeasures = new JSONArray();


      for(POJODoctor doctor : doctors){
            jsonDoctor = new JSONObject();
            jsonDoctor.put(PatientDBAdapter.KEY_DOCTOR_ID,
doctor.getDoctorID());
            jsonDoctor.put(PatientDBAdapter.KEY_DOCTOR_FIRST_NAME,
doctor.getFirstName());
            jsonDoctor.put(PatientDBAdapter.KEY_DOCTOR_LAST_NAME,
doctor.getLastName());

            jsonDoctors.put(jsonDoctor);
```

```
	}



	for(POJOEvent event : events){
		jsonEvent = new JSONObject();
		jsonEvent.put(PatientDBAdapter.KEY_EVENT_ID, event.getEventID());
		jsonEvent.put(PatientDBAdapter.KEY_EVENT_FK_PATIENT_ID,
event.getPatientID());
		jsonEvent.put(PatientDBAdapter.KEY_EVENT_FK_DOCTOR_ID,
event.getDoctorID());
		jsonEvent.put(PatientDBAdapter.KEY_EVENT_TITLE, event.getTitle());
		jsonEvent.put(PatientDBAdapter.KEY_EVENT_DATE, event.getDate());
		jsonEvent.put(PatientDBAdapter.KEY_EVENT_NOTE, event.getNote());
		jsonEvent.put(PatientDBAdapter.KEY_EVENT_TYPE, event.getType());

		jsonEvents.put(jsonEvent);
	}

	for(POJOMedication medication : medications){
		jsonMedication = new JSONObject();
		jsonMedication.put(PatientDBAdapter.KEY_MEDICATION_ID,
medication.getMedicationID());
		jsonMedication.put(PatientDBAdapter.KEY_MEDICATION_END_DATE,
medication.getEndDate());
		jsonMedication.put(PatientDBAdapter.KEY_MEDICATION_DOSAGE,
medication.getDosage());

		jsonMedications.put(jsonMedication);
	}

	for(POJOPatient patient : patients){
		jsonPatient = new JSONObject();
		jsonPatient.put(PatientDBAdapter.KEY_PATIENT_ID,
patient.getPatientID());
		jsonPatient.put(PatientDBAdapter.KEY_PATIENT_FIRST_NAME,
patient.getFirstName());
		jsonPatient.put(PatientDBAdapter.KEY_PATIENT_LAST_NAME,
patient.getLastName());
		jsonPatient.put(PatientDBAdapter.KEY_PATIENT_LAST_KNOWN_ADDRESS,
patient.getLastKnownLocation());
		jsonPatient.put(PatientDBAdapter.KEY_PATIENT_FK_DOCTOR_ID,
patient.getDocotorID());

		jsonPatients.put(jsonPatient);
	}

	for(POJOMeasure measure : measures){
		jsonMeasure = new JSONObject();
		jsonMeasure.put(PatientDBAdapter.KEY_MEASURE_ID,
measure.getMeasureID());

		jsonMeasures.put(jsonMeasure);
	}

	jObj.put(PatientDBAdapter.DATABASE_TABLE_DOCTORS, jsonDoctors);
	jObj.put(PatientDBAdapter.DATABASE_TABLE_EVENTS, jsonEvents);
	jObj.put(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS, jsonMedications);
	jObj.put(PatientDBAdapter.DATABASE_TABLE_PATIENTS, jsonPatients);
```

```
        jObj.put(PatientDBAdapter.DATABASE_TABLE_MEASURES, jsonMeasures);


        return jObj;
    }

    //Updating Database.
    public static void updateDB(JSONObject jObj, PatientDBAdapter db) throws
JSONException{

        /**
         * Set 1: Get ArrayLists from JSONObjects.
         *
         * Set 2: Get ArrayLists from Database.
         *
         * Set 3: Check and Delete all Entries that are not in both lists.
         *
         * Set 4: Update Database and if update...() returns 0 then means it
doesn't
         *        exists in Database, so insert the new entry.
         */

        //Set 1
        ArrayList<POJOEvent> events = new ArrayList<POJOEvent>();
        ArrayList<POJOMedication> medications = new ArrayList<POJOMedication>();
        ArrayList<POJOMeasure> measures = new ArrayList<POJOMeasure>();
        ArrayList<POJOPatient> patients = new ArrayList<POJOPatient>();
        ArrayList<POJODoctor> doctors = new ArrayList<POJODoctor>();

        events = getPOJOEvents(jObj);
        medications = getPOJOMedications(jObj);
        measures = getPOJOPressures(jObj);
        patients = getPOJOPatients(jObj);
        doctors = getPOJODoctors(jObj);

        //Set 2
        ArrayList<POJOEvent> eventsDB = db.getEvents();
        ArrayList<POJOMedication> medicationsDB = db.getMedications();
        ArrayList<POJOMeasure> measuresDB = db.getMeasures();
        ArrayList<POJOPatient> patientsDB = db.getPatients();
        ArrayList<POJODoctor> doctorsDB = db.getDoctors();


        //Set 3
        boolean flag = false;

        if (eventsDB != null){
        for(int i=0; i<eventsDB.size(); i++){
              for(int j=0; j<events.size(); j++){
                    if(eventsDB.get(i).getEventID()==events.get(j).getEventID()){
                          flag = true;
                    }
              }
              if(!flag){
                    db.deleteEvent(eventsDB.get(i));
              }
              flag = false;
        }
        }
```

```
      //Set 4
      for(POJOEvent event : events){
            if(db.updateEvent(event)==0){
            db.insertEvent(event);
            }
      }

   ////////////////////////////////////////////////////////////////////*

      //Set 3
      if (medicationsDB != null){
      for(int i=0; i<medicationsDB.size(); i++){
            for(int j=0; j<medications.size(); j++){

      if(medicationsDB.get(i).getMedicationID()==medications.get(j).getMedication
ID()){
                        flag = true;
                  }
            }
            if(!flag){
                  db.deleteMedication(medicationsDB.get(i));
            }
            flag = false;
      }
      }

      //Set 4
      for(POJOMedication medication : medications){
            if(db.updateMedication(medication)==0){
            db.insertMedication(medication);
            }
      }

   ////////////////////////////////////////////////////////////////////

      //Set 3
      if (measuresDB != null){
      for(int i=0; i<measuresDB.size(); i++){
            for(int j=0; j<measures.size(); j++){

      if(measuresDB.get(i).getMeasureID()==measures.get(j).getMeasureID()){
                        flag = true;
                  }
            }
            if(!flag){
                  db.deletePressure(measuresDB.get(i));
            }
            flag = false;
      }
      }

      //Set 4
      for(POJOMeasure measure : measures){
            if(db.updateMeasure(measure)==0){
                  db.insertMeasure(measure);
            }
      }
```

```
    ////////////////////////////////////////////////////////////////////////////

      //Set 3
            if (patientsDB != null){
                  for(int i=0; i<patientsDB.size(); i++){
                        for(int j=0; j<patients.size(); j++){

      if(patients.get(i).getPatientID()==patients.get(j).getPatientID()){
                                          flag = true;
                                    }
                              }
                              if(!flag){
                                    db.deletePatient(patientsDB.get(i));
                              }
                              flag = false;
                  }
                  }

      for(POJOPatient patient : patients){
            if(db.updatePatient(patient)==0){
            db.insertPatient(patient);
            }
      }

    ////////////////////////////////////////////////////////////////////////////

            if (doctorsDB != null){
                  for(int i=0; i<doctorsDB.size(); i++){
                        for(int j=0; j<doctors.size(); j++){

      if(doctorsDB.get(i).getDoctorID()==doctors.get(j).getDoctorID()){
                                          flag = true;
                                    }
                              }
                              if(!flag){
                                    db.deleteDoctor(doctorsDB.get(i));
                              }
                              flag = false;
                  }
                  }

            for(POJODoctor doctor : doctors){
                  if(db.updateDoctor(doctor)==0){
                  db.insertDoctor(doctor);
                  }
            }
    }



    public static HttpResponse doPost(String url, JSONObject c) throws
ClientProtocolException, IOException

    {
      int TIMEOUT_MILLISEC = 10000;  // 10 seconds

      //Represents a collection of HTTP protocol and framework parameters.
      HttpParams httpParams = new BasicHttpParams();
```

```
      //HttpConnectionParams is an adaptor for accessing connection parameters in
HttpParams.

      // Set the timeout in milliseconds until a connection is established.
      HttpConnectionParams.setConnectionTimeout(httpParams, TIMEOUT_MILLISEC);

      // Set  the socket timeout for waiting for data.
      HttpConnectionParams.setSoTimeout(httpParams, TIMEOUT_MILLISEC);

      //Those 2 "sets" has changed the parameters in httpParams.
      DefaultHttpClient httpclient = new DefaultHttpClient(httpParams);

      HttpPost request = new HttpPost(url);

      //Put the String into the Entity using UTF-8 Encoding
        StringEntity entity = new StringEntity(c.toString(),"UTF-8");

        //set the Content Type and Encoding in the Header so the other side
        //will know what it's dealing with.
        entity.setContentType("application/json;charset=UTF-8");
        entity.setContentEncoding((Header) new BasicHeader(HTTP.CONTENT_TYPE,
"application/json"));


        request.setEntity(entity);

        HttpResponse response;

        response = httpclient.execute(request);

       return response;
    }


    //Get the JSONObject from the response.
    public static JSONObject responseToJson(HttpResponse httpResponse) throws
IllegalStateException, IOException, JSONException{

      JSONObject jObj = null;
      BufferedReader reader;
      StringBuilder sb;
      String line;
      String jsonSting = "";
      InputStream is = null;

            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();

            //Using UTF-8 to get the BufferReader with the String.
            reader = new BufferedReader(new InputStreamReader(
                                        is, "UTF-8"),10);

                  sb = new StringBuilder();
          line = null;

          while ((line = reader.readLine()) != null) {
                      sb.append(line + "\n");
                  }
```

```
            is.close();
            jsonSting = sb.toString();

               jObj = new JSONObject(jsonSting);

        return jObj;
    }
}
```

PatientDBAdapter.java

```
package com.chkyriacou.items;



import java.util.ArrayList;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;


public class PatientDBAdapter {

       /**
        * Class between the Database and the rest of the Application.
        *
        * All public static final Strings will be use for marking values in other
parts
        * in the Application.
        *
        * The field KEY_EVENT_TYPE holds one of their Table names:
        *
        * DATABASE_TABLE_EVENTS = The Event is a Note
        * DATABASE_TABLE_MEDICATIONS = The event is a Medication
        * so an Extra look up is needed to get the POJOMedication [1 Event : 1
Mediaction]
        *
        * DATABASE_TABLE_MEASURES = The event is Measure
        * so an Extra look up is needed to get the POJOMeasure [1 Event : 1
Mediaction]
        *
        *
        * All Times/Dates are insert "Backwords"
        * example: 10/9/2012 to 20120910
        * and      10:30 to 3010.
```

```
         *
         * logic: 1467 = 1000 > 400 > 60 > 7
         *    so: Years > Months > Days
         *      Hours > Mins > Secs
         *
         * This way we can get our Entries in Time/Date based order.
         *
         *
         *
         */

        public static final String DATABASE_TABLE_PATIENTS="patients";
        public static final String KEY_PATIENT_ID="_patient_id";
        public static final String KEY_PATIENT_FIRST_NAME="patient_first_name";
        public static final String KEY_PATIENT_LAST_NAME="patient_last_name";
        public static final String
KEY_PATIENT_LAST_KNOWN_ADDRESS="patient_last_known_address";
        public static final String KEY_PATIENT_FK_DOCTOR_ID="doctor_fk_id";

        public static final String DATABASE_TABLE_DOCTORS="doctors";
        public static final String KEY_DOCTOR_ID="_doctor_id";
        public static final String KEY_DOCTOR_FIRST_NAME="doctor_first_name";
        public static final String KEY_DOCTOR_LAST_NAME="doctor_last_name";

        public static final String DATABASE_TABLE_EVENTS="events";
        public static final String KEY_EVENT_ID="_event_id";
        public static final String KEY_EVENT_FK_PATIENT_ID="patient_fk_id";
        public static final String KEY_EVENT_FK_DOCTOR_ID="doctor_fk_id";
        public static final String KEY_EVENT_TITLE="title";
        public static final String KEY_EVENT_DATE="date";
        public static final String KEY_EVENT_NOTE="note";
        public static final String KEY_EVENT_TYPE="type";


        public static final String DATABASE_TABLE_MEDICATIONS="medications";
        public static final String KEY_MEDICATION_ID="_medication_id";
        public static final String KEY_MEDICATION_END_DATE="end_date";
        public static final String KEY_MEDICATION_DOSAGE="dosage";

        public static final String DATABASE_TABLE_MEASURES="measures";
        public static final String KEY_MEASURE_ID="_measure_id";
        public static final String KEY_MEASURE_TYPE = "type";
        public static final String KEY_MEASURE_DATA_STORE="data_store";


        public static final String DATABASE_TABLE_ALARMS="alarms";
        public static final String KEY_ALARM_ID = "_alarm_id";
        public static final String KEY_ALARM_TIME = "time";
        public static final String KEY_ALARM_END_DATE = "end_data";


        private static final String DATABASE_NAME = "PatientDB.db";
        private static final int DATABASE_VERSION = 11;

        private DbHelper myHelper;
        private SQLiteDatabase myDatabase;


            public PatientDBAdapter(Context c){
```

```
                        //Cunstractor
                        myHelper = new DbHelper(c);
                }

                public void open()throws SQLException{
                        //Open and get a Writable Database.
                        myDatabase = myHelper.getWritableDatabase();
                }

                public void close(){
                        myHelper.close();
                }


                //Doctor Modules
                public ArrayList<POJODoctor> getDoctors(){


                        String[] columns = new String[]{
                                        KEY_DOCTOR_ID, KEY_DOCTOR_FIRST_NAME,
                                        KEY_DOCTOR_LAST_NAME};

                                Cursor c = myDatabase.query(DATABASE_TABLE_DOCTORS,
columns,
                                                null, null, null, null, null);

                                if(c.moveToFirst()){
                                ArrayList<POJODoctor> doctors =  new
ArrayList<POJODoctor>();

                                int iDoctorID = c.getColumnIndex(KEY_DOCTOR_ID);
                                int iFirstName =
c.getColumnIndex(KEY_DOCTOR_FIRST_NAME);
                                int iLastName = c.getColumnIndex(KEY_DOCTOR_LAST_NAME);

                                for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){

                                        doctors.add(new POJODoctor(c.getLong(iDoctorID),
c.getString(iFirstName), c.getString(iLastName)));
                                }

                                return doctors;
                                }else{
                                        return null;
                                }
                        }

                public POJODoctor getDoctor(long id){
                        POJODoctor doctor;

                        String[] columns = new String[]{
                                        KEY_DOCTOR_ID, KEY_DOCTOR_FIRST_NAME,
                                        KEY_DOCTOR_LAST_NAME};

                        Cursor c = myDatabase.query(DATABASE_TABLE_DOCTORS,
                                        columns, KEY_DOCTOR_ID +" = ?",
                                        new String[]{Long.toString(id)}, null, null,
null);
```

```
                    if(c.moveToFirst()){
                    int iDoctorID= c.getColumnIndex(KEY_DOCTOR_ID);
                    int iDoctorFirstName=c.getColumnIndex(KEY_DOCTOR_FIRST_NAME);
                    int iDoctorLastName=c.getColumnIndex(KEY_DOCTOR_LAST_NAME);
                    doctor = new POJODoctor(c.getLong(iDoctorID),
c.getString(iDoctorFirstName), c.getString(iDoctorLastName));
                    return doctor;
                    }
                    else{return null;}
             }

             public long insertDoctor(POJODoctor doctor){
                    ContentValues cv = new ContentValues();
                    cv.put(KEY_DOCTOR_ID, doctor.getDoctorID());
                    cv.put(KEY_DOCTOR_FIRST_NAME, doctor.getFirstName());
                    cv.put(KEY_DOCTOR_LAST_NAME, doctor.getLastName());

                    return myDatabase.insert(DATABASE_TABLE_DOCTORS, null, cv);
             }

             public int updateDoctor(POJODoctor doctor){
                    ContentValues cv = new ContentValues();
                    cv.put(KEY_DOCTOR_ID, doctor.getDoctorID());
                    cv.put(KEY_DOCTOR_FIRST_NAME, doctor.getFirstName());
                    cv.put(KEY_DOCTOR_LAST_NAME, doctor.getLastName());

                    return myDatabase.update(DATABASE_TABLE_DOCTORS, cv,
KEY_DOCTOR_ID+" = ?",
                                  new String[]{
String.valueOf(doctor.getDoctorID()) });
             }

             public int deleteDoctor(POJODoctor doctor){
                    return
myDatabase.delete(DATABASE_TABLE_DOCTORS,KEY_DOCTOR_ID+" = ?",
                                  new String[]{
String.valueOf(doctor.getDoctorID()) });
             }

             public int emptyDoctorsTable(){
                    return myDatabase.delete(DATABASE_TABLE_DOCTORS, null, null);
             }

             //Patient Modules

             public ArrayList<POJOPatient> getPatients(){

                    String[] columns = new String[]{
                                  KEY_PATIENT_ID, KEY_PATIENT_FIRST_NAME,
                                  KEY_PATIENT_LAST_NAME,
KEY_PATIENT_LAST_KNOWN_ADDRESS, KEY_EVENT_FK_DOCTOR_ID};

                    Cursor c = myDatabase.query(DATABASE_TABLE_PATIENTS, columns,
                                        null, null, null, null, null);

                    if(c.moveToFirst()){
                           ArrayList<POJOPatient> patients =  new
ArrayList<POJOPatient>();
```

```
                        int iPatientID = c.getColumnIndex(KEY_PATIENT_ID);
                        int iFirstName =
c.getColumnIndex(KEY_PATIENT_FIRST_NAME);
                        int iLastName =
c.getColumnIndex(KEY_PATIENT_LAST_NAME);
                        int iLastKnownAddress =
c.getColumnIndex(KEY_PATIENT_LAST_KNOWN_ADDRESS);
                        int iFKDoctorID =
c.getColumnIndex(KEY_EVENT_FK_DOCTOR_ID);

                        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                              patients.add(new
POJOPatient(c.getLong(iPatientID), c.getString(iFirstName),
                                                      c.getString(iLastName),
c.getString(iLastKnownAddress), c.getLong(iFKDoctorID)));
                        }

                        return patients;
                  }else{
                              return null;
                  }
            }

            public POJOPatient getPatient(long id){
                  POJOPatient patient;

                  String[] columns = new String[]{
                              KEY_PATIENT_ID, KEY_PATIENT_FIRST_NAME,
                              KEY_PATIENT_LAST_NAME,
KEY_PATIENT_LAST_KNOWN_ADDRESS, KEY_EVENT_FK_DOCTOR_ID};

                  Cursor c = myDatabase.query(DATABASE_TABLE_PATIENTS,
                              columns, KEY_PATIENT_ID +" = ?",
                              new String[]{Long.toString(id)}, null, null,
null);

                  if(c.moveToFirst()){
                        int iPatientID = c.getColumnIndex(KEY_PATIENT_ID);
                        int iFirstName =
c.getColumnIndex(KEY_PATIENT_FIRST_NAME);
                        int iLastName =
c.getColumnIndex(KEY_PATIENT_LAST_NAME);
                        int iLastKnownAddress =
c.getColumnIndex(KEY_PATIENT_LAST_KNOWN_ADDRESS);
                        int iFKDoctorID =
c.getColumnIndex(KEY_PATIENT_FK_DOCTOR_ID);
                        patient = new POJOPatient(c.getLong(iPatientID),
c.getString(iFirstName),
                                          c.getString(iLastName),
c.getString(iLastKnownAddress),c.getLong(iFKDoctorID));
                  return patient;
                  }
                  else{return null;}
            }

            public long insertPatient(POJOPatient patient){
                  ContentValues cv = new ContentValues();
                  cv.put(KEY_PATIENT_ID, patient.getPatientID());
                  cv.put(KEY_PATIENT_FIRST_NAME, patient.getFirstName());
```

```
                    cv.put(KEY_PATIENT_LAST_NAME, patient.getLastName());
                    cv.put(KEY_PATIENT_LAST_KNOWN_ADDRESS,
patient.getLastKnownLocation());
                    return myDatabase.insert(DATABASE_TABLE_PATIENTS, null, cv);
             }

             public int updatePatient(POJOPatient patient){
                    ContentValues cv = new ContentValues();
                    cv.put(KEY_PATIENT_ID, patient.getPatientID());
                    cv.put(KEY_PATIENT_FIRST_NAME, patient.getFirstName());
                    cv.put(KEY_PATIENT_LAST_NAME, patient.getLastName());
                    cv.put(KEY_PATIENT_LAST_KNOWN_ADDRESS,
patient.getLastKnownLocation());
                    cv.put(KEY_PATIENT_FK_DOCTOR_ID, patient.getDocotorID());
                    return myDatabase.update(DATABASE_TABLE_PATIENTS, cv,
KEY_PATIENT_ID+" = ?",
                                  new String[]{
String.valueOf(patient.getPatientID()) });
             }

             public int deletePatient(POJOPatient patient){
                    return
myDatabase.delete(DATABASE_TABLE_PATIENTS,KEY_PATIENT_ID+" = ?",
                                  new String[]{
String.valueOf(patient.getPatientID()) });
             }

             public int emptyPatientsTable(){
                    return myDatabase.delete(DATABASE_TABLE_PATIENTS, null, null);
             }



