ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΥΠΗΡΕΣΙΩΝ ΒΑΣΕΙ ΘΕΣΗΣ ΜΕ ΑΝΑΛΥΣΗ ΤΟΥ ΠΛΑΙΣΙΟΥ ΧΡΗΣΤΗ ΣΕ ΣΥΣΤΗΜΑΤΑ ΚΙΝΗΤΗΣ ΥΠΟΛΟΓΙΣΤΙΚΗΣ

Μακρής Νικόλαος

Α.Μ.17

Επιβλέποντες Καθηγητές:

Κουτκιάς Βασίλειος

Αγγελίδης Παντελής

Ιούλιος 2013

Πρόλογος

Η διπλωματική εργασία αφορά στην ανάπτυξη υπηρεσιών βάσει θέσης με ανάλυση του πλαισίου χρήστη σε συστήματα κινητής υπολογιστικής. Ασχολείται με τις ανάγκες και τις ιδιαιτερότητες της τρίτης ηλικίας και στοχεύει στην ανάπτυξη υπηρεσιών που θα παρέχονται μέσω μιας εφαρμογής κινητού τηλεφώνου, οι οποίες θα επιβλέπουν και θα εποπτεύουν τον χρήστη σε διάφορα σενάρια. Ειδικότερα, το προφίλ του χρήστη-στόχου θα αφορά ηλικιωμένους που πάσχουν από άνοια και έχουν μειωμένες ικανότητες συγκέντρωσης και αντίληψης. Η εφαρμογή προς ανάπτυξη έχει ως στόχο την αναγνώριση των μεταβλητών του περιβάλλοντος του χρήστη μέσω τεχνικών αναγνώρισης πλαισίου, και την εκτέλεση συγκεκριμένων ενεργειών ως αντίδραση σε συγκεκριμένα σενάρια έκτακτης ανάγκης.

Η παρούσα εργασία (η οποία χωρίζεται στο γενικό μέρος όπου επισημαίνονται οι γενικές έννοιες και οι ορισμοί που αφορούν το θέμα, και στο ειδικό, όπου γίνεται η παρουσίαση της εφαρμογής) συνοδεύεται από τον κώδικα της εφαρμογής που αναπτύχθηκε, και αποτελεί τεκμηρίωσή του.

Σημειώνεται πως η εφαρμογή (που αναπτύχθηκε με βάση τη δημοφιλή πλατφόρμα για κινητά τηλέφωνα Android) αποτελεί προϊόν ερευνητικής δουλειάς και δεν αποσκοπεί με τη συγκεκριμένη μορφή της σε εμπορική ή παρόμοιου είδους εκμετάλλευση.

Στο σημείο αυτό, θα πρέπει να αποδοθούν ευχαριστίες στον κύριο Αγγελίδη Παντελή, Αναπληρωτή καθηγητή του Τμήματος Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, ως επιβλέπων καθηγητής της διπλωματικής, αλλά και τον Δρ. Κουτκιά Βασίλειο, εντεταλμένο καθηγητή του Τμήματος Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, ο οποίος αποτέλεσε τον εμπνευστή του θέματος της διπλωματικής εργασίας, τον κύριο πάροχο πληροφοριών και πηγών και επιπλέον τον καθοδηγητή καθ' όλη τη διάρκεια ανάπτυξης και επεξεργασίας της διπλωματικής.

Περιεχόμενα

Κατάλογος Ακρωνύμιων 5
1. Εισαγωγή 6
2. Κινητή Υπολογιστική και Υπηρεσίες Θέσης 8
2.1 Κινητή Υπολογιστική 8
2.2 Υπηρεσίες Θέσης 11
3. Η έννοια του πλαισίου στην Κινητή Υπολογιστική 13
3.1 Πλαίσιο (Context) 13
3.2 Επίγνωση Πλαισίου(Context Awareness) 13
3.3 Μέθοδοι Μοντελοποίησης και Αναπαράστασης Πλαισίου 14
4. Συστήματα Κινητής Υπολογιστικής με Επίγνωση Πλαισίου 17
4.1 Ubiquitous (or Pervasive) Computing (Διάχυτη Υπολογιστική) 17
4.2 Δομή και Αρχιτεκτονική Συστημάτων με Επίγνωση Πλαισίου 17
4.3 Use Case (Παράδειγμα): Desk Sharing Application 20
5. Σχεδίαση Πρότυπης Εφαρμογής για Άτομα με Άνοια 23
5.1 Εισαγωγή – Η Τρίτη ηλικία στο Μικροσκόπιο της Σύγχρονης Υπολογιστικής 23
5.2 Η Εφαρμογή **User Monitoring** 23
5.3 Use Cases (Σενάρια Χρήσης) 26
5.4 Σχεδιασμός Οθονών Εφαρμογής 27
5.5 Σκοπός Σχεδιασμού και Λειτουργία στο Παρασκήνιο 31
6. Αναπαράσταση Πλαισίου με Οντολογία 32
6.1 Σχεδίαση Οντολογίας 32
6.2 Παρουσίαση της οντολογίας μέσω του Protégé 34
6.3 Υιοθέτηση Οντολογικής Προσέγγισης 38
7. Εργαλεία και Τεχνολογίες Υλοποίησης 39
7.1 Protégé 39
7.2 OWL (Ontology Web Language) 40
7.3 Android 40
7.4 XML (Extensible Markup Language) 42

7.5 GPS (Global Positioning System) 42
7.6 Accelerometer 43
7.7 Prototyper 44
8. Παρουσίαση εφαρμογής 46
8.1 Από τη Σχεδίαση στην Υλοποίηση 46
8.2 Δραστηριότητες/Υπηρεσίες/Διεπαφή (Xml) 47
8.3 Οθόνη Καλωσορίσματος 48
8.4 Αρχική Οθόνη 49
8.5 Ενεργοποίηση/Απενεργοποίηση Υπηρεσιών και Είσοδος στις Προηγμένες Ρυθμίσεις52
8.6 Ατομικές Ρυθμίσεις Υπηρεσιών 54
8.7 Η Υπηρεσία Out of Range 57
8.8 Η Υπηρεσία Fall Detection 61
8.8 Η Υπηρεσία Loss Detection 62
8.9 Διεπαφές Χρήστη (XML) 64
8.10 Αρχείο Δήλωσης (Android Manifest) 64
9. Μελλοντικές Επεκτάσεις – Συμπεράσματα 65
Βιβλιογραφία 68
Παράρτημα: Κώδικας Android 70

Κατάλογος Ακρωνύμιων

AVD: Android Virtual Device (Εικονική Συσκευή Android)

GPS: Geographical Positioning System (Γεωγραφικό Σύστημα Εντοπισμού Θέσης)

GSM: Παγκόσμιο Σύστημα Κινητών Επικοινωνιών

IDE: Integrated Development Environment (Ολοκληρωμένο Περιβάλλον Ανάπτυξης)

JDK: Java Development Kit (Εργαλεία Ανάπτυξης σε Java)

LBS: Location-Based Services (Υπηρεσίες Βασισμένες στη Θέση)

OWL: Ontology Web Language (Διαδικτυακή Γλώσσα Αναπαράστασης Οντολογίας)

PDA: Personal Digital Assistant (Προσωπικός Ψηφιακός Βοηθός)

SDK: Software Development Kit (Εργαλεία Ανάπτυξης Λογισμικού)

SMS: Short Message Service (Υπηρεσία Σύντομων Μηνυμάτων)

VoIP: Voice over IP (Τηλεφωνία Μέσω Διαδικτύου)

XML: Extensible Markup Language (Επεκτάσιμη Γλώσσα Σήμανσης)

WAP: Wireless Application Protocol (Ασύρματο Πρωτόκολλο Εφαρμογών)

Γενικό Μέρος

1. Εισαγωγή

Η βελτίωση της καθημερινότητας και της ποιότητας ζωής των ανθρώπων της τρίτης ηλικίας αποτελεί σημαντικό πεδίο εφαρμογής των Τεχνολογιών Πληροφορικής και Επικοινωνιών (ΤΠΕ). Με την ραγδαία άνοδο του προσδόκιμου μέσου όρου ζωής τα τελευταία είκοσι χρόνια ειδικά στις χώρες της Ευρώπης και της Αμερικής, η χρήση της τεχνολογίας για την εποπτεία της καθημερινότητας αυτών των ανθρώπων μπορεί να διασφαλίσει την ευημερία τους και σε πολλές περιπτώσεις την υγεία τους.

Υπάρχουν πολλές παθήσεις και ασθένειες που παρουσιάζονται σε ηλικιωμένους, οι οποίες περιορίζουν την ποιότητα ζωής τους. Οι καρδιαγγειακές παθήσεις καθώς και η νόσος Altzheimer και άλλες μορφές άνοιας είναι μόνο μερικές από τις παθήσεις της τρίτης ηλικίας που χρήζουν ειδικής μεταχείρισης. Η προσαρμογή, λοιπόν, του συστήματος υγείας στη γήρανση του πληθυσμού απαιτεί και τη συνεισφορά της τεχνολογίας και δη των σύγχρονων υπολογιστών και smartphones. Πώς μπορεί να συμβεί αυτό;

Μία προσέγγιση στο ζήτημα αυτό επιχειρήθηκε στα πλαίσια της παρούσας εργασίας. Πιο συγκεκριμένα, αναπτύχθηκε μια εφαρμογή κινητού smartphone, η οποία δίνει τη δυνατότητα παρακολούθησης-επίβλεψης ασθενών της τρίτης ηλικίας που αντιμετωπίζουν προβλήματα άνοιας και με διάφορες τεχνικές επιχειρεί την πρόληψη και επίβλεψη κάποιων επιβλαβών καταστάσεων για την υγεία τους. Αυτές οι καταστάσεις συνοψίζονται ουσιαστικά σε τρία σενάρια:

1. ***Out Of Range Detection***: Ανιχνεύει εάν ο χρήστης απομακρυνθεί για ένα συγκεκριμένο εύρος από ένα συγκεκριμένο σημείο που έχει οριστεί ως αρχική τοποθεσία από τον γιατρό του ή κάποιο συγγενή του (συνήθως αυτό είναι η τοποθεσία του σπιτιού του) και ειδοποιεί κάποια οικεία πρόσωπα που έχουν προκαθοριστεί.

2. ***Fall Detection***: Ανιχνεύει εάν ο ασθενής έχει πέσει και ειδοποιεί τα οικεία πρόσωπα που έχουν προκαθοριστεί.

3. ***Loss Detection***: Ανιχνεύει εάν ο ασθενής χαθεί και ειδοποιεί πάλι κάποια ορισθέντα πρόσωπα όπως τους συγγενείς ή τον γιατρό του.

Βασική προϋπόθεση για την παροχή των παραπάνω είναι ο χρήστης να φέρει έξυπνη κινητή συσκευή (smartphone) κυρίως σε εξωτερικούς χώρους με εγκατεστημένη την προτεινόμενη εφαρμογή. Η υλοποίηση των παραπάνω υπηρεσιών επιτυγχάνεται μέσω της αναγνώρισης του πλαισίου χρήστη (context) και της «αντίδρασης» σε κάποια ερεθίσματα που θα δίνονται από το εκάστοτε πλαίσιο, χωρίς την ανάγκη να παρεμβαίνει ο χρήστης. Τα ερεθίσματα αυτά θα λαμβάνονται από διάφορους αισθητήρες και θα αποστέλλονται ως είσοδοι στην εφαρμογή, η οποία θα προβαίνει στις απαιτούμενες από τον σχεδιασμό ενέργειες. Η επίβλεψη που αναφέραμε επιτυγχάνεται κυρίως με τη χρήση υπηρεσιών θέσης, οι οποίες θα αναλυθούν στη συνέχεια. Θα παρουσιαστούν όλα τα βήματα ανάπτυξης της εφαρμογής, από τον σχεδιασμό της μέσα από διάφορα υπολογιστικά εργαλεία και σχεδιαγράμματα, μέχρι την υλοποίηση του κώδικα στην πλέον δημοφιλή πλατφόρμα για κινητά τηλέφωνα-smartphones παγκοσμίως, το Android.

Η εργασία έχει την ακόλουθη δομή: Στο Κεφάλαιο 2, γίνεται μια εισαγωγή στις έννοιες της Κινητής Υπολογιστικής και των Υπηρεσιών Θέσης (όπως, για παράδειγμα, το GPS). Στο Κεφάλαιο 3 εξηγείται η έννοια του πλαισίου και πώς αυτό αναπαριστάται, ενώ στο Κεφάλαιο 4 γίνεται αναφορά στα συστήματα κινητής υπολογιστικής με επίγνωση πλαισίου και παρουσιάζεται ως παράδειγμα μια τέτοιου είδους εφαρμογή. Στη συνέχεια, προχωρώντας στο ειδικό μέρος της εργασίας, στο Κεφάλαιο 5 αναπτύσσεται η διαδικασία σχεδίασης της πρότυπης εφαρμογής για άτομα με άνοια, με αναλυτική περιγραφή της και παρουσίαση διαφόρων σεναρίων χρήσης. Στο Κεφάλαιο 6, παρουσιάζεται (με τη βοήθεια σχεδιαγράμματος) η οντολογία της εφαρμογής και η σημασία χρήση της, ενώ στο Κεφάλαιο 7 αναφέρονται οι τεχνολογίες και τα εργαλεία υλοποίησης που χρησιμοποιήθηκαν για την εφαρμογή. Ακολούθως, στο Κεφάλαιο 8 γίνεται η παρουσίαση της εφαρμογής, με τη βοήθεια screenshots που πάρθηκαν από τις οθόνες της εφαρμογής αλλά και σημαντικά κομμάτια προγραμματιστικού κώδικα. Τέλος, στο Κεφάλαιο 9 καταγράφονται τα συμπεράσματα αλλά και πιθανές μελλοντικές επεκτάσεις της εφαρμογής.

2. Κινητή Υπολογιστική και Υπηρεσίες Θέσης

2.1 Κινητή Υπολογιστική

Η κινητή υπολογιστική (**mobile computing**) αποτελεί έναν κλάδο της πληροφορικής που ασχολείται με την ανάπτυξη υπολογιστικών συστημάτων τα οποία έχουν ως χαρακτηριστικά το μικρό μέγεθος και την φορητότητα. Αυτά τα υπολογιστικά συστήματα, ειδικά τον τελευταίο καιρό, έχουν επικαλύψει την αγορά των κινητών τηλεφώνων και αποτελούν σύμφωνα με τους ειδικούς το "next big thing" της παγκόσμιας αγοράς πληροφορικής.

Τι είναι όμως αυτό που τα κάνει τόσο δημοφιλή; Ο κυριότερος και ο πιο απλός λόγος είναι ο εξής: ο χρήστης μπορεί να συνδυάσει την χρηστικότητα και το πρακτικό μέγεθος του κινητού τηλεφώνου με τις τεράστιες δυνατότητες χρήσης, επεξεργασίας και ανταλλαγής δεδομένων και υπηρεσιών που παρέχει ένας υπολογιστής. Η εξέλιξη του hardware έχει επιτρέψει αυτή τη στιγμή σε μια συσκευή στο μέγεθος ενός κινητού τηλεφώνου να έχει αντίστοιχες υπολογιστικές ταχύτητες με ένα PC ηλικίας 2-3 ετών! Έτσι ο χρήστης ενός smartphone (κυριότερος αντιπρόσωπος συσκευών mobile computing) μπορεί να συνδέεται στο διαδίκτυο, να ελέγχει το email του, να εγκαθιστά και να χρησιμοποιεί χιλιάδες εφαρμογές (όπως κειμενογράφο, παιχνίδια, προγράμματα αναπαραγωγής ταινιών) και προφανώς να πραγματοποιεί κλήσεις και να αποστέλλει μηνύματα, και όλα αυτά να γίνονται μέσα από το κινητό του τηλέφωνο!

Εικόνα 1: Galaxy Nexus, η ναυαρχίδα της Google στα Android smartphones[1].

Χαρακτηριστικά

Γενικά, τα συστήματα κινητής υπολογιστικής έχουν τα ακόλουθα χαρακτηριστικά:

Κινητικότητα χρήστη (user mobility): Ο χρήστης θα πρέπει να μπορεί να μετακινείται από τη μία φυσική τοποθεσία σε μια άλλη και να χρησιμοποιεί την ίδια υπηρεσία. Η υπηρεσία θα μπορούσε να βρίσκεται στο οικιακό δίκτυο ή σε ένα απομακρυσμένο δίκτυο. Για παράδειγμα, ένας χρήστης θα μπορούσε να μετακινηθεί από το Λονδίνο στη Νέα Υόρκη και να χρησιμοποιεί το Διαδίκτυο, για να έχει πρόσβαση σε μια εταιρική εφαρμογή, με τον ίδιο τρόπο που το χρησιμοποιεί στο γραφείο ή στο σπίτι.

Κινητικότητα δικτύου (network mobility): Ο χρήστης θα πρέπει να μπορεί να μετακινείται από το ένα δίκτυο σε κάποιο άλλο και να κάνει χρήση της ίδιας υπηρεσίας. Για παράδειγμα, ένας χρήστης που μετακινείται συχνά να μπορεί να χρησιμοποιεί το ίδιο τηλέφωνο GSM για να έχει πρόσβαση σε μια εταιρική εφαρμογή μέσω WAP (Wireless Application Protocol) σε ένα μέρος, ενώ σε ένα άλλο να μπορεί να έχει πρόσβαση στην ίδια εφαρμογή μέσω GPRS (General Packet Radio Service).

Κινητικότητα φορέα (bearer mobility): Ο χρήστης θα πρέπει να μπορεί να μετακινείται από τον ένα φορέα στον άλλο και να χρησιμοποιεί την ίδια υπηρεσία.

Κινητικότητα συσκευής (device mobility): Ο χρήστης θα πρέπει να μπορεί να μετακινείται από τη μία συσκευή στην άλλη και να χρησιμοποιεί την ίδια υπηρεσία. Για παράδειγμα, αντιπρόσωποι πωλήσεων που χρησιμοποιούν επιτραπέζιο υπολογιστή στο οικιακό τους γραφείο, κατά τη διάρκεια της ημέρας, όταν βρίσκονται στο δρόμο, θα ήθελαν να χρησιμοποιήσουν το tablet τους για να έχουν πρόσβαση σε μια εφαρμογή πωλήσεων.

Κινητικότητα συνεδρίας (session mobility): Η συνεδρία ενός χρήστη με μια υπηρεσία θα πρέπει να μπορεί να μετακινηθεί από ένα περιβάλλον χρήσης σε ένα άλλο και να μπορεί να γίνεται ανάκτησή της σε περίπτωση διακοπής της επικοινωνίας.

Κινητικότητα υπηρεσίας (service mobility): Ο χρήστης θα πρέπει να μπορεί να μεταβαίνει από μία υπηρεσία σε μια άλλη. Για παράδειγμα, έστω ότι ένας χρήστης γράφει ένα μήνυμα. Για να το ολοκληρώσει πρέπει να ανοίξει μια εφαρμογή για να συλλέξει κάποιες πληροφορίες. Σε έναν επιτραπέζιο υπολογιστή, ο χρήστης ανοίγει διάφορες εφαρμογές και τις εναλλάσσει χρησιμοποιώντας τη γραμμή εργασιών· αυτό πρέπει να συμβαίνει και σε συσκευές κινητής υπολογιστικής.

Οι κυριότερες κατηγορίες συσκευών κινητής υπολογιστικής είναι οι εξής:

1. Smartphones, που όπως αναφέραμε παραπάνω είναι ίσως ο πιο δημοφιλής εκπρόσωπος. Οι συσκευές αυτές, πέρα από τις κλασσικές δυνατότητες ενός κινητού (τηλέφωνο και sms), μέσω της εγκατάστασης χιλιάδων applications έχουν τη δυνατότητα για πρόσβαση σε ηλεκτρονικό ταχυδρομείο, κοινωνικά δίκτυα (facebook, twitter), ψυχαγωγία μέσω gaming κ.ά. Συνδέονται στο διαδίκτυο μέσω δικτύων δεδομένων (wi-fi, 3G/4G, κ.λπ.) και ενσωματώνουν ένα σύνολο αισθητήρων (π.χ. GPS, επιταχυνσιόμετρα, κ.λπ.) που μπορούν να φανούν ιδιαίτερα χρήσιμοι ανάλογα με την εφαρμογή και τις περιστάσεις.

2. PDA (Personal Digital Assistant): Είναι ένας υπολογιστής τσέπης με περιορισμένες δυνατότητες. Η κύρια λειτουργικότητά του είναι ο συγχρονισμός με ένα desktop PC για πρόσβαση στις επαφές, mail, σημειώσεις κ.ά. (Εικόνα 2).

3. Tablet: Έχει μεγαλύτερη οθόνη από ένα smartphone. Ως μέσο αλληλεπίδρασης ο χρήστης χρησιμοποιεί τυπικά οθόνη αφής. Η συσκευή ακολουθεί την ίδια φιλοσοφία με τα smartphones αλλά, λόγω της μεγάλης οθόνης, είναι καταλληλότερη για άλλες δραστηριότητες όπως η ανάγνωση online εφημερίδων και e-books, η προβολή ταινιών, κ.λπ. (Εικόνα 3).

4. Wearable computers: Αποτελούν υπολογιστικές συσκευές (συνοδευόμενες από αισθητήρες) που έχουν τη μορφή ρουχισμού και μπορούν να «φορεθούν» από τον χρήστη [2]. Χαρακτηριστικά τους είναι η συνέπεια (αλληλεπιδρούν συνεχώς με τον χρήστη) και η παρασκηνιακή τους λειτουργία (ο χρήστης δεν αντιλαμβάνεται συνήθως τη λειτουργία τους).

Βέβαια, πέρα από τις παραπάνω δυνατότητες των συσκευών κινητής υπολογιστικής, υπάρχουν και κάποιοι περιορισμοί οι οποίοι πρέπει να λαμβάνονται υπόψη κατά το σχεδιασμό εφαρμογών κινητής υπολογιστικής αλλά και τη λειτουργία τους. Τέτοιοι είναι:

- Bandwidth: Το mobile internet είναι κατά κανόνα πιο αργό από τις καλωδιακές συνδέσεις.
- Ασφάλεια: Οι υποκλοπές και η παρακολούθηση των δεδομένων είναι πιο εύκολο να συμβούν στις ασύρματες επικοινωνίες. Επίσης μια κινητή συσκευή είναι εύκολο να κλαπεί ή δεδομένα που εμπεριέχονται σε αυτή να καταστραφούν μερικώς ή ολικώς.
- Κατανάλωση ενέργειας: Ως επί το πλείστον οι κινητές συσκευές λειτουργούν αποκλειστικά με μπαταρία και η κατανάλωσή της επηρεάσει άμεσα την αυτονομία της συσκευής.
- Υγεία: Κακή χρήση των κινητών συσκευών (για παράδειγμα στην οδήγηση) και εκπομπή ραδιενέργειας.

Εικόνα 2: PDA με λειτουργικό Windows [3]

Εικόνα 3: Tablet με λειτουργικό Android [4]

2.2 Υπηρεσίες Θέσης

Μαζί με τη θεαματική άνοδο του mobile computing τα τελευταία χρόνια έχει ανθίσει και ένα σύνολο υπηρεσιών που ονομάζονται *υπηρεσίες θέσης* **(Location-Based Services/LBS**). Οι υπηρεσίες θέσης είναι ένα σύνολο από υπηρεσίες προγραμματιστικού επιπέδου οι οποίες χρησιμοποιούν και επεξεργάζονται δεδομένα γεωγραφικών θέσεων και τοποθεσιών ως χαρακτηριστικά ελέγχου διαφόρων εφαρμογών. Ουσιαστικά, η λειτουργία τους συνίσταται στην παροχή υπηρεσιών στον χρήστη μέσω της ανάκτησης της θέσης της συσκευής που έχουν στην κατοχή τους.

Τα παραδείγματα χρήσης των LBS είναι πολλά και μπορούν να δώσουν μια καλύτερη εικόνα για τη χρησιμότητά τους:

- Αναγνώριση τοποθεσίας ενός ανθρώπου ή αντικειμένου, όπως αναζήτηση για το πλησιέστερο ATM μιας τράπεζας ή η τοποθεσία ενός φίλου.
- Ψυχαγωγία, όπως η αναζήτηση του πλησιέστερου εστιατορίου ή καφετέριας σε μια συγκεκριμένη τοποθεσία.
- Γεωγραφική εποπτεία μεταφορικών οχημάτων (π.χ. πλοίων) και πακέτων αποστολής.
- Πλοήγηση σε διάφορες τοποθεσίες μέσω χάρτη.
- Προσωποποιημένη διαφήμιση και εμπόριο απευθυνόμενα σε πελάτες σύμφωνα με την τοποθεσία τους.
- Ενημερώσεις για την κίνηση και τον καιρό στην τοποθεσία του χρήστη.
- Ειδοποιήσεις λόγω εγγύτητας, όπως εμφάνιση friend list και «ταίριασμα» κοινών προφίλ.
- Πραγματοποίηση αυτοματοποιημένων ενεργειών λόγω εγγύτητας, όπως αυτόματο check-in σε αεροδρόμιο ή αυτόματη πληρωμή διοδίων.

Η πιο δημοφιλής υλοποίηση των LBS σήμερα είναι μέσω **GPS** (Geographical Positioning System). Το GPS βασίζεται ουσιαστικά στην επικοινωνία της κινητής συσκευής με τους

τεχνητούς δορυφόρους που περιστρέφονται σε τροχιές γύρω από την Γη και μέσω τριγωνομετρικών υπολογισμών υπολογίζει τις γεωγραφικές συντεταγμένες της συσκευής. Οι συντεταγμένες αυτές συνήθως αναπαριστώνται μέσω των latitude/longitude που αποτελούν ουσιαστικά τους άξονες x και y σε έναν επίπεδο χάρτη της Γης (κάτι που προφανώς αποτελεί μια εξιδανίκευση διότι η Γη δεν είναι επίπεδη). Το GPS είναι από τα πλέον διαδεδομένα συστήματα εντοπισμού θέσης και υπολογίζεται ότι μέχρι το 2015 το 90% των κινητών συσκευών θα έχουν ενσωματωμένο GPS.

Ο εντοπισμός μέσω GSM είναι ένας άλλος τρόπος εντοπισμού θέσης ενός χρήστη ή μιας συσκευής. Βασίζεται στην εύρεση θέσης της συσκευής σε σχέση με τον αναμεταδότη του τηλεφωνικού παρόχου του χρήστη, με τον οποίο ανταλλάσσει σήματα και σύμφωνα με τους χρόνους καθυστέρησης προσδιορίζει τη θέση του.

Το GPS και το GSM localization έχουν ένα σημαντικό μειονέκτημα: δε λειτουργούν καλά σε κλειστούς χώρους, ενώ και το σφάλμα τους (μπορεί να φτάσει τα 10-15 μέτρα) δεν είναι αποδεκτό σε μικρούς χώρους. Επομένως σε αυτές τις περιπτώσεις χρησιμοποιούνται άλλες τεχνικές εντοπισμού θέσης όπως Bluetooth και wi-fi, τα οποία λειτουργούν σε κλειστούς χώρους και έχουν περισσότερη ακρίβεια στα αποτελέσματά τους από τις outdoor τεχνικές.

3. Η έννοια του πλαισίου στην Κινητή Υπολογιστική

3.1 Πλαίσιο (Context)

Η εφαρμογή την οποία αναπτύχθηκε στην παρούσα διπλωματική στηρίζεται στην «ανάγνωση» του πλαισίου χρήστη για να εξάγει και να επεξεργαστεί δεδομένα. Τι είναι όμως το πλαίσιο χρήστη και πώς μπορεί να επηρεάσει τη λειτουργία μιας εφαρμογής κινητής υπολογιστικής;

Ως πληροφορία πλαισίου ορίζεται «οποιαδήποτε πληροφορία μπορεί να περιγράψει την κατάσταση μιας οντότητας. Οντότητα μπορεί να αποτελεί μια συσκευή, μια εφαρμογή, ή και ένας άνθρωπος» [5]. Έτσι λοιπόν ως **πλαίσιο (context)** μπορούμε να ορίσουμε οτιδήποτε περιβάλλει τον χρήστη ή τη συσκευή σε φυσικό επίπεδο: που βρίσκεται ο χρήστης, πόσοι και ποιοι είναι οι γύρω του, ποια ερεθίσματα και πληροφορίες λαμβάνει από τα γύρω αντικείμενα, κ.ά.

