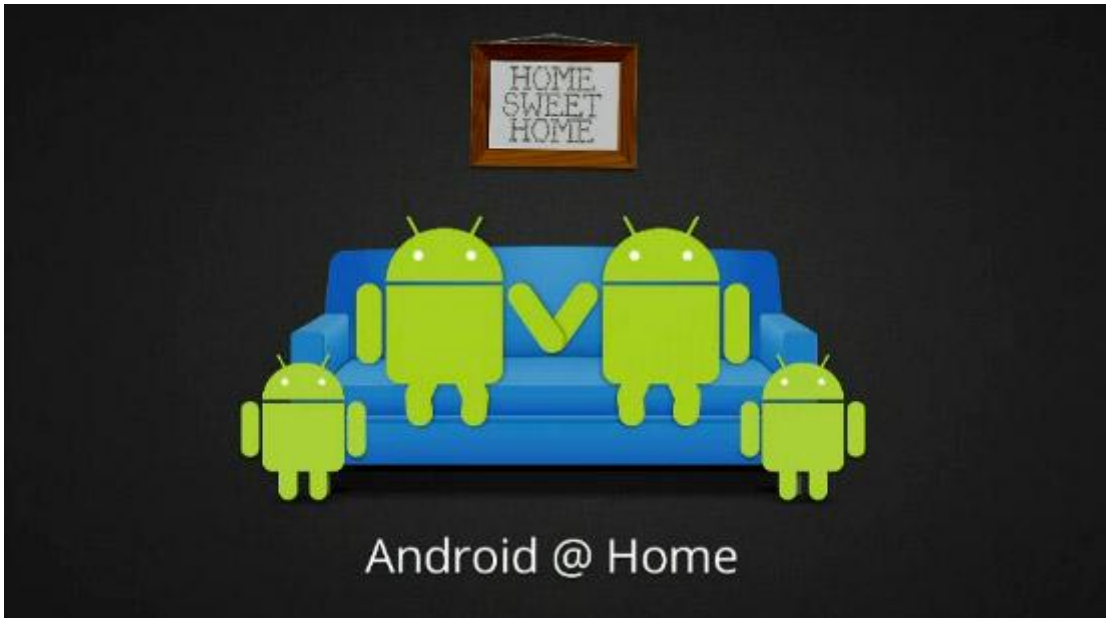
Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων
Π.Μ.Σ. " Διδακτική της Τεχνολογίας & Ψηφιακά Συστήματα "

Διπλωματική Εργασία

Σχεδίαση και Ανάπτυξη Εφαρμογής Android για Διαχείριση Έξυπνου Σπιτιού



Ζαχαριάδης Στέλιος

Α.Μ.: ΜΕ12063

ΠΕΙΡΑΙΑΣ 2014

Ευχαριστίες

Παρόλο που το ενδιαφέρον μου για τα θέματα που άπτονται των κινητών τερματικών, ξεκίνησε από τα προπτυχιακά μου χρόνια στο τμήμα Διοίκησης Τεχνολογίας του Πανεπιστημίου Μακεδονίας, δεν ήταν παρά στις μεταπτυχιακές μου σπουδές στο τμήμα Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς, που απέκτησα τη δυνατότητα να εμβαθύνω στην ανάπτυξη εφαρμογών για κινητές συσκευές, έναν κλάδο ιδιαίτερα ελκυστικό, αν αναλογιστεί κανείς τις συνεχείς εξελίξεις στο διεθνές τεχνολογικό επίπεδο.

Πριν ξεκινήσω να εκφράζω τις θερμές μου ευχαριστίες σε ορισμένους ανθρώπους, η συμβολή και η συμπαράσταση των οποίων ήταν πολύτιμη και καθοριστική στην εκπόνηση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα της πτυχιακής μου εργασίας Αναπληρωτή Καθηγητή κ. Κασκάλη Χ. Θεόδωρο. Ο συνδυασμός επιστήμης και ανθρωπιάς δυστυχώς δεν συναντάται πάντοτε και ήμουν τυχερός που δούλεψα με έναν άνθρωπο από τον οποίο πήρα πολλά περισσότερα, πέρα από την επιστημονική καθοδήγηση. Βοηθώντας με ακούραστα σε όλα τα στάδια της εργασίας μου, φρόντιζε πάντα να μου ενεργοποιεί το ενδιαφέρον και το φιλότιμο για δουλειά. Τον ευχαριστώ βαθύτατα διότι, μέσα από την πολυδιάστατη επιστημονική του σκέψη και τις εξόχως εποικοδομητικές συζητήσεις, εμπλούτισε τις γνώσεις μου και μου κέντρισε το ενδιαφέρον να ακολουθήσω μεταπτυχιακές σπουδές.