             //Event Modules
             public ArrayList<POJOEvent> getEvents(){

                    String[] columns = new String[]{
                                  KEY_EVENT_ID, KEY_EVENT_FK_PATIENT_ID,
                                  KEY_EVENT_FK_DOCTOR_ID, KEY_EVENT_TITLE,
KEY_EVENT_DATE,
                                  KEY_EVENT_NOTE,KEY_EVENT_TYPE};

                    Cursor c = myDatabase.query(DATABASE_TABLE_EVENTS, columns,
                                        null, null, null, null, KEY_EVENT_DATE +"
DESC");

                    if(c.moveToFirst()){
                           ArrayList<POJOEvent> events =  new
ArrayList<POJOEvent>();

                           int iEventID = c.getColumnIndex(KEY_EVENT_ID);
                           int iFKPatientID =
c.getColumnIndex(KEY_EVENT_FK_PATIENT_ID);
                           int iFKDoctorID =
c.getColumnIndex(KEY_EVENT_FK_DOCTOR_ID);
                           int iTitle = c.getColumnIndex(KEY_EVENT_TITLE);
                           int iDate = c.getColumnIndex(KEY_EVENT_DATE);
                           int iNote = c.getColumnIndex(KEY_EVENT_NOTE);
                           int iType = c.getColumnIndex(KEY_EVENT_TYPE);
```

```
                        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                              events.add(new POJOEvent(c.getLong(iEventID),
c.getLong(iFKPatientID),
                                                      c.getLong(iFKDoctorID),
c.getString(iTitle), c.getInt(iDate),

      c.getString(iNote),c.getString(iType)));
                        }

                        return events;
                  }else{
                        return null;
                  }
            }



            public ArrayList<POJOEvent> getEventsNotes(){

                  String[] columns = new String[]{
                              KEY_EVENT_ID, KEY_EVENT_FK_PATIENT_ID,
                              KEY_EVENT_FK_DOCTOR_ID, KEY_EVENT_TITLE,
KEY_EVENT_DATE,
                              KEY_EVENT_NOTE,KEY_EVENT_TYPE};

                  Cursor c = myDatabase.query(DATABASE_TABLE_EVENTS, columns,
                              KEY_EVENT_TYPE+" = ?",
                              new String[]{DATABASE_TABLE_EVENTS}, null, null,
KEY_EVENT_DATE +" DESC");

                  if(c.moveToFirst()){
                        ArrayList<POJOEvent> events =  new
ArrayList<POJOEvent>();

                        int iEventID = c.getColumnIndex(KEY_EVENT_ID);
                        int iFKPatientID =
c.getColumnIndex(KEY_EVENT_FK_PATIENT_ID);
                        int iFKDoctorID =
c.getColumnIndex(KEY_EVENT_FK_DOCTOR_ID);
                        int iTitle = c.getColumnIndex(KEY_EVENT_TITLE);
                        int iDate = c.getColumnIndex(KEY_EVENT_DATE);
                        int iNote = c.getColumnIndex(KEY_EVENT_NOTE);
                        int iType = c.getColumnIndex(KEY_EVENT_TYPE);

                        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                              events.add(new POJOEvent(c.getLong(iEventID),
c.getLong(iFKPatientID),
                                                      c.getLong(iFKDoctorID),
c.getString(iTitle), c.getInt(iDate),

      c.getString(iNote),c.getString(iType)));
                        }

                        return events;
                  }else{
                        return null;
                  }
            }
```

```
            public ArrayList<POJOEvent> getEventsMedications(){

                  String[] columns = new String[]{
                  KEY_EVENT_ID, KEY_EVENT_FK_PATIENT_ID,
                  KEY_EVENT_FK_DOCTOR_ID, KEY_EVENT_TITLE, KEY_EVENT_DATE,
                  KEY_EVENT_NOTE,KEY_EVENT_TYPE};

                  Cursor c = myDatabase.query(DATABASE_TABLE_EVENTS, columns,
                  KEY_EVENT_TYPE+" = ?",
                  new String[]{DATABASE_TABLE_MEDICATIONS}, null, null,
KEY_EVENT_DATE +" DESC");

                  if(c.moveToFirst()){
                        ArrayList<POJOEvent> events =  new
ArrayList<POJOEvent>();

                  int iEventID = c.getColumnIndex(KEY_EVENT_ID);
                  int iFKPatientID = c.getColumnIndex(KEY_EVENT_FK_PATIENT_ID);
                  int iFKDoctorID = c.getColumnIndex(KEY_EVENT_FK_DOCTOR_ID);
                  int iTitle = c.getColumnIndex(KEY_EVENT_TITLE);
                  int iDate = c.getColumnIndex(KEY_EVENT_DATE);
                  int iNote = c.getColumnIndex(KEY_EVENT_NOTE);
                  int iType = c.getColumnIndex(KEY_EVENT_TYPE);

                  for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                  events.add(new POJOEvent(c.getLong(iEventID),
c.getLong(iFKPatientID),
                              c.getLong(iFKDoctorID), c.getString(iTitle),
c.getInt(iDate),
                              c.getString(iNote),c.getString(iType)));
                  }

                  return events;
            }else{
                  return null;
                  }
            }



            public ArrayList<POJOEvent> getEventsMeasures(){

                  String[] columns = new String[]{
                              KEY_EVENT_ID, KEY_EVENT_FK_PATIENT_ID,
                              KEY_EVENT_FK_DOCTOR_ID, KEY_EVENT_TITLE,
KEY_EVENT_DATE,
                              KEY_EVENT_NOTE,KEY_EVENT_TYPE};

                  Cursor c = myDatabase.query(DATABASE_TABLE_EVENTS, columns,
                              KEY_EVENT_TYPE+" = ?",
                              new String[]{DATABASE_TABLE_MEASURES}, null,
null, KEY_EVENT_DATE +" DESC");

                  if(c.moveToFirst()){
                        ArrayList<POJOEvent> events =  new
ArrayList<POJOEvent>();
```

```
                        int iEventID = c.getColumnIndex(KEY_EVENT_ID);
                        int iFKPatientID =
c.getColumnIndex(KEY_EVENT_FK_PATIENT_ID);
                        int iFKDoctorID =
c.getColumnIndex(KEY_EVENT_FK_DOCTOR_ID);
                        int iTitle = c.getColumnIndex(KEY_EVENT_TITLE);
                        int iDate = c.getColumnIndex(KEY_EVENT_DATE);
                        int iNote = c.getColumnIndex(KEY_EVENT_NOTE);
                        int iType = c.getColumnIndex(KEY_EVENT_TYPE);

                        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                              events.add(new POJOEvent(c.getLong(iEventID),
c.getLong(iFKPatientID),
                                                c.getLong(iFKDoctorID),
c.getString(iTitle), c.getInt(iDate),

      c.getString(iNote),c.getString(iType)));
                        }

                        return events;
                  }else{
                        return null;
                  }
            }



            public POJOEvent getEvent(long id){
                  POJOEvent event;

                  String[] columns = new String[]{
                              KEY_EVENT_ID, KEY_EVENT_FK_PATIENT_ID,
                              KEY_EVENT_FK_DOCTOR_ID, KEY_EVENT_TITLE,
KEY_EVENT_DATE,
                              KEY_EVENT_NOTE, KEY_EVENT_TYPE};

                  Cursor c = myDatabase.query(DATABASE_TABLE_EVENTS,
                              columns, KEY_EVENT_ID+" = ?",
                              new String[]{Long.toString(id)}, null, null,
null);

                  if(c.moveToFirst()){
                        int iEventID = c.getColumnIndex(KEY_EVENT_ID);
                        int iFKPatientID =
c.getColumnIndex(KEY_EVENT_FK_PATIENT_ID);
                        int iFKDoctorID =
c.getColumnIndex(KEY_EVENT_FK_DOCTOR_ID);
                        int iTitle = c.getColumnIndex(KEY_EVENT_TITLE);
                        int iDate = c.getColumnIndex(KEY_EVENT_DATE);
                        int iNote = c.getColumnIndex(KEY_EVENT_NOTE);
                        int iType = c.getColumnIndex(KEY_EVENT_TYPE);

                        event = new POJOEvent(c.getLong(iEventID),
c.getLong(iFKPatientID),
                                          c.getLong(iFKDoctorID),
c.getString(iTitle), c.getInt(iDate),
                                          c.getString(iNote),c.getString(iType));
                  return event;
                  }
```

```
                else{return null;}
        }

        public long insertEvent(POJOEvent event){
                ContentValues cv = new ContentValues();
                cv.put(KEY_EVENT_ID, event.getEventID());
                cv.put(KEY_EVENT_FK_PATIENT_ID, event.getPatientID());
                cv.put(KEY_EVENT_FK_DOCTOR_ID, event.getDoctorID());
                cv.put(KEY_EVENT_TITLE, event.getTitle());
                cv.put(KEY_EVENT_DATE, event.getDate());
                cv.put(KEY_EVENT_NOTE, event.getNote());
                cv.put(KEY_EVENT_TYPE, event.getType());
                return myDatabase.insert(DATABASE_TABLE_EVENTS, null, cv);
        }

        public int updateEvent(POJOEvent event){
                ContentValues cv = new ContentValues();
                cv.put(KEY_EVENT_ID, event.getEventID());
                cv.put(KEY_EVENT_FK_PATIENT_ID, event.getPatientID());
                cv.put(KEY_EVENT_FK_DOCTOR_ID, event.getDoctorID());
                cv.put(KEY_EVENT_TITLE, event.getTitle());
                cv.put(KEY_EVENT_DATE, event.getDate());
                cv.put(KEY_EVENT_NOTE, event.getNote());
                cv.put(KEY_EVENT_TYPE, event.getType());
                return myDatabase.update(DATABASE_TABLE_EVENTS, cv,
KEY_EVENT_ID+" = ?",
                                new String[]{ String.valueOf(event.getEventID())
});
        }

        public int deleteEvent(POJOEvent event){
                return myDatabase.delete(DATABASE_TABLE_EVENTS,KEY_EVENT_ID+"
= ?",
                                new String[]{ String.valueOf(event.getEventID())
});
        }

        public int emptyEventsTable(){
                return myDatabase.delete(DATABASE_TABLE_EVENTS, null, null);
        }



        //Medication Modules

        public ArrayList<POJOMedication> getMedications(){

                String[] columns = new String[]{
                                KEY_MEDICATION_ID, KEY_MEDICATION_END_DATE,
                                KEY_MEDICATION_DOSAGE};

                Cursor c = myDatabase.query(DATABASE_TABLE_MEDICATIONS,
columns,
                                        null, null, null, null, null);

                if(c.moveToFirst()){
                        ArrayList<POJOMedication> medications =  new
ArrayList<POJOMedication>();
```

```
                        int iMedicationID =
c.getColumnIndex(KEY_MEDICATION_ID);
                        int iEndDate=
c.getColumnIndex(KEY_MEDICATION_END_DATE);
                        int iDosage = c.getColumnIndex(KEY_MEDICATION_DOSAGE);


                        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                              medications.add(new
POJOMedication(c.getLong(iMedicationID),

      c.getInt(iEndDate),c.getInt(iDosage)));
                        }

                        return medications;
                  }else{
                        return null;
                  }
            }

            public POJOMedication getMedication(long id){
                  POJOMedication medication;

                  String[] columns = new String[]{
                              KEY_MEDICATION_ID, KEY_MEDICATION_END_DATE,
                              KEY_MEDICATION_DOSAGE};

                  Cursor c = myDatabase.query(DATABASE_TABLE_MEDICATIONS,
                              columns, KEY_MEDICATION_ID+" = ?",
                              new String[]{Long.toString(id)}, null, null,
null);

                  if(c.moveToFirst()){
                        int iMedicationID =
c.getColumnIndex(KEY_MEDICATION_ID);
                        int iEndDate=
c.getColumnIndex(KEY_MEDICATION_END_DATE);
                        int iDosage = c.getColumnIndex(KEY_MEDICATION_DOSAGE);

                        medication = new
POJOMedication(c.getLong(iMedicationID),
                                    c.getInt(iEndDate),c.getInt(iDosage));
                  return medication;
                  }
                  else{return null;}
            }

            public long insertMedication(POJOMedication medication){
                  ContentValues cv = new ContentValues();
                  cv.put(KEY_MEDICATION_ID, medication.getMedicationID());
                  cv.put(KEY_MEDICATION_END_DATE, medication.getEndDate());
                  cv.put(KEY_MEDICATION_DOSAGE, medication.getDosage());

                  return myDatabase.insert(DATABASE_TABLE_MEDICATIONS, null,
cv);
            }

            public int updateMedication(POJOMedication medication){
                  ContentValues cv = new ContentValues();
```

```
                cv.put(KEY_MEDICATION_ID, medication.getMedicationID());
                cv.put(KEY_MEDICATION_END_DATE, medication.getEndDate());
                cv.put(KEY_MEDICATION_DOSAGE, medication.getDosage());

                return myDatabase.update(DATABASE_TABLE_MEDICATIONS, cv,
KEY_MEDICATION_ID+" = ?",
                                new String[]{
String.valueOf(medication.getMedicationID()) });
        }

        public int deleteMedication(POJOMedication medication){
                return
myDatabase.delete(DATABASE_TABLE_MEDICATIONS,KEY_MEDICATION_ID+" = ?",
                                new String[]{
String.valueOf(medication.getMedicationID()) });
        }

        public int emptyMedicationTable(){
                return myDatabase.delete(DATABASE_TABLE_MEDICATIONS, null,
null);
        }




    //Measure Modules
        public ArrayList<POJOMeasure> getMeasures(){

                String[] columns = new String[]{
                                KEY_MEASURE_ID, KEY_MEASURE_TYPE
,KEY_MEASURE_DATA_STORE};

                Cursor c = myDatabase.query(DATABASE_TABLE_MEASURES, columns,
                                        null, null, null, null, null);

                if(c.moveToFirst()){
                        ArrayList<POJOMeasure> measures =  new
ArrayList<POJOMeasure>();

                        int iMeasureID= c.getColumnIndex(KEY_MEASURE_ID);
                        int iType= c.getColumnIndex(KEY_MEASURE_TYPE);
                        int iDataStore =
c.getColumnIndex(KEY_MEASURE_DATA_STORE);


                        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                                measures.add(new
POJOMeasure(c.getLong(iMeasureID),
                                                c.getInt(iType),
                                                c.getBlob(iDataStore)) );
                        }

                        return measures;
                }else{
                        return null;
                }
        }

        public POJOMeasure getMeasure(long id){
```

```
                    POJOMeasure measure;

                    String[] columns = new String[]{
                                KEY_MEASURE_ID, KEY_MEASURE_TYPE
,KEY_MEASURE_DATA_STORE};

                    Cursor c = myDatabase.query(DATABASE_TABLE_MEASURES,
                                columns, KEY_MEASURE_ID+" = ?",
                                new String[]{Long.toString(id)}, null, null,
null);

                    if(c.moveToFirst()){
                          int iPressureID = c.getColumnIndex(KEY_MEASURE_ID);
                          int iType = c.getColumnIndex(KEY_MEASURE_TYPE);
                          int iDataStore =
c.getColumnIndex(KEY_MEASURE_DATA_STORE);


                          measure = new POJOMeasure(c.getLong(iPressureID),
                                      c.getInt(iType),
                                      c.getBlob(iDataStore));
                    return measure;
                    }
                    else{return null;}
              }

              public long insertMeasure(POJOMeasure measure){
                    ContentValues cv = new ContentValues();
                    cv.put(KEY_MEASURE_ID, measure.getMeasureID());
                    cv.put(KEY_MEASURE_TYPE, measure.getType());
                    cv.put(KEY_MEASURE_DATA_STORE, measure.getData());

                    return myDatabase.insert(DATABASE_TABLE_MEASURES, null, cv);
              }

              public int updateMeasure(POJOMeasure measure){
                    ContentValues cv = new ContentValues();
                    cv.put(KEY_MEASURE_ID, measure.getMeasureID());
                    cv.put(KEY_MEASURE_TYPE, measure.getType());
                    cv.put(KEY_MEASURE_DATA_STORE, measure.getData());


                    return myDatabase.update(DATABASE_TABLE_MEASURES, cv,
KEY_MEASURE_ID+" = ?",
                                new String[]{
String.valueOf(measure.getMeasureID()) });
              }

              public int deletePressure(POJOMeasure measure){
                    return
myDatabase.delete(DATABASE_TABLE_MEASURES,KEY_MEASURE_ID+" = ?",
                                new String[]{
String.valueOf(measure.getMeasureID()) });
              }

              public int emptyPressureTable(){
                    return myDatabase.delete(DATABASE_TABLE_MEASURES, null, null);
              }
```

```
            //ALAMS
            public ArrayList<POJOAlarm> getAlarms(){

                  String[] columns = new String[]{
                              KEY_ALARM_ID,KEY_ALARM_TIME, KEY_ALARM_END_DATE};

                  Cursor c = myDatabase.query(DATABASE_TABLE_ALARMS, columns,
                                    null, null, null, null, KEY_ALARM_TIME);

                  if(c.moveToFirst()){
                        ArrayList<POJOAlarm> alarms =  new
ArrayList<POJOAlarm>();

                        int iAlarmID = c.getColumnIndex(KEY_ALARM_ID);
                        int iAlarmTime = c.getColumnIndex(KEY_ALARM_TIME);
                        int iAlarmEndDate =
c.getColumnIndex(KEY_ALARM_END_DATE);


                        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
                              alarms.add(new POJOAlarm(c.getLong(iAlarmID),

      c.getInt(iAlarmTime),c.getInt(iAlarmEndDate)));
                        }

                        return alarms;
                  }else{
                        return null;
                  }
            }

            public POJOAlarm getAlarm(long id){
                  POJOAlarm alarm;

                  String[] columns = new String[]{
                              KEY_ALARM_ID,KEY_ALARM_TIME, KEY_ALARM_END_DATE};

                  Cursor c = myDatabase.query(DATABASE_TABLE_ALARMS,
                              columns, KEY_ALARM_ID+" = ?",
                              new String[]{Long.toString(id)}, null, null,
null);

                  if(c.moveToFirst()){
                        int iAlarmID = c.getColumnIndex(KEY_ALARM_ID);
                        int iAlarmTime = c.getColumnIndex(KEY_ALARM_TIME);
                        int iAlarmEndDate =
c.getColumnIndex(KEY_ALARM_END_DATE);

                        alarm = new POJOAlarm(c.getLong(iAlarmID),

      c.getInt(iAlarmTime),c.getInt(iAlarmEndDate));

                        return alarm;
                  }
                  else{return null;}
            }
```

```
	public ArrayList<POJOAlarm> getAlarms(int time, int correntDate){

		String[] columns = new String[]{
				KEY_ALARM_ID,KEY_ALARM_TIME, KEY_ALARM_END_DATE};

		// get Alarms with time and date higher than current.
		Cursor c = myDatabase.query(DATABASE_TABLE_ALARMS,
				columns, KEY_ALARM_TIME+" >= ? AND "+
KEY_ALARM_END_DATE+" >= ?",
				new String[]{Integer.toString(time),
Integer.toString(correntDate)}, null, null, KEY_ALARM_TIME);



			if(c.moveToFirst()){
				ArrayList<POJOAlarm> alarms =  new
ArrayList<POJOAlarm>();

				int iAlarmID = c.getColumnIndex(KEY_ALARM_ID);
				int iAlarmTime =
c.getColumnIndex(KEY_ALARM_TIME);
				int iAlarmEndDate =
c.getColumnIndex(KEY_ALARM_END_DATE);



				for(c.moveToFirst();
!c.isAfterLast();c.moveToNext()){
					alarms.add(new
POJOAlarm(c.getLong(iAlarmID),

	c.getInt(iAlarmTime),c.getInt(iAlarmEndDate)));
				}

				return alarms;
		}
		else{return null;}
	}




	public long insertAlarm(POJOAlarm alarm){
		ContentValues cv = new ContentValues();
		cv.put(KEY_ALARM_ID, alarm.getAlarmID());
		cv.put(KEY_ALARM_TIME, alarm.getTime());
		cv.put(KEY_ALARM_END_DATE, alarm.getEndDate());


		return myDatabase.insert(DATABASE_TABLE_ALARMS, null, cv);
	}

	public int updateAlarm(POJOAlarm alarm){
		ContentValues cv = new ContentValues();
		cv.put(KEY_ALARM_ID, alarm.getAlarmID());
		cv.put(KEY_ALARM_TIME, alarm.getTime());
		cv.put(KEY_ALARM_END_DATE, alarm.getEndDate());
```