Όπως γίνεται αντιληπτό, το πλαίσιο είναι μια γενική έννοια και περιλαμβάνει ένα τεράστιο όγκο πληροφορίας που μπορεί να αφορά τον χρήστη. Αυτό μπορεί να σχετίζεται με πληροφορίες για το προφίλ του χρήστη (ηλικία, φύλο, ενδιαφέροντα κτλ.), για το δίκτυο που χρησιμοποιεί η συσκευή του (ταχύτητα και πάροχος), για τη συσκευή του χρήστη(επεξεργαστής, μοντέλο, χωρητικότητα, δεδομένα), για την κατάσταση του χρήστη(δραστηριότητα, ταχύτητα), για το περιβάλλον του χρήστη (τοποθεσία, θερμοκρασία, υγρασία, επίπεδο φωτεινότητας κ.ά.). Επομένως, το πλαίσιο χρήστη περιλαμβάνει μια τεράστια γκάμα πληροφοριών, οι οποίες μπορούν να χρησιμοποιηθούν προς όφελος του χρήστη όπως θα δούμε στη συνέχεια.

Αξίζει, τέλος, να τονίσουμε ότι ένα συγκεκριμένο στιγμιότυπο πλαισίου μπορεί γενικά να αναπαρασταθεί με δύο χαρακτηριστικά. Αυτά συνήθως είναι: Τύπος *πλαισίου* (context type) και τιμή πλαισίου (context value). Ο τύπος πλαισίου αναφέρεται στην κατηγορία(είδος) του πλαισίου, όπως θερμοκρασία, χρόνος, ταχύτητα, τοποθεσία κτλ. ενώ η τιμή πλαισίου αναφέρεται στα πρωτογενή δεδομένα που λαμβάνονται από αισθητήρες που αναγνωρίζουν το εκάστοτε πλαίσιο. Από εκεί και πέρα υπάρχουν και άλλα δευτερεύοντα χαρακτηριστικά, όπως το timestamp (υποδηλώνει τη χρονική στιγμή του πλαισίου) και η πηγή του πλαισίου, στοιχείο που χρησιμοποιείται από κάποιον που ενδιαφέρεται να αντλήσει πληροφορία πλαισίου από μια συγκεκριμένη πηγή.

3.2 Επίγνωση Πλαισίου(Context Awareness)

Ένα σύστημα το οποίο έχει πλήρη επίγνωση του πλαισίου του μπορεί να έχει τεράστιες δυνατότητες. Ας αναλογιστούμε πόσο εντυπωσιακό θα είναι ένα σύστημα το οποίο θα

μπορεί να αντιδρά με επιτυχία στα ερεθίσματα που δέχεται από το περιβάλλον του και να μπορεί να προσαρμόζεται με επιτυχία στις αλλαγές που συμβαίνουν γύρω του.

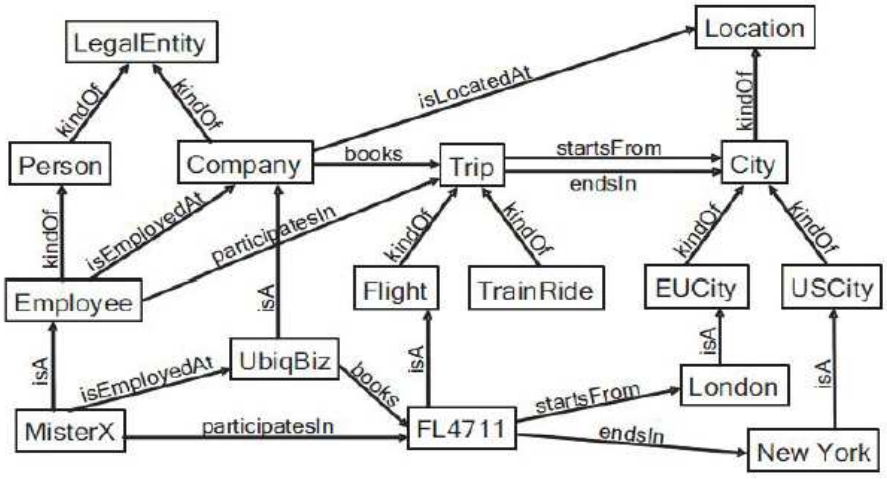
Ως επίγνωση πλαισίου (**context awareness**) ορίζεται η ικανότητα ενός συστήματος να ανακαλύπτει, να διερμηνεύει, να συμπεραίνει, να αξιοποιεί και να συλλογίζεται βάσει της περιρρέουσας πληροφορίας ώστε να λαμβάνει αποφάσεις, να προβαίνει σε προκαθορισμένες ενέργειες και να προσαρμόζεται σε διάφορες καταστάσεις. Μια σωστή εισαγωγή της επίγνωσης του πλαισίου στα σύγχρονα υπολογιστικά εργαλεία μπορεί να κάνει πολλές δραστηριότητες πιο απλές και αποδοτικές, με ελάχιστη ή και καθόλου παρέμβαση από τον χρήστη. Ένα σύστημα επίγνωσης πλαισίου πρέπει ταυτόχρονα να υιοθετεί το κατάλληλο μοντέλο αναπαράστασης πλαισίου και συμπερασμού γνώσης(context reasoning), ώστε να είναι ικανό να συλλογίζεται με βάση την πληροφορία πλαισίου που έχει ανακτήσει, να εμφανίζει μια διεισδυτική συμπεριφορά στην αυτόνομη λήψη αποφάσεων και κατ' επέκταση ενεργειών που πρέπει να διεκπεραιώνει, και τέλος να αναθεωρεί εσφαλμένες και μη επιτυχημένες ενέργειες μέσω ενός μηχανισμού προσαρμογής στον τρόπο λήψης αποφάσεων και διεκπεραίωσης ενεργειών προκειμένου να γίνει περισσότερο διεισδυτικό και προσιτό στον τελικό χρήστη.

3.3 Μέθοδοι Μοντελοποίησης και Αναπαράστασης Πλαισίου

Μέσω της μοντελοποίησης και της αναπαράστασης του πλαισίου προσπαθούμε να προσδιορίσουμε ποια είναι η καταλληλότερη πληροφορία πλαισίου (μεταβλητές, παράμετροι, γνωρίσματα) που μπορεί να αναπαραστήσει όσο το δυνατό καλύτερα την πληροφορία (χρόνος, θέση, περιβάλλον κλπ.) για ένα συγκεκριμένο επιστημονικό πεδίο, καθώς και να προσδιορίσουμε τις σχέσεις και τις εξαρτήσεις μεταξύ αυτών των μεταβλητών που αναφέραμε. Ακολουθούν διάφορες μέθοδοι αναπαράστασης πλαισίου.

Σημασιολογικά δίκτυα και πλαίσια

Ένα σημασιολογικό δίκτυο είναι ένα γράφημα του οποίου οι κόμβοι αντιπροσωπεύουν έννοιες και οι ακμές αντιπροσωπεύουν τις σχέσεις και τις αλληλεπιδράσεις μεταξύ αυτών των εννοιών. Στην Εικόνα 4 παρατίθεται ένα γράφημα που αποτελεί παράδειγμα ενός τέτοιου δικτύου.



Εικόνα 4: Παράδειγμα σημασιολογικού δικτύου για τον τομέα «επαγγελματικά ταξίδια».

Όπως βλέπουμε στο σχήμα παραπάνω, τυπικές έννοιες οι οποίες αναπαριστώνται ως κόμβοι είναι οι Company, Employee, Flight, City κτλ. Ενώ οι σχέσεις μεταξύ τους που αναπαριστώνται ως ακμές είναι isEmployedAt, startsFrom, isLocatedAt κτλ. Βλέπουμε λοιπόν ότι με αρκετά κατανοητό τρόπο (οι σχέσεις μεταξύ τους είναι άκρως επεξηγηματικές από μόνες τους), αναπαριστάται η γνώση για ένα συγκεκριμένο τομέα σε μορφή εύκολη για υπολογισμό, αντικαθιστώντας τη φυσική γλώσσα την οποία θα μπορούσε να χρησιμοποιήσει κάποιος για επεξήγηση.

Κανόνες

Μια άλλη μέθοδος αναπαράστασης γνώσης είναι οι κανόνες που αντανακλούν την έννοια της συνέπειας. Οι κανόνες έχουν την μορφή δομών IF-THEN-ELSE και επιτρέπουν την έκφραση διαφόρων ειδών πολύπλοκων καταστάσεων.

Λογική πρώτης τάξης(FOL)

Η λογική πρώτης τάξης (First-order Logic) επιτρέπει την περιγραφή του πεδίου ενδιαφέροντος και αποτελείται από λογικά και μη λογικά σύμβολα. Τα λογικά σύμβολα αντιπροσωπεύουν την ποσοτικοποίηση (quantification), τη συνεπαγωγή, τη σύζευξη και τη διάζευξη, ενώ τα μη λογικά σύμβολα είναι σταθερές, συναρτησιακά σύμβολα και μεταβλητές. Τα παραπάνω συνδυάζονται μεταξύ τους για την κατασκευή εκφράσεων οι οποίες και θα περιγράφουν τον τομέα που θέλουμε να αναπαραστήσουμε.

Οντολογίες (Ontologies)

Μια **οντολογία** είναι μια αυστηρά τυποποιημένη περιγραφή ενός πεδίου γνώσης (επιστημολογικό πεδίο) και περιλαμβάνει ένα σύνολο από όρους και συσχετίσεις μεταξύ τους. Οι όροι περιγράφουν κλάσεις αντικειμένων, δηλαδή έννοιες σχετικές με αντικείμενα, ενώ οι συσχετίσεις συνήθως αφορούν ιεραρχικές μεταβατικές εξαρτήσεις μεταξύ των όρων. Άλλες πληροφορίες που μπορεί να υπάρχουν σε μια οντολογία είναι οι ιδιότητες των εννοιών, διάφοροι περιορισμοί που μπορεί να αφορούν τις έννοιες ή τις συσχετίσεις μεταξύ τους, σχέσεις ισοδυναμίας, στιγμιότυπα των κλάσεων, καθώς και σημασιολογικοί συσχετισμοί μεταξύ των εννοιών με τη χρήση της λογικής. Η πιο κοινή και διαδεδομένη μορφή οντολογίας είναι η «ταξινόμηση» (taxonomy) μαζί με ένα σύνολο κανόνων συμπερασμού (reasoning rules). Σε μια ταξινόμηση ορίζονται κλάσεις, υποκλάσεις, οι σχέσεις μεταξύ τους και οι κανόνες συμπερασμού.

Τα πλεονεκτήματα της ανάπτυξης οντολογιών είναι τα εξής :

- Κοινή χρήση της πληροφορίας ανάμεσα σε ανθρώπους και μηχανές: Είναι ο βασικότερος λόγος ανάπτυξης μιας οντολογίας. Με τη βοήθεια μιας οντολογίας δίνεται η δυνατότητα στις εφαρμογές να μπορούν να συνδυάσουν πληροφορίες για να απαντήσουν στις αναζητήσεις του χρήστη.

- Επαναχρησιμοποίηση της γνώσης: Όταν μια οντολογία είναι καλά ορισμένη και υλοποιημένη, τότε μπορεί να επαναχρησιμοποιηθεί σε άλλες περιστάσεις για να συνδράμει στην επίλυση ενός άλλου προβλήματος.

- Σαφής ορισμός των αξιωμάτων μιας γνωστικής περιοχής: Ο σαφής ορισμός των εννοιών βοηθά τους νέους χρήστες να κατανοήσουν την υπάρχουσα γνώση και καθιστά ευκολότερες τις αλλαγές στην οντολογία.

- Διαχωρισμός της γνώσης μιας περιοχής από τη λειτουργική γνώση: Πολλές συσκευές χρησιμοποιούν την ίδια διαδικασία στη λειτουργία τους. Μια οντολογία που περιγράφει την λειτουργική διαδικασία μιας συσκευής, μπορεί να χρησιμοποιηθεί αυτούσια για κάποιες άλλες συσκευές, αν αυτές προστεθούν στην χρησιμοποιούμενη οντολογία, χωρίς όμως να χρειάζονται αλλαγές στον ορισμό της διαδικασίας.

Η οντολογία είναι και η μορφή αναπαράστασης πλαισίου που θα χρησιμοποιήσουμε στην παρούσα διπλωματική, οπότε θα επανέρθουμε στη δομή και την περιγραφή της στη συνέχεια.

4. Συστήματα Κινητής Υπολογιστικής με Επίγνωση Πλαισίου

Τα **συστήματα κινητής υπολογιστικής με επίγνωση πλαισίου (context-aware systems)** υλοποιούν την αναγνώριση του πλαισίου χρήστη που αναφέραμε προηγουμένως. Μέσα από ένα σύνολο αισθητήρων, αντλούν πληροφορίες από το περιβάλλον του χρήστη ή της συσκευής με «σκιώδη» τρόπο (χωρίς δηλαδή να το αντιλαμβάνεται ο χρήστης) και καθορίζουν τη συμπεριφορά της εφαρμογής.

4.1 Ubiquitous (or Pervasive) Computing (Διάχυτη Υπολογιστική)

Τα context-aware systems αποτελούν κομμάτι του **ubiquitous computing** (διάχυτης υπολογιστικής). Η διάχυτη υπολογιστική ουσιαστικά εκφράζει την ιδέα της ενσωμάτωσης των υπολογιστών στον φυσικό μας κόσμο. Σε ελεύθερη μετάφραση, ο όρος αυτός εκφράζει τον «πανταχού παρόντα» υπολογιστή που θα αποτελεί κομμάτι της ζωής του χρήστη και θα μπορεί να αλληλεπιδρά και να προσφέρει τις υπηρεσίες του στον χρήστη με αόρατο και συνεχή τρόπο (ο χρήστης σε πολλές περιπτώσεις θα αγνοεί και την ύπαρξη του υπολογιστή!). Η διάχυτη υπολογιστική αποτελεί για πολλούς επιστήμονες το μέλλον στην εξέλιξη των υπολογιστών και σίγουρα η εξάπλωση και οι εφαρμογές της θα μας απασχολήσουν αρκετά στο μέλλον.

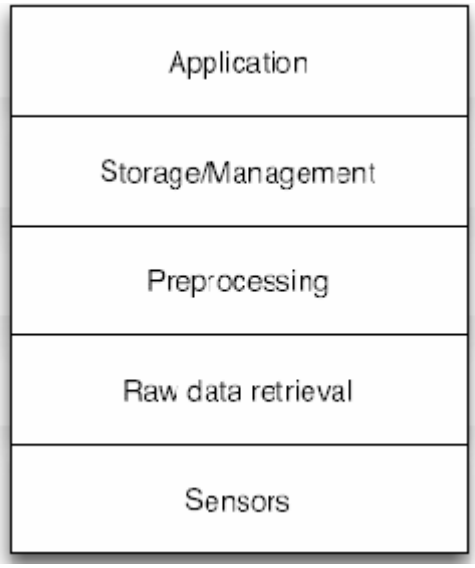
Τα context-aware συστήματα, λοιπόν, είναι ικανά να προσαρμόσουν τις ενέργειές τους στο παρόν πλαίσιο χωρίς άμεση παρέμβαση από τον χρήστη, στοχεύοντας να αυξήσουν την χρηστικότητα και την αποτελεσματικότητά τους, με το να λάβουν υπόψη το πλαίσιο του περιβάλλοντός τους. Ειδικά όταν αναφερόμαστε στη χρήση κινητών συσκευών, είναι επιθυμητό τα προγράμματα και οι υπηρεσίες να αντιδρούν με ειδικό τρόπο στην τρέχουσα τοποθεσία, ώρα και άλλες περιβαλλοντικές μεταβλητές· ενώ πρέπει να προσαρμόζουν τη συμπεριφορά τους σύμφωνα με τις περιστάσεις διότι τα δεδομένα πλαισίου μπορούν να αλλάξουν ταχύτατα (και συνήθως αυτό συμβαίνει). Η απαιτούμενη πληροφορία πλαισίου μπορεί να ανακτηθεί μέσω πολλών τρόπων, όπως αισθητήρες (sensors), πληροφορίες δικτύου, κατάσταση συσκευής, αναζήτηση σε προφίλ χρηστών και χρησιμοποιώντας άλλες πηγές.

4.2 Δομή και Αρχιτεκτονική Συστημάτων με Επίγνωση Πλαισίου

Αρχιτεκτονική

Ενώ τα context-aware συστήματα γενικά μπορούν να διαφέρουν μεταξύ τους σε κάποια κομμάτια της σχεδίασής τους, κατά βάση ακολουθείται ένα συγκεκριμένο μοντέλο

αρχιτεκτονικής με πολλαπλά στρώματα, το οποίο παρουσιάζεται στην Εικόνα 5 (επόμενη σελίδα).



Εικόνα 5: Μοντέλο αρχιτεκτονικής context-aware mobile systems [7]

Το πρώτο υπόστρωμα αφορά στους *αισθητήρες* (sensors). Οι αισθητήρες μπορούν να διακριθούν σε φυσικούς και σε εικονικούς(virtual).

Φυσικοί αισθητήρες: Εικόνα 6 – Στην αριστερή στήλη βλέπουμε τον τύπο του πλαισίου και στη δεξιά τους αντίστοιχους διαθέσιμους αισθητήρες.

Type of context	Available Sensors
Light	Photodiodes, color sensors, IR and UV-sensors etc.
Visual Context	Various cameras
Audio	Microphones
Motion, Acceleration	Mercury switches, angular sensors, accelerometers, motion detectors, magnetic fields
Location	Outdoor: Global Positioning System (GPS), Global System for Mobile Communications (GSM); Indoor: Active Badge system, etc.
Touch	Touch sensors implemented in mobile devices
Temperature	Thermometers
Physical attributes	Biosensors to measure skin resistance, blood pressure

Εικόνα 6: Τύποι πλαισίου και διαθέσιμοι αισθητήρες[7]

Εικονικοί αισθητήρες: Ανακτούν δεδομένα πλαισίου από εφαρμογές λογισμικού ή υπηρεσίες. Για παράδειγμα, ο εντοπισμός ενός εργαζόμενου μπορεί να γίνει εκτός από GPS, μέσω πρόσβασης στο ηλεκτρονικό ημερολόγιό του, στα e-mail του, κ.λπ.

Το δεύτερο υπόστρωμα είναι υπεύθυνο για την *ανάκτηση των ακατέργαστων δεδομένων πλαισίου (raw context data)*. Σε αυτό το επίπεδο χρησιμοποιούνται οδηγοί λογισμικού(drivers) για τους φυσικούς αισθητήρες και API (application programming interface) για τους εικονικούς. Επίσης, η διαδικασία της ανάκτησης υλοποιείται συχνά με επαναχρησιμοποιήσιμα κομμάτια λογισμικού, θέτοντας χαμηλού επιπέδου δεδομένα

hardware προσπελάσιμα από πιο γενικές (ή αφηρημένες/abstract) μεθόδους (παράδειγμα η getLocation() για ανάκτηση δεδομένων GPS).

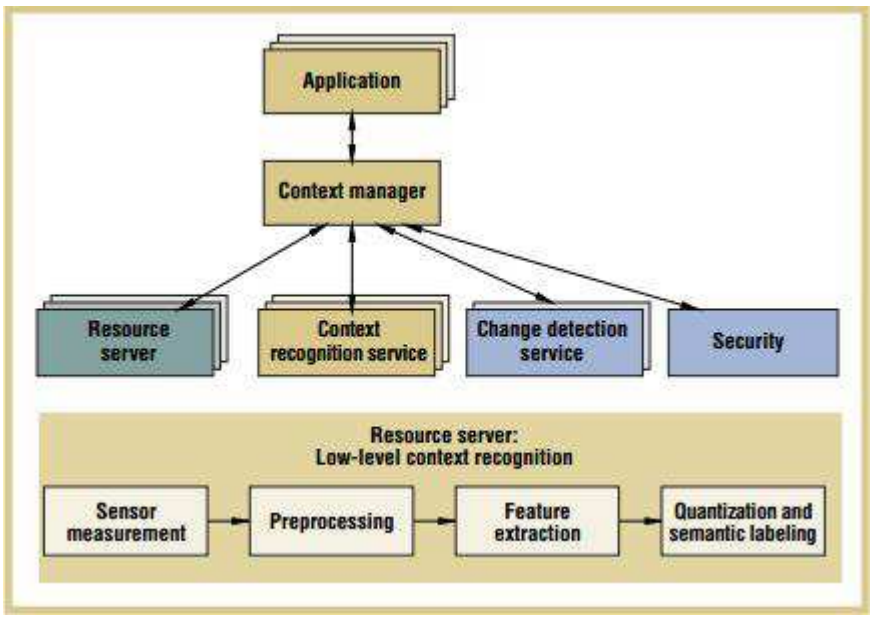
Το τρίτο υπόστρωμα, το *Preprocessing (Προεπεξεργασία)*, είναι υπεύθυνο για την επεξεργασία και κανονικοποίηση των πρωτογενών δεδομένων που παρέλαβε από το προηγούμενο υπόστρωμα. Αναλαμβάνει, δηλαδή, να ερμηνεύει και να μετασχηματίζει τα δεδομένα σε μορφή σύμφωνη με τις απαιτήσεις και τις προδιαγραφές της εφαρμογής.

Το τέταρτο υπόστρωμα, Αποθήκευση/Διαχείριση(Storage/Management) οργανώνει τα δεδομένα και τα προσφέρει στο χρήστη μέσω ενός περιβάλλοντος αλληλεπίδρασης με σύγχρονο τρόπο (ο χρήστης ειδοποιεί για αλλαγή σε δεδομένα και η εφαρμογή κάνει παύση μέχρι να τα στείλει) αλλά και ασύγχρονο τρόπο (με τη μέθοδο της συνδρομής).

Στο πέμπτο υπόστρωμα (Application) πραγματοποιείται μέσω διεπαφής η ουσιαστική επικοινωνία χρήστη-εφαρμογής. Εδώ, δηλαδή, υλοποιείται η πραγματική αντίδραση της εφαρμογής σε διάφορα γεγονότα και αλλαγές του πλαισίου (εμφάνιση μηνυμάτων, alerts κτλ.).

Context Framework

Έχοντας δει το μοντέλο αρχιτεκτονικής μιας εφαρμογής με αναγνώριση πλαισίου και πριν προχωρήσουμε σε ένα παράδειγμα ανάπτυξης μιας τέτοιας εφαρμογής, θα ήταν καλό να παρουσιάσουμε και το διάγραμμα του context framework (Εικόνα 7), δηλαδή της διαδικασίας ανάκτησης και επεξεργασίας του πλαισίου στα συστήματα κινητής υπολογιστικής.



Εικόνα 7:Context Framework [8]

Τέσσερις κύριες λειτουργικές οντότητες συνιστούν το context framework που απεικονίζεται στην Εικόνα 7: context manager (διαχειριστής πλαισίου), resource server, context recognition service (υπηρεσία αναγνώρισης πλαισίου) και application (εφαρμογή). Όπως φαίνεται και στην Εικόνα, η πληροφορία μεταφέρεται όπως υποδηλώνουν τα βέλη στο framework. Όταν διάφορες οντότητες επικοινωνούν, ο διαχειριστής πλαισίου λειτουργεί ως κεντρικός server και οι υπόλοιπες οντότητες χρησιμοποιούν υπηρεσίες τις οποίες διαθέτει ο συγκεκριμένος server. Έτσι, ο διαχειριστής πλαισίου αποθηκεύει πλαίσια, στέλνει απαντήσεις και αλλάζει τις ειδοποιήσεις που απευθύνονται στις υπόλοιπες οντότητες.

Από την άλλη πλευρά, οι resource servers συνδέονται στις διάφορες πηγές δεδομένων πλαισίου και σύμφωνα με αυτές ενημερώνουν τον διαχειριστή πλαισίου για τις αλλαγές στα δεδομένα. Αυτός, με τη σειρά του, επανεπεξεργάζεται τα δεδομένα και τα αποστέλλει στους clients του (δηλαδή στις υπόλοιπες εφαρμογές), σύμφωνα με τις απαιτήσεις τους.

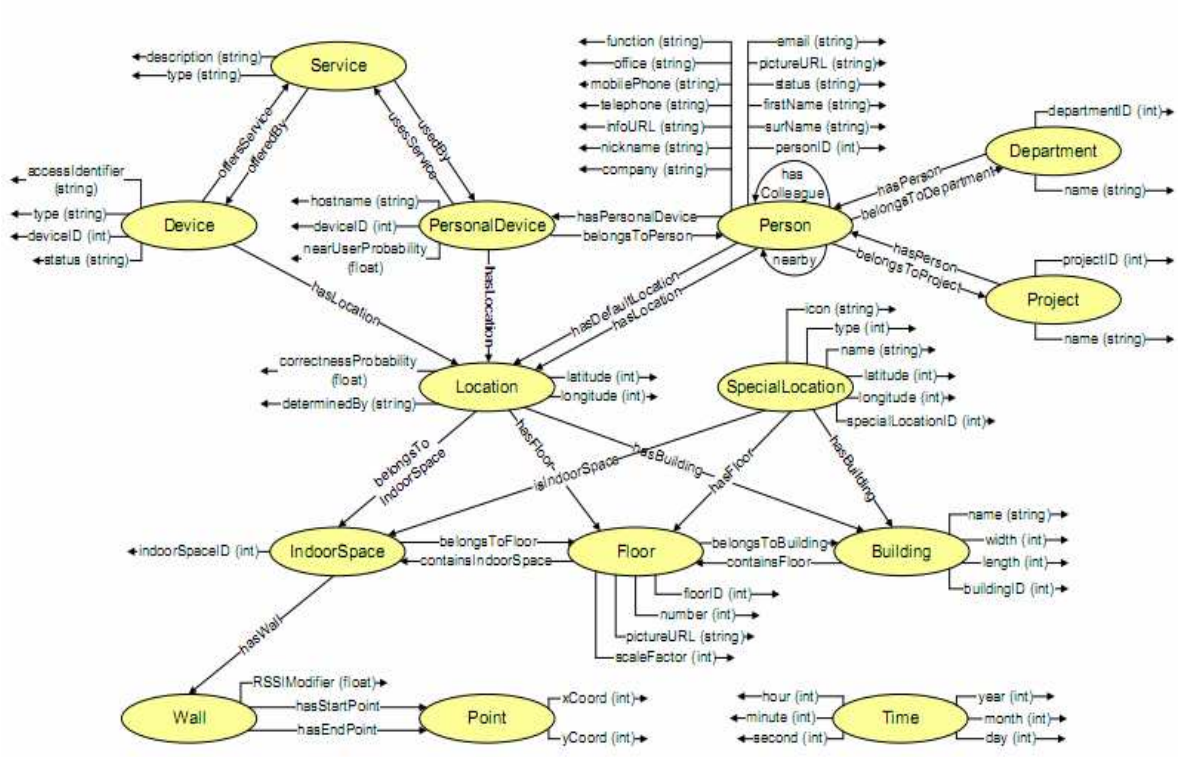
Τέλος, στο κάτω μέρος του γραφήματος, παρουσιάζεται η διαδικασία που εκτελείται μέσα στον resource server για την χαμηλού επιπέδου αναγνώριση του πλαισίου. Η διαδικασία αυτή, συνίσταται στην μετατροπή των raw data (ακατέργαστα δεδομένα) που λαμβάνουν οι αισθητήρες σε μορφή δεδομένων προσπελάσιμη από την εφαρμογή.

4.3 Use Case (Παράδειγμα): Desk Sharing Application

Στο σημείο αυτό παρουσιάζεται ένα παράδειγμα εφαρμογής κινητής υπολογιστικής με αναγνώριση πλαισίου, όπου εστιάζοντας στο πρόβλημα και στην οντολογία που δημιουργήθηκε θα μπορέσουμε να αναδείξουμε πρακτικά τη δομή και τη διαδικασία ανάπτυξης μιας τέτοιας εφαρμογής.

Το συγκεκριμένο use case ονομάζεται Desk Sharing Office Environment [9]. Αναπτύχθηκε για τις ανάγκες μιας εταιρείας στο Βέλγιο στην οποία μετά τη μετεγκατάσταση των γραφείων της, γεννήθηκε η ιδέα του desk sharing. Η ιδέα αυτή συνίσταται στο να μην έχουν οι εργαζόμενοι ένα συγκεκριμένο χώρο εργασίας, αλλά άμα τη αφήξει τους στην εταιρία να πηγαίνουν στον πρώτο διαθέσιμο χώρο για να εργαστούν. Έτσι λοιπόν ο κύριος σκοπός της εφαρμογής προς σχεδίαση είναι να εντοπίζει τους εργαζομένους για λογαριασμό άλλων εργαζομένων και πελατών και να παρουσιάζει τρόπους εύρεσής τους. Ο χρήστης συμπληρώνοντας το όνομα του εργαζομένου μπορεί να δει το μονοπάτι προς την τοποθεσία που βρίσκεται. Όταν ο εργαζόμενος δεν είναι παρών τότε επιστρέφεται η θέση κάποιου συνεργάτη του που θα μπορέσει να δώσει πληροφορίες ή το γραφείο υποδοχής. Επίσης υπάρχει η δυνατότητα να κάνεις μια κλήση μέσω του συστήματος VoIP.