Ιδιαίτερες ευχαριστίες οφείλω, καταρχάς, στον Λέκτορα και επιβλέποντα της διπλωματικής μου εργασίας κ. Μηλιώνη Απόστολο, για την πολύτιμη βοήθεια και συμπαράστασή του. Ήταν ο άνθρωπος που με ενέπνευσε και με βοήθησε στην επιλογή του συγκεκριμένου αντικειμένου έρευνας.

Σε αυτό το σημείο θέλω να αναφέρω τους ανθρώπους, εκτός του στενού ακαδημαϊκού περιβάλλοντος, που υπήρξαν σημαντικοί πόλοι στη ζωή μου. Επιθυμώ, αρχικά, να ευχαριστήσω τους φίλους μου Σκιαδά Παναγιώτη και Κριτζιλάκη Κωνσταντίνο που, ενισχύοντάς με, διατήρησαν την ισορροπία μου ανάμεσα στον εργασιακό και κοινωνικό χώρο. Επίσης, θα ήθελα να ευχαριστήσω τον Αλέξανδρο, που τόσα χρόνια στέκεται στο πλάι μου, σαν μικρότερος Αδελφός, δίνοντας μου δύναμη και κουράγιο να συνεχίζω κάθε προσπάθεια. Βέβαια, το μεγαλύτερο ευχαριστώ το οφείλω στους γονείς μου, Αριστείδη και Παναγιώτα, των οποίων η πίστη στις δυνατότητές μου αποτέλεσε αρωγός σε όλους τους στόχους και τα όνειρά μου. Αν καταφέρω στο μέλλον να δώσω στα δικά μου παιδιά τα μισά από όσα μου προσέφεραν, θα έχω δικαιολογήσει την παρουσία μου στο έπακρο.

Περιεχόμενα

Κατάλογος Εικόνων

Περίληψη

Το «Διαδίκτυο των Πραγμάτων» (Internet of Things - IoT) είναι μια μελλοντική κατάσταση, κατά την οποία καθημερινά αντικείμενα - όπως κινητά τηλέφωνα, αυτοκίνητα, οικιακές συσκευές, ρούχα, ακόμη και τρόφιμα - θα συνδέονται ασύρματα στο διαδίκτυο, μέσω έξυπνων μικροκυκλωμάτων, και θα μπορούν να συλλέγουν και να ανταλλάσσουν δεδομένα. Εξελισσόμενες τεχνολογίες δικτύωσης, όπως η ταυτοποίηση μέσω ραδιοσυχνοτήτων (RFID tags), θα μπορούν να εφαρμοστούν σε δισεκατομμύρια αντικείμενα, για μια σχεδόν απεριόριστη γκάμα εφαρμογών.

Με τις παραπάνω προοπτικές, οι εφαρμογές που αφορούν το έξυπνο σπίτι (Smart Home) αποκτούν ιδιαίτερο ενδιαφέρον. Συσκευές με ενσωματωμένες πλακέτες, που θα μπορούν να φέρουν αισθητήρες και να συνδεθούν ασύρματο στο δίκτυο, μπορούν να επιτρέψουν την απομακρυσμένη διαχείριση και την μείωση της κατανάλωσης ενέργειας.

Τα έξυπνα συστήματα μπορούν να ελέγχουν, εκτός από τις ηλεκτρολογικές εγκαταστάσεις, τις μηχανολογικές εγκαταστάσεις αλλά και τις οικιακές συσκευές και τις συσκευές πολυμέσων (Multimedia), δημιουργώντας ένα ενοποιημένο σύστημα. Στις τελευταίες εντάσσονται οι συσκευές τηλεπικοινωνιών, τα ηχοσυστήματα αλλά και οι τηλεοράσεις του σπιτιού. Συνδυάζοντας όλες αυτές τις ανεξάρτητες, αρχικά, εγκαταστάσεις σε μία κοινή βάση αποκτάται πλήρης έλεγχος της οικίας, ο οποίος μπορεί να διεξαχθεί ακόμα και από μακριά.

Στα πλαίσια της εργασίας αυτής δημιουργήθηκε μια εφαρμογή Android για τη διαχείριση ενός έξυπνου σπιτιού. Από το "έξυπνο σπίτι" υλοποιήθηκε μέρος του Gateway, ο οποίος λειτουργεί ως διεπαφή ανάμεσα στην εφαρμογή και τις συσκευές προς διαχείριση.