```
                return myDatabase.update(DATABASE_TABLE_ALARMS, cv,
KEY_ALARM_ID+" = ?",
                                new String[]{ String.valueOf(alarm.getAlarmID())
});
        }

        public int deleteAlarm(POJOAlarm alarm){
                return myDatabase.delete(DATABASE_TABLE_ALARMS,KEY_ALARM_ID+"
= ?",
                                new String[]{ String.valueOf(alarm.getAlarmID())
});
        }

        public int deleteOutdatedAlarms(int date){
                return
myDatabase.delete(DATABASE_TABLE_ALARMS,KEY_ALARM_END_DATE+" < ?",
                                new String[]{ String.valueOf(date) });
        }

        public int emptyAlarmTable(){
                return myDatabase.delete(DATABASE_TABLE_ALARMS, null, null);
        }



        //DbHelper
        // Our static class from creating and updating the Database.
        private static class DbHelper extends SQLiteOpenHelper{

                public DbHelper(Context context) {
                        //Creates the Databases and if the Version is higher
than
                        //the Version of the of the Database that's all ready
in Calls
                        //the upGrade.
                        super(context, DATABASE_NAME , null, DATABASE_VERSION);
                        // TODO Auto-generated constructor stub
                }

                @Override
                public void onCreate(SQLiteDatabase db) {
                        // TODO Auto-generated method stub
                        try{


                        db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_PATIENTS + " (" +
                                        KEY_PATIENT_ID + " INTEGER PRIMARY KEY, "
+
                                        KEY_PATIENT_FIRST_NAME + " TEXT, " +
                                        KEY_PATIENT_LAST_NAME + " TEXT, " +
                                        KEY_PATIENT_LAST_KNOWN_ADDRESS + " TEXT,
"+
                                        KEY_PATIENT_FK_DOCTOR_ID + " INTEGER);"
                        );
```

```
                        db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_DOCTORS + " (" +
                                        KEY_DOCTOR_ID + " INTEGER PRIMARY KEY, " +
                                        KEY_DOCTOR_FIRST_NAME + " TEXT, " +
                                        KEY_DOCTOR_LAST_NAME + " TEXT);"
                        );

                        db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_EVENTS + " (" +
                                        KEY_EVENT_ID + " INTEGER PRIMARY KEY, " +
                                        KEY_EVENT_FK_PATIENT_ID + " INTEGER," +
                                        KEY_EVENT_FK_DOCTOR_ID + " INTEGER," +
                                        KEY_EVENT_TITLE+ " TEXT, " +
                                        KEY_EVENT_DATE + " INTEGER, " +
                                        KEY_EVENT_NOTE + " TEXT, " +
                                        KEY_EVENT_TYPE + " TEXT);"
                        );

                        db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_MEDICATIONS + " (" +
                                        KEY_MEDICATION_ID + " INTEGER PRIMARY KEY,
" +
                                        KEY_MEDICATION_END_DATE + " INTEGER, " +
                                        KEY_MEDICATION_DOSAGE + " INTEGER);"
                        );

                        db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_MEASURES + " (" +
                                        KEY_MEASURE_ID + " INTEGER PRIMARY KEY, "
+
                                        KEY_MEASURE_TYPE + " INTEGER, " +
                                        KEY_MEASURE_DATA_STORE + " BLOB);"
                        );


                        db.execSQL("CREATE TABLE IF NOT EXISTS " +
DATABASE_TABLE_ALARMS + " (" +
                                        KEY_ALARM_ID + " INTEGER PRIMARY KEY, " +
                                        KEY_ALARM_TIME + " INTEGER, " +
                                        KEY_ALARM_END_DATE + " INTEGER);"

                        );

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

                @Override
                public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
                        // TODO Auto-generated method stub

                        //Drops the all tables and make new ones.
                        //In case we want to change things in old Database
Change this method.

                        db.execSQL("DROP TABLE IF EXISTS
"+DATABASE_TABLE_DOCTORS);
```

```
                        db.execSQL("DROP TABLE IF EXISTS
"+DATABASE_TABLE_EVENTS);
                        db.execSQL("DROP TABLE IF EXISTS
"+DATABASE_TABLE_MEDICATIONS);
                        db.execSQL("DROP TABLE IF EXISTS
"+DATABASE_TABLE_PATIENTS);
                        db.execSQL("DROP TABLE IF EXISTS
"+DATABASE_TABLE_MEASURES);
                        db.execSQL("DROP TABLE IF EXISTS
"+DATABASE_TABLE_ALARMS);
                        onCreate(db);
                    }
            }
            }
```

POJOAlarm.java

```
package com.chkyriacou.items;

/**
 *
 * Is used only for setting the alarms (see AlarmService)
 *
 */
public class POJOAlarm {
	private long alarmID;
	private int time;
	private int endDate;



	public POJOAlarm(){
		this.alarmID=-1;
		this.time=-1;
		this.endDate=-1;

	}


	public POJOAlarm(long alarmID, int time, int endDate) {
		super();
		this.alarmID = alarmID;
		this.time=time;
		this.endDate = endDate;

	}

	public long getAlarmID() {
		return alarmID;
	}
	public void setAlarmID(long alarmID) {
		this.alarmID = alarmID;
	}
```

```
	public int getTime() {
		return time;
	}


	public void setTime(int time) {
		this.time = time;
	}


	public int getEndDate() {
		return endDate;
	}
	public void setEndDate(int endDate) {
		this.endDate = endDate;
	}



}
```

POJODoctor.java

```
package com.chkyriacou.items;

public class POJODoctor {
	private long doctorID;
	private String firstName;
	private String lastName;


	public POJODoctor() {
		// TODO Auto-generated constructor stub
		this.doctorID = -1;
		this.firstName = null;
		this.lastName = null;
	}


	public POJODoctor(long doctorID, String firstName, String lastName) {
		this.doctorID = doctorID;
		this.firstName = firstName;
		this.lastName = lastName;
	}


	public long getDoctorID() {
		return doctorID;
	}


	public void setDoctorID(long doctorID) {
		this.doctorID = doctorID;
	}
```

```
	public String getFirstName() {
		return firstName;
	}


	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}


	public String getLastName() {
		return lastName;
	}


	public void setLastName(String lastNakme) {
		this.lastName = lastNakme;
	}
}
```

POJOEvent.java

```
package com.chkyriacou.items;



public class POJOEvent{

	private long eventID;
	private long patientID;
	private long doctorID;
	private String title;
	private int date;
	private String note;
	private String type;


	public POJOEvent() {
		this.eventID = -1;
		this.patientID=-1;
		this.doctorID=-1;
		this.title = null;
		this.date = -1;
		this.note = null;
		this.type = null;
	}


	public POJOEvent( long eventID, long patientID, long doctorID, String title
,int date, String note, String type) {
		this.eventID = eventID;
		this.patientID=patientID;
		this.doctorID=doctorID;
		this.title = title;
		this.date = date;
		this.note = note;
```

```
		this.type = type;
	}

	public long getPatientID() {
		return patientID;
	}



	public void setPatientID(long patientID) {
		this.patientID = patientID;
	}



	public long getDoctorID() {
		return doctorID;
	}



	public void setDoctorID(long doctorID) {
		this.doctorID = doctorID;
	}



	public void setTitle(String title) {
		this.title = title;
	}


	public String getTitle() {
		return title;
	}
	public void setText(String title) {
		this.title = title;
	}
	public long getEventID() {
		return eventID;
	}
	public void setEventID(long eventID) {
		this.eventID = eventID;
	}
	public int getDate() {
		return date;
	}
	public void setDate(int date) {
		this.date = date;
	}
	public String getNote() {
		return note;
	}
	public void setNote(String note) {
		this.note = note;
	}
	public String getType() {
		return type;
	}
```

```
	public void setType(String type) {
		this.type = type;
	}


	public String getFormedDate(){
		String date= String.valueOf(getDate());
		date = date.substring(6,
8)+"/"+date.substring(4,6)+"/"+date.substring(0, 4);

		return date;
	}

	public int getYear(){
		return getDate()/10000;
	}

	public int getDay(){
		return getDate()%100;
	}

	public int getMonth(){
		return (getDate()%10000)/100;
	}

}
```

POJOMeasure.java

```
package com.chkyriacou.items;

public class POJOMeasure {

	private long measureID;
	private int type;
	private byte[] data;


	public POJOMeasure() {
		this.measureID = -1;
		this.data = null;
	}

	public POJOMeasure(long measureID, int type){
		this.measureID = measureID;
		this.type = type;
	}

	public POJOMeasure(long measureID, int type, byte[] data) {
		this.measureID = measureID;
		this.type = type;
		this.data = data;
	}

	public long getMeasureID() {
		return measureID;
```

```
	}

	public void setMeasureID(long measureID) {
		this.measureID = measureID;
	}

	public byte[] getData() {
		return data;
	}

	public void setData(byte[] data) {
		this.data = data;
	}

	public void setType(int type) {
		this.type = type;
	}

	public int getType() {
		return type;
	}


}
```

POJOMediaction.java

```
package com.chkyriacou.items;


public class POJOMedication {
	long medicationID;
	int endDate;
	int dosage;


	public POJOMedication() {
		this.medicationID = -1;
		this.endDate = -1;
		this.dosage = -1;
	}

	public POJOMedication(long medicationID, int endDate, int dosage) {
		this.medicationID = medicationID;
		this.endDate = endDate;
		this.dosage = dosage;

	}

	public long getMedicationID() {
		return medicationID;
	}

	public void setMedicationID(long medicationID) {
```

```
		this.medicationID = medicationID;
	}

	public int getEndDate() {
		return endDate;
	}

	public void setEndDate(int endDate) {
		this.endDate = endDate;
	}

	public int getDosage() {
		return dosage;
	}

	public void setDosage(int dosage) {
		this.dosage = dosage;
	}



	public String getFormedEndDate(){
		String date= String.valueOf(getEndDate());
		date = date.substring(6,
8)+"/"+date.substring(4,6)+"/"+date.substring(0, 4);

		return date;
	}

	public int getEndYear(){
		String date= String.valueOf(getEndDate());
		return Integer.decode(date.substring(0, 4));
	}

	public int getEndDay(){
		String date= String.valueOf(getEndDate());
		return Integer.decode(date.substring(6, 8));
	}

	public int getEndMonth(){
		String date= String.valueOf(getEndDate());
		return Integer.decode(date.substring(4, 6));
	}

}
```

POJOPatient.java

```
package com.chkyriacou.items;

public class POJOPatient {
	private long patientID;
	private String firstName;
	private String lastName;
	private String lastKnownLocation;
	private long doctorID;
```

```
	public POJOPatient() {
		this.patientID = -1;
		this.firstName = null;
		this.lastName = null;
		this.lastKnownLocation = null;
		this.doctorID = -1;
	}

	public POJOPatient(long patientID, String firstName, String lastName,
			String lastKnownLocation, long doctorID) {
		this.patientID = patientID;
		this.firstName = firstName;
		this.lastName = lastName;
		this.lastKnownLocation = lastKnownLocation;
		this.doctorID=doctorID;
	}

	public long getPatientID() {
		return patientID;
	}

	public void setPatientID(long patientID) {
		this.patientID = patientID;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getLastKnownLocation() {
		return lastKnownLocation;
	}

	public void setLastKnownLocation(String lastKnownLocation) {
		this.lastKnownLocation = lastKnownLocation;
	}

	public void setDoctorID(long docotorID){
		this.doctorID=docotorID;
	}

	public long getDocotorID(){
		return doctorID;
	}

}
```

ProjectDoc src com.chkyriacou.services

AlarmService.java

```
package com.chkyriacou.services;

import java.util.ArrayList;
import java.util.Calendar;

import com.chkyriacou.activities.Schedule;
import com.chkyriacou.items.AlarmList;
import com.chkyriacou.items.POJOAlarm;
import com.chkyriacou.items.PatientDBAdapter;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.app.Service;

import android.content.Context;
import android.content.Intent;

import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.SystemClock;

/**
 *
 * Checks and Sets the next Alarm
 * Using an Handler to handle the Message from
 * EventScreen Result in Medication Case.
 *
 *
 *
 */

public class AlarmService extends Service {
private static PatientDBAdapter db;
public static final int MSG_ADD_ALARMS = 101;
public static final int MSG_REMOVE_ALARMS = 102;

private static AlarmList alarms;
private AlarmManager myAlarmManager;


private static class IncomingHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {

            case MSG_ADD_ALARMS:
```

```
            //Taking the alarms from the message
            //and insert them the table.
             alarms = (AlarmList) msg.obj;

             for(POJOAlarm alarm : alarms){

                    //update and if not, insert
                    if( db.updateAlarm(alarm) == 0){
                           db.insertAlarm(alarm);

                    }
             }

             break;

            case MSG_REMOVE_ALARMS:
             alarms = (AlarmList) msg.obj;
             for(POJOAlarm alarm : alarms){
                    //delete the specific Alarms
                    db.deleteAlarm(alarm);
             }

                break;


            default:
                super.handleMessage(msg);
        }
    }
}


final Messenger mMessenger = new Messenger(new IncomingHandler());

       @Override
       public void onCreate() {
              // TODO Auto-generated method stub
              super.onCreate();
              db = new PatientDBAdapter(this);
              db.open();

              //Gets the alarm manager from System
              myAlarmManager =
(AlarmManager)getSystemService(Context.ALARM_SERVICE);

              //Using the Calender to delete out of date alarms

              int date = 0;
              Calendar cal = Calendar.getInstance();
              date = cal.get(Calendar.DAY_OF_MONTH);
              date = date + ((cal.get(Calendar.MONTH)+1)*100);
              date = date + (cal.get(Calendar.YEAR)*10000);

              db.deleteOutdatedAlarms(date);

              alarms=null;

       }
```

```
@Override
public void onDestroy() {
	// TODO Auto-generated method stub
	super.onDestroy();
	db.close();
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
	// TODO Auto-generated method stub

	setNextAlarm();


	//START_STICKY means that is will be called each time someone
	//calls the startService with this Service.
	return START_STICKY;
}

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

	//Return the Messenger
	 return mMessenger.getBinder();
}



@Override
public void onRebind(Intent intent) {
	// TODO Auto-generated method stub
	super.onRebind(intent);

	//When EventScreen (Medication Case)
	//is resume two times, it will be called.
	//Perfect to go with EventScreen onResultActivity.
	//The onResultActivity happens BEFORE onResume

	setNextAlarm();
}

@Override
public boolean onUnbind(Intent intent) {
	// TODO Auto-generated method stub

	//return true means that it will
	//use the onRebind
	return true;
}


//in setNextAlarm

private void setNextAlarm(){
	ArrayList<POJOAlarm> alarms = null;
```

```
//using the Calender to get the Alarms we need
//using the time and date (correntCal)
int correntCal = 0;
Calendar cal = Calendar.getInstance();
correntCal = cal.get(Calendar.DAY_OF_MONTH);
correntCal = correntCal + ((cal.get(Calendar.MONTH)+1)*100);
correntCal = correntCal + (cal.get(Calendar.YEAR)*10000);


int calMin = cal.get(Calendar.MINUTE);
int calHour = cal.get(Calendar.HOUR_OF_DAY);
if(calHour==24)calHour=0;
int calSec = cal.get(Calendar.SECOND);
int time= (calHour*10000)+(calMin*100)+calSec;

alarms = db.getAlarms(time, correntCal);


if(alarms!=null){

int hour = alarms.get(0).getTime()/10000;
int min = (alarms.get(0).getTime()/100)%100;
int sec = alarms.get(0).getTime()%100;



min=min-calMin;
sec=sec-calSec;
hour=hour-calHour;


//Finding the Hours Mins and Secs between
//our time and the alarm time.

if(sec<0){
      sec=60+sec;
      min= min - 1;
}

if(min<0){
      min=60+min;
      hour=hour-1;
}

if(hour<0){
      hour=24+hour;
}


//Creating the Intent that will call the wanted
//EventScreen.

Intent data = new Intent(this, Schedule.class);

data.putExtra("ring", true);

//is an Intent that will call an other Intent(data)
//Allowing the AlarmManager to start our Intent(data)
```

```
            PendingIntent pi = PendingIntent.getActivity(this, 0, data, 0);

            //Giving 1 sec plus (+ 1000 to ensure that will not report
            //it self more times.

            time = 3600000*hour +60000*min+ 1000*sec + 1000;

            //Using the Elepsed real time to will wake up the device and
            //it will go off that time we tell it.
            myAlarmManager.set(AlarmManager.ELAPSED_REALTIME_WAKEUP,
SystemClock.elapsedRealtime() + time, pi);

            }else{
                  //if alarms is empty that means no Entries in the Alarms
Table.
                  //using the same to make
                  PendingIntent pi = PendingIntent.getActivity(this, 0, new
Intent(this, Schedule.class), 0);

                  //It will cancel the Alarm with the same Intent.
                  myAlarmManager.cancel(pi);
            }
      }
      }
```

ProjectDoc src com.chkyriacou.activities

AlarmSetup.java

```
package com.chkyriacou.activities;


import java.util.ArrayList;

import com.chkyriacou.R;
import com.chkyriacou.items.PatientDBAdapter;


import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;
```

```
/**
 *
 * An Screen to setup the Alarms, is called when in EventScreen Medication Case.
 *
 * Saves old setups from the timePickers using SharedPreferences
 *
 * Return Mins and Hours us a resualt when Button Done is pressed
 * If Cansel is pressed then EventScreen will remove those Alarms from
AlarmService
 *
       */

public class AlarmSetup extends Activity {


      private SharedPreferences mySharedPreferences;
      private SharedPreferences.Editor editor;

      private TimePicker tpOne, tpTwo, tpThree,tpFour, tpFive, tpSix;
      private ArrayList<TimePicker> tps;

      private TextView tvOne, tvTwo, tvThree, tvFour, tvFive, tvSix;
      private ArrayList<TextView> tvs;

      private Button btDone, btCancel;

      private int dosage;

      private int[] mins, hours;


      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub

            super.onCreate(savedInstanceState);
            setContentView(R.layout.alarm_setup);

            Intent data = this.getIntent();
             mins = new int [6];
             hours = new int [6];

            dosage = data.getIntExtra(PatientDBAdapter.KEY_MEDICATION_DOSAGE,
0);
            tps = new ArrayList<TimePicker>();

            tpOne = (TimePicker) findViewById(R.id.tp_alarm_setup_1);
            tpTwo = (TimePicker) findViewById(R.id.tp_alarm_setup_2);
            tpThree = (TimePicker) findViewById(R.id.tp_alarm_setup_3);
            tpFour = (TimePicker) findViewById(R.id.tp_alarm_setup_4);
            tpFive = (TimePicker) findViewById(R.id.tp_alarm_setup_5);
            tpSix = (TimePicker) findViewById(R.id.tp_alarm_setup_6);

            tps.add(tpOne);
            tps.add(tpTwo);
            tps.add(tpThree);
```

```
tps.add(tpFour);
tps.add(tpFive);
tps.add(tpSix);

tvs = new ArrayList<TextView>();
tvOne = (TextView) findViewById(R.id.tv_alarm_setup_1);
tvTwo = (TextView) findViewById(R.id.tv_alarm_setup_2);
tvThree = (TextView) findViewById(R.id.tv_alarm_setup_3);
tvFour = (TextView) findViewById(R.id.tv_alarm_setup_4);
tvFive = (TextView) findViewById(R.id.tv_alarm_setup_5);
tvSix = (TextView) findViewById(R.id.tv_alarm_setup_6);

tvs.add(tvOne);
tvs.add(tvTwo);
tvs.add(tvThree);
tvs.add(tvFour);
tvs.add(tvFive);
tvs.add(tvSix);


//Set Visible only the Views we need.
for(int i=0;i<dosage;i++){
	tvs.get(i).setVisibility(View.VISIBLE);
	tps.get(i).setVisibility(View.VISIBLE);
	tps.get(i).setIs24HourView(true);
}

btDone = (Button) findViewById(R.id.bt_alarm_setup_done);
btDone.setOnClickListener(new OnClickListener() {

	public void onClick(View v) {
		// TODO Auto-generated method stub

		//Get the Values and set -1 to the other Mins and Hours
		for(int i=0;i<6;i++){
			if(i<dosage){
			mins[i] = tps.get(i).getCurrentMinute();
			hours[i] = tps.get(i).getCurrentHour();
			}else{
				mins[i] = -1;
				hours[i] = -1;
			}
		}

		Intent data = getIntent();
		save();
		data.putExtra("tp_mins", mins);
		data.putExtra("tp_hours", hours);
		setResult(RESULT_OK, data);
		finish();
	}
});

btCancel = (Button) findViewById(R.id.bt_alarm_setup_cancel);
btCancel.setOnClickListener(new OnClickListener() {

	public void onClick(View v) {
		// TODO Auto-generated method stub
```

```
                        setResult(RESULT_CANCELED);
                        Toast.makeText(getApplicationContext(), "All Alarms
have been canceled", Toast.LENGTH_LONG).show();
                        finish();
                  }
            });

            mySharedPreferences =

      getSharedPreferences("com.chkyriacoy.projectdoc",Context.MODE_PRIVATE);

            editor = mySharedPreferences.edit();

      }

      @Override
      protected void onPause() {
            // TODO Auto-generated method stub
            super.onPause();
            this.finish();