Στην Εικόνα 8 παρουσιάζεται η οντολογία της συγκεκριμένης εφαρμογής, η οποία αποτελεί ουσιαστικά τη βάση στην οποία στηρίχθηκαν οι ερευνητές για αναπτύξουν τον κώδικά τους.



Εικόνα 8: Το διάγραμμα που αναπαριστά την οντολογία της desk sharing εφαρμογής [9]

Τα κίτρινα οβάλ σχήματα αναπαριστούν έννοιες (κλάσεις), τα βελάκια συσχετίσεις, ενώ γύρω από κάθε έννοια παρουσιάζονται και τα χαρακτηριστικά (attributes) τους. Όπως φαίνεται, μέσα σε ένα διάγραμμα εισάγονται όλες οι έννοιες, τα χαρακτηριστικά και οι συνδέσεις μεταξύ τους, ώστε να περιγραφεί πλήρως το περιβάλλον και τα δεδομένα της εφαρμογής. Η κυριότερη έννοια είναι η person (άτομο), η οποία έχει και τα περισσότερα χαρακτηριστικά (attributes). Κάθε person έχει και μια ή παραπάνω προσωπικές συσκευές (personal devices), ενώ και οι δύο ορίζονται από μια συγκεκριμένη τοποθεσία (location).

Προχωρώντας σε αυτή καθεαυτή τη λειτουργία της εφαρμογής, το σύστημα μέσω του εντοπισμού των συσκευών που έχει στη διάθεσή του ο εργαζόμενος, μπορεί να ανακτήσει τη θέση του και να την εμφανίσει στον χρήστη. Ο εντοπισμός γίνεται με δύο τρόπους: μπορεί να επιτευχθεί είτε με το Simple Network Management Protocol (SNMP), που χαρτογραφεί τη θέση των θυρίδων του δικτύου και των MAC διευθύνσεων, ή με μια τεχνική εύρεσης της τοποθεσίας μέσω του ασύρματου δικτύου της εταιρίας (wi-fi).

Το γεγονός ότι το άτομο μπορεί να εντοπιστεί από διαφορετικές συσκευές σημαίνει ότι το σύστημα θα επιστρέφει όλες τις θέσεις τους. Αυτό σημαίνει ότι χρειάζεται περαιτέρω επεξεργασία των πληροφοριών πλαισίου που λαμβάνει η εφαρμογή, ούτως ώστε να δώσει ακριβή αποτελέσματα. Για την εύρεση του εργαζόμενου λοιπόν, εφαρμόζεται ένα σύνολο από *κανόνες που λαμβάνουν υπόψη τους τα μέρη της οντολογίας*. Για παράδειγμα, η *nearUserProbability* επισημαίνει την πιθανότητα μια συσκευή να γειτνιάζει με τον εργαζόμενο. Επιπλέον, η ιδιότητα *corectnessPropability* περιγράφει την ακρίβεια της

εύρεσης της τοποθεσίας. Επομένως η επιλογή γίνεται με βάση τη μέγιστη *corectnessPropability* για τη συσκευή με τη μέγιστη τιμή *nearUserProbability*.

Το παραπάνω παράδειγμα αποτελεί ουσιαστικά μια «εισαγωγή» στον κόσμο των εφαρμογών κινητής υπολογιστικής με επίγνωση θέσης. Στο δεύτερο μέρος της διπλωματικής, που θα αναφερθούμε ειδικότερα στο θέμα προς ανάπτυξη, θα παρουσιαστεί μια πιο αναλυτική περιγραφή της διαδικασίας ανάπτυξης μιας τέτοιου είδους εφαρμογής.

Ειδικό Μέρος

5. Σχεδίαση Πρότυπης Εφαρμογής για Άτομα με Άνοια

5.1 Εισαγωγή – Η Τρίτη ηλικία στο Μικροσκόπιο της Σύγχρονης Υπολογιστικής

Όπως αναφέρθηκε και στην Εισαγωγή, η ανάπτυξη της σύγχρονης ιατρικής έχει ως αποτέλεσμα το αυξανόμενο προσδόκιμο ζωής στο μεγαλύτερο μέρος των ανεπτυγμένων χωρών. Η γήρανση του πληθυσμού είναι γεγονός (το ποσοστό της τρίτης ηλικίας αυξάνεται με αλματώδεις ρυθμούς), κάτι που έχει δημιουργήσει υψηλές ανάγκες για παροχή φροντίδας και γενικότερη βελτίωση της ποιότητας ζωής αυτών των ανθρώπων.

Η φροντίδα των ηλικιωμένων μέσω μεταφοράς και διαβίωσης σε κέντρα παροχής φροντίδας τείνει να αποτελεί έσχατο σενάριο, λόγω των παρατηρημένων περιπτώσεων κατάθλιψης που προκαλεί. Επομένως, ζητούμενο είναι η φροντίδα και η επίβλεψη ασθενών με όσο το δυνατό λιγότερη παρέμβαση στην καθημερινότητά τους.

Η βοήθεια των φαρμάκων είναι αδιαμφισβήτητη, καθώς εκατομμύρια άνθρωποι ζούνε και θα συνεχίσουν να ζούνε σήμερα χάρη στη χορήγησή τους· πλέον όμως δε φτάνει αυτό. Οι άνθρωποι όσο γερνάνε, τόσο πιο δυνατά εργαλεία απαιτούν για να διατηρήσουν τη καθημερινότητά τους. Θέλουν να μπορούν να κάνουν καθημερινές δραστηριότητες, όπως για παράδειγμα έναν περίπατο, με τον τρόπο που το κάνανε όταν ήταν νεότεροι, χωρίς να θέτουν σε κίνδυνο την υγεία τους. Εδώ είναι που στοχεύει η σύγχρονη υπολογιστική επιστήμη· να παρέχει αυτά τα «εργαλεία» στην Τρίτη ηλικία, ώστε να έχουν μια απρόσκοπτη καθημερινότητα.

*5.2 Η Εφαρμογή **User Monitoring***

Η εφαρμογή που θα αναπτυχθεί στην παρούσα διπλωματική, ονομάζεται **User Monitoring** (επίβλεψη χρήστη). Είναι μια εφαρμογή κινητής υπολογιστικής με επίγνωση πλαισίου που αναπτύχθηκε πάνω στην πλατφόρμα Android και μπορεί να εγκατασταθεί σε κινητά τύπου smartphone. Απευθύνεται κυρίως σε ανθρώπους τρίτης ηλικίας οι οποίοι υποφέρουν από άνοια[1], αλλά μπορεί να χρησιμοποιηθεί και γενικότερα από ηλικιωμένους. Είναι σχεδιασμένη να λειτουργεί στο παρασκήνιο, χωρίς τις περισσότερες φορές να απαιτεί την

[1] *Άνοια*: Η άνοια είναι ένα σύνδρομο που οφείλεται σε οργανική βλάβη του εγκεφάλου. Συνήθως συνοδεύεται από έκπτωση πολλών ανώτερων διανοητικών λειτουργιών, όπως η μνήμη, η κρίση, ο λόγος, η σκέψη, ο προσανατολισμός, η κατανόηση, η εκτέλεση αριθμητικών πράξεων, η ικανότητα για μάθηση[10].

παρέμβαση του ηλικιωμένου-χρήστη, και προσφέρει τριών ειδών υπηρεσίες, οι οποίες περιγράφονται στη συνέχεια.

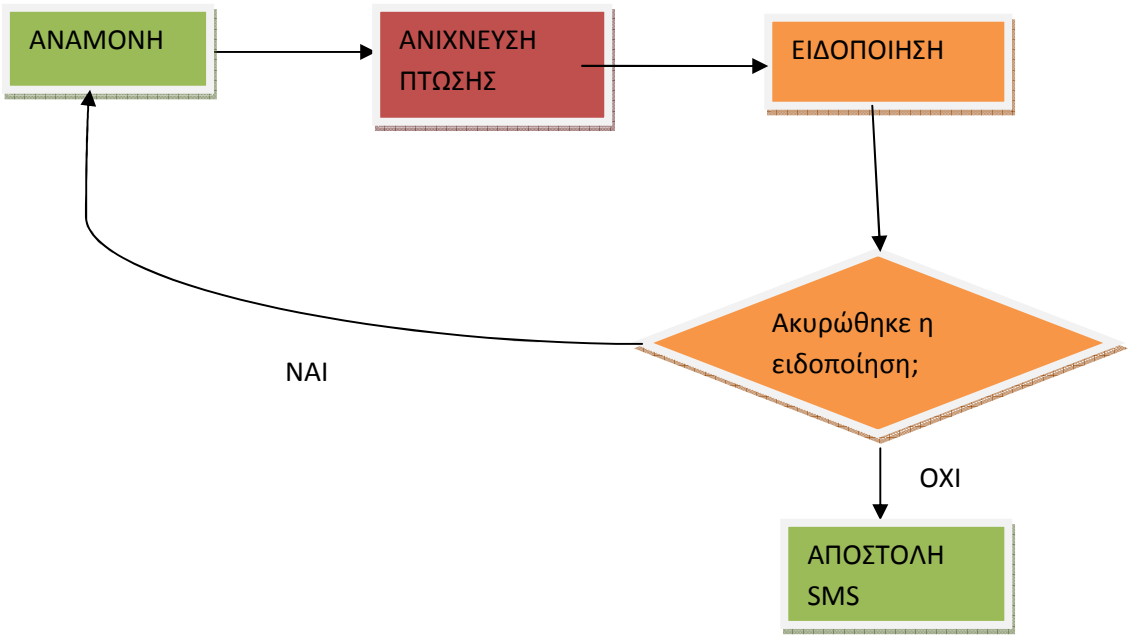
1) Out of range detection (Ανίχνευση Παραβίασης Επιτρεπόμενης Εμβέλειας)

Σε αυτή την υπηρεσία, η εφαρμογή ανιχνεύει με τη βοήθεια υπηρεσιών θέσης, το ενδεχόμενο ο χρήστης να υπερβεί μια συγκεκριμένη εμβέλεια από ένα καθορισμένο αρχικό σημείο του χάρτη. Η εφαρμογή υπολογίζει ανά τακτά χρονικά διαστήματα την απόσταση του αρχικού σημείου από το τρέχον σημείο που βρίσκεται ο χρήστης και μόλις ανιχνεύσει μια τέτοιου είδους παραβίαση, εμφανίζει μια ειδοποίηση στην οθόνη του χρήστη, μαζί με ηχητική ειδοποίηση και δόνηση, επισημαίνοντάς την κατάλληλα, ενώ παράλληλα αποστέλλει ένα μήνυμα κειμένου (sms) σε ένα ή δύο φιλικά προσκείμενα πρόσωπα στον χρήστη (μπορεί δηλαδή να είναι συγγενής, φίλος ή γιατρός του), ειδοποιώντας τους για αυτή την παραβίαση, παρέχοντας και τη θέση του στον χάρτη. Σημειώνεται πως οι υπηρεσίες θέσης ανακτώνται με τη βοήθεια του ενσωματωμένου GPS της κινητής συσκευής του χρήστη και μεταφράζεται σε συντεταγμένες latitude και longitude. Τονίζουμε ότι η επιτρεπόμενη εμβέλεια του χρήστη (που μετριέται σε μέτρα), η αρχική τοποθεσία (με συντεταγμένες latitude και longitude) και τα δύο φιλικά προσκείμενα πρόσωπα και τα τηλέφωνά τους, μπορούν να αλλάξουν μέσα από τις οθόνες ρυθμίσεων της εφαρμογής, μετά το κατάλληλο authentication (εξακρίβωση στοιχείων), στο οποίο θα αναφερθούμε σε λίγο.

2) Fall Detection (Ανίχνευση Πτώσης)

Η δεύτερη υπηρεσία που υλοποιεί η εφαρμογή ονομάζεται Ανίχνευση Πτώσης. Εδώ, με τη βοήθεια ενός ειδικού αισθητήρα του smartphone που ονομάζεται **επιταχυνσιόμετρο (accelerometer)** ανιχνεύεται η περίπτωση ο χρήστης να υποστεί μια πτώση. Η ανίχνευση αυτή γίνεται μέσω του επιταχυνσιόμετρου, που καταγράφει τις επιταχύνσεις της συσκευής· εάν ανιχνεύσει σε ένα πολύ μικρό διάστημα (χιλιοστά του δευτερολέπτου) μια απότομη αλλαγή στην ταχύτητα της συσκευής, η εφαρμογή «υποθέτει» πως ο χρήστης έπεσε.

Σε αυτή την περίπτωση, και επειδή πρέπει να λάβουμε υπόψη μας την περίπτωση false alarm (λανθασμένου συναγερμού), δηλαδή την περίπτωση να μην είχαμε πτώση του χρήστη αλλά, για παράδειγμα, πτώση του κινητού, το μήνυμα ειδοποίησης που εμφανίζεται στην οθόνη περιέχει και έναν countdown timer (αντίστροφη μέτρηση) και κουμπί με δυνατότητα ακύρωσης του συναγερμού. Σε περίπτωση που λήξει το χρονόμετρο και ο χρήστης *δεν ακυρώσει τον συναγερμό* (κάτι που πιθανότατα σημαίνει ότι υπέστη πτώση), εκκινούν οι διαδικασίες ενημέρωσης των φιλικά προσκείμενων προσώπων, ακριβώς όπως στο προηγούμενο σενάριο (ειδοποίηση μέσω sms). Να σημειώσουμε πάλι πως οι συγγενείς, τα τηλέφωνά τους, αλλά και ο χρόνος της αντίστροφης μέτρησης, είναι διαχειρίσιμα μέσω της οθόνης ρυθμίσεων της υπηρεσίας. Στην Εικόνα 9 απεικονίζεται η λογική αυτή.



Εικόνα 9: Διαδικασία λειτουργίας της fall detection μέσω διαγράμματος λήψης αποφάσεων

3)Loss Detection(Ανίχνευση Πτώσης)

Η τρίτη υπηρεσία που υλοποιεί η εφαρμογή ανιχνεύει το σενάριο ο χρήστης να χαθεί κατά τη διάρκεια μιας εξόδου του από το σπίτι. Όταν συμβεί κάτι τέτοιο, με τις ίδιες μεθόδους ειδοποιήσεων που αναφέραμε στις δύο προηγούμενες υπηρεσίες, η συσκευή δονείται με παράλληλη ηχητική ειδοποίηση και ειδοποιούνται οι προκαθορισμένοι συγγενείς ή φίλοι.

Τι ακριβώς σημαίνει όμως αυτό το σενάριο; Ουσιαστικά, προσπαθούμε να ανιχνεύσουμε κάθε περίπτωση αποπροσανατολισμού (disorientation) του ασθενούς-χρήστη. Πάλι με τη βοήθεια του επιταχυνσιόμετρου, η εφαρμογή προσπαθεί να ανιχνεύσει συνεχείς αλλαγές στον προσανατολισμό της συσκευής σε ένα συγκεκριμένο χρονικό πλαίσιο. Εάν αυτό συμβεί, υποθέτουμε πως ο χρήστης έχει αποπροσανατολιστεί και, πανικοβλημένος, προσπαθεί να αντιληφθεί το που βρίσκεται με συνεχείς αλλαγές κατεύθυνσης του εαυτού του και συνεπώς της εφαρμογής. Σε τέτοιο ενδεχόμενο λοιπόν, η εφαρμογή υποθέτει πως ο ασθενής-χρήστης χάθηκε και ακολουθεί τις προαναφερθέντες διαδικασίες.

Όπως γίνεται εύκολα κατανοητό, αυτή είναι κατά κάποιο τρόπο η πιο «αμφιλεγόμενη» υπηρεσία από τις τρεις. Η ανίχνευση του γεγονότος που την ενεργοποιεί είναι δύσκολη (αν μη τι άλλο, ο αποπροσανατολισμός και το ενδεχόμενο ο χρήστης να χαθεί αποτελούν αρκετά γενικές έννοιες για να «μεταφραστούν» σε λογικό κώδικα) και το σενάριο λανθασμένου συναγερμού είναι αρκετά πιθανό (ο χρήστης για παράδειγμα μπορεί να πραγματοποιεί κάποιου είδους σωματική δραστηριότητα που μπορεί να προκαλεί αυτές τις αλλαγές κατεύθυνσης), επομένως η πλήρης λειτουργικότητά της είναι ακόμα προς διερεύνηση.

Παρακάτω θα παρουσιάσουμε τρία σενάρια χρήσης (use cases), ένα για κάθε υπηρεσία, για να δώσουμε ένα μικρό δείγμα χρήσης της εφαρμογής.

5.3 Use Cases (Σενάρια Χρήσης)

Σενάριο 1

Ο ασθενής που υποφέρει από άνοια, βγαίνει για περίπατο στην αυλή του σπιτιού του. Μη έχοντας επίγνωση του χώρου που πρέπει να κυκλοφορήσει για τη δική του ασφάλεια, ξεχνιέται και ξεκινά να απομακρυνθεί από την αυλή στους γύρω δρόμους. Ο επιβλέπων του ασθενούς, γνωρίζοντας τους κινδύνους, του έχει παραχωρήσει ένα κινητό με την εφαρμογή εγκατεστημένη, και έχει ρυθμίσει την επιτρεπόμενη εμβέλεια που μπορεί να κυκλοφορήσει ο ασθενής του(με αρχική τοποθεσία βέβαια το σπίτι του). Ο ασθενής, λοιπόν, με το που υπερβεί το συγκεκριμένο όριο απόστασης από το σπίτι του, λαμβάνει ηχητικό μήνυμα και δόνηση από το κινητό. Αμέσως αντιλαμβάνεται ότι κάτι δεν πάει καλά και σταματάει. Παράλληλα, έχει σταλεί το sms συναγερμού στον γιατρό ή συγγενή του με παράλληλη ενημέρωση της τοποθεσίας του, ο οποίος προβαίνει στις απαραίτητες ενέργειες για να επαναφέρει τον ασθενή του μέσα στα όρια ασφαλείας. Έτσι, με τη βοήθεια της εφαρμογής, υπάρχει συνεχής εποπτεία του χρήστη, ακόμα και όταν βρίσκεται μόνος του χωρίς να τον προσέχει κάποιο φυσικό πρόσωπο, και προλαμβάνονται δυσάρεστες καταστάσεις, όπως η εξαφάνιση του.

Σενάριο 2

Ο ασθενής-χρήστης, έχοντας στην κατοχή του το κινητό ρυθμισμένο από τον γιατρό του, βρίσκεται πάλι σε περίπατο στο πάρκο κοντά στο σπίτι του. Λόγω μειωμένης όρασης και απροσεξίας, σε ένα σημείο σκοντάφτει σε μια πέτρα και πέφτει στο έδαφος. Η πτώση είναι σφοδρή και ο χρήστης τραυματίζεται και είναι ανήμπορος να ζητήσει βοήθεια, καθώς δεν υπάρχουν εκείνη τη στιγμή περαστικοί να τον βοηθήσουν. Η εφαρμογή, όμως, ανιχνεύοντας την πτώση, στέλνει ειδοποίηση στον γιατρό του για την πτώση(μετά το πέρας ενός μικρού διαστήματος μεταξύ του μηνύματος ειδοποίησης και της αυτόματης αποστολής του sms) και ο γιατρός μπορεί άμεσα να καλέσει αστυνομία ή/και ασθενοφόρο. Σ’ αυτή την περίπτωση, λοιπόν, μπορούμε να πούμε πως η εφαρμογή αποβαίνει σωτήρια για τον χρήστη, καθώς προλαμβάνει πιθανότατα πολύ σοβαρές καταστάσεις.

Σενάριο 3

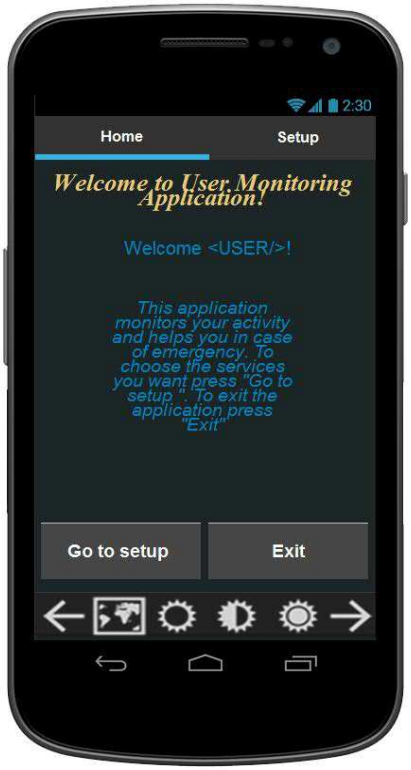
Ο ασθενής-χρήστης της εφαρμογής βρίσκεται σε εκδρομή με τον οίκο ευγηρίας του σε ένα μέρος άγνωστο σε αυτόν. Κάποια στιγμή, ο ασθενής ξεφεύγει από την επίβλεψη των υπευθύνων και απομακρύνεται ασυναίσθητα από το σύνολο των υπόλοιπων ηλικιωμένων. Κάποια στιγμή και ενώ έχει προχωρήσει αρκετά, συνειδητοποιεί ότι δε γνωρίζει που

βρίσκεται και δε μπορεί να βρει τον δρόμο της επιστροφής. Πανικόβλητος, αρχίζει και στριφογυρίζει προσπαθώντας να καταλάβει που είναι το υπόλοιπο γκρουπ. Η εφαρμογή αντιλαμβάνεται αυτές τις απότομες αλλαγές κατεύθυνσης και ενεργοποιείται· στέλνει sms στον γιατρό ή/και στον συγγενή του και αυτοί με τη σειρά τους ειδοποιούν τους υπευθύνους του γκρουπ ώστε να κινήσουν τις διαδικασίες εύρεσής του. Έτσι, με τη βοήθεια της εφαρμογής, προλαμβάνονται τυχόν δυσάρεστες καταστάσεις.

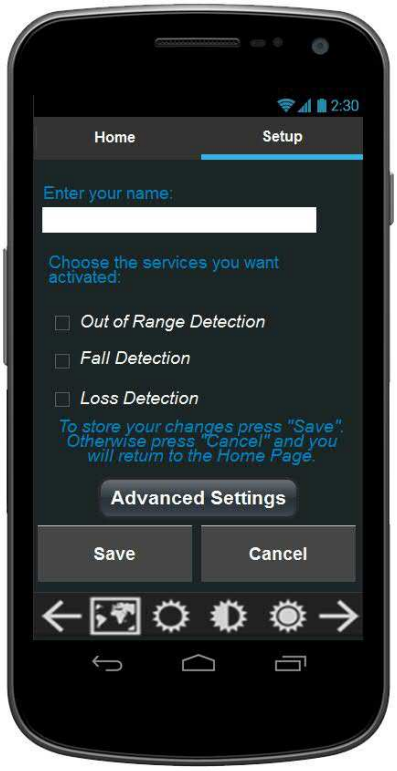
5.4 Σχεδιασμός Οθονών Εφαρμογής

Παρακάτω παρουσιάζουμε με επεξηγηματικές λεζάντες κάποιες από τις οθόνες της εφαρμογής, οι οποίες δημιουργήθηκαν κατά τα αρχικά στάδια της σχεδίασης. Οι οθόνες αυτές προέρχονται από ένα εργαλείο δημιουργίας γραφικών περιβαλλόντων (user interfaces),το Protoyper, το οποίο και θα παρουσιάσουμε αργότερα (Κεφάλαιο 7).

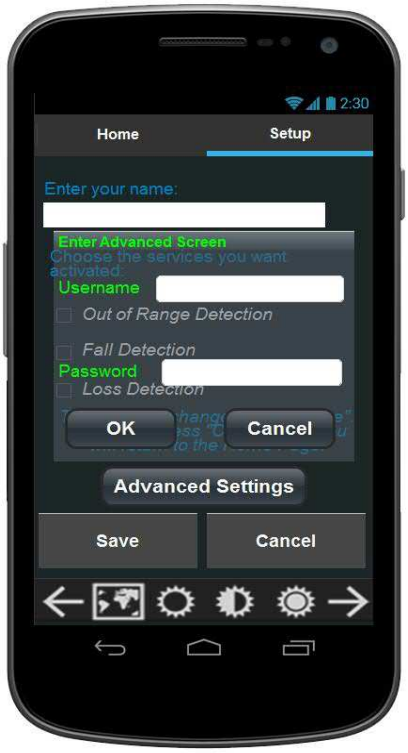
Σημείωση: Οι παρακάτω οθόνες δεν αντιπροσωπεύουν το ακριβές user interface της εφαρμογής (όπως θα δείτε και στη συνέχεια). Δημιουργήθηκαν μέσω εργαλείου σχεδίασης συγκεκριμένων δυνατοτήτων και σκοπό έχουν να δώσουν μια πρώτη ιδέα για το look & feel (εικόνα και αίσθηση) της εφαρμογής, αλλά και να αποτελέσουν ένα σημείο αναφοράς για την ανάπτυξή της.



Εικόνα 10: Η αρχική οθόνη(Home Screen) της εφαρμογής. Περιέχει τον τίτλο της εφαρμογής και ένα μήνυμα καλοσωρίσματος. Διακρίνονται τα κουμπιά "Go to Setup"(πηγαίνει στην οθόνη ρυθμίσεων) και "Exit"(έξοδος από την εφαρμογή).

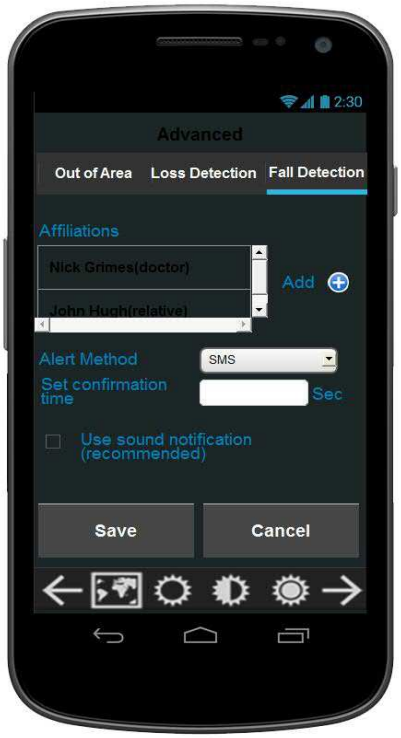


Εικόνα 11: Η οθόνη ρυθμίσεων (Setup Screen) της εφαρμογής. Διακρίνονται η επιλογή για προσθήκη του ονόματος χρήστη, τρία checkboxes όπου μπορούν να επιλεγούν ποια/ες από τις τρεις υπηρεσίες θα είναι ενεργοποιημένες, τα κουμπιά Save και Cancel (για αποθήκευση και ακύρωση επιλογών αντίστοιχα) και κουμπί για πρόσβαση στην οθόνη των προχωρημένων ρυθμίσεων (Advanced Settings).



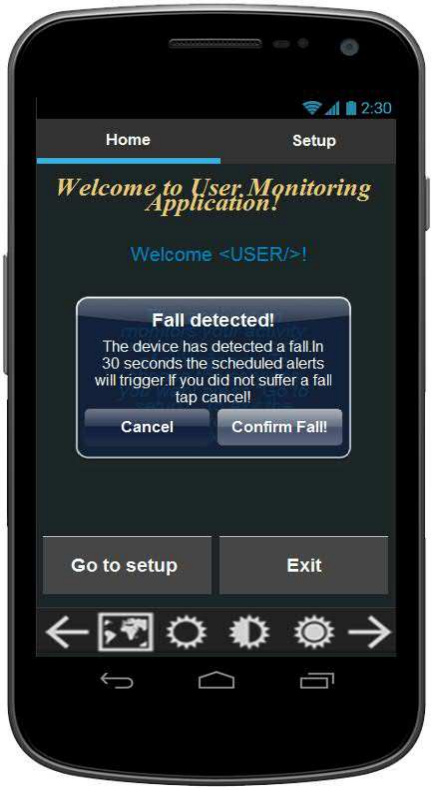
Εικόνα 12:Η πρόσβαση στην οθόνη προχωρημένων ρυθμίσεων γίνεται μέσω authentication, δηλαδή απαιτείται η προσθήκη username και password για την είσοδο σε αυτή.