1. Εισαγωγή

Το «Διαδίκτυο των Πραγμάτων» (Internet of Things - IoT) είναι μια μελλοντική κατάσταση, κατά την οποία καθημερινά αντικείμενα - όπως κινητά τηλέφωνα, αυτοκίνητα, οικιακές συσκευές, ρούχα, ακόμη και τρόφιμα - θα συνδέονται ασύρματα στο διαδίκτυο, μέσω έξυπνων μικροκυκλωμάτων, και θα μπορούν να συλλέγουν και να ανταλλάσσουν δεδομένα. Εξελισσόμενες τεχνολογίες δικτύωσης, όπως η ταυτοποίηση μέσω ραδιοσυχνοτήτων (RFID tags), θα μπορούν να εφαρμοστούν σε δισεκατομμύρια αντικείμενα, για μια σχεδόν απεριόριστη γκάμα εφαρμογών.

Συναφής έννοια του IoT είναι η έννοια του Web of Things. Στην περίπτωση αυτή, κάθε αντικείμενο είναι προσβάσιμο μέσω του πρωτοκόλλου HTTP και ταυτοποείται μοναδικά μέσω ενός URI. Αρχιτεκτονικές αρχές, όπως REpresentational State Transfer (REST) και Service Oriented Architecture (SOA), χρησιμοποιούν ως βάση το Web για τη δημιουργία υπηρεσιών λογισμικού, που χρησιμοποιούνται από εκατομμύρια ανθρώπους και επιχειρήσεις σε όλο τον κόσμο.

Με τις παραπάνω προοπτικές, οι εφαρμογές που αφορούν το έξυπνο σπίτι (Smart Home) αποκτούν ιδιαίτερο ενδιαφέρον. Συσκευές με ενσωματωμένες πλακέτες, που θα μπορούν να φέρουν αισθητήρες και να συνδεθούν ασύρματο στο δίκτυο, μπορούν να επιτρέψουν την απομακρυσμένη διαχείριση και την μείωση της κατανάλωσης ενέργειας.

Με τον όρο «έξυπνα σπίτια (Smart Home)» ή «συστήματα αυτοματισμών κατοικιών (Home Automation System)» περιγράφονται οι ηλεκτρικές εγκαταστάσεις που τοποθετούνται σε σπίτια με σκοπό να προσφέρουν άνεση, ασφάλεια, εξοικονόμηση ενέργειας και χρημάτων στους ενοίκους.

Οι έξυπνες εγκαταστάσεις αλληλεπιδρούν με το περιβάλλον χρησιμοποιώντας ένα μέσο επικοινωνίας, με τη βοήθεια του οποίου ανταλλάσσουν δεδομένα, προκειμένου να διεξάγουν κάποιες λειτουργίες, όπως να ενεργοποιήσουν το φωτισμό ενός χώρου ή να ρυθμίσουν τη θερμοκρασία. Έξυπνα συστήματα εγκαθίστανται και σε εμπορικές εφαρμογές, όπου αναφέρονται με τον όρο «αυτοματισμοί κτηρίων (Building Automation)».

Τα έξυπνα συστήματα μπορούν να ελέγχουν, εκτός από τις ηλεκτρολογικές εγκαταστάσεις, τις μηχανολογικές εγκαταστάσεις αλλά και τις οικιακές συσκευές και τις συσκευές πολυμέσων (Multimedia), δημιουργώντας ένα ενοποιημένο σύστημα. Στις τελευταίες εντάσσονται οι συσκευές τηλεπικοινωνιών, τα ηχοσυστήματα αλλά και οι τηλεοράσεις του

σπιτιού. Συνδυάζοντας όλες αυτές τις ανεξάρτητες, αρχικά, εγκαταστάσεις σε μία κοινή βάση αποκτάται πλήρης έλεγχος της οικίας, ο οποίος μπορεί να διεξαχθεί ακόμα και από μακριά.

Ένα χαρακτηριστικό των έξυπνων σπιτιών είναι ότι τα ίδια περιφερειακά χρησιμοποιούνται για πολλές χρήσεις. Χαρακτηριστικό παράδειγμα είναι ότι οι αισθητήρες παρουσίας μπορούν να χρησιμοποιηθούν για τον έλεγχο του φωτισμού και του συστήματος θέρμανσης, αλλά χρησιμεύουν και για το σύστημα του συναγερμού. Ένα άλλο παράδειγμα αφορά στις οθόνες των τηλεοράσεων, οι οποίες μπορούν να προβάλουν και την εικόνα της θυροτηλεόρασης.