      }


      // onResume Get the last used Number user had set before.
      @Override
      protected void onResume() {
            // TODO Auto-generated method stub
            super.onResume();

            mins[0] = mySharedPreferences.getInt("tp_min_one", 0);
            mins[1] = mySharedPreferences.getInt("tp_min_two", 0);
            mins[2] = mySharedPreferences.getInt("tp_min_three", 0);
            mins[3] = mySharedPreferences.getInt("tp_min_four", 0);
            mins[4] = mySharedPreferences.getInt("tp_min_five", 0);
            mins[5] = mySharedPreferences.getInt("tp_min_six", 0);


            hours[0] = mySharedPreferences.getInt("tp_hour_one", 0);
            hours[1] = mySharedPreferences.getInt("tp_hour_two", 0);
            hours[2] = mySharedPreferences.getInt("tp_hour_three", 0);
            hours[3] = mySharedPreferences.getInt("tp_hour_four", 0);
            hours[4] = mySharedPreferences.getInt("tp_hour_five", 0);
            hours[5] = mySharedPreferences.getInt("tp_hour_six", 0);

            for(int i=0;i<dosage;i++){


                  if(mins[i]!=-1&&hours[i]!=-1){
                  tps.get(i).setCurrentMinute(mins[i]);
                  tps.get(i).setCurrentHour(hours[i]);
                  }
            }

      }

      //Put all values from timePickers in to SharedPreferences
      //if the value is -1 don't put it in.
      private void save(){
```

```
            if(mins[0]!=-1&&hours[0]!=-1){
                  editor.putInt("tp_min_one", mins[0]);
                  editor.putInt("tp_hour_one", hours[0]);

            }
            if(mins[1]!=-1&&hours[1]!=-1){
                  editor.putInt("tp_min_two", mins[1]);
                  editor.putInt("tp_hour_two", hours[1]);
            }
            if(mins[2]!=-1&&hours[2]!=-1){
                  editor.putInt("tp_min_three", mins[2]);
                  editor.putInt("tp_hour_three", hours[2]);
            }
            if(mins[3]!=-1&&hours[3]!=-1){
                  editor.putInt("tp_min_four", mins[3]);
                  editor.putInt("tp_hour_four", hours[3]);
            }
            if(mins[4]!=-1&&hours[4]!=-1){
                  editor.putInt("tp_min_five", mins[4]);
                  editor.putInt("tp_hour_five", hours[4]);
            }
            if(mins[5]!=-1&&hours[5]!=-1){
                  editor.putInt("tp_min_six", mins[5]);
                  editor.putInt("tp_hour_six", hours[5]);
            }


            editor.commit();

            Toast.makeText(getApplicationContext(), "The Alarms has been set",
Toast.LENGTH_LONG).show();
      }
}
```

BluetoothScreen.java

```
package com.chkyriacou.activities;

import java.util.ArrayList;
import java.util.Set;
import com.chkyriacou.R;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
```

```
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;


/**
 *
 * Not finished Activity, needs Android 4 and above.
 *
 * Asking user to pair with the Bluetooth Health Device.
 * In result shows all paired Diveces.
 *
 */


public class BluetoothScreen extends Activity {

	private BluetoothAdapter bluetoothAdapter;
	private Button btbutton;
	private ArrayList<BluetoothDevice> arrayOfDevices = new
ArrayList<BluetoothDevice>();
	private ArrayAdapter<BluetoothDevice> devicesAdapter;
	private ListView listView;


	private BluetoothDevice wantedPairedDevice;



	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);

	  setContentView(R.layout.bluetooth_view);



		listView = (ListView)findViewById(R.id.lv_devices);
		arrayOfDevices = new ArrayList<BluetoothDevice>();
		devicesAdapter = new DeviceAdapter();
		listView.setAdapter(devicesAdapter);

		listView.setOnItemClickListener(new OnItemClickListener() {

			public void onItemClick(AdapterView<?> parent, View view, int
position,
					long id) {
				// TODO Auto-generated method stub
				wantedPairedDevice = arrayOfDevices.get(position);
				Toast.makeText(getApplicationContext(),
wantedPairedDevice.getName(), Toast.LENGTH_LONG).show();
			}
		});

		Toast.makeText(getApplicationContext(),"Press the Button to pair
your devices.", Toast.LENGTH_LONG).show();
```

```
            btbutton = (Button) findViewById(R.id.bt_turnbton);
        btbutton.setOnClickListener(new OnClickListener() {

                    public void onClick(View v) {
                          // TODO Auto-generated method stub
                          Toast.makeText(getApplicationContext(),"Make sure your
Bluetooth is on, then scan and pair your device", Toast.LENGTH_LONG).show();

                          Intent intentBluetooth = new Intent();

intentBluetooth.setAction(android.provider.Settings.ACTION_BLUETOOTH_SETTINGS);
                        startActivity(intentBluetooth);
                    }
            });



      }


      @Override
      protected void onStart() {
            // TODO Auto-generated method stub
            super.onStart();

            //Check if there the Device supports Bluetooth.
            //If not it Exit.
            bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (bluetoothAdapter == null) {
            Toast.makeText(this, "Bluetooth is not supported",
Toast.LENGTH_LONG).show();
            finish();
            return;
        }
      }


      @Override
      protected void onResume() {
            // TODO Auto-generated method stub
            super.onResume();

            if(!arrayOfDevices.isEmpty()){
                  arrayOfDevices.clear();
            }

            Set<BluetoothDevice> pairedDevices =
bluetoothAdapter.getBondedDevices();
            for (BluetoothDevice pairedDevice : pairedDevices){
                  arrayOfDevices.add(pairedDevice);
            }
            devicesAdapter.notifyDataSetChanged();
      }



      @Override
      protected void onStop() {
            // TODO Auto-generated method stub
```

```
		super.onStop();

	}


	@Override
	protected void onDestroy() {
		// TODO Auto-generated method stub
		super.onDestroy();

	}




	class DeviceAdapter extends ArrayAdapter<BluetoothDevice>{

		DeviceAdapter(){
			super(BluetoothScreen.this,
android.R.layout.simple_list_item_1, arrayOfDevices);
		}

		@Override
		public View getView(int position, View convertView, ViewGroup
parent) {
			// TODO Auto-generated method stub
			ViewHolder holder;

			if(convertView == null){
				LayoutInflater inflater = getLayoutInflater();

	convertView=inflater.inflate(R.layout.bluetooth_listitem, null);
				holder = new ViewHolder(convertView);
				convertView.setTag(holder);
			}else{
				holder=(ViewHolder) convertView.getTag();
			}
			holder.populateFrom(arrayOfDevices.get(position));
			return(convertView);
		}
	}

	class ViewHolder{

		public TextView name = null;
		public TextView address = null;

	ViewHolder(View row) {
		name = (TextView)row.findViewById(R.id.tv_blue_name);
		address = (TextView)row.findViewById(R.id.tv_blue_address);
	}

	void populateFrom(BluetoothDevice device){
		name.setText(device.getName());
		address.setText(device.getAddress());
	}
}
}
```

EventScreen.java

```
package com.chkyriacou.activities;


import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.SocketAddress;
import java.net.UnknownHostException;

import com.chkyriacou.R;
import com.chkyriacou.items.AlarmList;
import com.chkyriacou.items.POJOAlarm;
import com.chkyriacou.items.POJOEvent;
import com.chkyriacou.items.POJOMeasure;
import com.chkyriacou.items.POJOMedication;
import com.chkyriacou.items.PatientDBAdapter;
import com.chkyriacou.services.AlarmService;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;

import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.ServiceConnection;

import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;



/**
 *
 * EventScreen use a Single layout for all three Type of Events
 *
```

```
 * Type Note :
 * Just show: title, data and Notes, no need to look up the Database
 *
 * Type Medication :
 * Show both the values in the Events and find the Medication using
 * the Event's ID. Offers the Setup Alarms Button that leads to
 * AlarmSetup Activity and gets an Result from it. Then using the
 * alarmMessenger Sends messenges to AlarmService. It also Binds
 * with AlarmService.
 *
 * Type Measure :
 * Show both the values in the Events and find the Meausure using
 * the Event's ID. Offers a stanger Button that leads to
 * BluetoothScreen and if there are any Data in that Measure,
 * the btSend will be appear to send them using a Socket.
 *
 *
 *
 */

public class EventScreen extends Activity {

	private final int BLUETOOTH_CODE = 1001;
	private final int ALARM_SETUP_CODE = 1002;

	private	PatientDBAdapter db;
	private TextView tvTitle, tvDate, tvEndDate, tvDosageLabel, tvDosageVolume,
tvNote;
	private Button btButton,btSend;
	private ImageView ivIcon;
	private long eventID;

	private POJOEvent event;
	private POJOMedication medication;
	private POJOMeasure measure;

	private ProgressDialog loadingDialog;

	private Messenger alarmMessenger;



	private ServiceConnection mConnection = new ServiceConnection() {

		public void onServiceDisconnected(ComponentName name) {
			// TODO Auto-generated method stub
			alarmMessenger = null;

		}

		public void onServiceConnected(ComponentName name, IBinder service)
{
			// TODO Auto-generated method stub
			alarmMessenger = new Messenger(service);
		}
	};
```

```
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
	    setContentView(R.layout.event_screen);
	    db = new PatientDBAdapter(this);
		Intent intent = this.getIntent();
		eventID = intent.getLongExtra(PatientDBAdapter.KEY_EVENT_ID, -1);
	    db.open();
	    setupIU();

	}



		@Override
	protected void onResume() {
		// TODO Auto-generated method stub
		super.onResume();

		//Bind this AlarmService Only when it's Medication.

	if(event.getType().equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){


			bindService(new Intent(this, AlarmService.class),
					mConnection, Context.BIND_AUTO_CREATE);
			}
		}



		@Override
	protected void onPause() {
		// TODO Auto-generated method stub
		super.onPause();

		//unBind

	if(event.getType().equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){


		unbindService(mConnection);
		}
	}




		@Override
	protected void onDestroy() {
		// TODO Auto-generated method stub
		super.onDestroy();
		db.close();
	}


//Changes the Layout based on the Event Type.
```

```
            private void setupIU(){


            tvTitle = (TextView) findViewById(R.id.tv_event_title);
            tvDate = (TextView) findViewById(R.id.tv_event_date);
            tvEndDate = (TextView) findViewById(R.id.tv_event_enddate);
            tvDosageLabel = (TextView) findViewById(R.id.tv_event_dosage_label);
            tvDosageVolume = (TextView)
findViewById(R.id.tv_event_dosage_volume);
            tvNote = (TextView) findViewById(R.id.tv_event_note);
            btButton = (Button) findViewById(R.id.bt_event_button);
            btSend = (Button) findViewById(R.id.bt_event_send);
            ivIcon = (ImageView) findViewById(R.id.iv_event_icon);



            //Getting the Event Type.
            Intent intent = this.getIntent();
            eventID = intent.getLongExtra(PatientDBAdapter.KEY_EVENT_ID, -1);
            event = db.getEvent(eventID);

            tvTitle.setText(event.getTitle());
            tvTitle.setMovementMethod(new ScrollingMovementMethod());
            tvDate.setText(event.getFormedDate());
            tvNote.setText(event.getNote());
            tvNote.setMovementMethod(new ScrollingMovementMethod());

            String type = event.getType();

            if(type.equals(PatientDBAdapter.DATABASE_TABLE_EVENTS)){

                  tvEndDate.setText("");
                  tvDosageLabel.setText("");
                  tvDosageVolume.setText("");
                  btButton.setVisibility(Button.GONE);
                  btSend.setVisibility(Button.GONE);

      ivIcon.setImageDrawable(getResources().getDrawable(R.drawable.note));


            }else if(type.equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){

                  medication = db.getMedication(eventID);

                  tvEndDate.setText(medication.getFormedEndDate());
                  tvDosageLabel.setText("Dosage: ");


      tvDosageVolume.setText(String.valueOf(medication.getDosage()));
                  btButton.setVisibility(Button.VISIBLE);
                  btSend.setVisibility(Button.GONE);
                  btButton.setText("Setup Alarms");


      ivIcon.setImageDrawable(getResources().getDrawable(R.drawable.drug));

                  btButton.setOnClickListener( new OnClickListener() {

                        public void onClick(View v) {
```

```
                                    // TODO Auto-generated method stub
                                    Intent data = new Intent(EventScreen.this,
AlarmSetup.class);


      data.putExtra(PatientDBAdapter.KEY_MEDICATION_DOSAGE,
medication.getDosage());
                                    startActivityForResult(data, ALARM_SETUP_CODE);

                              }
                        });

                  }else if(type.equals(PatientDBAdapter.DATABASE_TABLE_MEASURES)){


                        measure = db.getMeasure(eventID);

                        tvEndDate.setText("");
                        tvDosageLabel.setText("");
                        tvDosageVolume.setText("");
                        btButton.setVisibility(Button.VISIBLE);
                        btSend.setVisibility(Button.VISIBLE);
                        btButton.setText("Get Pressure");

      ivIcon.setImageDrawable(getResources().getDrawable(R.drawable.blutooth));

                        if(measure.getData()==null){
                              btSend.setVisibility(Button.GONE);
                        }else{
                              btSend.setVisibility(Button.VISIBLE);
                        }

                        btButton.setOnClickListener( new OnClickListener() {

                              public void onClick(View v) {
                                    Intent intent = new
Intent(EventScreen.this,BluetoothScreen.class);
                                    intent.putExtra(PatientDBAdapter.KEY_MEASURE_ID,
measure.getMeasureID());
                                    startActivityForResult(intent, BLUETOOTH_CODE);

                              }
                        });


                        btSend.setOnClickListener( new OnClickListener() {

                              public void onClick(View v) {

                                  ConnectivityManager connMgr = (ConnectivityManager)
                                getSystemService(Context.CONNECTIVITY_SERVICE);
                                    NetworkInfo networkInfo =
connMgr.getActiveNetworkInfo();
                                    if (networkInfo != null){
                                    if(networkInfo.isConnected()) {

                                    new LoadingTask().execute();

                                    }else{
```

```
                                        Toast.makeText(getApplicationContext(),
"Your device is not connected to the internet", Toast.LENGTH_LONG).show();
                                    }
                                }else{
                                    Toast.makeText(getApplicationContext(), "Your
device is not connected to the internet", Toast.LENGTH_LONG).show();
                                }
                            }
                        });
                }
        }


//When return from AlarmSetup Activity.
        @Override
        protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
                        // TODO Auto-generated method stub
                        super.onActivityResult(requestCode, resultCode, data);

                        switch(requestCode){
                        case ALARM_SETUP_CODE:
                                if(resultCode==RESULT_OK){

                                // Get mins and hours from the Intent
                                // the same Intent AlarmSetup sent.
                                int[]  mins = data.getIntArrayExtra("tp_mins");

                                int[]  hours = data.getIntArrayExtra("tp_hours");

                                        AlarmList alarms = new AlarmList();
                                        int sec=0;
                                        for(int i=0;i<6;i++){

                                                //fill the alarms (ArrayLists) with
POJOAlarms
                                                if(hours[i]>-1&&mins[i]>-1){
                                                alarms.add( new
POJOAlarm((event.getEventID()*10)+1+i,(hours[i]*10000)+(mins[i]*100)+sec,medicatio
n.getEndDate()));
                                                }
                                        }


                                        try {

                                                //Try to send the alarms with a Messege
MSG_ADD_ALARMS
                                                alarmMessenger.send(
Message.obtain(null,AlarmService.MSG_ADD_ALARMS, alarms));
                                        } catch (RemoteException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }

                                }

                                if(resultCode==RESULT_FIRST_USER){
```

```
                                    AlarmList alarms = new AlarmList();

                                    for(int i=0;i<6;i++){
                                          alarms.add( new
POJOAlarm((event.getEventID()*10)+1+i,0,medication.getEndDate()));

                                    }


                                    try {

                                          //In case of Canceled we send
MSG_REMOVE_ALARMS to AlarmService.
                                          alarmMessenger.send(
Message.obtain(null,AlarmService.MSG_REMOVE_ALARMS, alarms));
                                    } catch (RemoteException e) {
                                          // TODO Auto-generated catch block
                                          e.printStackTrace();
                                    }
                              }
                              break;
                        case BLUETOOTH_CODE:

                              //In case the BluetoothScreen we check if there is any
Data
                              //If yes we change Visibility of the Button btSend
                              if(resultCode==RESULT_OK||resultCode==RESULT_CANCELED){
                                    measure= db.getMeasure(eventID);
                                    if(measure.getData()!=null){
                                          btSend.setVisibility(Button.VISIBLE);
                                    }else{
                                          btSend.setVisibility(Button.GONE);
                                    }
                              }


                        }
                  }

      //AsyncTask for the Socket
      private class LoadingTask extends AsyncTask<Void, Void, String>{

            @Override
            protected String doInBackground(Void... params) {
                  // TODO Auto-generated method stub

                  Socket socket = null;
                  InputStream ins = null;
                  OutputStream outs = null;
                  DataOutputStream dos = null;
                  DataInputStream dis = null;
                  String message = null;

                  String dstAddress = "www.sitehere.com";
            int dstPort = 0;


            try {
```

```
            SocketAddress remoteAddr = new InetSocketAddress(dstAddress,
dstPort);
                        socket = new Socket();

                        if(socket!=null){
                              socket.connect(remoteAddr, 5000);
            if(socket.isConnected()){


                              byte[] preData = measure.getData();

                              outs = socket.getOutputStream();
                              dos = new DataOutputStream(outs);

                              //Send the id in UTF-8
                              dos.writeUTF(
                                          String.valueOf(
                                          measure.getMeasureID()));

                              //Now sends the Data
                              dos.write(preData,0,preData.length);


                              ins = socket.getInputStream();
                              dis = new DataInputStream(ins);

                              message = dis.readUTF();

                        }
            }

            } catch (UnknownHostException e2) {
                              // TODO Auto-generated catch block
                              e2.printStackTrace();
                        } catch (IOException e2) {
                              // TODO Auto-generated catch block
                              e2.printStackTrace();
                        }

            try {
                              if(dos!=null)
                                    dos.close();
                              if(dis!=null)
                                    dis.close();
                              if(ins!=null)
                                    ins.close();
                              if(outs!=null)
                                    outs.close();
                              if(socket!=null)
                                    socket.close();

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


                  return message;
            }
```

```
		@Override
		protected void onPreExecute() {
			// TODO Auto-generated method stub
			super.onPreExecute();

		  loadingDialog = new ProgressDialog(EventScreen.this);
			loadingDialog.show();
			loadingDialog.setMessage("Send Data, Please Wait . . .");
		}

		@Override
		protected void onPostExecute(String result) {
			// TODO Auto-generated method stub
			super.onPostExecute(result);

			//After getting the result we Pop a AlartDialog
			//with what the server sent if not null.


			loadingDialog.dismiss();

			AlertDialog.Builder adBuilder = new
AlertDialog.Builder(EventScreen.this);
			adBuilder.setTitle("REPORT");

			if(result!=null){

				adBuilder.setMessage(result)
				.setCancelable(false)
				.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

					public void onClick(DialogInterface dialog, int
which) {

						// TODO Auto-generated method stub
						dialog.cancel();
					}
				});
				}else{
					adBuilder.setMessage("Server is not responding")
					.setCancelable(false)
					.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

						public void onClick(DialogInterface
dialog, int which) {

							// TODO Auto-generated method stub
							dialog.cancel();
						}
					});
				}
				AlertDialog alDialog = adBuilder.create();

				alDialog.show();
		}
	}
}
```

Intro.java

```
package com.chkyriacou.activities;

import com.chkyriacou.R;
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;

/**
 *  Timed Intro
 *  After 1 sec it calls the Login Activity and ends it self (finish())
 */

public class Intro extends Activity {


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.intro);
        Thread introTimer = new Thread (){

             public void run(){

                   try{
                          short countTimer = 0;
                                 while(countTimer<1000){
                                        sleep(100);
                                        countTimer = (short) (countTimer +100);
                                 }

                                 startActivity(new
Intent(Intro.this,Login.class));
                   }
                   catch(InterruptedException e){

                   }finally{

                          finish();
                   }
             }
         };

       introTimer.start();
    }
      }
```

Login.java

```
package com.chkyriacou.activities;
```

```
import java.io.IOException;
import java.util.ArrayList;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.json.JSONException;
import org.json.JSONObject;

import com.chkyriacou.R;
import com.chkyriacou.items.JSONHelper;
import com.chkyriacou.items.POJODoctor;
import com.chkyriacou.items.POJOEvent;
import com.chkyriacou.items.POJOMeasure;
import com.chkyriacou.items.POJOMedication;
import com.chkyriacou.items.POJOPatient;
import com.chkyriacou.items.PatientDBAdapter;



import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

/**
 * The patient types hes ID. If there is a connection to the internet,
 * then it will send a request. If not Massage will appear and tell him
 * to try the Off-line mode or the Demo.
 *
 * SharedPreferences holds the ID he types, holding always the last ID.
 *
 * On every No-OffLine login the Database will be refreshed with new
 * entries. Entries that are not updated or inserted will be deleted,
 * ending up with the same Database the server have.
 *
 */

public class Login extends Activity {

	private SharedPreferences mySharedPreferences;
	private SharedPreferences.Editor editor;

	private ProgressDialog loadingDialog;

	private EditText etID;
	private Button btLogin;
	private PatientDBAdapter db;
```

```
	private long id;
	private AlertDialog alDialog;


	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.login_screen);

		db = new PatientDBAdapter(this);
		db.open();

		etID = (EditText) findViewById(R.id.ef_login);
		btLogin = (Button) findViewById(R.id.bt_login);



		//adBuilder to create the AlertDialog we want.
	    AlertDialog.Builder adBuilder = new AlertDialog.Builder(Login.this);
		adBuilder.setCancelable(true)
		.setPositiveButton("OK", new DialogInterface.OnClickListener() {

			public void onClick(DialogInterface dialog, int which) {
				// TODO Auto-generated method stub
				dialog.cancel();
			}
		})
		.setNegativeButton("Off-line Mode", new
DialogInterface.OnClickListener() {

			public void onClick(DialogInterface dialog, int which) {
				// TODO Auto-generated method stub
				//Checks if This ID all ready in
				//If yes it let's the user in.


				if(db.getPatient(id)!=null){
					String text = etID.getText().toString();
					Intent intent = new Intent(Login.this,
TabSelection.class);
					intent.putExtra(PatientDBAdapter.KEY_PATIENT_ID,
text);
					startActivity(intent);
				}else{
					Toast.makeText(getApplicationContext(), "Wrong
ID", Toast.LENGTH_LONG).show();
				}

			}
		}).setNeutralButton("Demo", new DialogInterface.OnClickListener() {

			public void onClick(DialogInterface dialog, int which) {
				// TODO Auto-generated method stub

				Intent intent = new Intent(Login.this,
TabSelection.class);
```

```
                        demo();
                        id = 109876543210L;

                        Toast.makeText(getApplicationContext(),"DEMO MODE ",
Toast.LENGTH_LONG).show();

                        startActivity(intent);


                }
            });


            alDialog = adBuilder.create();



            btLogin.setOnClickListener(new OnClickListener() {

                  public void onClick(View v) {
                        // TODO Auto-generated method stub

                        id = Long.parseLong(etID.getText().toString());

                        //Flag to check if there is a connection
                        //to the internet.
                        boolean internetFlag = true;
                        //Getting the ConnectivityManager from the Device.
                        ConnectivityManager connMgr = (ConnectivityManager)

      getSystemService(Context.CONNECTIVITY_SERVICE);

                        //Getting Info.
                        NetworkInfo networkInfo =
connMgr.getActiveNetworkInfo();

                        //if is Null or is not connected to the Internet
                        //It will change the flag to false.
                        //Otherwise it will execute an AsyncTask.