Εικόνα 13: Η οθόνη προχωρημένων ρυθμίσεων της υπηρεσίας Out of Range(Out of Range Advanced Settings). Εμπεριέχει τη λίστα με τα ονόματα των ατόμων που θα ειδοποιηθούν σε περίπτωση συναγερμού και το κουμπί Add, που προσθέτει νέες καταχωρήσεις στη λίστα. Επίσης υπάρχει επιλογή για τη μέθοδο ειδοποίησης (στοιχείο που καταργήθηκε στον τελικό σχεδιασμό), επιλογή της επιτρεπόμενης εμβέλειας και το κουμπί Choose Area(Επιλογή περιοχής) που επιλέγει την αρχική θέση(το σημείο αναφοράς για την εμβέλεια).



Εικόνα 14: Η οθόνη προχωρημένων ρυθμίσεων της υπηρεσίας Fall Detection (Fall Advanced Settings). Βλέπουμε ότι στην πλειοψηφία της, εκτελεί παρόμοιες λειτουργίες με την οθόνη ρυθμίσεων της out of range(λίστα συγγενών /γιατρών κλπ), με διαφορά το πεδίο set confirmation time, όπου ορίζεις το χρόνο μεταξύ της εμφάνισης της ειδοποίησης σε περίπτωση πτώσης στην οθόνη και της αποστολής του sms.

Εικόνα 15:Το πλαίσιο διαλόγου "Add Affiliate"(Προσθήκη φιλικά προσκείμενου προσώπου). Διακρίνονται τα πεδία Όνομα, Τηλέφωνο και Σχέση. Πατώντας το κουμπί Add, η επαφή θα αποθηκευτεί στη λίστα με τις επαφές που θα ειδοποιηθούν σε περίπτωση ανάγκης.



Εικόνα 16: Εδώ έχουμε παράδειγμα εμφάνισης του πλαισίου διαλόγου σε περίπτωση γεγονότος-εν προκειμένω, ανιχνεύθηκε πτώση. Υπάρχουν δύο κουμπιά: το Confirm Fall(θα σταλεί άμεσα sms) και το Cancel(ακυρώνει ο χρήστης σε περίπτωση false alarm). Εάν δεν υπάρξει καμία ενέργεια μετά το πέρας του χρόνου που εμφανίζεται(ο οποίος έχει καθοριστεί από τις ρυθμίσεις) θα αποσταλεί αυτόματα sms.

5.5 Σκοπός Σχεδιασμού και Λειτουργία στο Παρασκήνιο

Πριν κλείσουμε το κεφάλαιο του σχεδιασμού της εφαρμογής και προχωρήσουμε στην διαδικασία ανάπτυξης της εφαρμογής, οφείλουμε να τονίσουμε δύο βασικά σημεία που πρέπει να προσεχθούν και αφορούν την ευχρηστία της εφαρμογής.

Κατά την πορεία ανάπτυξης της εφαρμογής, και αφού είχα ήδη δημιουργήσει το προσχεδιασμένο interface που παρουσιάστηκε στην προηγούμενη ενότητα μέσω των screenshots, αποφάσισα να τα παρουσιάσω στο target group της εφαρμογής, δηλαδή σε ηλικιωμένους με παθήσεις τύπου άνοιας. Το feedback ήταν σχετικά αρνητικό: φαίνεται πως η νέα τεχνολογία και ο χειρισμός μιας τυπικής εφαρμογής με μερικά κουμπιά, λίστες και καρτέλες(tabs) αποτελούν κάτι το εξεζητημένο γι αυτούς τους ανθρώπους· γεγονός που η αλήθεια είναι πως είναι και αρκετά αναμενόμενο.

Έτσι λοιπόν, και όπως θα φανεί στη συνέχεια στην παρουσίαση της εφαρμογής στο Κεφάλαιο 8, αποφασίστηκε αλλαγή γραμμής. Απλούστατη διεπαφή χρήστη (interface) κα όσο το δυνατόν λιγότερες απαιτήσεις από αυτόν. Τι σημαίνει αυτό;

Απλούστατα, ο χρήστης ουσιαστικά θα λαμβάνει τις υπηρεσίες της εφαρμογής χωρίς καν να ξέρει ότι υπάρχουν. Οι υπηρεσίες θα εκτελούνται στο παρασκήνιο και ο χρήστης θα πρέπει να αλληλεπιδρά **μόνο** σε περίπτωση συμβάντος· και εκεί, το μόνο που θα μπορεί να απαιτηθεί θα είναι το πάτημα ενός κουμπιού (όπως στην περίπτωση που θα χρειαστεί να ακυρώσει έναν λανθασμένο συναγερμό). Έτσι, θα μπορεί να ασχολείται με διάφορες δραστηριότητες στο κινητό του και η εφαρμογή θα κάνει την επίβλεψή της χωρίς αυτός να το αντιλαμβάνεται.

Κάποιος, όμως, θα πρέπει να ρυθμίσει την εφαρμογή σωστά (δηλαδή να «παίξει» με τις οθόνες ρυθμίσεων) ώστε να τη φέρει στα μέτρα του εκάστοτε ασθενή-χρήστη. Αυτή λοιπόν είναι δουλεία του συγγενή ή του γιατρού. Αυτός μέσω και του απαραίτητου authentication (για να μην υπάρχει ενδεχόμενο ο χρήστης να «πειράξει» ρυθμίσεις χωρίς να το καταλάβει) θα έχει πρόσβαση στις ρυθμίσεις των υπηρεσιών, ώστε να τις προσαρμόσει κατάλληλα στις ανάγκες του χρήστη. Συνεπώς, τις περισσότερες φορές, θα αρκεί ένα αρχικό σετάρισμα στις ρυθμίσεις και από εκεί και πέρα η εφαρμογή θα μπορεί να αναλάβει τα υπόλοιπα.

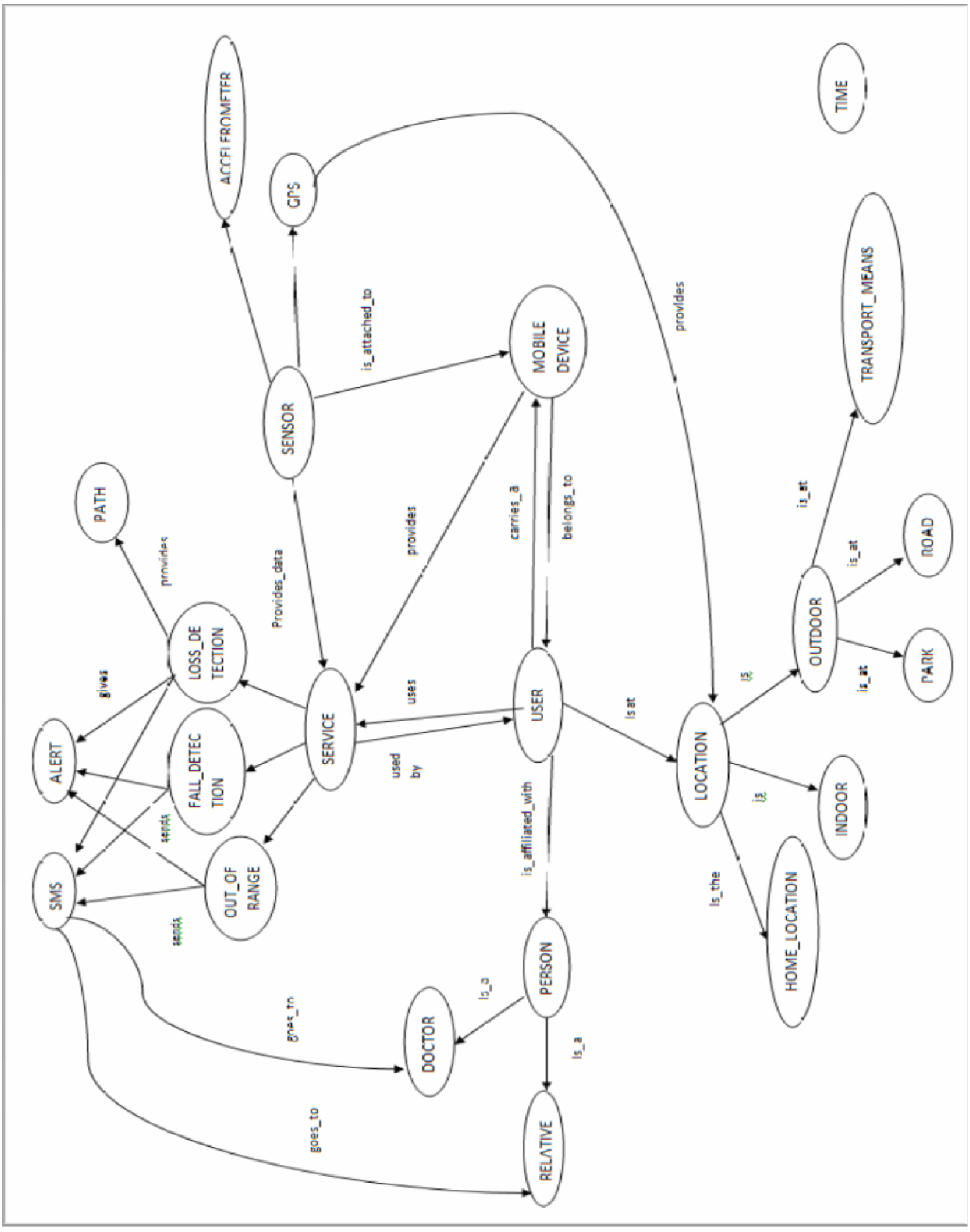
6. Αναπαράσταση Πλαισίου με Οντολογία

6.1 Σχεδίαση Οντολογίας

Η οντολογία που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας περιγράφει (σε προσιτή μορφή) το πλαίσιο μέσα στο οποίο λειτουργεί η εφαρμογή, περιγράφοντας σχετικές έννοιες και συσχετίσεις μεταξύ αυτών. Έτσι λοιπόν, και μιας και η εφαρμογή είναι σχεδόν αποκλειστικά εξαρτώμενη από το πλαίσιο και την επίγνωσή του, αποτελεί σημαντικό βοήθημα η δημιουργία μιας καλά ορισμένης οντολογίας. Στην **Εικόνα 17** παρουσιάζεται το σχήμα της οντολογίας με τις βασικές έννοιες που περιγράφουν το συγκεκριμένο πλαίσιο.

Τα σημαντικότερα στοιχεία που παρουσιάζονται στο σχήμα είναι τα εξής:

- Τα οβάλ σχήματα αναπαριστούν τις έννοιες της οντολογίας.
- Οι έννοιες συνίστανται σε κλάσεις και υποκλάσεις. Για παράδειγμα οι έννοιες USER, MOBILE DEVICE και LOCATION αποτελούν κλάσεις ενώ οι έννοιες OUT_OF_RANGE, FALL_DETECTION και LOSS_DETECTION αποτελούν υποκλάσεις της κλάσης SERVICE.
- Τα βέλη που συνδέουν τις κλάσεις/υποκλάσεις μεταξύ τους αποτελούν τις σχέσεις που τις συνδέουν μεταξύ τους. Η περιγραφή αυτής της σχέσης δηλώνεται από τις λέξεις που υπάρχουν κατά μήκος των ακμών. Για παράδειγμα, οι κλάσεις SENSOR και MOBILE_DEVICE συνδέονται μέσω της σχέσης “is_attached_to” (κάτι που σημαίνει πρακτικά ότι ο αισθητήρας συσχετίζεται με το κινητό επειδή είναι ενσωματωμένος σε αυτό).
- Μπορούν να υπάρχουν και αμφίδρομες σχέσεις, όπως μεταξύ USER και MOBILE_DEVICE που συνδέονται με τις σχέσεις “carries_a” και “belongs_to”.
- Στο σχήμα δεν παρουσιάζονται οι ιδιότητες των κλάσεων. Ο σκοπός της είναι να δώσει μια γενική πρώτη εικόνα του πλαισίου ώστε να μπορέσουμε να κατανοήσουμε τις έννοιες που το αποτελούν καθώς και τις σχέσεις μεταξύ τους. Αμέσως μετά μέσω του εργαλείου Protégé θα παρουσιάσουμε αναλυτικότερα την οντολογία προσθέτοντας ιδιότητες στις κλάσεις και άλλα στοιχεία.



Εικόνα 17: Σχεδιάγραμμα οντολογίας του User Monitoring

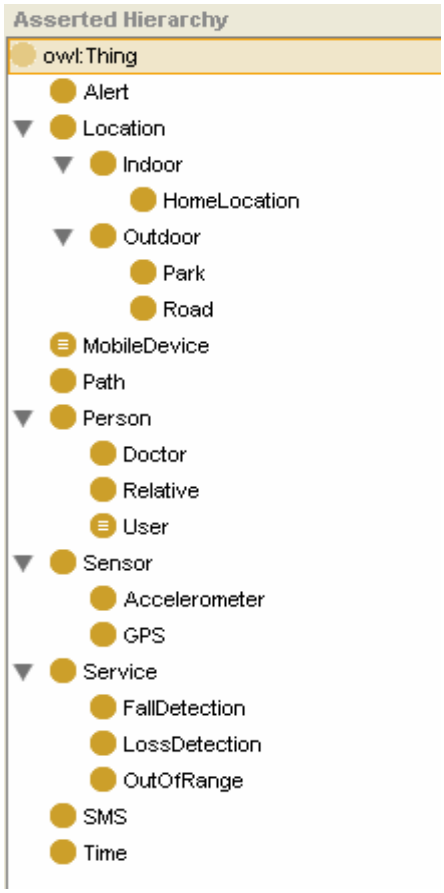
6.2 Παρουσίαση της οντολογίας μέσω του Protégé

Η οντολογία όπως προαναφέραμε, έχει και άλλα στοιχεία πέρα από αυτά που παρουσιάσαμε στο παραπάνω γράφημα, όπως για παράδειγμα ιδιότητες των κλάσεων αλλά και περιορισμοί στις σχέσεις μεταξύ τους αλλά και στις τιμές που μπορούν να πάρουν. Δημιουργήσαμε λοιπόν μια λεπτομερέστερη και πιο περιγραφική μορφή της οντολογίας μέσω του εργαλείου Protégé [11].

Το **Protégé** είναι ένα εργαλείο ανοιχτού λογισμικού που επιτρέπει την ανάπτυξη και τη διαχείριση οντολογιών με όλα τα απαραίτητα στοιχεία τους (κλάσεις, συνδέσεις, ιδιότητες κλάσεων, περιορισμούς, στιγμιότυπα των κλάσεων, σημασιολογικούς κανόνες). Υποστηρίζει την πρότυπη γλώσσα αναπαράστασης οντολογιών **OWL** (*βλπ. Κεφάλαιο 7*). Στη συνέχεια, παρουσιάζεται η οντολογία που αναπτύχθηκε παραθέτοντας οθόνες του αντίστοιχου περιεχομένου στο Protégé και σχετικές επεξηγήσεις.

Κλάσεις

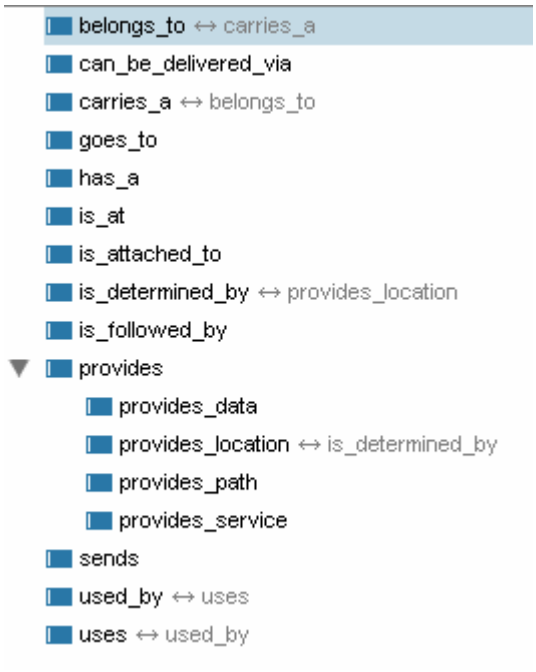
Στην Εικόνα 18 παρουσιάζεται η ιεραρχία κλάσεων της οντολογίας (κλάσεις-υποκλάσεις). Εδώ βλέπουμε απλά τις γνωστές κλάσεις και υποκλάσεις από το προηγούμενο γράφημα (οι υποκλάσεις βρίσκονται πιο «μέσα» στη στοίχιση σε σχέση με τις κλάσεις, και οι κλάσεις που έχουν «παιδιά» έχουν ένα βελάκι στα αριστερά του ονόματός τους).



Εικόνα 18:Κλάσεις

Συσχετίσεις-Συνδέσεις

Στην Εικόνα 19 παρουσιάζεται η λίστα σχέσεων (object properties) που ορίστηκε, μέσω των οποίων συνδέονται έννοιες (κλάσεις) μεταξύ τους. Η κάθε μια σύνδεση εάν επιλεγεί εμφανίζει τις κλάσεις τις οποίες συνδέει μεταξύ τους (Εικόνα 20). Για παράδειγμα, η "is_at" σχέση συνδέει την κλάση User με την κλάση Location (δηλαδή, ο χρήστης συνδέεται με την τοποθεσία μέσω της λογικής σχέσης «βρίσκεται στην»). Τα ονόματα των σχέσεων εξηγούν από μόνα τους την εννοιολογική σχέση μεταξύ των κλάσεων.



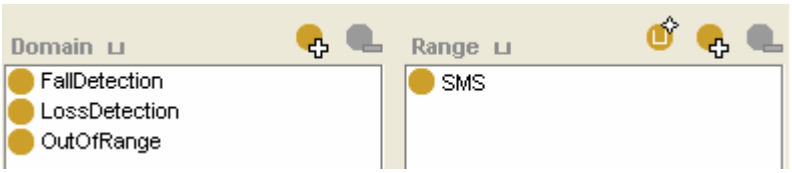
Εικόνα 19:Συσχετίσεις-Συνδέσεις (object properties)

Βλέπουμε πως μερικές σχέσεις έχουν δίπλα τους μια άλλη σχέση, που εμφανίζεται με αχνά γράμματα (όπως η belongs_to με την carries_a). Αυτές είναι οι αμφίδρομες συνδέσεις που αναφέραμε και στην υλοποίηση με το γράφημα, δηλαδή συνδέσεις που είναι συμπληρωματικές μεταξύ τους (inverse properties).

Παρατηρούμε επίσης πως υπάρχει και εδώ μια έννοια ιεραρχίας, με «υπερσυνδέσεις» και «υποσυνδέσεις» (subproperties). Αυτό φαίνεται στη σχέση provides (παρέχει) και τις σχέσεις provides_data (παρέχει δεδομένα), provides_path (παρέχει μονοπάτι), provides_location (παρέχει τοποθεσία) και provides_service (παρέχει υπηρεσία). Επειδή η σχέση provides εμπεριέχει από εννοιολογικής άποψης τις υπόλοιπες τρεις σχέσεις, λειτουργεί ως «υπερσύνδεση» και οι άλλες λειτουργούν ως «υποσυνδέσεις».

Εικόνα 20: Σύνδεση "is_at"

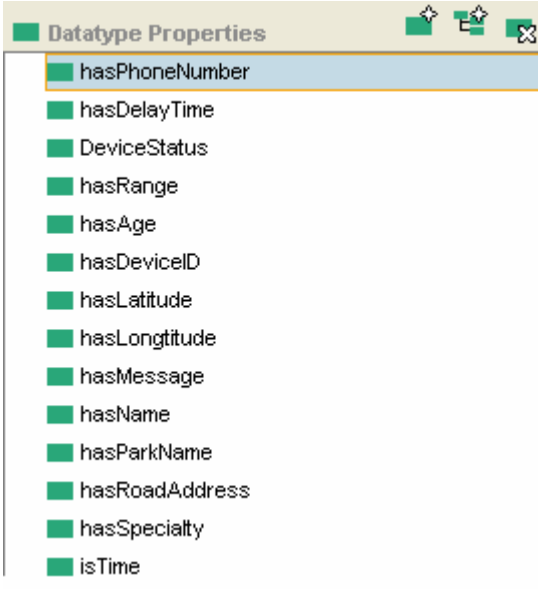
Σημειώνουμε πως, όπως και στο γράφημα της οντολογίας, μπορούν πολλαπλές κλάσεις να συνδέονται μεταξύ τους μέσω μιας σχέσης. Στην Εικόνα 21, για παράδειγμα, οι τρεις υπηρεσίες (OutOfRange, FallDetection, LossDetection) συνδέονται με το SMS μέσω της ιδιότητας sends (αποστέλλει).



Εικόνα 21: Σύνδεση “sends”

Ιδιότητες

Στη διπλανή εικόνα παρουσιάζεται ο κατάλογος με τις ιδιότητες των κλάσεων (datatype properties) της οντολογίας. Αυτές περιγράφουν περαιτέρω την κάθε κλάση ξεχωριστά καθώς εμπεριέχουν περισσότερες πληροφορίες για την κάθε κλάση και πολλές από αυτές είναι απαραίτητες για τη λειτουργικότητα της εφαρμογής. Οι ιδιότητες έχουν αρκετά επεξηγηματικά (για τη λειτουργία τους) ονόματα και, επιλέγοντάς τες, μπορούμε να δούμε σε ποια κλάση είναι εγγεγραμμένη η κάθε μια τους (Εικόνα 22), αλλά και τον τύπο των δεδομένων που εμπεριέχει η κάθε ιδιότητα (εάν είναι δηλαδή αλφαριθμητικός (string), δεκαδικός (float), ακέραιος (int) κτλ.).



Εικόνα 22: Ιδιότητες datatype

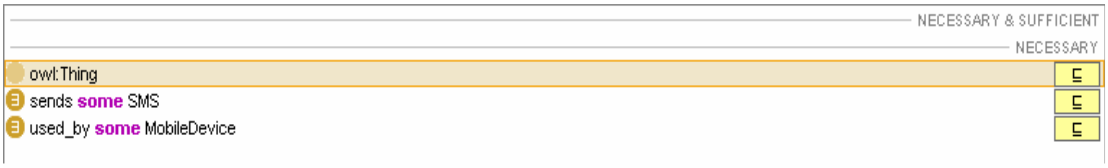
Εικόνα 23: Τύπος και συνοδευτική κλάση της hasRange ιδιότητας.

Ένα παράδειγμα είναι η ιδιότητα hasRange. Αυτή, όπως φαίνεται στην Εικόνα 23, είναι ιδιότητα της κλάσης HomeLocation και περιέχει το επιτρεπόμενο εύρος από την αρχική τοποθεσία στο οποίο μπορεί να κυκλοφορεί ο χρήστης. Στο δεξί μέρος της εικόνας, στο πεδίο Range, βλέπουμε την τιμή float, κάτι που σημαίνει πως η συγκεκριμένη ιδιότητα μπορεί να πάρει πραγματικούς αριθμούς ως τιμή.

Περιορισμοί

Οι περιορισμοί είναι αυτό ακριβώς που υποδηλώνει το όνομά τους: περιορίζουν τις σχέσεις μεταξύ των κλάσεων με βάση κάποιους κανόνες τους οποίους θέτουμε ώστε να μην παραβιάζεται η λογική στην εφαρμογή μας. Θέτουν, λοιπόν, κάποιους κανόνες τους οποίους πρέπει να ακολουθούν **υποχρεωτικά** τα διάφορα στιγμιότυπα (instances) των κλάσεων. Με αυτόν τον τρόπο, λοιπόν, η οντολογία οριοθετείται και ορίζεται ακόμα πιο καλά.

Στην Εικόνα 24 παρουσιάζονται δύο περιορισμοί που αφορούν στην κλάση Service. Στον πρώτο περιορισμό, έχουμε τη σχέση sends και την κλάση SMS και τη λογική σύζευξη some, που σημαίνει «κάποια». Και εδώ οι συμβολισμοί είναι επεξηγηματικοί: *Τα στιγμιότυπα της κλάσης Service πρέπει να μπορούν να συνδέονται με τη σχέση sends) με* ***τουλάχιστον ένα (some)*** *στιγμιότυπο της κλάσης SMS.* Με άλλα λόγια, πρέπει κάθε υπηρεσία της εφαρμογής να μπορεί να στέλνει ένα τουλάχιστον μήνυμα κειμένου. Επίσης, ο επόμενος περιορισμός (used_by some MobileDevice) δηλώνει ότι η κάθε υπηρεσία της εφαρμογής πρέπει να χρησιμοποιείται από μια κινητή συσκευή.



Εικόνα 24: Περιορισμοί της κλάσης Service

Οι παραπάνω περιορισμοί δηλώθηκαν ως NECESSARY. Αυτό σημαίνει πως εάν ένα στιγμιότυπο αποτελεί μέλος της κλάσης Service, τότε είναι απαραίτητο (necessary) να εκπληρώνει αυτούς τους περιορισμούς.

Επιπρόσθετα, στην Εικόνα 25 παρουσιάζονται οι περιορισμοί της κλάσης User ως NECESSARY & SUFFICIENT.

Εικόνα 25: Περιορισμοί της κλάσης User

Αυτό σημαίνει όχι μόνο ότι τα αντικείμενα της κλάσης είναι απαραίτητο να εκπληρώνουν αυτόν τον περιορισμό, αλλά και ότι *εάν ένα στιγμιότυπο εκπληρώνει αυτόν τον περιορισμό, τότε* ***αναγκαστικά*** *πρέπει να υπάγεται σε αυτήν την κλάση*. Επομένως, σε αυτό το παράδειγμα ο περιορισμός "carries_a some MobileDevice" σημαίνει ότι εάν κάποιο στιγμιότυπο πληροί αυτόν τον περιορισμό, τότε αυτό είναι αρκετό για να συμπεράνουμε ότι είναι στιγμιότυπο της κλάσης User.

Τέλος, αξίζει να σημειώσουμε ότι οι περιορισμοί «κληρονομούνται» από τις υπερκλάσεις στις υποκλάσεις (όπως συμβαίνει και στις κλάσεις της αντικειμενοστραφής γλώσσας προγραμματισμού Java). Για παράδειγμα, εάν ένας περιορισμός ισχύει για την κλάση Service, τότε αυτός θα κληρονομηθεί και στις τρεις υποκλάσεις της, δηλαδή στις OutOfRange, FallDetection και LossDetection.

6.3 Υιοθέτηση Οντολογικής Προσέγγισης

Η δημιουργία και η ανάπτυξη μιας εφαρμογής με επίγνωση πλαισίου είναι ένα αρκετά πολύπλοκο και δύσκολο επιχείρημα. Ενέχει αρκετές σχεδιαστικές προκλήσεις για να που αφορούν στη δυναμικότητα του περιβάλλοντος πλαισίου (συνεχείς μεταβολές) αλλά και τις αλλαγές στις απαιτήσεις των χρηστών. Οι απαιτήσεις αυτές συνίστανται κυρίως στη συλλογή, μοντελοποίηση, αποθήκευση και επίβλεψη των πληροφοριών πλαισίου. Επομένως, αυτές οι απαιτήσεις δημιουργούν την ανάγκη για μια σωστή σχεδιαστική προσέγγιση αυτής της ανάγκης.

Αυτό ακριβώς κάνει η χρήση της οντολογίας: χρησιμοποιώντας ένα σύνολο σημασιολογικών κανόνων αλλά και κάποιες αποδεκτές συμβάσεις, αναπαριστά με ακρίβεια και σε μορφή αποδεκτή από υπολογιστικά συστήματα τις ιδιότητες, τη δομή και τις σχέσεις μεταξύ των πληροφοριών πλαισίου.

Ένα ακόμα σημαντικό πλεονέκτημα της χρήσης των οντολογιών είναι ότι λόγω της ανεξαρτητοποίησής τους από το προγραμματιστικό μέρος της διαδικασίας ανάπτυξης εφαρμογής, λειτουργούν ως αυτόνομες δημιουργίες και μπορούν να επαναχρησιμοποιηθούν σε πολλές άλλες εφαρμογές. Εφόσον η οντολογία ουσιαστικά αναπαριστά πληροφορία με κοινά αποδεκτούς όρους και κανόνες, αυτή η αναπαράσταση μπορεί να χρησιμοποιηθεί και από άλλες εφαρμογές πέρα από τη δική μας, οι οποίες θα υλοποιούν διαφορετικού είδους υπηρεσίες.