                        if (networkInfo != null){
                              if(networkInfo.isConnected()){
                               new LoadingTask().execute();

                             }else{
                                 internetFlag=false;
                            }
                        }else{
                              internetFlag=false;
                        }


                       if(!internetFlag){

                          alDialog.setTitle("No Internet Connection");
                          alDialog.setMessage("Please Check if you are
Connected to the Internet");
                          alDialog.show();
                        }
```

```
                }
            });

            //used to read from SharedPreferences
            //Mode private, accessed only from this Application.
            mySharedPreferences =

      this.getSharedPreferences("com.chkyriacoy.projectdoc",Context.MODE_PRIVATE)
;

            //used to write to SharedPreferences
            editor = mySharedPreferences.edit();

      }



      @Override
      protected void onPause() {
            // TODO Auto-generated method stub
            super.onPause();
            //write the id
            editor.putLong(PatientDBAdapter.KEY_PATIENT_ID, id);
            editor.commit();

      }



      @Override
      protected void onResume() {
            // TODO Auto-generated method stub
            super.onResume();

            //read Id

            id = mySharedPreferences.getLong(PatientDBAdapter.KEY_PATIENT_ID, -
1L);

            //if exists then change the text in our View Text (etID)
            if (id != -1L){
            etID.setText(String.valueOf(id));
            }
      }



      @Override
      protected void onDestroy() {
            // TODO Auto-generated method stub
            super.onDestroy();
            db.close();
      }


      //AsyncTask execute in hes own process
      //using a Booling to check if has found any JSON
      private class LoadingTask extends AsyncTask<Void, Void, Boolean>{
```

```
@Override
protected Boolean doInBackground(Void... params) {
	// TODO Auto-generated method stub


	String url="http://www.sitehere.com";
HttpResponse httpResponse;
JSONObject jObj;
JSONObject c;
Boolean errorFlag = false;

try {

c = JSONHelper.idJson(id);
	httpResponse = JSONHelper.doPost(url, c);


	//Look at JSONHelper Class for those Methods
	jObj = JSONHelper.responseToJson(httpResponse);

	if(jObj.length()>0 ){
		JSONHelper.updateDB(jObj, db);
	}else{
		errorFlag = true;
	}




} catch (JSONException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
		loadingDialog.dismiss();
		errorFlag = true;
} catch (ClientProtocolException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
		loadingDialog.dismiss();
		errorFlag = true;
	} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
		loadingDialog.dismiss();
		errorFlag = true;
	}


	return errorFlag;
}

@Override
protected void onPreExecute() {
	// TODO Auto-generated method stub
	super.onPreExecute();

	//ProgressDialog to show whats going on to the user.
  loadingDialog = new ProgressDialog(Login.this);
	loadingDialog.setMessage("Logging In Please Wait . . .");
	loadingDialog.show();
```

```
            }

            @Override
            protected void onPostExecute(Boolean result) {
                  // TODO Auto-generated method stub
                  super.onPostExecute(result);

                  //If no error then move on to the next Activity.
                  //Else Pop the alDialog with new title and message.

                  loadingDialog.dismiss();

                  if(!result){
                        Intent intent = new Intent(Login.this,
TabSelection.class);
                        startActivity(intent);
                  }else{

                        alDialog.setTitle("Error");
                        alDialog.setMessage("Can't find your Entry");
                        alDialog.show();



                  }
            }
      }


      private void demo(){

            //Entry

            //Events
            ArrayList<POJOEvent> events = new ArrayList<POJOEvent>();
            events.add(new POJOEvent(12345678901L, 109876543210L, 101234567890L,
"Note", 20121212, "Please avoid red meat",
PatientDBAdapter.DATABASE_TABLE_EVENTS));
            events.add(new POJOEvent(12345678902L, 109876543210L, 101234567890L,
"Medication Scroll <--- to read more", 20121213, " Long Scrolling Text ",
PatientDBAdapter.DATABASE_TABLE_MEDICATIONS));
            events.add(new POJOEvent(12345678906L, 109876543210L, 101234567890L,
"Medication 2", 20120801, "Text Something",
PatientDBAdapter.DATABASE_TABLE_MEDICATIONS));

            events.add(new POJOEvent(12345678903L, 109876543210L, 101234567890L,
"Bluetooth", 20121210, "Take your pressure",
PatientDBAdapter.DATABASE_TABLE_MEASURES));

            //Medications
            ArrayList<POJOMedication> medications = new
ArrayList<POJOMedication>();
            medications.add(new POJOMedication(12345678902L, 20130311, 3));
            medications.add(new POJOMedication(12345678906L, 20131121, 5));

            //Patients
            ArrayList<POJOPatient> patients = new ArrayList<POJOPatient>();
```

```
            patients.add(new POJOPatient(109876543210L, "Christos", "Kyriaco",
"", 101234567890L));

            //Doctors
            ArrayList<POJODoctor> doctors = new ArrayList<POJODoctor>();
            doctors.add(new POJODoctor(101234567890L, "Panagiotis",
"Alefragis"));

            //Measures
            ArrayList<POJOMeasure> measures = new ArrayList<POJOMeasure>();
            measures.add(new POJOMeasure(12345678903L, 0x1007));



////////////////////////////////////////////////////////////////////////

            //Old Database

            ArrayList<POJOEvent> eventsDB = db.getEvents();
            ArrayList<POJOMedication> medicationsDB = db.getMedications();
            ArrayList<POJOMeasure> measuresDB = db.getMeasures();
            ArrayList<POJOPatient> patientsDB = db.getPatients();
            ArrayList<POJODoctor> doctorsDB = db.getDoctors();


////////////////////////////////////////////////////////////////////////

            //Check for what is not in the new Entries

            boolean flag = false;

            if (eventsDB != null){
            for(int i=0; i<eventsDB.size(); i++){
                  for(int j=0; j<events.size(); j++){

      if(eventsDB.get(i).getEventID()==events.get(j).getEventID()){
                              flag = true;
                        }
                  }
                  if(!flag){
                        db.deleteEvent(eventsDB.get(i));
                  }
                  flag = false;
            }
            }

            //Update and if can't, Insert

            for(POJOEvent event : events){
                  if(db.updateEvent(event)==0){
                  db.insertEvent(event);
                  }
            }

////////////////////////////////////////////////////////////////////*

            if (medicationsDB != null){
            for(int i=0; i<medicationsDB.size(); i++){
                  for(int j=0; j<medications.size(); j++){
```

```
	if(medicationsDB.get(i).getMedicationID()==medications.get(j).getMedication
ID()){
					flag = true;
				}
			}
			if(!flag){
				db.deleteMedication(medicationsDB.get(i));
			}
			flag = false;
		}
		}

		for(POJOMedication medication : medications){
			if(db.updateMedication(medication)==0){
			db.insertMedication(medication);
			}
		}

	////////////////////////////////////////////////////////////////

		if (measuresDB != null){
		for(int i=0; i<measuresDB.size(); i++){
			for(int j=0; j<measures.size(); j++){

	if(measuresDB.get(i).getMeasureID()==measures.get(j).getMeasureID()){
					flag = true;
				}
			}
			if(!flag){
				db.deletePressure(measuresDB.get(i));
			}
			flag = false;
		}
		}

	for(POJOMeasure measure : measures){
		if(db.updateMeasure(measure)==0){
			db.insertMeasure(measure);
		}
	}


	////////////////////////////////////////////////////////////////////////
/


		if (patientsDB != null){
			for(int i=0; i<patientsDB.size(); i++){
				for(int j=0; j<patients.size(); j++){

	if(patients.get(i).getPatientID()==patients.get(j).getPatientID()){
						flag = true;
					}
				}
				if(!flag){
					db.deletePatient(patientsDB.get(i));
				}
				flag = false;
```

```
                    }
                    }

      for(POJOPatient patient : patients){
            if(db.updatePatient(patient)==0){
            db.insertPatient(patient);
            }
      }


      ////////////////////////////////////////////////////////////////////////
/

                    if (doctorsDB != null){
                          for(int i=0; i<doctorsDB.size(); i++){
                                for(int j=0; j<doctors.size(); j++){

      if(doctorsDB.get(i).getDoctorID()==doctors.get(j).getDoctorID()){
                                            flag = true;
                                      }
                                }
                                if(!flag){
                                      db.deleteDoctor(doctorsDB.get(i));
                                }
                                flag = false;
                          }
                          }

                    for(POJODoctor doctor : doctors){
                          if(db.updateDoctor(doctor)==0){
                          db.insertDoctor(doctor);
                          }
                    }

      }
}
```

MainScreen.java

```
package com.chkyriacou.activities;


import java.util.ArrayList;

import com.chkyriacou.R;


import com.chkyriacou.items.POJOEvent;
import com.chkyriacou.items.PatientDBAdapter;
import com.chkyriacou.services.AlarmService;
```

```
import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;
import android.view.LayoutInflater;

import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TabHost;
import android.widget.TextView;


/**
 * Holds a ListView to display the wanted Events
 * that belog in the same Tab.
 *
 */

public class MainScreen extends Activity {

	private PatientDBAdapter db;
	private ArrayList<POJOEvent> arrayOfEvents = new ArrayList<POJOEvent>();
	private ArrayAdapter<POJOEvent> eventAdapter;
	private ListView listView=null;
	private TabHost tabHost;


	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main_screen);

		//getting the db from TabSelection.
		TabSelection parent = (TabSelection) getParent();
		db = parent.getDB();
		tabHost	= parent.getTabHost();

		listView = (ListView)findViewById(R.id.lv_events);
		arrayOfEvents = new ArrayList<POJOEvent>();
		eventAdapter = new EventAdapter();
		listView.setAdapter(eventAdapter);



		Intent serviceIntent= new Intent(this, AlarmService.class);
	  startService(serviceIntent);


	  //list item click listener, starts the next Activity (EventScreen)
	  //putting in the Extra the ID from the Clicked list item, the POJOEvent.
		listView.setOnItemClickListener(new OnItemClickListener() {
```

```
                    public void onItemClick(AdapterView<?> parent, View view, int
position,
                                    long id) {
                            // TODO Auto-generated method stub
                            POJOEvent event = arrayOfEvents.get(position);
                            Intent data = new
Intent(MainScreen.this,EventScreen.class);
                            data.putExtra(PatientDBAdapter.KEY_EVENT_ID,
event.getEventID());
                            startActivity(data);


                    }
            });


      }

//Called each time the we switch Tabs
      @Override
      protected void onResume() {
            // TODO Auto-generated method stub
            super.onResume();

            if(!arrayOfEvents.isEmpty()){
            arrayOfEvents.clear();
            }

            ArrayList<POJOEvent> events = null;

            //Changing the Dates and get the Events from the type
            //we need.
            if(tabHost.getCurrentTabTag().equals("All")){
                  events = db.getEvents();
            } else if(tabHost.getCurrentTabTag().equals("Notes")){
                  events = db.getEventsNotes();
            } else if(tabHost.getCurrentTabTag().equals("Medications")){
                  events = db.getEventsMedications();
            } else if(tabHost.getCurrentTabTag().equals("Measures")){
                  events = db.getEventsMeasures();
            } else {
                  events = null;
            }

            for(POJOEvent event: events){
                  arrayOfEvents.add(event);
            }

            //notify the adapter
            eventAdapter.notifyDataSetChanged();
      }



// Creating the Our ArrayAdapter that gets POJOEvents Objects
      class EventAdapter extends ArrayAdapter<POJOEvent>{

            EventAdapter(){
```

```
                        super(MainScreen.this, android.R.layout.simple_list_item_1,
arrayOfEvents);
                        //We Inflate this later. We just using it to begin.
                }

                @Override
                public View getView(int position, View convertView, ViewGroup
parent) {
                        // TODO Auto-generated method stub
                        ViewHolder holder;

                        if(convertView == null){
                        //inflater the convertView with the our Layout.
                                LayoutInflater inflater = getLayoutInflater();

        convertView=inflater.inflate(R.layout.listview_item_event, null);
                                holder = new ViewHolder(convertView);
                        //tagging our convertView so we can get it later
                                convertView.setTag(holder);
                        }else{
                        //if the convertView is tagged get it.
                                holder=(ViewHolder) convertView.getTag();
                        }
                        //Call populate to change the views in our holder.
                        //This will change the convertView too.
                        holder.populateFrom(arrayOfEvents.get(position));
                        return(convertView);
                }
        }

//Using this to hold our Views.
        class ViewHolder{
                public ImageView icon = null;
                public TextView title = null;
                public TextView brief = null;

        ViewHolder(View row) {
                icon = (ImageView)row.findViewById(R.id.iv_list_item);
                title = (TextView)row.findViewById(R.id.tv_list_item_title);
                brief = (TextView)row.findViewById(R.id.tv_list_item_brief);
        }

        void populateFrom(POJOEvent event){

                if(event.getType().equals(PatientDBAdapter.DATABASE_TABLE_EVENTS)){

        icon.setImageDrawable(getResources().getDrawable(R.drawable.note));
                }else
if(event.getType().equals(PatientDBAdapter.DATABASE_TABLE_MEDICATIONS)){

        icon.setImageDrawable(getResources().getDrawable(R.drawable.drug));
                }else
if(event.getType().equals(PatientDBAdapter.DATABASE_TABLE_MEASURES)){

        icon.setImageDrawable(getResources().getDrawable(R.drawable.blutooth));

                }else {

        icon.setImageDrawable(getResources().getDrawable(R.drawable.me));
```

```
            }

            title.setText(event.getTitle());
            brief.setText(event.getFormedDate());
            }
      }
}
```

MyFirstGoogleMap.java

```
package com.chkyriacou.activities;

import java.io.BufferedReader;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;


import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import android.app.ProgressDialog;
import android.content.Context;

import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.graphics.Point;
import android.graphics.drawable.Drawable;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;

import android.provider.Settings;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;

import com.chkyriacou.R;
```

```
import com.chkyriacou.items.POJOPatient;
import com.chkyriacou.items.PatientDBAdapter;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.OverlayItem;
import com.google.android.maps.Projection;

/**
 * Showing a google map.
 * Giving show the user where he is and how much Accurate
 * is his position.
 *
 * in the mune, the user can find where are hospitals and
 * dragstors. Checks for Postal Code to be used from more accurate
 * searching. For the searching the web site www.xo.gr is used.
 *
 *
 */

public class MyFirstGoogleMap extends MapActivity implements LocationListener{

	private MapController controller;
	private LocationManager manager;
	private GeoPoint myGeoPoint;

	//used to take the latitude and longitude
	private Geocoder myGeocoder;

	//hold the pins that will showed in the map.
	private List<Overlay> listOfOverlays;

	// used to represent the Hospitals and Drugstores
	private PinsOverlay pinsOverlay;

	//used only to represent the user
	private PinsOverlay mePin;
	private AccuracyOverlay accuracyOverlay;

	//Flags
	//if there is Postal Code use Range
	private Boolean rangeFlag;

	//if the location has been found
	private Boolean locationFlag;

	//To make the onLocationChange not execute while is
	//not finished.
	private Boolean locationUpdating;


	private String provider;
	private String accuracy;

	private MapView myMap;
```

```
	private PatientDBAdapter db;
	private SharedPreferences mySharedPreferences;
	private ProgressDialog loadingDialog;

	private List<String> myAddresses;


	@Override
	protected void onCreate(Bundle icicle) {
		// TODO Auto-generated method stub
		super.onCreate(icicle);
		setContentView(R.layout.map_view);

		locationFlag = false;
		locationUpdating = false;


		//Check if there is connection to the internet.
	   ConnectivityManager connMgr = (ConnectivityManager)
	  getSystemService(Context.CONNECTIVITY_SERVICE);
	   NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

	   if (networkInfo != null && networkInfo.isConnected()) {

		myMap = (MapView) findViewById(R.id.mapView);
		myMap.setBuiltInZoomControls(true);

	   listOfOverlays = myMap.getOverlays();

	   controller = myMap.getController();
	   controller.setZoom(15);

	   //if GPS and NETWORK is not on ask the user to turn it on
	   manager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);

	   if(
!(manager.isProviderEnabled(android.location.LocationManager.GPS_PROVIDER)||
manager.isProviderEnabled(android.location.LocationManager.NETWORK_PROVIDER)) )
	{
		Intent myIntent = new Intent( Settings.ACTION_SECURITY_SETTINGS );
		startActivity(myIntent);
	}

	   //Using Criteria to get the Provider that suits
	Criteria criteria = new Criteria();
	criteria.setAltitudeRequired(false);
	criteria.setCostAllowed(false);
	criteria.setAccuracy(Criteria.ACCURACY_COARSE);
	   provider = manager.getBestProvider(criteria, true);


	   //Because Geocoder is a Proxy Object we use getBaseContext instend of
	   //getApplicationContext
	   myGeocoder = new Geocoder(getBaseContext(), Locale.getDefault());

	   //Check if there is an Location already
	   Location location = manager.getLastKnownLocation(provider);
```

```
          if (location != null) {
            onLocationChanged(location);
        } else {
            Toast.makeText(getApplicationContext(),
                           "No last known locations has been found",
Toast.LENGTH_LONG).show();
        }


          } else {
            //no Connection to the internet.
            Toast.makeText(getApplicationContext(), "Your Device is not
connected to the internet", Toast.LENGTH_LONG).show();
            finish();
          }

          db = new PatientDBAdapter(this);

          mySharedPreferences =

      getSharedPreferences("com.chkyriacoy.projectdoc",Context.MODE_PRIVATE);


      }



      @Override
      protected void onStart() {
            // TODO Auto-generated method stub
            super.onStart();
            //inform the user about the provider
          Toast.makeText(getApplicationContext(), "You use "+provider+"
provider", Toast.LENGTH_LONG).show();

          db.open();

      }



      @Override
      protected void onResume() {
            // TODO Auto-generated method stub
            super.onResume();

                  //request update every 8 sec or 50 meters
              manager.requestLocationUpdates(provider, 8000, 50f, this);
              locationUpdating = true;
      }

      @Override
      protected void onPause() {
            // TODO Auto-generated method stub
            super.onPause();
            manager.removeUpdates(this);
            locationUpdating = false;
```

```
	}



	@Override
	protected void onStop() {
		// TODO Auto-generated method stub
		super.onStop();


		//put the last know location of with accuracy
		//in the table in String form.

		if(locationFlag){
			String lastKnownLocation =
String.valueOf(myGeoPoint.getLatitudeE6())+