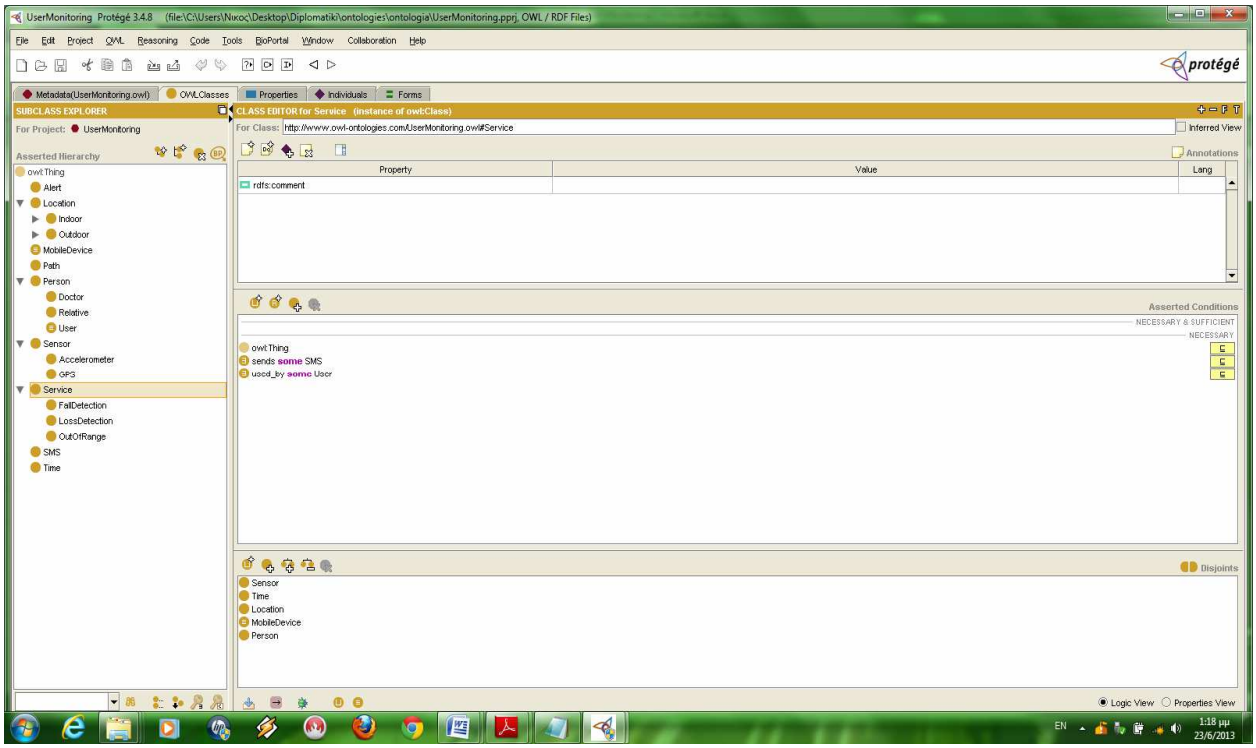
Τέλος, η ευχρηστία και η μεταβλητότητα αποτελούν σημαντικούς λόγους χρήσης της οντολογίας. Η περιγραφή των πληροφοριών πλαισίου σε hard-coding και μέσω προγραμματιστικής διαδικασίας, όχι μόνο θα την καθιστούσε μια δύσκολη και εξειδικευμένη δουλειά, αλλά θα έκανε δύσκολη και την επεξεργασία της πληροφορίας σε περίπτωση αλλαγής του πλαισίου που περιγράφεται. Αντίθετα, στην περίπτωση χρήσης οντολογίας, η ανάπτυξη αλλά και η επεξεργασία της αποτελεί μια πιο εύκολη και προσιτή διαδικασία.

7. Εργαλεία και Τεχνολογίες Υλοποίησης

Στο παρόν κεφάλαιο παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της προτεινόμενης εφαρμογής. Κάποιες από αυτές έχουν ήδη αναφερθεί (Protégé, Prototyper) και θα παρουσιαστούν εδώ πιο αναλυτικά, και κάποιες θα αναφερθούν στη συνέχεια κατά την παρουσίαση της ανάπτυξης της εφαρμογής (Κεφάλαιο 8).

7.1 Protégé

Το Protégé, όπως παρουσιάστηκε και στο Κεφάλαιο 7, αποτελεί ένα ισχυρό εργαλείο για ανάπτυξη και επεξεργασία οντολογιών αναπαράστασης πλαισίου. Δημιουργήθηκε από το πανεπιστήμιο του Stanford και είναι ένα εύχρηστο πρόγραμμα, με γραφικό περιβάλλον φιλικό προς το χρήστη, το οποίο δίνει τη δυνατότητα δημιουργίας κλάσεων/υποκλάσεων, σχέσεων και συνδέσεων μεταξύ αυτών, δημιουργία και επεξεργασία ιδιοτήτων των κλάσεων αλλά και περιορισμών.



Εικόνα 26: Το περιβάλλον εργασίας του Protégé

7.2 OWL (Ontology Web Language)

Η γλώσσα OWL (Ontology Web Language) δημιουργήθηκε από τον διεθνή οργανισμό προτύπων W3C (World Wide Web Consortium), και αποτελεί μια γλώσσα αναπαράστασης γνώσης η οποία χρησιμοποιείται για να ορίζει και να περιγράφει οντολογίες. Προσφέρει, όπως αναφέρθηκε και στο Κεφάλαιο 6, τη δυνατότητα ορισμού σημασιολογικών εκφράσεων και μπορεί να περιγράψει ρητά κλάσεις αντικειμένων, διάφορα χαρακτηριστικά και ιδιότητές τους, περιορισμούς και συνθήκες που αφορούν τις συγκεκριμένες κλάσεις (με τη βοήθεια λογικών συνδέσεων και συναρτήσεων). Βασίζεται στη γλώσσα σήμανσης XML, εμπλουτίζοντάς την με τα στοιχεία που αναφέραμε παραπάνω.

Ας παρουσιάσουμε ένα παράδειγμα της σύνταξης της γλώσσας OWL:

<rdfs:Classrdf:ID=“Bear">

<rdfs:subClassOf rdf:resource="#Animal"/>

</rdfs:Class>

Εδώ δηλώνεται η κλάση Bear ως υποκλάση της κλάσης Animal. Με παρόμοια φιλοσοφία ορίζεται σε OWL μία οντολογία.

Το Protégé υποστηρίζει την κωδικοποίηση οντολογιών σε OWL. Η OWL αναπαράσταση της οντολογίας δημιουργείται αυτόματα μέσω του Protégé, καθώς το ίδιο δημιουργεί τα κομμάτια του κώδικα, ενώ ο χρήστης δημιουργεί την οντολογία στο γραφικό περιβάλλον, όπως παρουσιάστηκε στο προηγούμενο κεφάλαιο.

7.3 Android

Το Android είναι μια πλατφόρμα για συσκευές κινητής τηλεφωνίας η οποία εμπεριέχει λειτουργικό σύστημα το οποίο έχει ως βάση τον πυρήνα του λειτουργικού συστήματος Linux. Αναπτύσσεται και υποστηρίζεται από την Google, και επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού. Είναι ανοιχτού κώδικα (open source) και σήμερα θεωρείται η μεγαλύτερη και ταχύτερα αναπτυσσόμενη πλατφόρμα για κινητά τηλέφωνα.

Η ανάπτυξη εφαρμογών στο Android, όπως προαναφέραμε, γίνεται μέσω της γλώσσας προγραμματισμού Java. Υπάρχουν, όμως, κάποιες διαφορές στις λέξεις-κλειδιά που χρησιμοποιούνται στον κώδικα (για παράδειγμα, οι κλάσεις στο Android ονομάζονται Activities), καθώς και μια πλειάδα βιβλιοθηκών ειδικές για Android που μπορούν να χρησιμοποιηθούν. Παρ’ όλα αυτά, η αντικειμενοστραφής φιλοσοφία στον προγραμματισμό παραμένει, και όλες σχεδόν οι βιβλιοθήκες και οι κλάσεις της Java μπορούν να χρησιμοποιηθούν και εδώ.

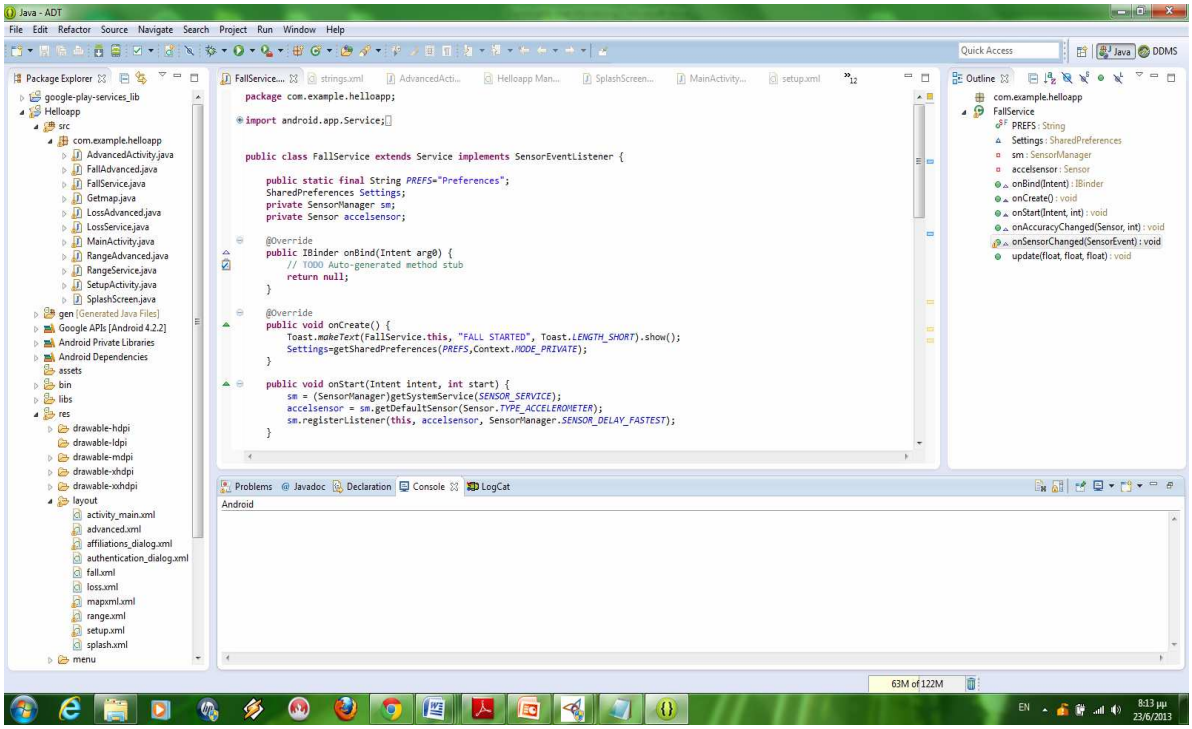
Το επίσημο εργαλείο ανάπτυξης για εφαρμογές Android είναι το **Eclipse**. Το Eclipse είναι μια σουίτα προγραμματισμού (όπως το Net Beans για την Java) το οποίο παρέχει σε ένα εύχρηστο γραφικό περιβάλλον επεξεργαστή κειμένου, ευρετήριο κλάσεων και αρχείων δόμησης, αποσφαλματωτή (debugger), αλλά και ενσωματωμένη AVD (Android Virtual Machine), δηλαδή μια «εικονική» συσκευή κινητού, όπου μπορεί να δοκιμαστεί η εφαρμογή που αναπτύσσουμε. Επίσης, υποστηρίζει δημιουργία xml αρχείων για τον ορισμό των δομικών στοιχείων της γραφικής διεπαφής των εφαρμογών, αλλά και γραφική απεικόνισή τους με δυνατότητα drag & drop των διαφόρων στοιχείων διεπαφής.

Κλείνοντας, παρακάτω είναι μια λίστα με τα απαραίτητα εργαλεία για την ανάπτυξη και τη δημιουργία εφαρμογών Android:

1. JDK (Java Developer Kit): Περιβάλλον μέσα στο οποίο μπορεί να τρέξει ένα πρόγραμμα σε Java.

2. Eclipse IDE: Το εργαλείο ανάπτυξης εφαρμογών Android.

3. Android SDK: Αντίστοιχο με το JDK, αλλά αφορά την πλατφόρμα Android.



Εικόνα 27: Το περιβάλλον εργασίας του Eclipse. Διακρίνονται στο κέντρο ο text editor, αριστερά η ιεραρχία με όλα τα αρχεία και τις κλάσεις του project, κάτω βρίσκεται η κονσόλα μηνυμάτων και δεξιά λίστα με μεταβλητές και μεθόδους της κάθε κλάσης.

7.4 XML (Extensible Markup Language)

Η XML είναι μια γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Σχεδιάστηκε αρχικά δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο διαδίκτυο, όπου πλέον είναι αρκετά διαδεδομένη. Μοιάζει πολύ στην HTML, και χρησιμοποιεί όπως και εκείνη ετικέτες (tags) για να δηλώνει στοιχεία (τα οποία περιβάλλονται από τα σύμβολα < >).

Η χρήση της XML στην εφαρμογή μας συνίσταται κυρίως στην αναπαράσταση του γραφικού περιβάλλοντος χρήστη της εφαρμογής (user interface). Ένα αρχείο xml μπορεί να αποτελεί μια οθόνη το κινητού, και οι διάφορες ετικέτες του αποτελούν διάφορα στοιχεία του interface (για παράδειγμα οι ετικέτες <EditText> , <Image>, <ListView> αφορούν αντίστοιχα στα στοιχεία «πλαίσιο εισαγωγής κειμένου», «εικόνα», «λίστα»). Ανάμεσα στις ετικέτες είναι δυνατό να προστεθούν διάφορες εντολές που μορφοποιούν ποικιλοτρόπως το αντικείμενο αυτό, και υπάρχει και η δυνατότητα συνδέσεων με προγραμματιστικές λειτουργίες και μεθόδους (για παράδειγμα, να συνδεθεί ένα κουμπί με την είσοδο σε μια νέα Activity).

Η χρήση της XML είναι εκτεταμένη στην εφαρμογή της παρούσας διπλωματικής για τη δημιουργία και τη μορφοποίηση του γραφικού περιβάλλοντός της και θα την συναντήσουμε αρκετά στο Κεφάλαιο 8.

7.5 GPS (Global Positioning System)

Η εφαρμογή μας χρησιμοποιεί την τεχνολογία GPS που είναι ενσωματωμένη στα σημερινά smartphone για να προσδιορίσει τη θέση του χρήστη ανά πάσα χρονική στιγμή. Ακολουθεί μια περιγραφή της τεχνολογίας αυτής και της λειτουργίας της.

Το παγκόσμιο σύστημα προσδιορισμού γεωγραφικής θέσης ή GPS (Global Positioning System) είναι ένα σύστημα ράδιο - πλοήγησης, το οποίο αποτελείται από ένα δίκτυο 24 δορυφόρων και από επίγειους σταθμούς κατανεμημένους σε όλο τον κόσμο. Το σύστημα χρησιμοποιεί τους δορυφόρους αυτούς και τους επίγειους σταθμούς ως σημεία αναφοράς για να υπολογίσει τη θέση που βρισκόμαστε, με ακρίβεια λίγων μέτρων. Τα δεδομένα της θέσης στέλνονται σε δέκτες GPS, οι οποίοι καθώς πλέον έχουν μειώσει κατά πολύ το μέγεθός τους, μπορούν και ενσωματώνονται και σε συσκευές όπως κινητά τηλέφωνα. Έτσι, στην περίπτωσή μας, έχουμε τη δυνατότητα να εκμεταλλευτούμε τον δέκτη GPS του κινητού και να προσδιορίσουμε τη θέση του ασθενούς.

Η λειτουργία του GPS περιλαμβάνει τα εξής βήματα [12]:

1) Διαδικασία “τριγωνισμού” (triangulation) από τους δορυφόρους, μια γεωμετρική ρουτίνα που προσδιορίζει τη θέση μας μετρώντας την απόσταση από τρεις δορυφόρους.

2) Μέτρηση απόστασης από τους δορυφόρους χρησιμοποιώντας τον χρόνο μετάδοσης των ραδιοσημάτων.

3) Συγχρονισμός ρολογιών δέκτη-δορυφόρου.

4) Εύρεση θέσης των δορυφόρων στον ουρανό (απαραίτητο να τη γνωρίζει ο δέκτης για να λειτουργήσει σωστά η διαδικασία του τριγωνισμού).

5) Διόρθωση σφαλμάτων.

Η θέση του χρήστη ερμηνεύεται μέσω δύο μεταβλητών: του γεωγραφικού πλάτους (latitude – προσδιορίζει τη θέση ενός σημείου πάνω στη Γη σε σχέση με τον Ισημερινό) και του γεωγραφικού μήκους (longitude – προσδιορίζει τη θέση ενός σημείου πάνω στη Γη σε σχέση με τον Μεσημβρινό του Γκρίνουιτς).

Το GPS λοιπόν, παρέχει αρκετά πλεονεκτήματα με τη χρήση του:

- Προσδιορίζει με ακρίβεια μέτρων τη θέση ενός σημείου πάνω στην επιφάνεια της Γης
- Οδηγεί με ακρίβεια προς ένα σημείο προορισμού
- Οι μετρήσεις που κάνει δεν επηρεάζονται από τις μεταβολές του καιρού
- Οι δέκτες του έχουν μικρό μέγεθος και μπορούν να ενσωματωθούν σε κινητά τηλέφωνα

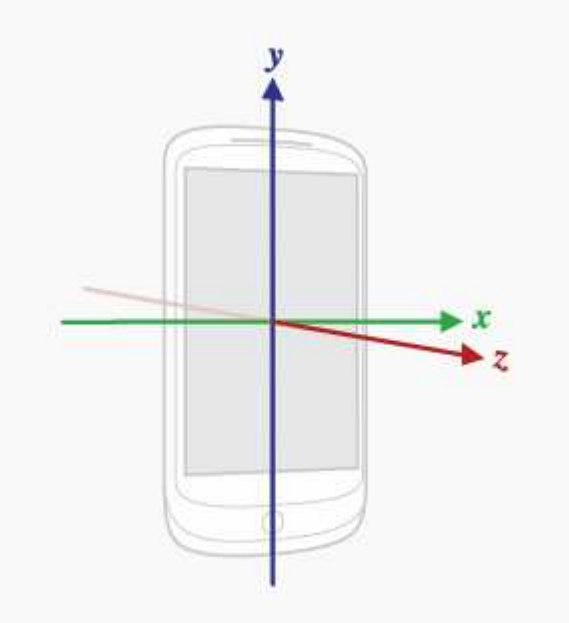
Υπάρχουν, όμως, και ορισμένα μειονεκτήματα:

- Η ακρίβεια των δεκτών εξαρτάται με το πόσους δορυφόρους συνδέεται: εάν είναι λιγότεροι από τρεις, τότε θα υπάρχουν μεγάλα σφάλματα.
- Δεν λειτουργεί σε κλειστούς χώρους.
- Είναι ιδιαίτερα ενεργοβόρο τις κινητές συσκευές.
- Δεν συνιστάται για εντοπισμό θέσης όπου απαιτούμε ακρίβεια εκατοστών.

7.6 Accelerometer

Το **επιταχυνσιόμετρο (accelerometer)** είναι μια ηλεκτρομηχανική συσκευή που έχει την ικανότητα να μετρά δυνάμεις επιτάχυνσης. Αυτές οι δυνάμεις μπορεί να είναι στατικές, όπως είναι η επιτάχυνση της βαρύτητας, ή δυναμικές όταν προκαλούνται/προέρχονται από αλλαγές στην ταχύτητα ή στην διεύθυνση της κίνησης (επιταχύνσεις, επιβραδύνσεις, στροφές). Επίσης, έχουν τη δυνατότητα να μετρούν δυνάμεις βάσει της επιτάχυνσης της βαρύτητας (η λεγόμενη «επιτάχυνση g»).

Πιο συγκεκριμένα, στα επιταχυνσιόμετρα που είναι ενσωματωμένα στα κινητά τηλέφωνα, επιτρέπουν στο smartphone να ανιχνεύει την κατεύθυνση της συσκευής και να αναπροσαρμόζει το περιεχόμενο που εμφανίζει (η αλλαγή του προσανατολισμού της οθόνης που συμβαίνει όταν τη γυρνάμε στο πλάι) αλλά και να υπολογίζει τις επιταχύνσεις στο τρισδιάστατο σύστημα αξόνων (ξεχωριστά δηλαδή σε κάθε άξονα), όπως φαίνεται στην Εικόνα 28.



Εικόνα 28: Οι άξονες στους οποίους μετρά επιταχύνσεις το επιταχυνσιόμετρο [13]

Το επιταχυνσιόμετρο, λοιπόν, μετρά τις επιταχύνσεις (Ad) που εφαρμόζονται στην συσκευή μέσω του τύπου: **Ad = -g - ∑F / m (m=μάζα)**

Μερικά παραδείγματα:

- Όταν η συσκευή είναι σε ένα τραπέζι ακίνητη (χωρίς επιτάχυνση), η συσκευή μετράει g=9,81 m/s (η ταχύτητα της συσκευής είναι ίση με το μηδέν, μείον την δύναμη της βαρύτητας).
- Όταν η συσκευή βρίσκεται σε ελεύθερη πτώση, η συσκευή μετράει g=0m/s.

Στα πλαίσια της εργασίας, χρησιμοποιήθηκε το επιταχυνσιόμετρο του κινητού τηλεφώνου για τα σενάρια Fall Detection και Loss Detection. Οι επιταχύνσεις που καταγράφει αναλύονται και ερμηνεύονται κατάλληλα για την εξαγωγή συμπερασμάτων ως προς τη συμπεριφορά τη χρήστη.

7.7 Prototyper

Το Prototyper (screenshots του οποίου παρουσιάστηκαν εκτενώς στην ενότητα 5.4) αποτελεί ένα εργαλείο δημιουργίας γραφικών διεπαφών για πρωτότυπα εφαρμογών χαμηλής-μέσης πιστότητας. Αποτελεί ουσιαστικά το πρώτο βήμα πριν την πρώτη έκδοση μιας εφαρμογής, ως το προσχέδιο για να αναπαρασταθεί πρακτικά αυτό που έχει στο μυαλό του ο σχεδιαστής. Παρέχει δυνατότητες όπως drag & drop τοποθέτηση πλειάδας οπτικών αντικειμένων, μεταβλητότητα μεγέθους και προσθήκη widgets (έτοιμες μικροεφαρμογές). Επίσης, δίνεται η δυνατότητα για σύνδεση οθονών μεταξύ τους, δημιουργία animations αλλά και προσομοίωση βάσεων δεδομένων. Το γεγονός πως δεν εμπλέκει προγραμματιστικές διαδικασίες (τα στοιχεία ελέγχου που προστίθενται στις οθόνες δεν υλοποιούν κάποιου είδους λογική), καθιστά δυνατή τη χρήση του Prototyper και από μη προγραμματιστές.

8. Παρουσίαση εφαρμογής

8.1 Από τη Σχεδίαση στην Υλοποίηση

Όπως αναφέρθηκε στο Κεφάλαιο 5 κατά την παρουσίαση του πρωτοτύπου της εφαρμογής, υπήρξαν αρκετές διαφοροποιήσεις στην τελική μορφή του περιβάλλοντος αλληλεπίδρασης με το χρήστη σε σχέση με τον αρχικό σχεδιασμό. Αυτό συνέβη για αρκετούς λόγους, οι περισσότεροι από τους οποίους έχουν να κάνουν με την αλληλεπίδραση με το πιθανό σύνολο χρηστών της συγκεκριμένης εφαρμογής.

Μετά από την επίδειξη των οθονών αυτών σε ένα σύνολο ανθρώπων τρίτης ηλικίας (όχι απαραίτητα πάσχοντες από ασθένεια μορφής άνοιας) αλλά και γενικότερα σε ανθρώπους ανειδίκευτους με την τεχνολογία, τα αποτελέσματα ήταν αποθαρρυντικά. Όταν επιχειρήθηκε να εξηγηθεί ο τρόπος που θα λειτουργεί η εφαρμογή και ο τρόπος που θα χρησιμοποιείται, υπήρξε προβληματισμός. Οι εν δυνάμει χρήστες της εφαρμογής δυσκολεύτηκαν να εξοικειωθούν με το interface και δεν μπορούσαν να καταλάβουν τη λειτουργία των στοιχείων ελέγχου της οθόνης. Όταν ζητήθηκε επίσης η συμβουλή ενός επαγγελματία υγείας, έγινε αντιληπτό πως υπήρχε πρόβλημα. Ο ίδιος εξήγησε ότι το target group της εφαρμογής, οι άνθρωποι δηλαδή με άνοια, σε αρκετές περιπτώσεις δυσκολεύονται να καταλάβουν το που βρίσκονται και να θυμηθούν το όνομά τους· πόσο μάλλον να μπορούν να χειριστούν και να αλληλεπιδράσουν με ένα κινητό.

Έτσι, λοιπόν, αποφασίστηκε η αλλαγή σε κάποια μέρη του interface, για να ανταποκρίνεται καλύτερα στις απαιτήσεις και τις προδιαγραφές των εν δυνάμει χρηστών της. Η αλλαγή αυτή θα πραγματοποιούνταν γύρω από δύο άξονες: αφαίρεση εξεζητημένων στοιχείων του γραφικού περιβάλλοντος (λίστες, αναδιπλούμενα μενού) και προσπάθεια για λειτουργία χωρίς την παρέμβαση του χρήστη. Η εφαρμογή θα πρέπει προσφέρει την επιθυμητή λειτουργικότητα στο χρήστη, ενώ αυτός θα παρεμβαίνει μόνο όταν είναι απαραίτητο, και οι ενέργειες που θα πρέπει να κάνει δε θα ξεπερνούν το πάτημα ενός κουμπιού. Για τα υπόλοιπα θα πρέπει να φροντίσει ο συγγενής/γιατρός.

Κάποιες από τις αλλαγές που γίνανε είναι:

- Η αρχική οθόνη, η οποία είναι η μοναδική στην οποία μπορεί να εισέρθει ο χρήστης, απαλλάχθηκε από περιττά στοιχεία (πλαίσια κειμένου όπου μπορεί να

  βάλει το όνομά του και μεγάλα εισαγωγικά σημειώματα) και αποτελείται από ένα κουμπί εξόδου και ένα κουμπί εισόδου στις οθόνες ρυθμίσεων.

- Το κουμπί Setup βέβαια, παραπέμπει πρώτα σε ένα παράθυρο διαλόγου όπου πρώτα ο χρήστης πρέπει να εισάγει username και password για να προχωρήσει

περαιτέρω. Αυτό συμβαίνει, βέβαια, για να έχουν ουσιαστικά πρόσβαση σε αυτές μόνο εξειδικευμένα άτομα, όπως ο γιατρός ή ο συγγενής του ασθενούς.

- Επίσης, αφαιρέθηκε η δυνατότητα στο χρήστη να επιλέγει ποιες υπηρεσίες θα είναι ενεργοποιημένες κάθε φορά. Η ενεργοποίησή τους εισήλθε στις οθόνες των ρυθμίσεων, στα ασφαλή χέρια των διαπιστευμένων χειριστών της εφαρμογής.
- Από εκεί και πέρα, οι υπόλοιπες αλλαγές στο interface θα παρουσιαστούν στη συνέχεια στην αναλυτική παρουσίαση της εφαρμογής.

8.2 Δραστηριότητες/Υπηρεσίες/Διεπαφή (Xml)

Πριν δοθεί η λεπτομερής παρουσίαση της εφαρμογής, παρατίθεται μία σύντομη επεξήγηση των δομικών της στοιχείων. Ουσιαστικά, το προγραμματιστικό μέρος της εφαρμογής αποτελείται από τριών ειδών αρχεία: Δραστηριότητες-Υπηρεσίες-Διεπαφές (xml).

Η **Δραστηριότητα (Activity)** αποτελεί ένα συστατικό της εφαρμογής το οποίο παρέχει μια οθόνη με την οποία ο χρήστης μπορεί να αλληλεπιδράσει, ώστε να πραγματοποιήσει μια ενέργεια ή να λάβει κάποιου είδους πληροφορία. Μια εφαρμογή συνήθως αποτελείται από πολλές δραστηριότητες, οι οποίες συνδέονται μεταξύ τους μέσω διαφόρων στοιχείων ελέγχου. Συνήθως, μια δραστηριότητα ορίζεται ως κύρια, (Main Activity) η οποία είναι η οθόνη που παρουσιάζεται στον χρήστη με την εκκίνηση της εφαρμογής. Στη συνέχεια, η δραστηριότητα μπορεί να εκκινήσει μια άλλη δραστηριότητα ώστε να πραγματοποιήσει κάποια λειτουργία. Μια δραστηριότητα ουσιαστικά αντιστοιχεί περίπου σε ένα αρχείο κλάσης στην Java (και ένα αρχείο Activity έχει επέκταση Java), και μπορούμε να πούμε πως αποτελεί το κυριότερο συστατικό μιας εφαρμογής σε Android, καθώς εκεί είναι που γίνεται όλη η προγραμματιστική δραστηριότητα (ορισμός λογικής της εφαρμογής).

Αναφέραμε πως οι δραστηριότητες ουσιαστικά αντιστοιχούν στις οθόνες της εφαρμογής. Το Android παρέχει τη δυνατότητα στον προγραμματιστή να υλοποιήσει τη μορφή της διεπαφής χρήστη (user interface) μιας οθόνης ανεξάρτητα από το προγραμματιστικό μέρος, και αυτό υλοποιείται μέσω **xml** περιγραφών. Ο προγραμματιστής μπορεί να δημιουργήσει ένα αρχείο xml, να το εμπλουτίσει (με την βοήθεια των αντίστοιχων tags) με διάφορα κομμάτια του interface (γραφικά, πλαίσια κειμένου, λίστες, κουμπιά κλπ.), να τα μορφοποιήσει όπως ακριβώς θέλει και να τους καταχωρήσει μοναδικά ονόματα, τα οποία χρησιμοποιούνται ως αναφορά από την σχετική δραστηριότητα για να προστεθεί λογική στην αλληλεπίδραση. Στη συνέχεια, μπορεί με μια απλή εντολή να αντιστοιχίσει το αρχείο xml που δημιούργησε με τη δραστηριότητα που επιθυμεί. Υπάρχει ακόμα και η δυνατότητα (μέσω του Eclipse) να βλέπει ένα προσχέδιο της οθόνης σε γραφική αναπαράσταση, ώστε να ελέγχει την πρόοδο της δουλειάς που κάνει στο αρχείο xml.

Το τρίτο κύριο συστατικό της εφαρμογής μας, είναι η υλοποίηση **υπηρεσιών (Services)**. Η υπηρεσία είναι ένα συστατικό μιας εφαρμογής, το οποίο υλοποιεί την «επιθυμία» μιας

εφαρμογής να πραγματοποιήσει μια διαδικασία για ένα παρατεταμένο χρονικό διάστημα, χωρίς να αλληλεπιδρά με το χρήστη. Ουσιαστικά, μια υπηρεσία πραγματοποιεί μια συγκεκριμένη διεργασία στο παρασκήνιο και ο χρήστης δεν αντιλαμβάνεται την ύπαρξή της. Είναι εύκολα κατανοητό, λοιπόν, πως μια υπηρεσία δεν απαιτεί αντιστοίχηση με γραφικό περιβάλλον, και μπορεί να εκκινηθεί από τον χρήστη ή από μια δραστηριότητα, καταλαμβάνοντας τους αντίστοιχους πόρους.

Στη συνέχεια, θα παρουσιαστούν μια-μια οι δραστηριότητες και οι υπηρεσίες της εφαρμογής μας με επεξήγηση των σημαντικών σημείων και των τεχνικών που χρησιμοποιούνται για την παροχή των υπηρεσιών στον χρήστη, όπως αναφέρθηκαν στην ενότητα 5.2.

8.3 Οθόνη Καλωσορίσματος

Η αρχική οθόνη της εφαρμογής (Εικόνα 29). Παρουσιάζει ένα γραφικό με ένα μήνυμα καλωσορίσματος στην εφαρμογή. Δεν υιοθετεί κάποιου είδους αλληλεπίδραση με τον χρήστη, και μετά από 5 δευτερόλεπτα εξαφανίζεται και αυτόματα καλείται η επόμενη δραστηριότητα (η οποία είναι η ουσιαστική αρχική οθόνη της εφαρμογής) SetupActivity.java. Η κλήση της γίνεται με τη βοήθεια της κλάσης *Intent* του Android, η οποία χρησιμοποιείται για να καλέσει μια δραστηριότητα στο προσκήνιο της εφαρμογής.

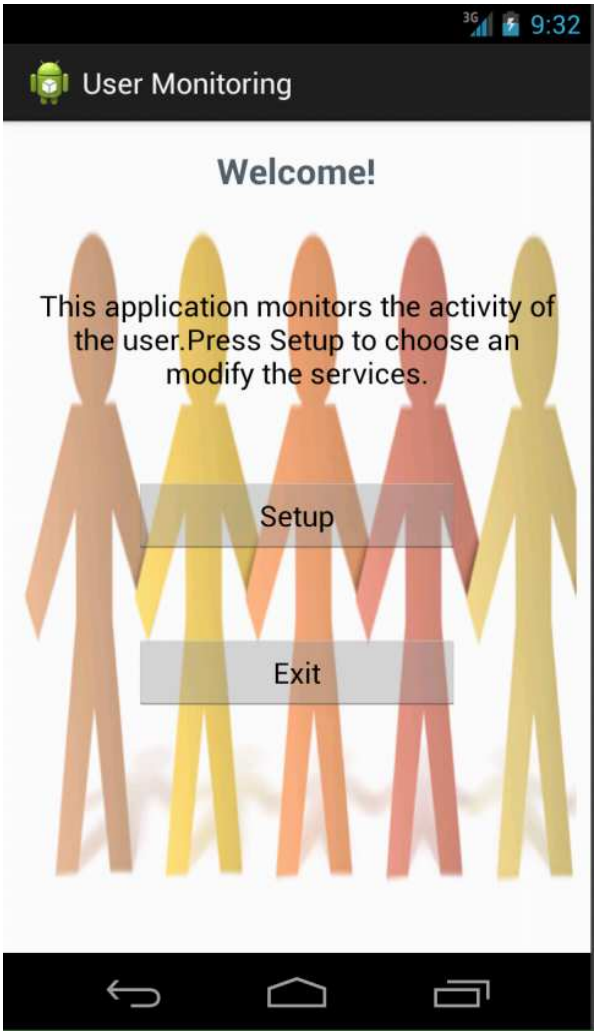
```
Intent intent = new Intent(SplashScreen.this, SetupActivity.class);
            SplashScreen.this.startActivity(intent);
            SplashScreen.this.finish();
```

Με το παραπάνω κομμάτι κώδικα δημιουργείται ένα αντικείμενο τύπου Intent με το οποίο γίνεται κλήση της SetupActivity.java. Το συγκεκριμένο κομμάτι θα χρησιμοποιείται συχνά, κυρίως σε κουμπιά, για την κλήση δραστηριοτήτων και τη μετάβαση από οθόνη σε οθόνη.

Εικόνα 29: Η SplashScreen της εφαρμογής

8.4 Αρχική Οθόνη

Η επόμενη οθόνη της εφαρμογής είναι η SetupActivity.java, η οποία αποκτά τον ουσιαστικό ρόλο της αρχικής οθόνης της εφαρμογής (εδώ υπάρχει αλληλεπίδραση). Αυτή υλοποιεί ένα απλό user interface με δύο κουμπιά, το Setup που μεταβαίνει στην οθόνη ρυθμίσεων και το Exit που πραγματοποιεί έξοδο από την εφαρμογή. Αυτή η οθόνη είναι ουσιαστικά η μοναδική που μπορεί να έχει πρόσβαση ο ασθενής-χρήστης, καθώς από εκεί και πέρα για είσοδο στις ρυθμίσεις απαιτείται authentication διότι αυτά τα στοιχεία της εφαρμογής είναι προσπελάσιμα από κατάλληλο άτομο που θα φροντίζει τον τελικό χρήστη (συγγενή/ επαγγελματία υγείας).



Εικόνα 30: Η SetupActivity.java. Διακρίνεται το μήνυμα καλωσορίσματος και τα κουμπιά Setup και Exit.

Η μορφή της οθόνης σχεδιάστηκε μέσω του setup.xml αρχείου, το οποίο αντιστοιχήθηκε στην Activity μέσω της παρακάτω εντολής

```
setContentView(R.layout.setup);
```

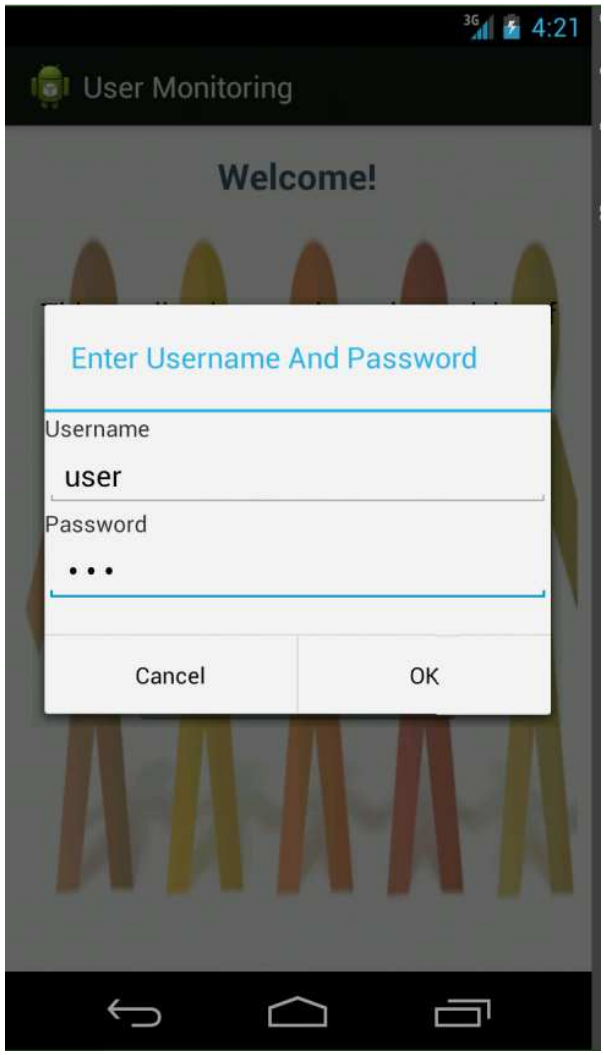
Το κουμπί Exit υλοποιεί την λειτουργία του μέσω μιας απλής κλήσης της μεθόδου finish() μέσα στη μέθοδο onClick, που όπως επεξηγεί το όνομά της χειρίζεται συμβάντα πατήματος κουμπιών. Η αναφορά στο στοιχείο button του αρχείου xml έγινε μέσω της παρακάτω εντολής:

```
Button button4=(Button)findViewById(R.id.button4);
```

Στην παραπάνω εντολή, δημιουργείται ένα αντικείμενο (button4) τύπου Button, το οποίο αντιστοιχεί στην ετικέτα xml με όνομα *button4.*

Παράθυρο διαλόγου Authentication

Αναφέρθηκε πριν ότι με την επιλογή Setup, πραγματοποιείται είσοδος στην οθόνη των ρυθμίσεων της εφαρμογής. Επειδή όμως έχει απαιτηθεί από το σχεδιασμό της εφαρμογής η πρόσβαση σε αυτή να γίνεται μόνο από τον γιατρό ή το συγγενή του χρήστη, με το πάτημα του κουμπιού εμφανίζεται πρώτα ένα παράθυρο διαλόγου όπου ζητείται ταυτοποίηση χρήστη με εισαγωγή username και password (Εικόνα 31).



Εικόνα 31: Το παράθυρο διαλόγου για είσοδο στις οθόνες ρυθμίσεων

Αρχικά, ορίζεται ένα αντικείμενο View με τη βοήθεια της κλάσης LayoutInflater, το οποίο θα αποτελεί τη μορφή του παραθύρου διαλόγου. Μετά, αντιστοιχίζεται σε αυτό το αντικείμενο και το αρχείο xml στο οποίο έχει δημιουργηθεί το επιθυμητό γραφικό περιβάλλον του παραθύρου και ολοκληρώνεται η διαδικασία της γραφικής αναπαράστασης του παραθύρου. Ο αντίστοιχος κώδικας παρατίθεται παρακάτω:

```
LayoutInflater inflater=(LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
final View
layout1=inflater.inflate(R.layout.authentication_dialog,(ViewGroup)findView
ById(R.id.root));
AlertDialog.Builder builder=new AlertDialog.Builder(this);
builder.setView(layout1);
```

```
builder.setTitle(R.string.Authentication);
```

Διακρίνεται το αντικείμενο View layout1, το οποίο δέχεται ως παράμετρο το αρχείο xml authentication_dialog όπου έχει υλοποιηθεί η όψη του παραθύρου διαλόγου. Στη συνέχεια, δημιουργείται το παράθυρο διαλόγου μέσω της AlertDialog και του αντιστοιχίζεται το αντικείμενο View που δημιουργήθηκε πριν, ενώ παράλληλα, μέσω της μεθόδου setTitle, αντιστοιχίζεται το αλφαριθμητικό Authentication που περιέχει τον επιθυμητό τίτλο του παραθύρου.

Στη συνέχεια, πρέπει να γίνει έλεγχος της ορθότητας του username και του password που θα εισάγει ο χρήστης. Οι αρχικές ρυθμίσεις στα πλαίσια του πρωτοτύπου ορίζουν πως τα αποδεκτά username και password είναι αντίστοιχα **“user”** και **“123”.** Επομένως, η διαδικασία είναι η εξής: με το πάτημα του κουμπιού “OK” γίνεται έλεγχος των τιμών που εισήχθησαν από το χρήστη στα δύο πεδία. Εάν οι τιμές στα πεδία αντιστοιχούν στα αποδεκτά username και password, τότε μέσω κλήσης της Intent η εφαρμογή μεταβαίνει στην οθόνη προηγμένων ρυθμίσεων. Εάν τα πεδία εισόδου έχουν λανθασμένα στοιχεία, τότε εμφανίζεται ένα μήνυμα λάθους και το παράθυρο κλείνει. Με αυτόν τον τρόπο, λοιπόν, εξασφαλίζεται πως ο ασθενής δε θα αλλάξει τις ρυθμίσεις κατά λάθος και δε θα μπερδευτεί, ενώ παράλληλα η εφαρμογή θα ρυθμίζεται μόνο από αρμόδια πρόσωπα και με συνειδητό τρόπο.

8.5 Ενεργοποίηση/Απενεργοποίηση Υπηρεσιών και Είσοδος στις Προηγμένες Ρυθμίσεις

Μετά τον έλεγχο των στοιχείων στο παράθυρο διαλόγου, η εφαρμογή μεταβαίνει στην AdvancedActivity.java. Εδώ συναντώνται τρία toggle buttons (κουμπιά καταστάσεων) και δίπλα από αυτά τρία κουμπιά με τα ονόματα των υπηρεσιών (Εικόνα 32). Τα κουμπιά ενεργοποιούν και απενεργοποιούν την υπηρεσία που αναφέρεται δίπλα τους, και τα κουμπιά με τα ονόματα των υπηρεσιών εισάγουν τον χρήστη στην αντίστοιχη ατομική οθόνη ρυθμίσεων της κάθε υπηρεσίας.

Τα κουμπιά έχουν τη δυνατότητα να «θυμούνται» την κατάστασή τους ακόμα και μετά το κλείσιμο της εφαρμογής. Έτσι, εάν ο χρήστης ενεργοποιήσει μια υπηρεσία, αυτή θα συνεχίσει να τρέχει στο παρασκήνιο ακόμα και αν κλείσει η εφαρμογή, και θα απενεργοποιηθεί μόνο όταν ξαναπατηθεί το κουμπί.

Αυτή η ικανότητα «μνήμης» που έχουν τα κουμπιά, οφείλεται στην κλάση SharedPreferences, η οποία ουσιαστικά υλοποιεί μια μικρή δομή δεδομένων όπου μπορούν να αποθηκεύονται και να ανακτώνται οι ρυθμίσεις που αφορούν μια εφαρμογή. Η κλήση της γίνεται μέσω της παρακάτω εντολής:

```
Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);
```

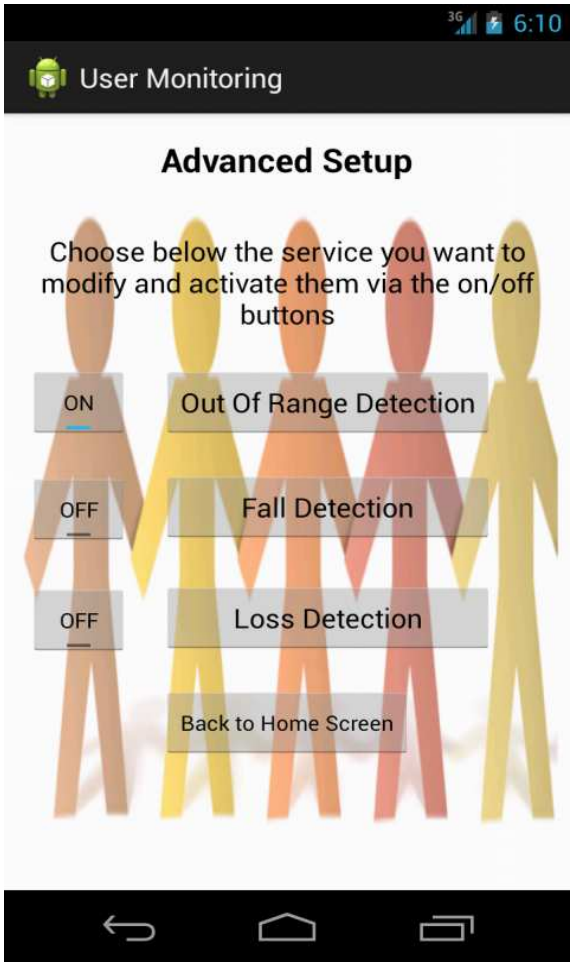
Όταν λοιπόν είναι επιθυμητή η αποθήκευση δεδομένων, αυτό επιτυγχάνεται με τη δημιουργία αντικειμένου της κλάσης Editor. Παρακάτω παρατίθεται ένα παράδειγμα αποθήκευσης της ενεργοποίησης του κουμπιού που αφορά την υπηρεσία “Out of Range”.

```
boolean on = ((ToggleButton) view).isChecked();

          if (on) {
             startService(new Intent(this, RangeService.class));

             Editor editor=Settings.edit();
                    editor.putBoolean(TOGGLE_RANGE, true);
                    editor.commit();

          } else {
             stopService(new
Intent(AdvancedActivity.this,RangeService.class));
             Editor editor=Settings.edit();
                    editor.putBoolean(TOGGLE_RANGE, false);
                    editor.commit();
          }
```

Στο παραπάνω παράδειγμα αποθηκεύεται ως λογική μεταβλητή (TOGGLE_RANGE) η κατάσταση του toggle button σε αληθής ή ψευδής, ανάλογα με την κατάσταση της μεταβλητής on (που καθορίζεται με το αν είναι ενεργοποιημένο το toggle button). Έτσι, όταν επιθυμούμε να δούμε εάν το κουμπί είναι ενεργοποιημένο, γίνεται ανάκτηση της TOGGLE_RANGE και διαβάζεται η τιμή της. Η ανάκτηση γίνεται μέσω της παρακάτω διαδικασίας:

```
Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);
boolean tgprefrange = Settings.getBoolean(TOGGLE_RANGE, false);
```

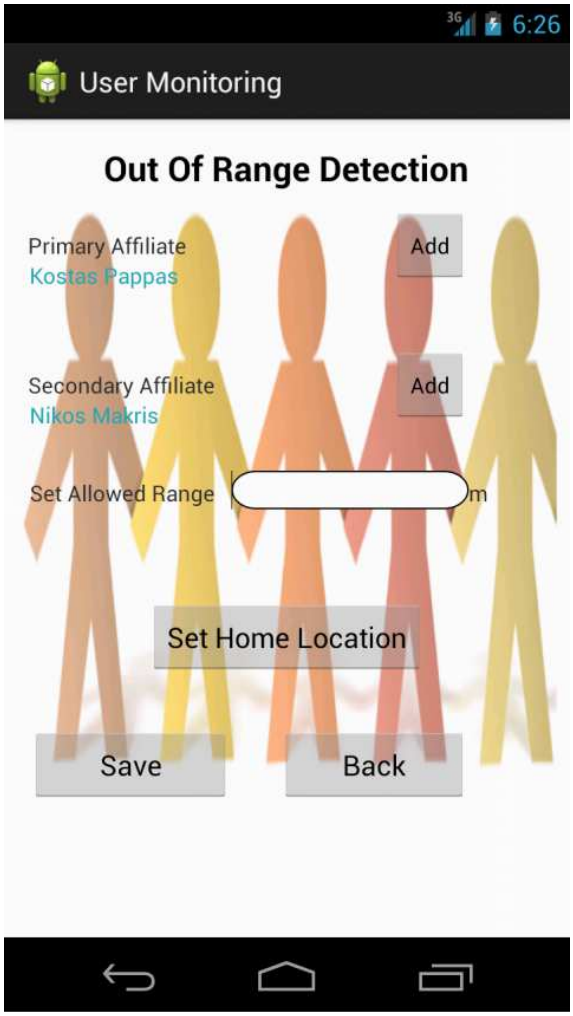
Πρώτα ανακτάται το αρχείο PREFS όπου είναι αποθηκευμένες όλες οι ρυθμίσεις και μετά αποθηκεύεται στη Boolean μεταβλητή tgprefrange η τιμή της αποθηκευμένης TOGGLE_RANGE.



Εικόνα 32: Η οθόνη προηγμένων ρυθμίσεων, με ενεργοποιημένο το Out of Range

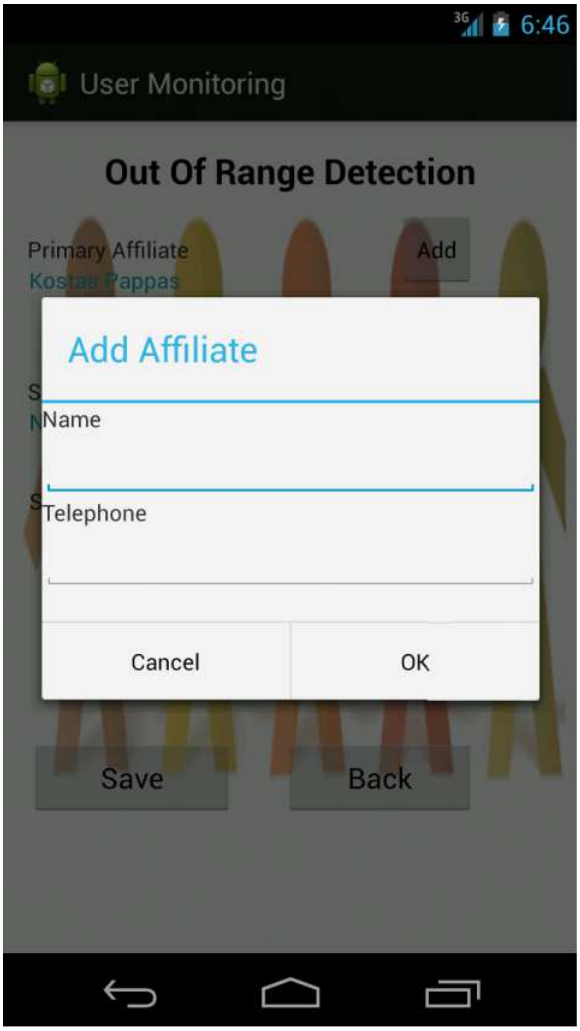
8.6 Ατομικές Ρυθμίσεις Υπηρεσιών

Στη συνέχεια παρουσιάζονται οι ατομικές ρυθμίσεις προτιμήσεων για κάθε υπηρεσία. Στην Εικόνα 33 παρουσιάζεται η οθόνη ρυθμίσεων της υπηρεσίας Out of Range.



Εικόνα 33: Οθόνη ρυθμίσεων Out of Range.

Όπως είναι φανερό, έχει αφαιρεθεί η λίστα με τους συγγενείς/γιατρούς προς ειδοποίηση, για το λόγο ότι σε μια περίπτωση συναγερμού θα πρέπει να ειδοποιούνται τα άκρως απαραίτητα και αρμόδια άτομα για να μη δημιουργούνται καταστάσεις πανικού. Αντί αυτής, έχουν προστεθεί δύο κουμπιά Add (Προσθήκη), με το πάτημα των οποίων ανοίγουν δύο πλαίσια διαλόγου (τα οποία υλοποιούνται όπως και το παράθυρο ταυτοποίησης) στα οποία μπορεί να προστεθεί όνομα και τηλέφωνο του φιλικά προσκείμενου προσώπου (Εικόνα 34). Με το πάτημα του κουμπιού, αποθηκεύονται στα SharedPreferences το όνομα αλλά και το τηλέφωνο, ώστε να ανακτηθούν στη συνέχεια (το τηλέφωνο ειδικά) από την αρμόδια υπηρεσία για να σταλεί το SMS ειδοποίησης.



Εικόνα 34: Εισαγωγή στοιχείων προσώπου προς ειδοποίηση σε περίπτωση συναγερμού

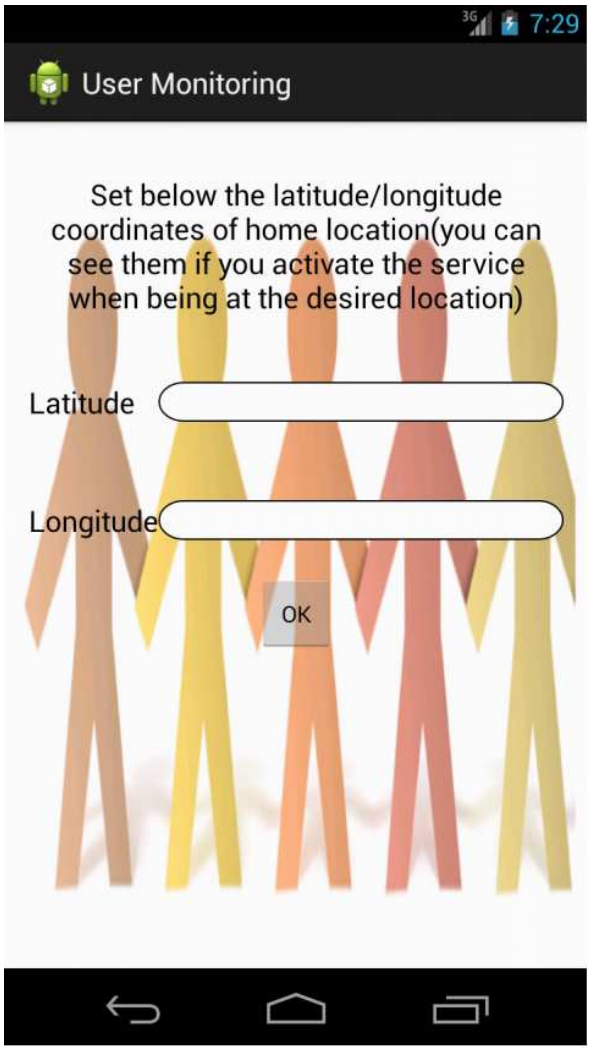
Εκτός από την αποθήκευση των προσώπων που θα ειδοποιηθούν σε περίπτωση ανάγκης, υπάρχει και το πλαίσιο όπου μπορεί να εισαχθεί η επιτρεπτή απόσταση κυκλοφορίας του χρήστη. Η αποθήκευση της τιμής αυτής (που αποθηκεύεται σε μέτρα) υλοποιείται στην μέθοδο onClick του κουμπιού Save, και πάλι με την κλήση των SharedPreferences. Επίσης, εμφανίζεται ένα μήνυμα για ένα σύντομο χρονικό διάστημα στην οθόνη που επιβεβαιώνει την αποθήκευση των αλλαγών. Η υλοποίηση του μηνύματος γίνεται μέσω του παρακάτω κώδικα:

```
Toast.makeText(RangeAdvanced.this, "Changes Saved",
Toast.LENGTH_SHORT).show();
```

Επίσης, γίνεται ένας έλεγχος για το εάν έχει συμπληρωθεί το πλαίσιο κειμένου που αφορά στο επιτρεπτό εύρος για να αποφευχθεί πιθανή δυσλειτουργία της εφαρμογής.

Τέλος, υπάρχει το κουμπί “Set Home Location”, το οποίο με το πάτημά του μας μεταφέρει σε μια νέα οθόνη. Εκεί, εμφανίζονται δύο πλαίσια κειμένου όπου εισάγονται οι συντεταγμένες latitude και longitude του επιθυμητού σημείου αναφοράς του χρήστη (αυτά ανακτώνται μέσω εμφάνισης της τωρινής τοποθεσίας, όπως θα δούμε στη συνέχεια στην

υλοποίηση της υπηρεσίας). Επιβεβαιώνοντας την επιλογή με το κουμπί OK, οι συντεταγμένες αποθηκεύονται στα SharedPreferences.