	"@"+String.valueOf(myGeoPoint.getLongitudeE6()+
							"#"+accuracy);

			long id =
mySharedPreferences.getLong(PatientDBAdapter.KEY_PATIENT_ID, -1L);

			if(id!=-1L){

				POJOPatient patient =	db.getPatient(id);

				patient.setLastKnownLocation(lastKnownLocation);
				db.updatePatient(patient);
			}
		}

		db.close();

	}




	//Called when the location has changed. With the new location, as a
Location object.

	public void onLocationChanged(Location location) {
		// TODO Auto-generated method stub
		if(locationUpdating){
	//to ensure that it will not execute again until is finished.
			locationUpdating = false;


		//make GeoPoint from my location
		myGeoPoint = new
GeoPoint((int)(location.getLatitude()*1E6),(int)(location.getLongitude()*1E6));

		controller.setCenter(myGeoPoint);

		 if(mePin!=null){
			listOfOverlays.remove(0);
```

```
                    listOfOverlays.remove(1);
             }else{
                    mePin = new  PinsOverlay(
                                 getResources().getDrawable(R.drawable.me));
             }

             accuracy = String.valueOf(location.getAccuracy());

             mePin.insertPin(new OverlayItem(myGeoPoint, "Me", "You are here"+"\n
Accuracy : "+accuracy+" meters"));
             accuracyOverlay = new AccuracyOverlay(myGeoPoint,
location.getAccuracy());

             listOfOverlays.add(0, accuracyOverlay);
             listOfOverlays.add(1, mePin);

             myMap.invalidate();

             locationFlag = true;

             locationUpdating = true;
             }
      }

      @Override
      protected boolean isRouteDisplayed() {
             // TODO Auto-generated method stub
             return false;
      }

      public void onProviderDisabled(String provider) {
             // TODO Auto-generated method stub

      }

      public void onProviderEnabled(String provider) {
             // TODO Auto-generated method stub

      }

      public void onStatusChanged(String provider, int status, Bundle extras) {
             // TODO Auto-generated method stub

      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
             // TODO Auto-generated method stub
             super.onCreateOptionsMenu(menu);
             MenuInflater menuInflater = getMenuInflater();
             menuInflater.inflate(R.menu.map_menu, menu);
             return true;
      }

      @Override
      public boolean onOptionsItemSelected(MenuItem item) {
             // TODO Auto-generated method stub
             //List<String> stringAddresses;
```

```
            switch (item.getItemId()){
            case R.id.menu_pharmacy:


             if (locationFlag) {


                  new LoadingTaskAddress().execute("drogstores");

             } else {
                  Toast.makeText(getApplicationContext(),
                              "You'r location is not found",
Toast.LENGTH_LONG).show();
             }
                  return true;

            case R.id.menu_hospital:

             if (locationFlag) {


           new LoadingTaskAddress().execute("hospitals");


        } else {
             Toast.makeText(getApplicationContext(),
                         "You'r location is not found",
Toast.LENGTH_LONG).show();
        }
            return true;
             }

             return false;
      }


      //Use to turn List of all the address in to Pins
      protected void addressesToPoints(List<String> stringAddresses, Drawable
draw ,boolean range ){
            GeoPoint myGeoPoint = this.myGeoPoint;

            //clear the old Pins
            //and re-enter the accuracy and user's Pin
            listOfOverlays.clear();
            listOfOverlays.add(0, accuracyOverlay);
            listOfOverlays.add(1, mePin);

            pinsOverlay = new PinsOverlay(draw);


            //check if there is any anything in the list.
            if(stringAddresses.size()==0){
                  Toast.makeText(this,
                              "No Address have been found in your location",
                              Toast.LENGTH_LONG).show();

                  }else{

            try {
```

```
                    //if there we have the Postal Code use the range
              if(range){

                    final int DISTANCE = (int)(00.27*1E6);
                    double lowerLeftLatitude = (myGeoPoint.getLatitudeE6()-
DISTANCE)/1E6;
                    double lowerLeftLongitude = (myGeoPoint.getLongitudeE6()-
DISTANCE)/1E6;
                    double upperRightLatitude =
(myGeoPoint.getLatitudeE6()+DISTANCE)/1E6;
                    double upperRightLongitude =
(myGeoPoint.getLongitudeE6()+DISTANCE)/1E6;

                    for (int i=0;i<stringAddresses.size();i++){


                          //Getting Address using the Geocoder and the Distance
will return
                          //only the Addresses that are in that "Box"
                          List<Address> myAddressList =
myGeocoder.getFromLocationName(stringAddresses.get(i), 1,
                                      lowerLeftLatitude, lowerLeftLongitude,
upperRightLatitude, upperRightLongitude);

                          if(myAddressList.size()>0){
                                //if all are ok put them in pinsOverlay.
                                OverlayItem overlayItem = new OverlayItem(new
GeoPoint((int)(myAddressList.get(0).getLatitude() *1E6),

      (int)(myAddressList.get(0).getLongitude()*1E6)),"", stringAddresses.get(i)
);
                                pinsOverlay.insertPin(overlayItem);


                                }
                          }
                    }else{

                          //no Postal Code. no range.

                          for (int i=0;i<stringAddresses.size();i++){


                                List<Address> myAddressList =
myGeocoder.getFromLocationName(stringAddresses.get(i), 1);

                                if(myAddressList.size()>0){
                                      OverlayItem overlayItem = new
OverlayItem(new GeoPoint((int)(myAddressList.get(0).getLatitude() *1E6),

      (int)(myAddressList.get(0).getLongitude()*1E6)),"", stringAddresses.get(i)
);
                                      pinsOverlay.insertPin(overlayItem);


                                      }
                                }
```

```
                }

            listOfOverlays.add(pinsOverlay);
            myMap.invalidate();

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



      //PinsOverlay for all our Pins
      class PinsOverlay extends ItemizedOverlay<OverlayItem>{

            private ArrayList<OverlayItem> pins;


            public PinsOverlay(Drawable defaultMarker) {
                //puts the bottom center of the Drawable
                //to the position in the map.
                super(boundCenterBottom(defaultMarker));

                this.pins = new ArrayList<OverlayItem>();
            }


            @Override
            protected OverlayItem createItem(int i) {
                // TODO Auto-generated method stub
                return pins.get(i);
            }

            public void insertPin(OverlayItem pin){
                pins.add(pin);
                this.populate();

            }

            @Override
            public int size() {
                // TODO Auto-generated method stub
                return pins.size();
            }



            @Override
            protected boolean onTap(int index) {
                // TODO Auto-generated method stub
                Toast.makeText(getApplicationContext(),
                             pins.get(index).getSnippet(),
                             Toast.LENGTH_LONG).show();

                return super.onTap(index) ;
            }
      }
```

```
      //for the circle that shows the Accuracy of our location.
       class AccuracyOverlay extends Overlay {

              private GeoPoint sourcePoint;
            private float accuracy;

           public AccuracyOverlay(GeoPoint geoPoint, float accuracy) {
               super();
              sourcePoint = geoPoint;
              this.accuracy = accuracy;
           }

           public void setSource(GeoPoint geoPoint, float accuracy) {
              sourcePoint = geoPoint;
              this.accuracy = accuracy;
              }


                  @Override
                  public void draw(Canvas canvas, MapView mapView, boolean shadow)
{
                     super.draw(canvas, mapView, false);

                     Projection projection = mapView.getProjection();

                     Point center = new Point();

                     int radius = (int)
(projection.metersToEquatorPixels(accuracy));

                     //the Point will be our object that will hold the X and Y in
the map
                     //the sourcePoint hold latitude and longitude and can't be
used in the map

                     projection.toPixels(sourcePoint, center);

                      Paint accuracyPaint = new Paint();

                      //Setting the outside of the circle
                      accuracyPaint.setAntiAlias(true);
                      accuracyPaint.setStrokeWidth(2.0f);
                      accuracyPaint.setColor(0xff6666ff);
                      accuracyPaint.setStyle(Style.STROKE);

                      canvas.drawCircle(center.x, center.y, radius,
accuracyPaint);

                      //Setting the inside of the circle
                      accuracyPaint.setColor(0x186666ff);
                      accuracyPaint.setStyle(Style.FILL);

                      canvas.drawCircle(center.x, center.y, radius,
accuracyPaint);

                  }
              }
```

```
        //This AsyncTask will get a String param that will be used to do the
research in
        //www.xo.gr (a get reguest)
              private class LoadingTaskAddress extends AsyncTask<String, Void,
Boolean>{

                    @Override
                    protected Boolean doInBackground(String... params) {
                          // TODO Auto-generated method stub

                    DefaultHttpClient client = new DefaultHttpClient();

                          HttpGet request = null;
                          HttpResponse response = null;
                          InputStream in = null;
                          BufferedReader reader = null ;
                          String url = "";
                          String line = "";
                          int endPointer = 0;
                          myAddresses = new ArrayList<String> ();
                          List<String> lines = new ArrayList<String> ();


                          //checking if we want Hospitals or Dragstores
                          Boolean hospitalsCase =
(params[0].equals("hospitals"));

                          try {
                                List<Address> myAddressList =
myGeocoder.getFromLocation((double)(myGeoPoint.getLatitudeE6()/1E6),
(double)(myGeoPoint.getLongitudeE6()/1E6), 1);

                                if(myAddressList.size()>0){


      url+="http://www.xo.gr/search/?what="+params[0]+"&where=";
                                      url+=myAddressList.get(0).getLocality();

                                      //Checking if we know the Postal Code

      if(myAddressList.get(0).getPostalCode()!=null){
                                            //if yes, we add it in our url and
set rangeFlag true

      url+="%20"+myAddressList.get(0).getPostalCode()+"&lang=en";
                                            rangeFlag=true;

                                      }else{
                                            //if not we set rangeFlag false.
                                            url+="&lang=en";
                                            rangeFlag=false;
                                      }

                                      //getting the whole page
                                      request = new HttpGet(url);
                                      response = client.execute(request);
                                      in = response.getEntity().getContent();
```

```
                                        reader = new BufferedReader(new
InputStreamReader(in),16384);

                                        while((line=reader.readLine())!= null){
                                              lines.add(line);
                                              }
                                  }
                  } catch (IOException e) {
                                  // TODO Auto-generated catch block
                                  e.printStackTrace();
                                  this.cancel(true);

                  }


                        //Reads the Page and finds the Addresses by finding
                        //chain characters ",</" and "nowrap"

                  String partOfLine = "";
                  for(String l : lines){
                        endPointer=l.lastIndexOf(",</");
                              if(endPointer!=-1){
                                    partOfLine = l.substring(endPointer-40,
endPointer);
                                    partOfLine =
partOfLine.substring(partOfLine.lastIndexOf("nowrap")+8);
                                    myAddresses.add(partOfLine);

                              }
                  }

                  //tell to onPostExecute what we here looking for
                        return hospitalsCase;
                  }

                  @Override
                  protected void onPreExecute() {
                        // TODO Auto-generated method stub
                        super.onPreExecute();

                    loadingDialog = new ProgressDialog(MyFirstGoogleMap.this);
                        loadingDialog.show();
                        loadingDialog.setMessage("Searching, Please Wait . .
.");
                  }

                  @Override
                  protected void onPostExecute(Boolean result) {
                        // TODO Auto-generated method stub
                        super.onPostExecute(result);

                        //if true put the hospital icon , if false put the
drugstore icon.
                        if(result){
                              addressesToPoints(myAddresses,
getResources().getDrawable(R.drawable.hospital) , rangeFlag);
                        } else {
                              addressesToPoints(myAddresses,
getResources().getDrawable(R.drawable.drugstore) , rangeFlag);
```

```
                        }

                        loadingDialog.dismiss();
                    }
                }
}
```

Schedule.java

```
package com.chkyriacou.activities;


import java.util.ArrayList;
import java.util.Calendar;

import com.chkyriacou.R;
import com.chkyriacou.items.POJOAlarm;
import com.chkyriacou.items.POJOEvent;
import com.chkyriacou.items.PatientDBAdapter;
import com.chkyriacou.services.AlarmService;


import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.Intent;

import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;


/**
 *
 * Shows the Daily Mediaction Alarms that has be Set.
 * All Items in the list that are with in the next
 * hour are show an Icon.
 * Is Activity is called by AlarmService when an Alarm
```

```
 * goes off. By that this Class with Vibe and Play a Tune.
 *
 *
 */



public class Schedule extends Activity {

	private PatientDBAdapter db;
	private ArrayList<POJOAlarm> arrayOfAlarms = new ArrayList<POJOAlarm>();
	private ArrayAdapter<POJOAlarm> alarmAdapter;
	private ListView listView=null;
	private Vibrator vib = null;

	private Ringtone r;
	private Uri uri;

	private boolean flag = false;

	private long[] pattern = {500,1000};

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main_screen);
		db = new PatientDBAdapter(this);
		db.open();




			//Getting the Ringtone with type "TYPE_ALARM"
		//This is the same Tone user use in the Mobile.

			uri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);
			r = RingtoneManager.getRingtone(getApplicationContext(), uri);

			//Getting the Device's Vibrator from the System.
		vib = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

		listView = (ListView)findViewById(R.id.lv_events);
		arrayOfAlarms= new ArrayList<POJOAlarm>();
		alarmAdapter = new AlarmAdapter();


		// Click an Item in the list will Start the proper
		listView.setOnItemClickListener(new OnItemClickListener() {

			public void onItemClick(AdapterView<?> parent, View view, int
position,
					long id) {
				// TODO Auto-generated method stub

				//The ID of Alarm is the ID of Event*10 Plus The
TimePicker Number
				//(From the AlarmSetup Activity).
```

```
                        long eventID =
(arrayOfAlarms.get(position).getAlarmID()/10);
                        POJOEvent event = db.getEvent(eventID);

                        Intent data = new
Intent(Schedule.this,EventScreen.class);
                        data.putExtra(PatientDBAdapter.KEY_EVENT_ID,
event.getEventID());
                        startActivity(data);
                    }
                });

        }

        @Override
        protected void onResume() {
        // TODO Auto-generated method stub
        super.onResume();


                listView.setAdapter(alarmAdapter);

        if(!arrayOfAlarms.isEmpty()){
                    arrayOfAlarms.clear();
                }


                ArrayList<POJOAlarm> alarms =  db.getAlarms();

                if (alarms!=null){
                    for(POJOAlarm alarm: alarms){
                        arrayOfAlarms.add(alarm);
                    }
                    alarmAdapter.notifyDataSetChanged();
                }


                //An Extra from the Intent from AlarmService
                //If flag is true it will play the Ringtone and Vib

                Intent intent = this.getIntent();
                ///flag = getIntent().getBooleanExtra("ring", false);

                flag= intent.getBooleanExtra("ring", false);


                if(flag){

                    //Play the righttone and vibe
                    r.play();

                    if(vib!=null){
                    vib.vibrate(pattern,0);
                    }

                    //Also Start the AlarmService to set the Next alarm
                    startService(new Intent(Schedule.this,AlarmService.class));
                }
    }
```

```
	@Override
	protected void onPause() {
	// TODO Auto-generated method stub
	super.onPause();

	if(flag){

		//Setting the Extra to false in case
		//the user press Back.
		//Making sure that is IF will happen
		//only once.
		//Also Stop Ringtone and Vibe

		getIntent().putExtra("ring", false);

		r.stop();
		if(vib!=null){
			vib.cancel();
		}
		}
	}

	@Override
	protected void onDestroy() {
		// TODO Auto-generated method stub
		super.onDestroy();
		db.close();
	}



class AlarmAdapter extends ArrayAdapter<POJOAlarm>{

		AlarmAdapter(){
			super(Schedule.this, android.R.layout.simple_list_item_1,
arrayOfAlarms);
		}

		@Override
		public View getView(int position, View convertView, ViewGroup
parent) {
			// TODO Auto-generated method stub
			ViewHolder holder;

			if(convertView == null){
				LayoutInflater inflater = getLayoutInflater();

	convertView=inflater.inflate(R.layout.listview_item_event, null);
				holder = new ViewHolder(convertView);

				convertView.setTag(holder);
			}else{
				holder=(ViewHolder) convertView.getTag();
			}
			holder.populateFrom(arrayOfAlarms.get(position),
					holder);
```

```
                    return convertView;
                    }
      }



      class ViewHolder{
            public ImageView icon = null;
            public TextView title = null;
            public TextView brief = null;

      ViewHolder(View row) {
            icon = (ImageView)row.findViewById(R.id.iv_list_item);
            title = (TextView)row.findViewById(R.id.tv_list_item_title);
            brief = (TextView)row.findViewById(R.id.tv_list_item_brief);
      }

      void populateFrom(POJOAlarm alarm, ViewHolder holder){

            long id = alarm.getAlarmID()/10;
            POJOEvent event = db.getEvent(id);
            title.setText(event.getTitle());

            int hour = alarm.getTime()/10000;
            int min = (alarm.getTime()/100)%100;

            String textmin = String.valueOf(min);

            if(min<10){
                  textmin ="0"+textmin;
            }

            String when = String.valueOf(hour)+":"+ textmin;
            brief.setText(when);

            //Using Calender to take the current Time
            //If our Item's time is within the current hour
            //then we set the Image.
            Calendar cal = Calendar.getInstance();

            int calTime =
(100*cal.get(Calendar.HOUR_OF_DAY))+cal.get(Calendar.MINUTE);
            int time = (100*hour)+min;

            if(calTime<=time&&calTime+100>=time){

      icon.setImageDrawable(getResources().getDrawable(R.drawable.schedule));
                  holder.brief.setTextSize(22);
            }else{
                  icon.setVisibility(View.INVISIBLE);
                  holder.brief.setTextSize(22);
            }

      }
   }



  //Menu Option to Delete all Alarms.
```

```
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// TODO Auto-generated method stub
		super.onCreateOptionsMenu(menu);
		MenuInflater menuInflater = getMenuInflater();
		menuInflater.inflate(R.menu.schedule_menu, menu);
		return true;
	}


	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// TODO Auto-generated method stub

		switch (item.getItemId()){

		case R.id.menu_schedule_delete:


		    Builder builder = new AlertDialog.Builder(this);
		    builder.setMessage("Do you wish to DELETE ALL Alarms?");
		    builder.setCancelable(true);
		    builder.setPositiveButton("Yes", new OnClickListener() {

                                public void onClick(DialogInterface dialog, int
which) {
                                        // TODO Auto-generated method stub

                                        //Clean the Table and the ViewList
                                        db.emptyAlarmTable();
                                        alarmAdapter.clear();

                                        //Start the Service so it will call and
remove the next Alarm.
                                        startService(new
Intent(Schedule.this,AlarmService.class));
                                }
                        });
		    builder.setNegativeButton("No", new OnClickListener() {

                                public void onClick(DialogInterface dialog, int
which) {
                                        // TODO Auto-generated method stub

                                }
                        });
		    AlertDialog dialog = builder.create();
		    dialog.show();

		       return true;
		}

		return false;
	}

}
```

TabSelection.java

```
package com.chkyriacou.activities;

import java.io.IOException;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.json.JSONException;
import org.json.JSONObject;

import com.chkyriacou.R;
import com.chkyriacou.items.JSONHelper;
import com.chkyriacou.items.PatientDBAdapter;
import com.chkyriacou.services.AlarmService;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.app.TabActivity;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.TabHost;
import android.widget.Toast;
import android.widget.TabHost.TabSpec;


/**
 * Provates Menu and Tags
 * All Tags lead to the same Activity (MainScreen).
 * All Tags Share the same Menu.
 *
 */


public class TabSelection extends TabActivity {

	private PatientDBAdapter db;
	private ProgressDialog loadingDialog;
	private SharedPreferences mySharedPreferences;

	//Provates the same Database so we don't need to
	//open and close in each Tab
	protected PatientDBAdapter getDB(){
		return this.db;
	}


	@Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab_selection);

    // Start the AlarmService Check AlarmService for more info.
    startService(new Intent(this, AlarmService.class ));

       db = new PatientDBAdapter(this);
       db.open();

       mySharedPreferences =
getSharedPreferences("com.chkyriacoy.projectdoc",Context.MODE_PRIVATE);

       TabHost tabHost = getTabHost();

    // Tab for All
    TabSpec allSpec = tabHost.newTabSpec("All");
    // setting Title and Icon for the Tab
    allSpec.setIndicator("All",getResources().getDrawable(R.drawable.star));
    Intent allIntent = new Intent(this, MainScreen.class);
    allSpec.setContent(allIntent);



    // Tab for Notes
    TabSpec notesSpec = tabHost.newTabSpec("Notes");
    notesSpec.setIndicator("Notes", getResources().getDrawable(R.drawable.note));
    Intent notesIntent = new Intent(this, MainScreen.class);
    notesSpec.setContent(notesIntent);

    // Tab for Medications
    TabSpec medicationsSpec = tabHost.newTabSpec("Medications");
    medicationsSpec.setIndicator("Medications",
getResources().getDrawable(R.drawable.drug));
       Intent medicationsIntent = new Intent(this, MainScreen.class);
       medicationsSpec.setContent(medicationsIntent);


    // Tab for Measures
    TabSpec measuresSpesc = tabHost.newTabSpec("Measures");
    measuresSpesc.setIndicator("Measures",
getResources().getDrawable(R.drawable.blutooth));
       Intent measuresIntent = new Intent(this, MainScreen.class);
    measuresSpesc.setContent(measuresIntent);




    // Adding all TabSpec to TabHost
    tabHost.addTab(allSpec); // Adding All tab
    tabHost.addTab(notesSpec); // Adding Notes tab
    tabHost.addTab(medicationsSpec);// Adding Medications tab
       tabHost.addTab(measuresSpesc); // Adding Measures tab

    }



       @Override
```

```
	protected void onDestroy() {
		// TODO Auto-generated method stub
		super.onDestroy();
		db.close();
	}



	// MOBILE'S MENU
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// TODO Auto-generated method stub
		super.onCreateOptionsMenu(menu);
		MenuInflater menuInflater = getMenuInflater();
		menuInflater.inflate(R.menu.main_menu, menu);
		return true;
	}


	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// TODO Auto-generated method stub

		switch (item.getItemId()){
		case R.id.menu_map:

			Intent mapIntent = new Intent(this, MyFirstGoogleMap.class);

			startActivity(mapIntent);
			return true;

		case R.id.menu_schedule:

			Intent scheduleIntent = new Intent(this, Schedule.class);

			startActivity(scheduleIntent);

			return true;


		case R.id.menu_update:


			ConnectivityManager connMgr = (ConnectivityManager)
	  getSystemService(Context.CONNECTIVITY_SERVICE);
		NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
		if (networkInfo != null && networkInfo.isConnected()) {

			long id = mySharedPreferences.getLong(
					PatientDBAdapter.KEY_PATIENT_ID, -1);

			if (id>-1){
			new LoadingTask().execute(id);
			}else{
				Toast.makeText(getApplicationContext(), "Error 37. .
.", Toast.LENGTH_LONG).show();
			}

		}else{
```

```
                    Toast.makeText(getApplicationContext(), "Your device is not
connected to the internet", Toast.LENGTH_LONG).show();
                }

                    return true;
            }

            return false;
      }


      private class LoadingTask extends AsyncTask<Long, Void, Boolean>{

            @Override
            protected Boolean doInBackground(Long... params) {
                  // TODO Auto-generated method stub


                  String url="http://www.sitehere.com";
            HttpResponse httpResponse;
            JSONObject jObj;
            JSONObject c;
            Boolean errorFlag = false;

            try {


            c = JSONHelper.idJson(params[0]);
            httpResponse = JSONHelper.doPost(url, c);

                  jObj = JSONHelper.responseToJson(httpResponse);

                  if(jObj.length()>0 ){
                        JSONHelper.updateDB(jObj, db);
                  }else{
                        errorFlag = true;
                  }



            } catch (JSONException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        loadingDialog.dismiss();
                        errorFlag = true;
            } catch (ClientProtocolException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        loadingDialog.dismiss();
                        errorFlag = true;
                  } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        loadingDialog.dismiss();
                        errorFlag = true;
                  }


                  return errorFlag;
```

```
		}

		@Override
		protected void onPreExecute() {
			// TODO Auto-generated method stub
			super.onPreExecute();

		  loadingDialog = new ProgressDialog(TabSelection.this);

			loadingDialog.show();
			loadingDialog.setMessage("Updating Please Wait . . .");
		}

		@Override
		protected void onPostExecute(Boolean result) {
			// TODO Auto-generated method stub
			super.onPostExecute(result);

			loadingDialog.dismiss();

			if(result){

				AlertDialog.Builder adBuilder = new
AlertDialog.Builder(TabSelection.this);
				adBuilder.setTitle("ERROR");

				adBuilder.setMessage("Can't find Server")
				.setCancelable(false)
				.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

					public void onClick(DialogInterface dialog, int
which) {

						// TODO Auto-generated method stub
						dialog.cancel();
					}
				});

				AlertDialog alDialog = adBuilder.create();

				alDialog.show();
			}
		}
	}


}
```

AndroidManifest.xml

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

    <uses-sdk android:minSdkVersion="8" />
>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />



    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <uses-library android:name="com.google.android.maps"/>

        <activity
            android:name="com.chkyriacou.activities.MyFirstGoogleMap"
            android:label="GoogleMap"
            android:screenOrientation="portrait" >
            <intent-filter >
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>

        <service
            android:name="com.chkyriacou.activities.MyFirstGoogleMap"
        />

        <activity
            android:name=".activities.Intro"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

             <activity
             android:name=".activities.MainScreen"
             android:label="@string/app_name"
             android:screenOrientation="portrait"
             >

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

             <activity
```

```
            android:name=".activities.EventScreen"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            >

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


            <activity
            android:name=".activities.AlarmSetup"
            android:label="AlarmSetup"
            android:screenOrientation="portrait"
            >

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


            <activity
            android:name=".activities.Login"
            android:label="Login"
            android:screenOrientation="portrait">
                  <intent-filter >
                  <category android:name="android.intent.category.DEFAULT" />
                  </intent-filter>
            </activity>


            <activity
                  android:name=".activities.BluetoothScreen"
            android:label="BluetoothScreen"
            android:screenOrientation="portrait">
                  <intent-filter >
                        <category
android:name="android.intent.category.DEFAULT" />
                  </intent-filter>
            </activity>

            <service android:name=".services.AlarmService"
       />

            <activity android:name=".activities.TabSelection"
                    android:label="TabSelection"
                  android:screenOrientation="portrait">

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

            <activity
                  android:name=".activities.Schedule"
            android:label="DailySchedule"
            android:screenOrientation="portrait">
```

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



      </application>

      </manifest>
```

res layout

alarm_setup.xml

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

<ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_weight="0.10">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingTop="10dp"
         >


        <TextView
            android:id="@+id/tv_alarm_setup_1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dosage Time 1"
            android:visibility="gone"
            android:textSize="30sp" />

        <TimePicker
            android:id="@+id/tp_alarm_setup_1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:visibility="gone"
            android:paddingBottom="10dp" />

        <TextView
            android:id="@+id/tv_alarm_setup_2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dosage Time 2"
```

```
            android:visibility="gone"
            android:textSize="30sp" />

        <TimePicker
            android:id="@+id/tp_alarm_setup_2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:visibility="gone"
            android:paddingBottom="10dp" />

        <TextView
            android:id="@+id/tv_alarm_setup_3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dosage Time 3"
            android:visibility="gone"
            android:textSize="30sp" />

        <TimePicker
            android:id="@+id/tp_alarm_setup_3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:visibility="gone"
            android:paddingBottom="10dp" />

        <TextView
            android:id="@+id/tv_alarm_setup_4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dosage Time 4"
            android:visibility="gone"
            android:textSize="30sp" />

        <TimePicker
            android:id="@+id/tp_alarm_setup_4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:visibility="gone"
            android:paddingBottom="10dp" />

        <TextView
            android:id="@+id/tv_alarm_setup_5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dosage Time 5"
            android:visibility="gone"
            android:textSize="30dp" />

        <TimePicker
            android:id="@+id/tp_alarm_setup_5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:visibility="gone"
            android:paddingBottom="10dp" />

        <TextView
            android:id="@+id/tv_alarm_setup_6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:text="Dosage Time 6"
            android:visibility="gone"
            android:textSize="30sp" />

        <TimePicker
            android:id="@+id/tp_alarm_setup_6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:visibility="gone"
            android:paddingBottom="10dp" />
    </LinearLayout>
      </ScrollView>

      <LinearLayout
          android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="horizontal"
      android:layout_weight="0.90">
    <Button
        android:id="@+id/bt_alarm_setup_done"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
             android:layout_weight="0.50"
        android:text="Done"
        android:textSize="20sp" />

      <Button
        android:id="@+id/bt_alarm_setup_cancel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:text="Cancel"
        android:textSize="20sp" />
      </LinearLayout>

      </LinearLayout>
```

bluetooth_listitem

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



    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tv_blue_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingTop="10dp"
```

```
            android:paddingLeft="5dp"
            android:textColor="#FFFFFF"
            android:textStyle="bold"
            android:text="1234" android:textSize="30dp"/>

        <TextView
            android:id="@+id/tv_blue_address"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="5421" android:textSize="30dp"/>

    </LinearLayout>

      </LinearLayout>
```

bluetooth_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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/bt_turnbton"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Bluetooth Settings" />

    <ListView
        android:id="@+id/lv_devices"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" android:layout_weight="70">
    </ListView>
</LinearLayout>
```

event_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ll_event_background"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000"
    android:orientation="vertical" >


            <LinearLayout
                android:layout_width="match_parent"
```

```
                android:layout_height="0dip"
                android:orientation="vertical"
                android:layout_weight="1" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFFFFF"
        android:paddingBottom="1dp"
        android:gravity="top" >



            <LinearLayout
                android:layout_width="wrap_content"
           android:layout_height="match_parent"
           android:gravity="center"
           android:background="#000000">


           <ImageView
               android:id="@+id/iv_event_icon"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:src="@drawable/me" />

             </LinearLayout>


        <TextView
            android:id="@+id/tv_event_title"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#000000"
            android:gravity="center_vertical"
            android:paddingLeft="10dp"
            android:scrollHorizontally="true"
            android:text="TextView"
            android:textColor="#FFFFFF"
            android:textSize="28dp"
            android:textStyle="bold" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFFFFF"
        android:paddingBottom="1dp"
        android:paddingTop="1dp" >

        <TextView
            android:id="@+id/tv_event_date"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0.50"
            android:background="#000000"
            android:text="TextView"
```

```
            android:textColor="#FFFFFF"
            android:textSize="20dp" />

        <TextView
            android:id="@+id/tv_event_enddate"
            android:layout_width="158dp"
            android:layout_height="wrap_content"
            android:background="#000000"
            android:text="TextView"
            android:textColor="#FFFFFF"
            android:textSize="20dp"
            android:visibility="visible" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFFFFF"
        android:paddingBottom="1dp"
        android:paddingTop="1dp" >

        <TextView
            android:id="@+id/tv_event_dosage_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="#000000"
            android:text="Dosage:   "
            android:textColor="#FFFFFF"
            android:textSize="20dp" />

        <TextView
            android:id="@+id/tv_event_dosage_volume"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#000000"
            android:text="0"
            android:textColor="#FFFFFF"
            android:textSize="20dp" />
    </LinearLayout>



    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFFFF"
        android:paddingBottom="2dp"
        android:paddingTop="1dp" >



        <TextView
            android:id="@+id/tv_event_note"
            android:layout_width="match_parent"
            android:layout_height="140dp"
            android:background="#FFFFFF"
            android:scrollbars="vertical"
            android:text="@string/long_text"
```

```
            android:textColor="#000000"
            android:padding="5dp"
            android:textSize="22dp" />

    </LinearLayout>
</LinearLayout>



    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom"
        android:orientation="vertical" >

            <Button
                android:id="@+id/bt_event_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="top|center"
                android:text="Bluetooth"
                android:textSize="20dp" />

            <Button
                android:id="@+id/bt_event_send"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="top|center"
                android:text="Send Results"
                android:textSize="20dp" />

        </LinearLayout>

</LinearLayout>
```

intro.xml

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom"
        android:paddingTop="20dp"
        android:text="@string/author"
        android:textColor="#FFFFFF"
        android:textSize="30dp"/>

        <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
        android:text="@string/app_name"
        android:paddingTop="20dp"
        android:paddingBottom="20dp"
        android:textSize="40dp"
        android:textColor="#FFFFFF"
        android:gravity="left|center"
        android:layout_gravity="center" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="bottom"
            >

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="bottom"
            android:text="Dissertation "
            android:textColor="#FFFFFF"
            android:textSize="30dp"
             android:paddingTop="20dp"
             android:paddingBottom="1dp"
            />

             <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="30dp"
        android:gravity="center_vertical"

        android:layout_gravity="center" android:layout_weight="13"
android:text="AM: 0449" android:textColor="#FFFFFF"/>

             <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:gravity="center_vertical"
        android:paddingTop="1dp"
        android:paddingBottom="1dp"
        android:layout_gravity="center" android:layout_weight="12"
android:text="ΑΤΕΙ: ΜΕΣΟΛΟΓΓΙΟΥ" android:textColor="#FFFFFF"/>

             <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:gravity="center_vertical"
        android:paddingTop="1dp"
        android:paddingBottom="1dp"

        android:layout_gravity="center" android:layout_weight="12"
android:text="ΤΕΣΥΔ Παράτρημα Ναυπάκτου" android:textColor="#FFFFFF"/>


        </LinearLayout>
```

```
</LinearLayout>
```

listview_item_event.xml

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

       android:id = "@+id/ll_list_item_background"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:background="@drawable/list_selector"
    android:gravity="center_vertical"
>

    <ImageView
        android:id="@+id/iv_list_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/drug"
        android:contentDescription="@string/accessibility"
        android:layout_gravity="center_vertical" />


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="5dp">

        <TextView
            android:id="@+id/tv_list_item_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textSize="20sp"
            android:textColor="#FFFFFF"
             />

        <TextView
            android:id="@+id/tv_list_item_brief"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="#FFFFFF"
             />

    </LinearLayout>

</LinearLayout>
```

login_screen.xml

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

       android:id = "@+id/ll_list_item_background"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:background="@drawable/list_selector"
    android:gravity="center_vertical"
>

    <ImageView
        android:id="@+id/iv_list_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/drug"
        android:contentDescription="@string/accessibility"
        android:layout_gravity="center_vertical" />


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="5dp">

        <TextView
            android:id="@+id/tv_list_item_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textSize="20sp"
            android:textColor="#FFFFFF"
             />

        <TextView
            android:id="@+id/tv_list_item_brief"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="#FFFFFF"
             />

    </LinearLayout>

</LinearLayout>
```

main_screen.xml

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

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/lv_events"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"

        android:fastScrollEnabled="true"

        >

    </ListView>

</LinearLayout>
```

map_view.xml

```
<?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" >

    <com.google.android.maps.MapView

                          android:id="@+id/mapView"
                   android:enabled="true"
                   android:clickable="true"

                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:apiKey="0fh_I1kIJVRIwQmuFo5ww64B3aNOPEtcNO032evA"
                />

</RelativeLayout>
```

tab_selection.xml

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

```
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"/>
    </LinearLayout>
</TabHost>
```

res menu

main_menu.xml

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

    <item
        android:id="@+id/menu_map"
        android:icon="@drawable/map"
        android:title="@string/menu_map"
    />

        <item
        android:id="@+id/menu_schedule"
        android:icon="@drawable/schedule"
        android:title="@string/menu_schedule"
    />


    <item
        android:id="@+id/menu_update"
        android:icon="@drawable/wife"
        android:title="@string/menu_update"
    />



</menu>
```

map_menu.xml

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

    <item
        android:id="@+id/menu_pharmacy"
        android:title="@string/menu_pharmacy"
        android:icon="@drawable/drugstore"
    />
```

```
    <item
        android:id="@+id/menu_hospital"
        android:title="@string/menu_hospital"
        android:icon="@drawable/hospital"
    />

</menu>
```

schedule_menu.xml

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

   <item
        android:id="@+id/menu_schedule_delete"
        android:title="@string/menu_schedule_delete"
    />

</menu>
```

res values

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="light_blue">#00aaff</color>

</resources>
```

strings.xml

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

    <string name="hello">Hello World, Intro!</string>
    <string name="app_name">ProjectDoc</string>
    <string name="author">Christos Kyriacou presents</string>
    <string name="menu_map">MAP</string>
    <string name="menu_pharmacy">PHARMACY</string>
    <string name="menu_hospital">HOSPITAL</string>
    <string name="long_text">Text</string>
    <string name="menu_update">Update</string>
      <string name="menu_schedule">Daily Schedule</string>
      <string name="menu_schedule_delete">Delete All</string>
```

```
      <string name="accessibility">Accessibility</string>

</resources>
```

res drawable

list_selector.xml

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

   <item android:drawable="@color/light_blue" android:state_pressed="true"/>
   <item android:drawable="@android:color/transparent"
android:state_selected="true"/>


</selector>
```

BluetoothHDP

src com.example.bluetooth.health

BluetoothHDPActivity.java

```
/*
 * Copyright (C) 2011 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.bluetooth.health;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.content.res.Resources;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

/**
 * Main user interface for the Sample application.  All Bluetooth health-related
```

```
 * operations happen in {@link BluetoothHDPService}.  This activity passes
messages to and from
 * the service.
 */
public class BluetoothHDPActivity extends Activity {
    private static final String TAG = "BluetoothHealthActivity";

    // Use the appropriate IEEE 11073 data types based on the devices used.
    // Below are some examples.  Refer to relevant Bluetooth HDP specifications
for detail.
    //     0x1007 - blood pressure meter
    //     0x1008 - body thermometer
    //     0x100F - body weight scale
    private static final int HEALTH_PROFILE_SOURCE_DATA_TYPE = 0x1007;

    private static final int REQUEST_ENABLE_BT = 1;

    private TextView mConnectIndicator;
    private ImageView mDataIndicator;
    private TextView mStatusMessage;

    private BluetoothAdapter mBluetoothAdapter;
    private BluetoothDevice[] mAllBondedDevices;
    private BluetoothDevice mDevice;
    private int mDeviceIndex = 0;
    private Resources mRes;
    private Messenger mHealthService;
    private boolean mHealthServiceBound;

    // Handles events sent by {@link HealthHDPService}.
    private Handler mIncomingHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                // Application registration complete.
                case BluetoothHDPService.STATUS_HEALTH_APP_REG:
                    mStatusMessage.setText(
                            String.format(mRes.getString(R.string.status_reg),
                            msg.arg1));
                    break;
                // Application unregistration complete.
                case BluetoothHDPService.STATUS_HEALTH_APP_UNREG:
                    mStatusMessage.setText(
                            String.format(mRes.getString(R.string.status_unreg),
                            msg.arg1));
                    break;
                // Reading data from HDP device.
                case BluetoothHDPService.STATUS_READ_DATA:
                    mStatusMessage.setText(mRes.getString(R.string.read_data));
                    mDataIndicator.setImageLevel(1);
                    break;
                // Finish reading data from HDP device.
                case BluetoothHDPService.STATUS_READ_DATA_DONE:

mStatusMessage.setText(mRes.getString(R.string.read_data_done));
                    mDataIndicator.setImageLevel(0);
                    break;
                // Channel creation complete.  Some devices will automatically
establish
```

```
                // connection.
                case BluetoothHDPService.STATUS_CREATE_CHANNEL:
                    mStatusMessage.setText(

String.format(mRes.getString(R.string.status_create_channel),
                            msg.arg1));
                    mConnectIndicator.setText(R.string.connected);
                    break;
                // Channel destroy complete.  This happens when either the device
disconnects or
                // there is extended inactivity.
                case BluetoothHDPService.STATUS_DESTROY_CHANNEL:
                    mStatusMessage.setText(

String.format(mRes.getString(R.string.status_destroy_channel),
                            msg.arg1));
                    mConnectIndicator.setText(R.string.disconnected);
                    break;
                default:
                    super.handleMessage(msg);
            }
        }
    };

    private final Messenger mMessenger = new Messenger(mIncomingHandler);

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Check for Bluetooth availability on the Android platform.
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null) {
            Toast.makeText(this, R.string.bluetooth_not_available,
Toast.LENGTH_LONG);
            finish();
            return;
        }
        setContentView(R.layout.console);
        mConnectIndicator = (TextView) findViewById(R.id.connect_ind);
        mStatusMessage = (TextView) findViewById(R.id.status_msg);
        mDataIndicator = (ImageView) findViewById(R.id.data_ind);
        mRes = getResources();
        mHealthServiceBound = false;

        // Initiates application registration through {@link BluetoothHDPService}.
        Button registerAppButton = (Button)
findViewById(R.id.button_register_app);
        registerAppButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                sendMessage(BluetoothHDPService.MSG_REG_HEALTH_APP,
                        HEALTH_PROFILE_SOURCE_DATA_TYPE);
            }
        });

        // Initiates application unregistration through {@link
BluetoothHDPService}.
        Button unregisterAppButton = (Button)
findViewById(R.id.button_unregister_app);
```

```
        unregisterAppButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                sendMessage(BluetoothHDPService.MSG_UNREG_HEALTH_APP, 0);
            }
        });

        // Initiates channel creation through {@link BluetoothHDPService}.  Some
devices will
        // initiate the channel connection, in which case, it is not necessary to
do this in the
        // application.  When pressed, the user is asked to select from one of the
bonded devices
        // to connect to.
        Button connectButton = (Button) findViewById(R.id.button_connect_channel);
        connectButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mAllBondedDevices =
                        (BluetoothDevice[])
mBluetoothAdapter.getBondedDevices().toArray(
                                new BluetoothDevice[0]);

                if (mAllBondedDevices.length > 0) {
                    int deviceCount = mAllBondedDevices.length;
                    if (mDeviceIndex < deviceCount) mDevice =
mAllBondedDevices[mDeviceIndex];
                    else {
                        mDeviceIndex = 0;
                        mDevice = mAllBondedDevices[0];
                    }
                    String[] deviceNames = new String[deviceCount];
                    int i = 0;
                    for (BluetoothDevice device : mAllBondedDevices) {
                        deviceNames[i++] = device.getName();
                    }