Εικόνα 35: Η οθόνη προσθήκης της αρχικής τοποθεσίας

Οι υπόλοιπες οθόνες για τις άλλες δύο υπηρεσίες (Fall Detection, Loss Detection) ακολουθούν παρόμοια φιλοσοφία, αποθηκεύοντας μεμονωμένα τα δικά τους δεδομένα. Έτσι, υπάρχει η δυνατότητα να ειδοποιούνται διαφορετικά άτομα για την κάθε υπηρεσία.

8.7 Η Υπηρεσία Out of Range

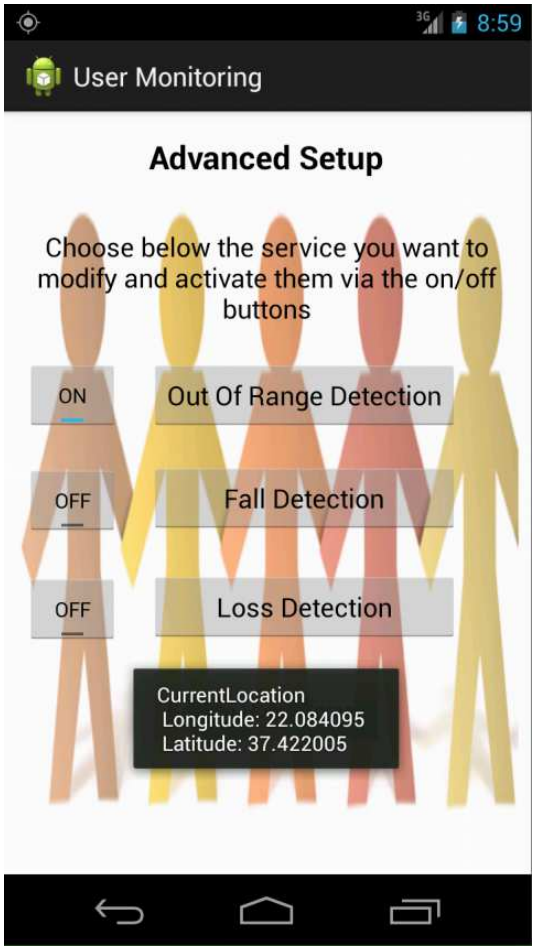
Παρακάτω θα παρουσιαστεί το αρχείο RangeService.java, όπου υλοποιείται η υπηρεσία Out of Range. Η γενική λογική που ακολουθείται συνοψίζεται στα εξής: Ανά τακτά χρονικά διαστήματα, ανακτάται μέσω του GPS η θέση του χρήστη. Έχοντας ανακτήσει την αρχική θέση που έχει αποθηκευτεί μέσω των ρυθμίσεων, γίνεται υπολογισμός της απόστασής τους μέσω αλγόριθμου που υπολογίζει αποστάσεις στην επιφάνεια της Γης [14]. Αυτή η απόσταση συγκρίνεται με την απόσταση που έχει τεθεί ως όριο μέσω των ρυθμίσεων· εάν είναι μεγαλύτερη, τότε ενεργοποιείται η περίπτωση συναγερμού και στέλνεται στον

ορισμένο σύνδεσμο ένα sms με ενημέρωση του γεγονότος και της θέσης του τελικού χρήστη.

Πιο αναλυτικά, στην μέθοδο onStart() της υπηρεσίας, γίνονται κάποιες αρχικοποιήσεις και δημιουργίες αντικειμένων που λαμβάνουν χώρα με το που ο χρήστης πατήσει το κουμπί ενεργοποίησης της υπηρεσίας. Ορίζονται ένα αντικείμενο «Διαχειριστής τοποθεσίας» (LocationManager) και ένας «Ακροατής Τοποθεσίας» (LocationListener) και μέσω της παρακάτω γραμμής ορίζεται η συχνότητα δειγματοληψίας της τοποθεσίας:

```
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,time_interval,0,ll);
```

όπου time_interval = 5 λεπτά στην περίπτωση αυτή. Επίσης, ορίζεται ένα αντικείμενο location το οποίο «ακροάζεται» την τοποθεσία του χρήστη. Τέλος, με την εκκίνηση της υπηρεσίας, εμφανίζεται ένα μήνυμα με τις συντεταγμένες της τρέχουσας τοποθεσίας (Εικόνα 36)· καταγράφοντας αυτές τις συντεταγμένες μπορεί ο γιατρός ή ο συγγενής του ασθενή, να καταχωρήσουν τις συντεταγμένες της αρχικής του τοποθεσίας (με την προϋπόθεση πως βρίσκονται στο επιθυμητό σημείο).



Εικόνα 36: Το μήνυμα που εμφανίζει τις συντεταγμένες της τρέχουσας τοποθεσίας

Στη συνέχεια, υλοποιείται η μέθοδος onLocationChanged που πραγματοποιεί τη μέτρηση και τη σύγκριση των αποστάσεων. Εκεί, ανακτώνται οι τρέχουσες συντεταγμένες του χρήστη μέσω των παρακάτω γραμμών:

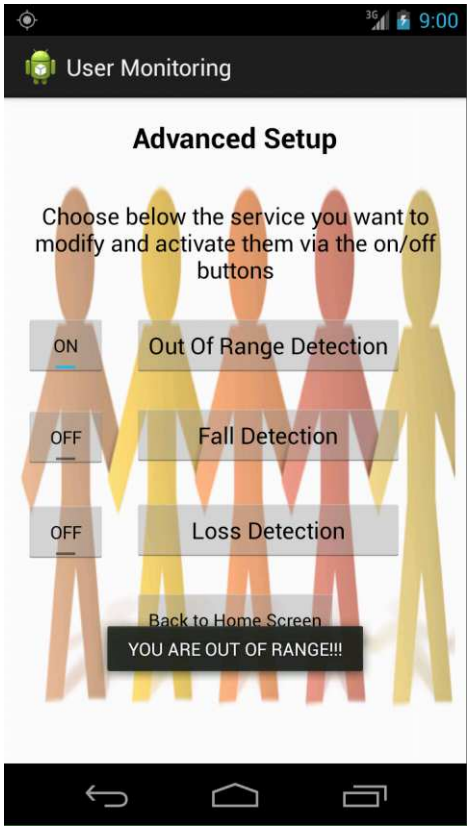
```
double lat=location.getLatitude();
double lon=location.getLongitude();
```

Ακολούθως, και με τη βοήθεια της μεθόδου parseDouble, μετατρέπονται σε μορφή double (δεκαδικός με πολλά ψηφία) οι συντεταγμένες του αρχικού σημείου αλλά και η επιτρεπόμενη απόσταση (όλα αυτά βεβαίως έχουν ανακτηθεί μέσω SharedPreferences) και στη συνέχεια καλείται η μέθοδος distance η οποία μέσω κάποιων υπολογισμών βρίσκει την απόσταση του χρήστη από το αρχικό σημείο. Το τελευταίο βήμα είναι, μέσω ενός απλού if case, η σύγκριση των δύο αποστάσεων (τρέχουσας και επιτρεπτής). Εάν η τρέχουσα απόσταση του χρήστη από την αρχική τοποθεσία υπερβαίνει το επιτρεπόμενο εύρος, τότε δημιουργείται alert και ξεκινά μια σειρά από διαδικασίες, όπως δόνηση, ηχητική ειδοποίηση, αλλά και αποστολή SMS στους ορισθέντες συνδέσμους, όπου και παρέχεται σύνδεσμος στο site των χαρτών της Google όπου εμφανίζεται η τοποθεσία του χρήστη. Το κομμάτι κώδικα με τις απαραίτητες ενέργειες είναι το παρακάτω:

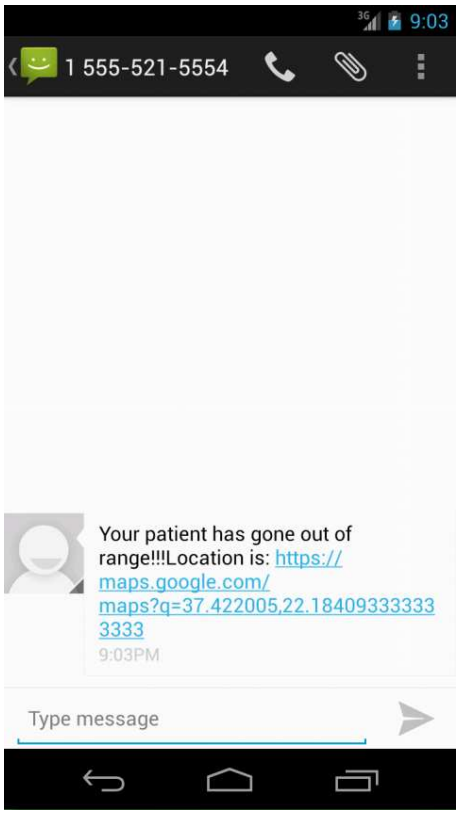
```
//sms alert
String smsmsg="Your patient has gone out of range!!!Location is:
https://maps.google.com/maps?q="+lat+","+lon;
sendSMS(Settings.getString(PHONERANGE1, ""),smsmsg);
sendSMS(Settings.getString(PHONERANGE2, ""),smsmsg);


//sound alert
Uri notification =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
Ringtone ring = RingtoneManager.getRingtone(getApplicationContext(),
notification);
ring.play();

//vibrate
Vibrator v = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
v.vibrate(2000);//2sec vibration
```

Ας σημειωθεί ότι η μέθοδος sendSMS δέχεται δύο μεταβλητές τύπου χαρακτήρα όπου στην πρώτη περιέχεται το τηλέφωνο και στη δεύτερη το κείμενο του SMS.

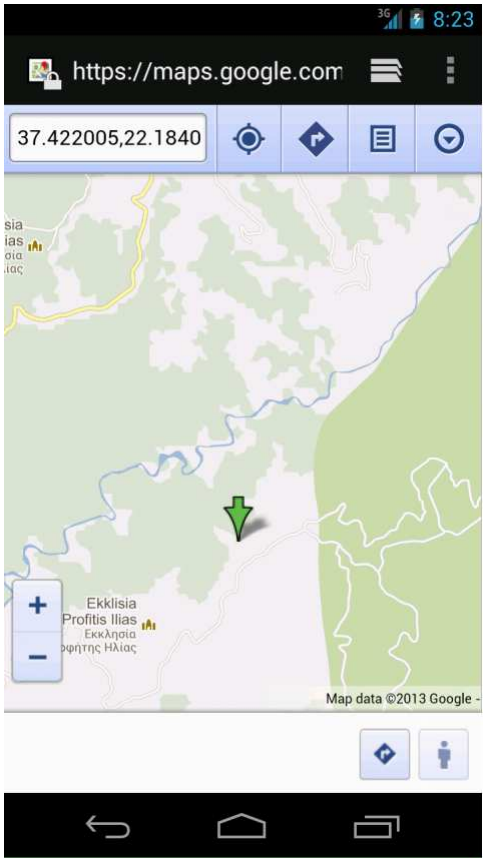
Παρακάτω παρατίθενται οι οθόνες του κινητού που περιγράφουν τα παραπάνω στάδια:



Εικόνα 37: Το μήνυμα σε περίπτωση που ο χρήστης βγει εκτός επιτρεπόμενων ορίων.



Εικόνα 38: Το μήνυμα που αποστέλλεται στον επιλεγμένο λήπτη. Φαίνεται και το link που εμφανίζει την τοποθεσία του χρήστη.



Εικόνα 39: Πατώντας το link, εμφανίζεται ο χάρτης, με το βέλος να υποδεικνύει την τοποθεσία του χρήστη.

8.8 Η Υπηρεσία Fall Detection

Η επόμενη υπηρεσία που υλοποιείται είναι η Fall Detection (Ανίχνευση Πτώσης). Η υλοποίηση γίνεται μέσω του αρχείου FallService.java, στο οποίο περιέχεται η λογική της ανίχνευσης πτώσης.

Η λογική διαδικασία για την ανίχνευση πτώσης συνοψίζεται στα εξής βήματα [15]:

1) Καταχωρούνται σε μεταβλητές οι επιταχύνσεις της συσκευής στους τρεις άξονες x,y,z (η ανανέωση των επιταχύνσεων γίνεται ανά 0.2 δευτερόλεπτα).

2) Υπολογίζεται η συνολική επιτάχυνση ως η ρίζα του αθροίσματος των τετραγώνων των τριών συνιστωσών.

3) Στη συνέχεια ανιχνεύεται η περίπτωση της ελεύθερης πτώσης. Η ανίχνευση γίνεται μέσω της σύγκρισης της τιμής της επιτάχυνσης της συσκευής με ένα συγκεκριμένο threshold (κατώφλι) το οποίο βρίσκεται κοντά στο μηδέν (διότι στην ελεύθερη πτώση η δύναμη της βαρύτητας εξουδετερώνεται και η συνολική επιτάχυνση της συσκευής αναμένεται να είναι κοντά στο μηδέν). Εάν ανιχνευτεί ελεύθερη πτώση, τότε θεωρούμε πως ο χρήστης υπέστη πτώση και ενεργοποιούνται οι ειδοποιήσεις.

Προγραμματιστικά, τα παραπάνω υλοποιούνται στις ακόλουθες γραμμές κώδικα (ο οποίος περιέχεται μέσα στη μέθοδο onSensorChanged, όπου πραγματοποιούνται οι έλεγχοι κάθε φορά που το επιταχυνσιόμετρο τροφοδοτεί την εφαρμογή με νέα δεδομένα): Αρχικά, ανακτώνται οι επιταχύνσεις στους τρεις άξονες:

```
float x = event.values[0];
float y = event.values[1];
float z = event.values[2];
```

Ακολούθως υπολογίζεται η συνιστώσα επιτάχυνση (με τη βοήθεια της κλάσης Math που περιέχει μαθηματικούς υπολογισμούς):

```
float gvec=Math.round(Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2)+Math.pow(z,
2)));
```

Μετά, μέσω των ελέγχων που προαναφέρθηκαν, ελέγχεται εάν η συνιστώσα επιτάχυνση έχει ξεπεράσει το κατώφλι (η σωστή τιμή του οποίου δεν είναι απόλυτη, μπορεί να διαφέρει λίγο από συσκευή σε συσκευή).

```
if (gvec<=6.0) {
            min=true;//free falls
             }
```

Σε περίπτωση, τέλος, που τα min και max πάρουν την τιμή true, τότε ανιχνεύεται η πτώση του χρήστη και όπως και στο σενάριο της ενότητας 8.7 ενεργοποιούνται οι προβλεπόμενες ειδοποιήσεις.

Είναι σημαντικό να τονιστεί πως η εμφάνιση ενός παραθύρου διαλόγου επιβεβαίωσης πτώσης (για να αποφευχθεί λανθασμένος συναγερμός) **δεν** έγινε εφικτή· αυτό συνέβη για δύο λόγους: Πρώτον εκ των πραγμάτων αυτό κατέστη αδύνατο καθώς μετά από έρευνα αποδείχθηκε πως είναι αδύνατον να κληθεί παράθυρο ειδοποίησης μέσα από υπηρεσία, και αυτό γιατί το παράθυρο απαιτεί αντιστοίχιση με γραφικό περιβάλλον, κάτι που δε μπορεί να συμβεί μέσω της υπηρεσίας. Δεύτερον, μέσα από αλληλεπίδραση με άλλους προγραμματιστές, κατέστη σαφές πως η διακοπή τη χρήσης της συσκευής με αναδυόμενα παράθυρα αποτελεί, όπως συνηθίζεται νε λέγεται, “bad programming” (κακή πρακτική προγραμματισμού). Επομένως, η ιδέα αυτή εγκαταλείφθηκε και η ειδοποίηση στέλνεται απευθείας με το που ανιχνευθεί πτώση.

8.8 Η Υπηρεσία Loss Detection

Η τελευταία υπηρεσία που υλοποιείται στην εφαρμογή είναι η ανίχνευση απώλειας προσανατολισμού (Loss Detection). Αυτό ουσιαστικά είναι το πιο εξεζητημένο κομμάτι της εφαρμογής, γιατί είναι δύσκολο να ερμηνευτεί προγραμματιστικά η πιθανότητα ο χρήστης να «χαθεί». Ρητή και αποτελεσματική μεθοδολογία δεν υπάρχει, όπως επίσης και παρόμοια εφαρμογή για να εξαχθούν πληροφορίες. Ακολούθως, θα προταθεί ένας αλγόριθμος ανίχνευσης που θα προσπαθήσει να παραθέσει μια λογική διαδικασία για την ανίχνευση της απώλειας προσανατολισμού του χρήστη, χωρίς βέβαια εγγυημένα αποτελέσματα.

Η ανίχνευση απώλειας προσανατολισμού στηρίζεται στη λογική ότι ο χρήστης, σε περίπτωση που χαθεί, αρχίζει να πανικοβάλλεται και να κάνει νευρικές περιστροφικές κινήσεις προσπαθώντας να καταλάβει που βρίσκεται και να βρει το δρόμο για το σπίτι. Σκοπός είναι να εντοπιστούν αυτές οι διαδοχικές αλλαγές κατεύθυνσης μέσω της συσκευής και να ερμηνευτούν ως φαινόμενο απώλειας προσανατολισμού.

Η ιδέα έχει ως εξής: θέτονται τρεις Boolean μεταβλητές flag1, flag2, flag3 και ένας μετρητής i. Η κάθε μια από τις μεταβλητές γίνεται αληθής για τρεις ξεχωριστές κατευθύνσεις στο τρισδιάστατο σύστημα αξόνων (δηλαδή συνδυασμός θετικών και αρνητικών μετρήσεων του επιταχυνσιόμετρου στους άξονες x, y, z). Κάθε φορά που μια μεταβλητή γίνεται αληθής ο μετρητής αυξάνεται. Όταν ο μετρητής φτάσει στο 10 (έχει πάρει δείγμα της κατεύθυνσης 10 φορές) ελέγχονται οι τιμές των μεταβλητών: εάν **και οι τρεις** έχουν αληθή τιμή, αυτό σημαίνει πως μέσα σε αυτό το διάστημα της δειγματοληψίας έχει αλλάξει τρεις διαφορετικές κατευθύνσεις· επομένως, συμπεραίνεται πως ο χρήστης χάθηκε και ενεργοποιούνται οι ειδοποιήσεις. Εάν ο μετρητής υπερβεί το νούμερο δέκα χωρίς και οι τρεις να είναι αληθής, μηδενίζεται. Παρακάτω παρατίθεται το κομμάτι του κώδικα που υλοποιεί την παραπάνω λογική:

```
if (y>0 && x>0) {
                    flag1=true;
                    i++;
              }
if (y<0 && x>0) {
                    flag2=true;
                    i++;
              }

if(y>0 && x<0){
                    flag3=true;
                    i++;
              }

if (flag1==true && flag2==true && flag3==true && i<=10){
Toast.makeText(LossService.this, "LOSS DETECTED!!!!",
Toast.LENGTH_LONG).show();

              }

if (i>10)  {
                    i=0;
              }
```

Ο παραπάνω κώδικας δεν κατέστη δυνατό να αξιολογηθεί σωστά για την αποτελεσματικότητά του, λόγω της αδυναμίας ελέγχου της σε πραγματικό περιβάλλον και για μεγάλο χρονικό διάστημα. Η αλγοριθμική ιδέα, πάντως, μπορεί να αποτελέσει μια βάση για μελλοντική επέκταση της υπηρεσίας και ελέγχου της σε πραγματικό περιβάλλον ώστε να καταλήξει σε πλήρη υλοποίηση.

8.9 Διεπαφές Χρήστη (XML)

Η καθεμία από τις παραπάνω δραστηριότητες που υλοποιήθηκαν κατά τη διάρκεια της εφαρμογής, «συνοδεύονται» και από ένα αρχείο περιγραφής διεπαφής χρήστη xml. Μέσα σε αυτά, όπως προαναφέρθηκε, υλοποιούνται όλα τα στοιχεία ελέγχου της οθόνης του κινητού: κουμπιά, λίστες, μενού, πλαίσια κειμένου κλπ. Μέσω πολλών διαφορετικών ετικετών, υπάρχει η δυνατότητα για πλήρη παραμετροποίηση τους ανάλογα με τις απαιτήσεις του προγραμματιστή και τις ανάγκες του χρήστη. Παρακάτω παρατίθεται ένα παράδειγμα δημιουργίας του κουμπιού Setup της δραστηριότητας SetupActivity.java:

```
<Button
android:id="@+id/button5"
android:layout_width="200dp"
android:layout_height="wrap_content"
android:layout_below="@+id/textView2"
android:layout_centerHorizontal="true"
android:layout_marginTop="53dp"
android:gravity="center"
android:text="@string/ButtonSetup" />
```

Τονίζεται ότι οι διάφοροι πόροι της εφαρμογής (όπως χρώματα, συμβολοσειρές, διαστάσεις κλπ.) καταχωρούνται και αυτοί μέσω της περιγραφής xml. Με αυτόν τον τρόπο, οι πόροι οργανώνονται ανά τύπο, είναι εγγυημένα μοναδικοί και ο κώδικας καθίσταται σαφέστερος και πιο ευανάγνωστος.

8.10 Αρχείο Δήλωσης (Android Manifest)

Κάθε εφαρμογή Android περιλαμβάνει ένα ειδικό αρχείο, το αρχείο δήλωσης (AndroidManifest.xml). Μέσα σε αυτό το αρχείο καθορίζονται οι ρυθμίσεις διαμόρφωσης τη εφαρμογής, που περιλαμβάνουν την ταυτότητα της εφαρμογής και τις άδειες που απαιτεί η εφαρμογή για να εκτελεστεί. Ουσιαστικά το αρχείο αυτό περιγράφει τι είναι η εφαρμογή, ποια είναι τα συστατικά της και τι άδειες ζητάει από τη συσκευή του χρήστη ώστε να εκτελεστεί. Πιο συγκεκριμένα, το σύστημα Android χρησιμοποιεί τις πληροφορίες αυτού του αρχείου ώστε να κάνει τα παρακάτω:

1. Εγκατάσταση και ενημέρωση του πακέτου της εφαρμογής
2. Εμφάνιση λεπτομερειών εφαρμογής σε χρήστες (τίτλος, εικονίδιο κλπ.)
3. Δήλωση και εκκίνηση δραστηριοτήτων και υπηρεσιών εφαρμογής
4. Διαχείριση αδειών εφαρμογής

9. Μελλοντικές Επεκτάσεις – Συμπεράσματα

Η εφαρμογή της παρούσας διπλωματικής δεν αποτελεί ένα συνηθισμένο πρότυπο εφαρμογής σε κινητό. Δεν πρόκειται για εφαρμογή ψυχαγωγίας ή ευρείας χρήσης, η οποία απαιτεί τη συνεχή αλληλεπίδραση με τον χρήστη. Πρόκειται για μια εφαρμογή υγείας που απευθύνεται σε ανειδίκευτους με την τεχνολογία και τα smartphone χρήστες, η οποία έχει ως στόχο να περιορίσει όσο το δυνατόν περισσότερο τον χρήστη σε παθητικό ρόλο. Επίσης, ακριβώς επειδή πρόκειται για εφαρμογή υγείας (τα σενάρια που υλοποιεί έχουν άμεση σχέση με την πρόληψη επιβλαβών καταστάσεων αλλά και την λήψη μέτρων σε περίπτωση ατυχημάτων), αποτελεί μείζον θέμα η πλήρης λειτουργικότητα και αποτελεσματικότητά της προτού διατεθεί προς χρήση σε ρεαλιστικά σενάρια.

Εν προκειμένω, η εφαρμογή αναπτύχθηκε αρχικά για εκπαιδευτικούς-ακαδημαϊκούς σκοπούς. Προσπαθεί να επιδείξει μέσα από τις λειτουργίες της πώς ένα σύνολο αρκετά σύγχρονων και εξειδικευμένων λειτουργιών μπορεί να χρησιμοποιηθεί για τους συγκεκριμένους σκοπούς· η αξιοποίησή της στην πράξη απαιτεί και κάποιες επεκτάσεις.

Είναι σημαντικό να τονιστεί πως μια σωστότερη διαδικασία ανάπτυξης θα απαιτούσε τη συνεχή αλληλεπίδραση με πιθανούς χρήστες της εφαρμογής καθ’ όλη τη διάρκεια της ανάπτυξής της (user-centered design), διότι με αυτόν τον τρόπο θα υπήρχε μια καλύτερη εικόνα όσον αφορά τις προδιαγραφές και τις απαιτήσεις των χρηστών που ανήκουν στο target group της εφαρμογής. Όμως, κάτι τέτοιο δεν κατέστη δυνατό (υπήρξαν μόνο μεμονωμένα δείγματα γραφής), επομένως κάποια συμπεράσματα για τη συμπεριφορά και την αλληλεπίδραση των χρηστών με την εφαρμογή εξήχθησαν προσεγγιστικά.

Θα πρέπει λοιπόν, **μελλοντικά**, να πραγματοποιηθεί μια αξιολόγηση της εφαρμογής από τους ίδιους τους χρήστες, ώστε να πραγματοποιηθεί μια βελτιστοποίηση όσον αφορά τη λειτουργικότητα και την συνέπειά της προς το σύνολο των χρηστών. Αυτή η αξιολόγηση μπορεί να επιτευχθεί ως εξής: δημιουργούνται από τον προγραμματιστή διάφορες εκδόσεις διεπαφής χρήστη (user interface) και παρουσιάζονται στους χρήστες της εφαρμογής. Αυτοί, με τη σειρά τους, χρησιμοποιούν και επεξεργάζονται τις διάφορες εκδόσεις της διεπαφής, αξιολογώντας κατάλληλα την καθεμία ανάλογα με το πόσο ευνόητη, οικεία και αποτελεσματική τους φαίνεται. Επομένως, ο προγραμματιστής επιλέγει να υιοθετήσει στην εφαρμογή την διεπαφή με την θετικότερη ανάδραση, όντας σίγουρος για την αποτελεσματικότητα και το αντίκτυπο που θα έχει στους χρήστες.

Επίσης, αποτελεί προτεραιότητα η πλήρης υλοποίηση του σεναρίου Loss Detection (Ανίχνευση Απώλειας Προσανατολισμού) μέσα από δοκιμές σε πραγματικούς χρήστες και η βελτιστοποίηση του αλγορίθμου. Επίσης, στόχος είναι η υλοποίηση των Google maps στην εφαρμογή για τη διευκόλυνση (ορισμός αρχικής τοποθεσίας μέσω χάρτη) και την επέκταση της (παροχή διαδρομής προς το σπίτι σε περίπτωση που ο χρήστης χαθεί). Τέλος, σχεδιάζεται να προστεθεί η δυνατότητα να αποτρέπονται λανθασμένοι συναγερμοί στην

περίπτωση της ανίχνευσης πτώσης, ίσως με τη χρήση ενός παραμετροποιημένου notification.

Τέλος, μια άλλη πιθανή επέκταση είναι η σύνδεση της οντολογίας με την εφαρμογή σε προγραμματιστικό επίπεδο. Αυτή η σύνδεση συνίσταται στο να τίθενται ερωτήματα στην οντολογία, ώστε να γίνεται δυναμική κτήση δεδομένων διαφόρων υπηρεσιών. Η δυνατότητα αυτή δίνεται από το Owl API [16] (προγραμματιστική διεπαφή) το οποίο δίνει τη δυνατότητα για επεξεργασία και διαχείριση της οντολογίας μέσω προγραμματιστικών διαδικασιών.

Περιορισμοί

Θα πρέπει αν μη τι άλλο να τονίσουμε πως η παροχή των υπηρεσιών της εφαρμογής συνοδεύεται από κάποιους περιορισμούς,. Αυτοί είναι:

- Η συσκευή πρέπει να διαθέτει απαραίτητα GPS (για την υλοποίηση της πρώτης υπηρεσίας) και επιταχυνσιόμετρο (για την υλοποίηση των άλλων δύο υπηρεσιών).