                  /*  SelectDeviceDialogFragment deviceDialog =
                            SelectDeviceDialogFragment.newInstance(deviceNames,
mDeviceIndex);
                    deviceDialog.show(getFragmentManager(), "deviceDialog");
                */
                }
            }
        });

        // Initiates channel disconnect through {@link BluetoothHDPService}.
        Button disconnectButton = (Button)
findViewById(R.id.button_disconnect_channel);
        disconnectButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                disconnectChannel();
            }
        });
        registerReceiver(mReceiver, initIntentFilter());
    }

    // Sets up communication with {@link BluetoothHDPService}.
    private ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName name, IBinder service) {
            mHealthServiceBound = true;
```

```
            Message msg = Message.obtain(null,
BluetoothHDPService.MSG_REG_CLIENT);
            msg.replyTo = mMessenger;
            mHealthService = new Messenger(service);
            try {
                mHealthService.send(msg);
            } catch (RemoteException e) {
                Log.w(TAG, "Unable to register client to service.");
                e.printStackTrace();
            }
        }

        public void onServiceDisconnected(ComponentName name) {
            mHealthService = null;
            mHealthServiceBound = false;
        }
    };

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mHealthServiceBound) unbindService(mConnection);
        unregisterReceiver(mReceiver);
    }

    @Override
    protected void onStart() {
        super.onStart();
        // If Bluetooth is not on, request that it be enabled.
        if (!mBluetoothAdapter.isEnabled()) {
            Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
        } else {
            initialize();
        }
    }

    /**
     * Ensures user has turned on Bluetooth on the Android device.
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
        switch (requestCode) {
        case REQUEST_ENABLE_BT:
            if (resultCode == Activity.RESULT_OK) {
                initialize();
            } else {
                finish();
                return;
            }
        }
    }

    /**
     * Used by {@link SelectDeviceDialogFragment} to record the bonded Bluetooth
device selected
     * by the user.
```

```
     *
     * @param position Position of the bonded Bluetooth device in the array.
     */
    public void setDevice(int position) {
        mDevice = this.mAllBondedDevices[position];
        mDeviceIndex = position;
    }

    private void connectChannel() {
        sendMessageWithDevice(BluetoothHDPService.MSG_CONNECT_CHANNEL);
    }

    private void disconnectChannel() {
        sendMessageWithDevice(BluetoothHDPService.MSG_DISCONNECT_CHANNEL);
    }

    private void initialize() {
        // Starts health service.
        Intent intent = new Intent(this, BluetoothHDPService.class);
        startService(intent);
        bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
    }

    // Intent filter and broadcast receive to handle Bluetooth on event.
    private IntentFilter initIntentFilter() {
        IntentFilter filter = new IntentFilter();
        filter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
        return filter;
    }

    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();
            if (BluetoothAdapter.ACTION_STATE_CHANGED.equals(action)) {
                if (intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
BluetoothAdapter.ERROR) ==
                    BluetoothAdapter.STATE_ON) {
                    initialize();
                }
            }
        }
    };

    // Sends a message to {@link BluetoothHDPService}.
    private void sendMessage(int what, int value) {
        if (mHealthService == null) {
            Log.d(TAG, "Health Service not connected.");
            return;
        }

        try {
            mHealthService.send(Message.obtain(null, what, value, 0));
        } catch (RemoteException e) {
            Log.w(TAG, "Unable to reach service.");
            e.printStackTrace();
        }
    }
```

```
    // Sends an update message, along with an HDP BluetoothDevice object, to
    // {@link BluetoothHDPService}.  The BluetoothDevice object is needed by the
channel creation
    // method.
    private void sendMessageWithDevice(int what) {
        if (mHealthService == null) {
            Log.d(TAG, "Health Service not connected.");
            return;
        }

        try {
            mHealthService.send(Message.obtain(null, what, mDevice));
        } catch (RemoteException e) {
            Log.w(TAG, "Unable to reach service.");
            e.printStackTrace();
        }
    }

    /**
     * Dialog to display a list of bonded Bluetooth devices for user to select
from.  This is
     * needed only for channel connection initiated from the application.
     */


    public static class SelectDeviceDialogFragment extends DialogFragment {

        public static SelectDeviceDialogFragment newInstance(String[] names, int
position) {
            SelectDeviceDialogFragment frag = new SelectDeviceDialogFragment();
            Bundle args = new Bundle();
            args.putStringArray("names", names);
            args.putInt("position", position);
            frag.setArguments(args);
            return frag;
        }

        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
            String[] deviceNames = getArguments().getStringArray("names");
            int position = getArguments().getInt("position", -1);
            if (position == -1) position = 0;
            return new AlertDialog.Builder(getActivity())
                    .setTitle(R.string.select_device)
                    .setPositiveButton(R.string.ok,
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int which)
{
                                ((BluetoothHDPActivity)
getActivity()).connectChannel();
                            }
                        })
                    .setSingleChoiceItems(deviceNames, position,
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int which)
{
                                ((BluetoothHDPActivity)
getActivity()).setDevice(which);
                            }
```

```
                        }
                    )
                    .create();
        }
    }


}
```

BluetoothHDPService.java

```
/*
 * Copyright (C) 2011 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.bluetooth.health;

import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothHealth;
import android.bluetooth.BluetoothHealthAppConfiguration;
import android.bluetooth.BluetoothHealthCallback;
import android.bluetooth.BluetoothProfile;
import android.content.Intent;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.ParcelFileDescriptor;
import android.os.RemoteException;
import android.util.Log;
import android.widget.Toast;

import java.io.FileInputStream;
import java.io.IOException;

/**
 * This Service encapsulates Bluetooth Health API to establish, manage, and
disconnect
 * communication between the Android device and a Bluetooth HDP-enabled device.
Possible HDP
 * device type includes blood pressure monitor, glucose meter, thermometer, etc.
 *
```

```
 * As outlined in the
 * <a
href="http://developer.android.com/reference/android/bluetooth/BluetoothHealth.html
l">BluetoothHealth</a>
 * documentation, the steps involve:
 * 1. Get a reference to the BluetoothHealth proxy object.
 * 2. Create a BluetoothHealth callback and register an application configuration
that acts as a
 *    Health SINK.
 * 3. Establish connection to a health device.  Some devices will initiate the
connection.  It is
 *    unnecessary to carry out this step for those devices.
 * 4. When connected successfully, read / write to the health device using the
file descriptor.
 *    The received data needs to be interpreted using a health manager which
implements the
 *    IEEE 11073-xxxxx specifications.
 * 5. When done, close the health channel and unregister the application.  The
channel will
 *    also close when there is extended inactivity.
 */
public class BluetoothHDPService extends Service {
    private static final String TAG = "BluetoothHDPService";

    public static final int RESULT_OK = 0;
    public static final int RESULT_FAIL = -1;

    // Status codes sent back to the UI client.
    // Application registration complete.
    public static final int STATUS_HEALTH_APP_REG = 100;
    // Application unregistration complete.
    public static final int STATUS_HEALTH_APP_UNREG = 101;
    // Channel creation complete.
    public static final int STATUS_CREATE_CHANNEL = 102;
    // Channel destroy complete.
    public static final int STATUS_DESTROY_CHANNEL = 103;
    // Reading data from Bluetooth HDP device.
    public static final int STATUS_READ_DATA = 104;
    // Done with reading data.
    public static final int STATUS_READ_DATA_DONE = 105;

    // Message codes received from the UI client.
    // Register client with this service.
    public static final int MSG_REG_CLIENT = 200;
    // Unregister client from this service.
    public static final int MSG_UNREG_CLIENT = 201;
    // Register health application.
    public static final int MSG_REG_HEALTH_APP = 300;
    // Unregister health application.
    public static final int MSG_UNREG_HEALTH_APP = 301;
    // Connect channel.
    public static final int MSG_CONNECT_CHANNEL = 400;
    // Disconnect channel.
    public static final int MSG_DISCONNECT_CHANNEL = 401;

    private BluetoothHealthAppConfiguration mHealthAppConfig;
    private BluetoothAdapter mBluetoothAdapter;
    private BluetoothHealth mBluetoothHealth;
    private BluetoothDevice mDevice;
```

```
    private int mChannelId;

    private Messenger mClient;

    // Handles events sent by {@link HealthHDPActivity}.
    private class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                // Register UI client to this service so the client can receive
messages.
                case MSG_REG_CLIENT:
                    Log.d(TAG, "Activity client registered");
                    mClient = msg.replyTo;
                    break;
                // Unregister UI client from this service.
                case MSG_UNREG_CLIENT:
                    mClient = null;
                    break;
                // Register health application.
                case MSG_REG_HEALTH_APP:
                    registerApp(msg.arg1);
                    break;
                // Unregister health application.
                case MSG_UNREG_HEALTH_APP:
                    unregisterApp();
                    break;
                // Connect channel.
                case MSG_CONNECT_CHANNEL:
                    mDevice = (BluetoothDevice) msg.obj;
                    connectChannel();
                    break;
                // Disconnect channel.
                case MSG_DISCONNECT_CHANNEL:
                    mDevice = (BluetoothDevice) msg.obj;
                    disconnectChannel();
                    break;
                default:
                    super.handleMessage(msg);
            }
        }
    }

    final Messenger mMessenger = new Messenger(new IncomingHandler());

    /**
     * Make sure Bluetooth and health profile are available on the Android device.
Stop service
     * if they are not available.
     */
    @Override
    public void onCreate() {
        super.onCreate();
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null || !mBluetoothAdapter.isEnabled()) {
            // Bluetooth adapter isn't available.  The client of the service is
supposed to
            // verify that it is available and activate before invoking this
service.
```

```
            stopSelf();
            return;
        }
        if (!mBluetoothAdapter.getProfileProxy(this, mBluetoothServiceListener,
                BluetoothProfile.HEALTH)) {
            Toast.makeText(this, R.string.bluetooth_health_profile_not_available,
                    Toast.LENGTH_LONG);
            stopSelf();
            return;
        }
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, "BluetoothHDPService is running.");
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mMessenger.getBinder();
    };

    // Register health application through the Bluetooth Health API.
    private void registerApp(int dataType) {
        mBluetoothHealth.registerSinkAppConfiguration(TAG, dataType,
mHealthCallback);
    }

    // Unregister health application through the Bluetooth Health API.
    private void unregisterApp() {
        mBluetoothHealth.unregisterAppConfiguration(mHealthAppConfig);
    }

    // Connect channel through the Bluetooth Health API.
    private void connectChannel() {
        Log.i(TAG, "connectChannel()");
        mBluetoothHealth.connectChannelToSource(mDevice, mHealthAppConfig);
    }

    // Disconnect channel through the Bluetooth Health API.
    private void disconnectChannel() {
        Log.i(TAG, "disconnectChannel()");
        mBluetoothHealth.disconnectChannel(mDevice, mHealthAppConfig, mChannelId);
    }

    // Callbacks to handle connection set up and disconnection clean up.
    private final BluetoothProfile.ServiceListener mBluetoothServiceListener =
            new BluetoothProfile.ServiceListener() {
        public void onServiceConnected(int profile, BluetoothProfile proxy) {
            if (profile == BluetoothProfile.HEALTH) {
                mBluetoothHealth = (BluetoothHealth) proxy;
                if (Log.isLoggable(TAG, Log.DEBUG))
                    Log.d(TAG, "onServiceConnected to profile: " + profile);
            }
        }

        public void onServiceDisconnected(int profile) {
            if (profile == BluetoothProfile.HEALTH) {
```

```
                mBluetoothHealth = null;
            }
        }
    };

    private final BluetoothHealthCallback mHealthCallback = new
BluetoothHealthCallback() {
        // Callback to handle application registration and unregistration events.
The service
        // passes the status back to the UI client.
        public void
onHealthAppConfigurationStatusChange(BluetoothHealthAppConfiguration config,
                int status) {
            if (status == BluetoothHealth.APP_CONFIG_REGISTRATION_FAILURE) {
                mHealthAppConfig = null;
                sendMessage(STATUS_HEALTH_APP_REG, RESULT_FAIL);
            } else if (status == BluetoothHealth.APP_CONFIG_REGISTRATION_SUCCESS)
{
                mHealthAppConfig = config;
                sendMessage(STATUS_HEALTH_APP_REG, RESULT_OK);
            } else if (status == BluetoothHealth.APP_CONFIG_UNREGISTRATION_FAILURE
||
                    status == BluetoothHealth.APP_CONFIG_UNREGISTRATION_SUCCESS) {
                sendMessage(STATUS_HEALTH_APP_UNREG,
                        status ==
BluetoothHealth.APP_CONFIG_UNREGISTRATION_SUCCESS ?
                        RESULT_OK : RESULT_FAIL);
            }
        }

        // Callback to handle channel connection state changes.
        // Note that the logic of the state machine may need to be modified based
on the HDP device.
        // When the HDP device is connected, the received file descriptor is
passed to the
        // ReadThread to read the content.
        public void onHealthChannelStateChange(BluetoothHealthAppConfiguration
config,
                BluetoothDevice device, int prevState, int newState,
ParcelFileDescriptor fd,
                int channelId) {
            if (Log.isLoggable(TAG, Log.DEBUG))
                Log.d(TAG, String.format("prevState\t%d ----------> newState\t%d",
                        prevState, newState));
            if (prevState == BluetoothHealth.STATE_CHANNEL_DISCONNECTED &&
                    newState == BluetoothHealth.STATE_CHANNEL_CONNECTED) {
                if (config.equals(mHealthAppConfig)) {
                    mChannelId = channelId;
                    sendMessage(STATUS_CREATE_CHANNEL, RESULT_OK);
                    (new ReadThread(fd)).start();
                } else {
                    sendMessage(STATUS_CREATE_CHANNEL, RESULT_FAIL);
                }
            } else if (
                    prevState == BluetoothHealth.STATE_CHANNEL_CONNECTING &&
                       newState == BluetoothHealth.STATE_CHANNEL_DISCONNECTED) {
                sendMessage(STATUS_CREATE_CHANNEL, RESULT_FAIL);
            } else if (newState == BluetoothHealth.STATE_CHANNEL_DISCONNECTED) {
                if (config.equals(mHealthAppConfig)) {
```

```
                    sendMessage(STATUS_DESTROY_CHANNEL, RESULT_OK);
                } else {
                    sendMessage(STATUS_DESTROY_CHANNEL, RESULT_FAIL);
                }
            }
        }
    };

    // Sends an update message to registered UI client.
    private void sendMessage(int what, int value) {
        if (mClient == null) {
            Log.d(TAG, "No clients registered.");
            return;
        }

        try {
            mClient.send(Message.obtain(null, what, value, 0));
        } catch (RemoteException e) {
            // Unable to reach client.
            e.printStackTrace();
        }
    }

    // Thread to read incoming data received from the HDP device.  This sample
application merely
    // reads the raw byte from the incoming file descriptor.  The data should be
interpreted using
    // a health manager which implements the IEEE 11073-xxxxx specifications.
    private class ReadThread extends Thread {
        private ParcelFileDescriptor mFd;

        public ReadThread(ParcelFileDescriptor fd) {
            super();
            mFd = fd;
        }

        @Override
        public void run() {
            FileInputStream fis = new FileInputStream(mFd.getFileDescriptor());
            final byte data[] = new byte[8192];
            try {
                while(fis.read(data) > -1) {
                    // At this point, the application can pass the raw data to a
parser that
                    // has implemented the IEEE 11073-xxxxx specifications.
Instead, this sample
                    // simply indicates that some data has been received.
                    sendMessage(STATUS_READ_DATA, 0);
                }
            } catch(IOException ioe) {}
            if (mFd != null) {
                try {
                    mFd.close();
                } catch (IOException e) { /* Do nothing. */ }
            }
            sendMessage(STATUS_READ_DATA_DONE, 0);
        }
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Copyright (C) 2011 The Android Open Source Project

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.bluetooth.health"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="14"
              android:targetSdkVersion="14" />
    <uses-permission android:name="android.permission.BLUETOOTH" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".BluetoothHDPActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".BluetoothHDPService" />
    </application>
</manifest>
```

res layout

console.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2011 The Android Open Source Project

     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at
```

```
          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
     See the License for the specific language governing permissions and
     limitations under the License.
-->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <LinearLayout android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="5dp">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/app_registration"
            android:textSize="20sp"
            android:textStyle="bold" />
        <LinearLayout android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:gravity="center" >
            <Button android:id="@+id/button_register_app"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:minWidth="130dp"
                android:text="@string/register" />
            <Button android:id="@+id/button_unregister_app"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:minWidth="130dp"
                android:text="@string/unregister" />
        </LinearLayout>

        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/channel_connection"
            android:textSize="20sp"
            android:textStyle="bold" />
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/channel_connection_desc"
            android:textSize="14sp" />
        <LinearLayout android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:gravity="center" >
            <Button android:id="@+id/button_connect_channel"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:minWidth="130dp"
                android:text="@string/connect" />
            <Button android:id="@+id/button_disconnect_channel"
                android:layout_height="wrap_content"
```

```
                android:layout_width="wrap_content"
                android:minWidth="130dp"
                android:text="@string/disconnect" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:orientation="horizontal" >
            <TextView android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/connection_state"
                android:textSize="20sp"
                android:textStyle="bold" />
            <Space android:layout_width="10dp"
                android:layout_height="0px" />
            <TextView android:id="@+id/connect_ind"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/disconnected"
                android:textSize="18sp"/>
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:orientation="horizontal" >
            <TextView android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/data_ind"
                android:textSize="20sp"
                android:textStyle="bold" />
            <Space android:layout_width="10dp"
                android:layout_height="0px" />
            <ImageView android:id="@+id/data_ind"
                android:layout_width="20dp"
                android:layout_height="20dp"
                android:layout_marginTop="4dp"
                android:src="@drawable/led_indicator" />
        </LinearLayout>
        <View android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#FFFFFF"/>
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:text="@string/status_msg"
            android:textSize="20sp"
            android:textStyle="bold" />
        <TextView android:id="@+id/status_msg"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/empty"
            android:textSize="18sp"
            android:layout_margin="10dp"/>
    </LinearLayout>
</ScrollView>
```

res values

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Copyright (C) 2011 The Android Open Source Project

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<resources>
    <string name="app_name">HDP Sample</string>
    <string name="app_registration">App Registration</string>
    <string name="bluetooth_not_available">Bluetooth is not available</string>
    <string name="bluetooth_health_profile_not_available">Bluetooth health profile
is not available</string>
    <string name="channel_connection">Channel Connection</string>
    <string name="channel_connection_desc">(some devices automatically initiate
connection)</string>
    <string name="connect">Connect</string>
    <string name="connected">CONNECTED</string>
    <string name="connection_state">Connection State</string>
    <string name="data_ind">Data Indicator</string>
    <string name="disconnect">Disconnect</string>
    <string name="disconnected">DISCONNECTED</string>
    <string name="empty"> </string>
    <string name="none">None...</string>
    <string name="ok">Okay</string>
    <string name="read_data">Reading data...</string>
    <string name="read_data_done">Done with reading data...</string>
    <string name="register">Register</string>
    <string name="select_device">Select a device</string>
    <string name="status_create_channel">Create channel status: %d</string>
    <string name="status_destroy_channel">Destroy channel status: %d</string>
    <string name="status_msg">Status Message</string>
    <string name="status_reg">App registration status: %d</string>
    <string name="status_unreg">App unregistration status: %d</string>
    <string name="unregister">Unregister</string>
</resources>
```

Βιβλιογραφία

Reto Meier “Professional Android 2 Application Development”

James Steele Nelson Addison-Wesley “The Android Developer's Cookbook”

Ιστοσελίδες

Android Developers, The Developer’s Guide:
http://developer.android.com/guide/components/index.html

Google Developers:
https://developers.google.com/android/

Map icons collection
http://mapicons.nicolasmollet.com/

ΧΡΥΣΟΣ ΟΔΗΓΟΣ ΕΛΛΑΔΑ
www.xo.gr

Android Intents - Tutorial
Copyright © 2009, 2010, 2011, 2012 Lars Vogel
http://www.vogella.com/articles/AndroidIntent/article.html

stackoverflow
http://stackoverflow.com

How Does Works
@1998 Smithsonian Istitution.
http://airandspace.si.edu/gps/work.html

Understanding Android: Three Ways to Find Your Location
http://blog.wadeburch.com/2012/03/understanding-android-three-ways-to-find-your-location.html

Android Location Providers gps, network, passive
http://android10.org/index.php/articleslocationmaps/226-android-location-providers-gps-network-passive

YouTube.com User profgustin
http://www.youtube.com/watch?v=X-Z4TiJTtlY&feature=plcp

Customizing Android ListView Items with Custom ArrayAdapter
Ravi Tamada
http://www.ezzylearning.com/tutorial.aspx?tid=1763429

Tecfrut Bioquimica
http://www.tbagrosensor.com/en/sink-node.html

Android JSON Parsing Tutorial
http://www.androidhive.info/2012/01/android-json-parsing-tutorial/