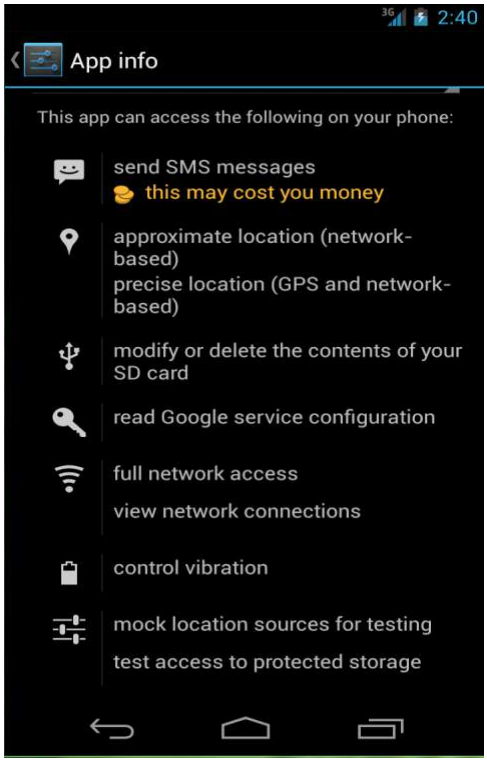
- Το GPS διαθέτει ακρίβεια 10-15 μέτρων (και στις περιπτώσεις των ενσωματωμένων GPS στα τηλέφωνα μπορεί να είναι και παραπάνω, λόγω χαμηλής ποιότητας του δέκτη), επομένως για πολύ μικρές αποστάσεις θα υπάρχουν σημαντικά σφάλματα στον εντοπισμό της θέσης χρήστη.

- Το GPS δεν λειτουργεί σωστά σε κλειστούς χώρους, επομένως η επίβλεψη ασθενούς μέσα στο σπίτι καθίσταται αδύνατη (χρησιμοποιούνται άλλες τεχνικές εντοπισμού θέσης, όπως για παράδειγμα μέσω Bluetooth).

Πάντως, στη σύγχρονη παγκόσμια αγορά smartphones η πλειοψηφία των συσκευών διαθέτουν GPS και επιταχυνσιόμετρα, κάτι που σημαίνει ότι η εφαρμογή δεν απαιτεί κάποιον εξεζητημένο εξοπλισμό για να υλοποιήσει τις υπηρεσίες της· ένα απλό smartphone θα κάνει σωστά τη δουλειά του.

Άδειες /Ιδιωτικότητα

Κλείνοντας, είναι σημαντικό να τεθεί προς συζήτηση το θέμα των αδειών που απαιτεί η εφαρμογή. Η Εικόνα 40 παραθέτει το σύνολο των αδειών που απαιτεί από τον χρήστη προκειμένου να εγκατασταθεί στο κινητό.



Εικόνα 40: Οθόνη αδειών (permissions page).

Παρατηρούμε λοιπόν, πως η εφαρμογή ζητά την άδεια (μεταξύ άλλων) να έχει πρόσβαση στην τοποθεσία του χρήστη και να αποστέλλει μηνύματα (υπηρεσία επί πληρωμή). Κατά πόσο όμως έχει το δικαίωμα η εφαρμογή να γνωρίζει ανά πάσα στιγμή το που βρίσκεται ο χρήστης; Γίνεται αντιληπτό λοιπόν πως εγείρονται ζητήματα παραβίασης της ιδιωτικότητας του χρήστη. Επίσης, είναι ευαίσθητο σημείο το γεγονός της αυτόματης αποστολής μηνύματος σε κινητό, καθώς μπορεί να ενοχληθεί ο χρήστης από την «πρωτοβουλία» της εφαρμογής να τον χρεώσει επιπλέον στο λογαριασμό που θα πληρώσει.

Βιβλιογραφία

[1] <http://en.wikipedia.org/wiki/File:Galaxy_nexus.jpg>(10/6/2013)

[2]Mann, Steve (2012): Wearable Computing. In: Soegaard, Mads and Dam, Rikke Friis (eds.). "Encyclopedia of Human-Computer Interaction". Aarhus, Denmark: The Interaction-Design.orgFoundation. <http://www.interactiondesign.org/encyclopedia/wearable_computing.html> (29/6/2013)

[3]<http://web.nmsu.edu/~kburke/Instrumentation/NMSU_PDA_Protocol.html> (10/6/2013)

[4]<http://androidhellas.com/wp-content/uploads/2012/08/Sony-Xperia-Tablet-S.jpg> (10/6/2013).

[5]A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness", Proc. CHI'00, The Netherlands, 2000.

[6]C. Anagnostopoulos, A. Tsounis, S. Hadjiefthymiades, "Context Awareness in Mobile Computing Environments", Wireless Personal Communications, vol. 42, pp.445-464, 2006.

[7]Matthias Baldauf, Schahram Dustdar* and Florian Rosenberg, "A Survey on Context-Aware Systems", International Journal of Ad Hoc and Ubiquitous Computing, Information Systems Institute, Vienna University of Technology, 2011.

[8]Panu Korpipää, Jani Mäntyjärvi, Juha Kela, Heikki Keränen, and Esko-Juhani Malm, "Managing Context Information in Mobile Devices", VTT Technical Research Centre of Finland, 2003.

[9] Matthias Strobbe, Olivier Van Laere, Femke Ongenae, Samuel Dauwe, Bart Dhoedt, Filip De Turck, Piet Demeester and Kris Luyten, "Integrating Location and Context Information for 6Novel Personalised Applications", Ghent University – IBBT, Department of Information Technology, 2011.

[10]Δόμνα Ζελένη, Παναγιώτα Σικλαφίδου, Μίλτος Λειβαδίτης, " Άνοια: Ψυχιατροδικαστικά ζητήματα", Εγκέφαλος 49, 47-50, 2012. <http://www.encephalos.gr/pdf/49-2-01g.pdf> (15/6/2013).

[11]<http://protege.stanford.edu/> (5/4/2012).

[12]Global Positioning System Βασικές Αρχές: Χριστόφορος Κουνιάκης <http://www.eosathinon.gr/GPS.pdf> (21/6/2013).

[13]<http://developer.android.com/reference/android/hardware/SensorEvent.html> (19/5/2013).

[14]: <http://stackoverflow.com/questions/837872/calculate-distance-in-meters-when-you-know-longitude-and-latitude-in-java?rq=1> (10/5/2013).

[15]:<http://developer.android.com/reference/android/hardware/SensorEvent.html> (22/5/2013).

[16]: <http://owlapi.sourceforge.net/documentation.html> (12/7/2012)

Παράρτημα: Κώδικας Android

SetupActivity.java

```
package com.example.myapp;

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.SharedPreferences;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;


public class SetupActivity extends Activity {

public static final String PREFS="Preferences";

public static final String PREFS_NAME="Name";

static final int AUTH_DIALOG_ID=1;

SharedPreferences Settings;


       protected void onCreate(Bundle savedInstanceState) {

         super.onCreate(savedInstanceState);

         setContentView(R.layout.setup);

         Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);
```

```
        exitapp();

        goAdvanced();

    }


       //dhmiourgia tou Authentication Dialog

       @Override

       protected Dialog onCreateDialog(int id) {

              switch(id){

              case AUTH_DIALOG_ID:

              LayoutInflater inflater=(LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);

              final View
layout1=inflater.inflate(R.layout.authentication_dialog,(ViewGroup)findView
ById(R.id.root));

              AlertDialog.Builder builder=new AlertDialog.Builder(this);

              builder.setView(layout1);

              builder.setTitle(R.string.Authentication);

              builder.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

                     @Override

                     public void onClick(DialogInterface arg0, int arg1) {

                            final String USERNAME="user";

                            final String PASS="123";

                            EditText user_input=(EditText)
layout1.findViewById(R.id.editTextName);

                            EditText pwd_input=(EditText)
layout1.findViewById(R.id.editTextPass);

                            TextView error=(TextView)
layout1.findViewById(R.id.errormsg);

                            String
struser=user_input.getText().toString();//APOTHIKEUO USERNAME

                            String
strpass=pwd_input.getText().toString();//APOTHIKEUO PASSWORD
```

```
        if(USERNAME.equals(struser)&&PASS.equals(strpass)){
                                    finish();
                Intent intent=new
Intent(SetupActivity.this,AdvancedActivity.class);
                startActivity(intent);
                SetupActivity.this.removeDialog(AUTH_DIALOG_ID);}
                else {
                error.setText("Wrong username or password!");
                            }
                      }
                });
                builder.setNegativeButton("Cancel",new
DialogInterface.OnClickListener() {
                @Override
                      public void onClick(DialogInterface arg0, int arg1) {
                            SetupActivity.this.removeDialog(AUTH_DIALOG_ID);
                      }
                });
                AlertDialog authenticationdialog=builder.create();
                authenticationdialog.show();
                return authenticationdialog;
                }
        return null;
        }
           //Koumpi Exit
           private void exitapp() {
                Button button4=(Button)findViewById(R.id.button4);
                button4.setOnClickListener(new Button.OnClickListener() {
                public void onClick(View v) {
```

```
            finish();
            }
            });
         }
        //Koumpi Setup
        public void goAdvanced(){
               Button button5=(Button)findViewById(R.id.button5);
               button5.setOnClickListener(new Button.OnClickListener(){
               @SuppressWarnings("deprecation")
            public void onClick(View v) {
               showDialog(AUTH_DIALOG_ID)  ;
               }
               });
        }
}
```

AdvancedActivity.java

```
package com.example.myapp;
import android.R.layout;
import android.os.Bundle;
import android.app.Activity;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

public class AdvancedActivity extends Activity {
       public static final String PREFS="Preferences";
       public static String TOGGLE_RANGE="range";
       public static String TOGGLE_FALL="fall";
       public static String TOGGLE_LOSS="loss";
       SharedPreferences Settings;
       private ToggleButton tgrange;
       private ToggleButton tgfall;
       private ToggleButton tgloss;
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.advanced);
        Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);
      goRange();
      goFall();
      goLoss();
      canceladvanced();
    }

       private void goRange() {
```

```
 Button buttongorange=(Button)findViewById(R.id.buttonrange);
buttongorange.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v) {
finish();
Intent intent=new Intent(AdvancedActivity.this,RangeAdvanced.class);
startActivity(intent);
}
});
}
   private void goFall() {
 Button buttongofall=(Button)findViewById(R.id.buttonfall);
buttongofall.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v) {
finish();
Intent intent=new Intent(AdvancedActivity.this,FallAdvanced.class);
startActivity(intent);
}});
}
   private void goLoss() {
 Button buttongoloss=(Button)findViewById(R.id.buttonloss);
buttongoloss.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v) {
finish();
Intent intent=new Intent(AdvancedActivity.this,LossAdvanced.class);
startActivity(intent);
}
});
}
   private void canceladvanced() {
```

```
        Button buttongomain=(Button)findViewById(R.id.buttoncncadv);
       buttongomain.setOnClickListener(new Button.OnClickListener() {
       public void onClick(View v) {
       finish();
       Intent intent=new Intent(AdvancedActivity.this,SetupActivity.class);
       startActivity(intent);
       }
       });
    }
       public void onToggleRangeClicked(View view) {
           boolean on = ((ToggleButton) view).isChecked();
           if (on) {
              startService(new Intent(this, RangeService.class));
              Editor editor=Settings.edit();
                     editor.putBoolean(TOGGLE_RANGE, true);
                     editor.commit();
           } else {
              stopService(new
Intent(AdvancedActivity.this,RangeService.class));
              Editor editor=Settings.edit();
                     editor.putBoolean(TOGGLE_RANGE, false);
                     editor.commit();
           }
       }
       public void onToggleFallClicked(View view) {
           boolean on = ((ToggleButton) view).isChecked();
           if (on) {
              startService(new Intent(this, FallService.class));
              Editor editor=Settings.edit();
                     editor.putBoolean(TOGGLE_FALL, true);
```

```
                editor.commit();
            } else {
               stopService(new
Intent(AdvancedActivity.this,FallService.class));
               Editor editor=Settings.edit();
                     editor.putBoolean(TOGGLE_FALL, false);
                     editor.commit();
            }
        }
        public void onToggleLossClicked(View view) {
            boolean on = ((ToggleButton) view).isChecked();
            if (on) {
               startService(new Intent(this, LossService.class));
               Editor editor=Settings.edit();
                     editor.putBoolean(TOGGLE_LOSS, true);
                     editor.commit();
            } else {
               stopService(new
Intent(AdvancedActivity.this,LossService.class));
               Editor editor=Settings.edit();
                     editor.putBoolean(TOGGLE_LOSS, false);
                     editor.commit();
            }
        }
        public void onResume(){
               super.onResume();
               tgrange = (ToggleButton) findViewById(R.id.toggleRange);
               tgfall = (ToggleButton) findViewById(R.id.toggleFall);
               tgloss = (ToggleButton) findViewById(R.id.toggleLoss);
               Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);
```

```
            boolean tgprefrange = Settings.getBoolean(TOGGLE_RANGE,
false);
            boolean tgpreffall = Settings.getBoolean(TOGGLE_FALL, false);
            boolean tgprefloss = Settings.getBoolean(TOGGLE_LOSS, false);
            if (tgprefrange){
                  tgrange.setChecked(true);}
                  else {
                        tgrange.setChecked(false);
                  }
            if (tgpreffall){
                  tgfall.setChecked(true);}
                  else {
                        tgfall.setChecked(false);
                  }
            if (tgprefloss){
                  tgloss.setChecked(true);}
                  else {
                        tgloss.setChecked(false);
                  }
      }
}
```

RangeAdvanced.java

```
package com.example.myapp;
import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
```

```
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class RangeAdvanced extends Activity  {
       public static final String PREFS="Preferences";
       public static final String PREFS_ALLOW_RANGE="AllowedRange";
       public static final String PHONERANGE1="phone1";
       public static final String NAMERANGE1="name1";
       public static final String PHONERANGE2="phone2";
       public static final String NAMERANGE2="name2";
       SharedPreferences Settings;
       private static final String tag1="rangesaved";
       static final int add_affil_range_1=2;
       static final int add_affil_range_2=3;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.range);

        Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);

    cancelrange();

    saveRange();

    sethomelocation();

    AddAffil1();

    AddAffil2();

    }

    protected Dialog onCreateDialog(int id) {

              switch(id){

              //1o dialog box

              case add_affil_range_1:

              LayoutInflater inflater1=(LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);

              final View
layoutrange=inflater1.inflate(R.layout.affiliations_dialog,(ViewGroup)findV
iewById(R.id.rootrange));

              AlertDialog.Builder builderrange=new
AlertDialog.Builder(this);

              builderrange.setView(layoutrange);

              builderrange.setTitle(R.string.addaffil);

              builderrange.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

                    @Override

                    public void onClick(DialogInterface arg0, int arg1) {

                          EditText newname1=(EditText)
layoutrange.findViewById(R.id.Afillname);

                          EditText newphone1=(EditText)
layoutrange.findViewById(R.id.Afillphone);

                          String
strname1=newname1.getText().toString();//APOTHIKEUO NAME

                          String
strphone1=newphone1.getText().toString();//APOTHIKEUO PHONE
```

```
                        TextView
showaffilrange1=(TextView)findViewById(R.id.rangeaffiliate1);

                        showaffilrange1.setText(strname1);

                        Editor editor=Settings.edit();

                           editor.putString(PHONERANGE1,strphone1);

                           editor.putString(NAMERANGE1, strname1);

                           editor.commit();

      RangeAdvanced.this.removeDialog(add_affil_range_1);

                  }

            });

            builderrange.setNegativeButton("Cancel",new
DialogInterface.OnClickListener() {

                  @Override

                  public void onClick(DialogInterface arg0, int arg1) {

                        //nothing

                  }

            });

            AlertDialog affiliationdialog=builderrange.create();

            affiliationdialog.show();

            return affiliationdialog;


            //2o dialog box

            case add_affil_range_2:

                  LayoutInflater inflater2=(LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);

                  final View
layoutrange2=inflater2.inflate(R.layout.affiliations_dialog,(ViewGroup)find
ViewById(R.id.rootrange));

                  AlertDialog.Builder builderrange1=new
AlertDialog.Builder(this);

                  builderrange1.setView(layoutrange2);

                  builderrange1.setTitle(R.string.addaffil);
```

```
                    builderrange1.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

                          @Override

                    public void onClick(DialogInterface arg0, int arg1) {

      EditText newname2=(EditText)
layoutrange2.findViewById(R.id.Afillname);

                                EditText newphone2=(EditText)
layoutrange2.findViewById(R.id.Afillphone);

                                String
strname2=newname2.getText().toString();//APOTHIKEUO NAME

                                String
strphone2=newphone2.getText().toString();//APOTHIKEUO PHONE

                                TextView
showaffilrange2=(TextView)findViewById(R.id.rangeaffiliate2);

                                showaffilrange2.setText(strname2);

                                Editor editor=Settings.edit();

                                  editor.putString(PHONERANGE2,strphone2);

                                   editor.putString(NAMERANGE2, strname2);

                                   editor.commit();

RangeAdvanced.this.removeDialog(add_affil_range_2);

                          }

                    });

                    builderrange1.setNegativeButton("Cancel",new
DialogInterface.OnClickListener() {

                          @Override

                          public void onClick(DialogInterface arg0, int
arg1) {

                                //nothing

                          }

                    });

                    AlertDialog affiliationdialog1=builderrange1.create();

                    affiliationdialog1.show();
```

```
                return affiliationdialog1;

            }

            return null;

      }

         private void cancelrange() {

              Button button8=(Button)findViewById(R.id.buttonbackrange);

            button8.setOnClickListener(new Button.OnClickListener() {

            public void onClick(View v) {

                  finish();

            Intent intent=new
Intent(RangeAdvanced.this,AdvancedActivity.class);

            startActivity(intent);

            }

            });

          }

         private void saveRange() {

                Button
btnsaverange=(Button)findViewById(R.id.buttonsaverange);

                btnsaverange.setOnClickListener(new
Button.OnClickListener(){

                public void onClick(View v) {

                final EditText
editTextrange=(EditText)findViewById(R.id.setrange);

                String allowrange=editTextrange.getText().toString();

                if(allowrange.trim().length()>0){

                Editor editor=Settings.edit();

                editor.putString(PREFS_ALLOW_RANGE, allowrange);

                editor.commit();

                /*Log.d(tag1, allowrange);*/

                    double range1=Double.parseDouble(allowrange);

                    Log.d(tag1, "Range is now"+range1);
```

```
                    Toast.makeText(RangeAdvanced.this, "Changes Saved",
Toast.LENGTH_SHORT).show();}

                }

                });

        }

        private void sethomelocation() {

             Button buttonmp=(Button)findViewById(R.id.buttonmap);

            buttonmp.setOnClickListener(new Button.OnClickListener() {

            public void onClick(View v) {

            Intent intent=new Intent(RangeAdvanced.this,GetMap.class);

            startActivity(intent);

            }

            });

         }

        private void AddAffil1(){

                Button buttonaar1=(Button)findViewById(R.id.btnaddrange1);

                buttonaar1.setOnClickListener(new Button.OnClickListener(){

                @SuppressWarnings("deprecation")

            public void onClick(View v) {

                showDialog(add_affil_range_1)  ;

                }

                });

        }

        private void AddAffil2(){

                Button buttonaar2=(Button)findViewById(R.id.btnaddrange2);

                buttonaar2.setOnClickListener(new Button.OnClickListener(){

                @SuppressWarnings("deprecation")

            public void onClick(View v) {

                showDialog(add_affil_range_2)  ;

                }
```

```
                });

        }

        public void onResume(){

                super.onResume();

                Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);

                TextView
showaffilrange1=(TextView)findViewById(R.id.rangeaffiliate1);

                showaffilrange1.setText(Settings.getString(NAMERANGE1,
""));

                TextView
showaffilrange2=(TextView)findViewById(R.id.rangeaffiliate2);

                showaffilrange2.setText(Settings.getString(NAMERANGE2,
""));}

}
```

RangeService.java

```
package com.example.myapp;

import android.app.Service;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.net.Uri;

import android.os.Bundle;

import android.os.IBinder;

import android.os.Vibrator;

import android.telephony.SmsManager;

import android.util.Log;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;
```

```
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.widget.Toast;
import java.lang.Double;
import java.lang.Math;
public class RangeService extends Service {
	private static final String Tag="RANGESERVICE";
	private static final long time_interval = 6000; // in Milliseconds
	protected LocationManager lm;
	public static final String PREFS="Preferences";
	public static final String PREFS_ALLOW_RANGE = "AllowedRange";
	public static final String PHONERANGE1="phone1";
	public static final String NAMERANGE1="name1";
	public static final String PHONERANGE2="phone2";
	public static final String NAMERANGE2="name2";
	public static final String PREFS_HLAT="hlatitude";
	public static final String PREFS_HLON="hlongitude";
	SharedPreferences Settings;
	@Override
	public IBinder onBind(Intent arg0) {
		// TODO Auto-generated method stub
		return null;
	}
	@Override
	public void onCreate() {
		Log.d(Tag,"onCreate");
		 Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);
	}
	//arxikopoihseis
```

```
	public void onStart(Intent intent, int start){

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

		LocationListener ll=new MyLocationListener();

	lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,time_interval
,0,ll);

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

		if(location!=null){

		String msg = String.format("CurrentLocation \n Longitude: %1$s
\n Latitude: %2$s",location.getLongitude(),

		location.getLatitude());

		Toast.makeText(RangeService.this,
msg,Toast.LENGTH_LONG).show();

		}

	}

	private class MyLocationListener implements LocationListener{

		@Override

		public void onLocationChanged(Location location) {

		Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);

		String locrange=Settings.getString(PREFS_ALLOW_RANGE, "");

		String shlat=Settings.getString(PREFS_HLAT, "");

		String shlon=Settings.getString(PREFS_HLON, "");

		double lat=location.getLatitude();

		double lon=location.getLongitude();

		double hlat=Double.parseDouble(shlat); // home location
latitude

		double hlon=Double.parseDouble(shlon); // home location
longitude

		double dist=distance(lat,hlat,lon,hlon,0.0,0.0);  //distance

		double range=Double.parseDouble(locrange); //allowed range

		if (dist>range){
```

```
                    try{

                    //sms alert

                    String smsmsg="Your patient has gone out of
range!!!Location is: https://maps.google.com/maps?q="+lat+","+lon;

                    sendSMS(Settings.getString(PHONERANGE1, ""),smsmsg);

                    sendSMS(Settings.getString(PHONERANGE2, ""),smsmsg);

                    //sound alert

                    Uri notification =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

                    Ringtone ring =
RingtoneManager.getRingtone(getApplicationContext(), notification);

                    ring.play();

                    //vibrate

                    Vibrator v = (Vibrator)
getSystemService(Context.VIBRATOR_SERVICE);

                    v.vibrate(2000);//2sec vibration

                    }catch (Exception e){

                          Toast.makeText(RangeService.this,"Invalid sms
receiver",Toast.LENGTH_SHORT).show();   }

                    Toast.makeText(RangeService.this,"YOU ARE OUT OF
RANGE!!!",Toast.LENGTH_LONG).show();

             }

             }


             private double distance(double lat1, double lat2, double lon1,
double lon2,

                      double el1, double el2) {

                final int R = 6371; // Radius of the earth

                Double latDistance = deg2rad(lat2 - lat1);

                Double lonDistance = deg2rad(lon2 - lon1);

                Double a = Math.sin(latDistance / 2) *
Math.sin(latDistance / 2)
```

```
                        + Math.cos(deg2rad(lat1)) *
Math.cos(deg2rad(lat2))
                        * Math.sin(lonDistance / 2) * Math.sin(lonDistance
/ 2);
                Double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
                double distance = R * c * 1000; // convert to meters
                double height = el1 - el2;
                distance = Math.pow(distance, 2) + Math.pow(height, 2);
                return Math.sqrt(distance);
            }
            private double deg2rad(double deg) {
                return (deg * Math.PI / 180.0);
            }
            public void sendSMS(String phoneNumber, String message)
               {
                   SmsManager sms = SmsManager.getDefault();
                   sms.sendTextMessage(phoneNumber, null, message, null,
null);}
            @Override
            public void onProviderDisabled(String arg0) {
            Toast.makeText(RangeService.this, "GPS
Disabled",Toast.LENGTH_LONG).show();
            }
            @Override
            public void onProviderEnabled(String arg0) {
            Toast.makeText(RangeService.this, "GPS
Enabled",Toast.LENGTH_LONG).show();
            }
            @Override
            public void onStatusChanged(String s, int i, Bundle b) {
            }
```

```
        }
}
```

FallService.java

```
package com.example.myapp;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.IBinder;
import android.os.Vibrator;
import android.telephony.SmsManager;
import android.util.Log;
import android.widget.Toast;
import java.lang.Math;
public class FallService extends Service implements SensorEventListener {
        static final int ID=1;
        public static final String PREFS="Preferences";
        public static final String PHONEFALL1="phone3";
        public static final String NAMEFALL1="name3";
        public static final String PHONEFALL2="phone4";
        public static final String NAMEFALL2="name4";
        SharedPreferences Settings;
```

```
	private SensorManager sm;

	private Sensor accelsensor;

	@Override

	public IBinder onBind(Intent arg0) {

		// TODO Auto-generated method stub

		return null;

	}

	@Override

	public void onCreate() {

		Toast.makeText(FallService.this, "FALL STARTED",
Toast.LENGTH_SHORT).show();

		Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);

	}

	public void onStart(Intent intent, int start) {

		sm = (SensorManager)getSystemService(SENSOR_SERVICE);

	    accelsensor = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

	    sm.registerListener(this, accelsensor,
SensorManager.SENSOR_DELAY_NORMAL);

	}

	@Override

	public void onAccuracyChanged(Sensor arg0, int arg1) {

		// TODO Auto-generated method stub

	}

	@Override

	public void onSensorChanged(SensorEvent event) {

		boolean min=false;

	boolean max=false;

	Settings=getSharedPreferences(PREFS,Context.MODE_PRIVATE);

		float x = event.values[0];

	 float y = event.values[1];
```

```
        float z = event.values[2];
        update(x, y, z);
        float gvec=Math.round(Math.sqrt(Math.pow(x, 2)
                +Math.pow(y, 2)
                +Math.pow(z, 2)));
             if (gvec<=6.0) {
            min=true;//free falls
            Toast.makeText(FallService.this,"FALL DETECTED!!!!!"
,Toast.LENGTH_LONG).show();
            String smsmsg="Your patient has suffered a fall!!!";
                    sendSMS(Settings.getString(PHONEFALL1, ""),smsmsg);
                    sendSMS(Settings.getString(PHONEFALL2, ""),smsmsg);
             Vibrator v = (Vibrator)
getSystemService(Context.VIBRATOR_SERVICE);
                    v.vibrate(2000);//2sec vibration
                    Uri notification =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
                    Ringtone ring =
RingtoneManager.getRingtone(getApplicationContext(), notification);
                    ring.play();
            }   }
      public void update(float x, float y, float z) {
          Log.v("vals", "x: " + x + " y: " + y + " z: " + z);
      }
      public void sendSMS(String phoneNumber, String message)
         {
            SmsManager sms = SmsManager.getDefault();
            sms.sendTextMessage(phoneNumber, null, message, null, null);
          }
      public void onDestroy(){
          sm.unregisterListener(this);
```

```
	}
}
```

LossService.java

```
package com.example.myapp;

import android.app.Service;

import android.content.Intent;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorListener;

import android.hardware.SensorManager;

import android.os.IBinder;

import android.widget.Toast;

public class LossService extends Service implements SensorEventListener {

	private SensorManager smloss;

	private Sensor accelsensorloss;

	@Override

	public IBinder onBind(Intent arg0) {

		return null;

	}

	public void onCreate() {

		Toast.makeText(LossService.this, "LOSS STARTED",
Toast.LENGTH_LONG).show();

	}

	public void onStart(Intent intent, int start) {

		smloss = (SensorManager)getSystemService(SENSOR_SERVICE);

	    accelsensorloss =
smloss.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

	    smloss.registerListener(this, accelsensorloss,
SensorManager.SENSOR_DELAY_NORMAL);

	}
```

```
	public void onAccuracyChanged(Sensor arg0, int arg1) {
		// TODO Auto-generated method stub
	}
	public void onSensorChanged(SensorEvent event1) {
		float x= event1.values[0];
		float y= event1.values[1];
		float z= event1.values[2];
		int i=0;
		boolean flag1=false;
		boolean flag2=false;
		boolean flag3=false;
		if (y>0 && x>0) {
			flag1=true;
			i++;
		}
		if (y<0 && x>0) {
			flag2=true;
			i++;
		}
		if(y>0 && x<0){
			flag3=true;
			i++;
		}
		if (flag1==true && flag2==true && flag3==true && i<=10){
			Toast.makeText(LossService.this, "LOSS DETECTED!!!!",
Toast.LENGTH_LONG).show();
		}
		if (i>10)  {
			i=0;
		}
```

```
        }
}
```