Τα πλεονεκτήματα, που προκύπτουν από τον αποτελεσματικό συντονισμό των συστημάτων, αφορούν στη διευκόλυνση της καθημερινότητας των χρηστών. Η βελτίωση της ποιότητας ζωής των ενοίκων, έπειτα από κατάλληλο προγραμματισμό του συστήματος, συνοδεύεται από εξοικονόμηση της καταναλισκόμενης ενέργειας και κατ' επέκταση και από εξοικονόμηση χρημάτων. Επίσης, τα έξυπνα συστήματα είναι δυνατό να εξασφαλίσουν ασφαλέστερες συνθήκες διαβίωσης. Κάποια ενδεικτικά παραδείγματα, σχετικά με τους τρόπους που επιτυγχάνονται αυτοί οι στόχοι, είναι τα εξής:

- Ποιότητα ζωής: Ο ένοικος, μέσω οποιουδήποτε τονικού τηλεφώνου, σταθερού ή κινητού ή μέσω του Internet, μπορεί να χειριστεί τις κύριες λειτουργίες της κατοικίας, κατά τη διάρκεια απουσίας του. Έτσι, έχει τη δυνατότητα να ανάψει το θερμοσίφωνα, λίγο πριν φτάσει σπίτι του, και να ρυθμίσει τη θερμοκρασία του σπιτιού. Επίσης, μπορεί να προγραμματίσει αυτοματοποιημένο πότισμα, κατά τη διάρκεια μακράς απουσίας.
- Εξοικονόμηση ενέργειας: Η κατανάλωση ενέργειας μειώνεται με τον αυτόματο έλεγχο των θερμαντικών σωμάτων. Εφόσον η θερμοκρασία δωματίου φτάσει σε κάποιο επιθυμητό επίπεδο, τα θερμαντικά σώματα απενεργοποιούνται αυτόματα. Ένας άλλος τρόπος για την αποφυγή άσκοπης κατανάλωσης ενέργειας είναι η απενεργοποίηση της θέρμανσης, όταν είναι ανοιχτά τα παράθυρα.
- Ασφάλεια: Τα σύγχρονα συστήματα προσφέρουν τη δυνατότητα παρακολούθησης της κατοικίας. Έτσι, ο ιδιοκτήτης έχει τη δυνατότητα, όχι μόνο να παρακολουθεί από όλες τις τηλεοράσεις του σπιτιού την εικόνα που καταγράφουν οι κάμερες, αλλά και να ενημερώνεται για την κατάσταση της οικίας, κατά την απουσία του, μέσω φωτογραφιών στο κινητό του. Σε περίπτωση που ενεργοποιηθούν οι αισθητήρες συναγερμού, λόγω παραβίασης, υπάρχει η δυνατότητα αυτόματης

καταγραφής εικόνων. Επιπλέον, ο ιδιοκτήτης μπορεί να ενημερώνεται αν προκύψει κάτι έκτακτο, όπως πυρκαγιά ή διαρροή νερού, κατά την απουσία του.

Προκειμένου η διαχείριση των συσκευών στο έξυπνο σπίτι να είναι εύκολη, και να μπορεί να γίνει από παντού, απαιτείται η δημιουργία μιας εφαρμογής για φορητές/κινητές συσκευές. Η διάδοση των Smartphone και Tablet, τα τελευταία χρόνια, ευνοεί την ανάπτυξη τέτοιων εφαρμογών και τις κάνεις προσιτές στους χρήστες.

1.1. Σκοπός και δομή της εργασίας

Ο στόχος, της συγκεκριμένης εργασίας, είναι ο σχεδιασμός και η ανάπτυξη εφαρμογής Android για τη διαχείριση έξυπνου σπιτιού. Στα πλαίσια της εργασίας, θα γίνουν τα εξής:

- Θα παρουσιαστούν οι έννοιες Internet of Things και Web of Things.
- Θα γίνει μελέτη της αρχιτεκτονικής του Android και του Android SDK, που χρησιμοποιείται για την ανάπτυξη εφαρμογών σε Android.
- Θα σχεδιαστεί ένα REST API για τη διαχείριση συσκευών στο σπίτι, με γνώμονα την ασφάλεια και την μείωση της κατανάλωσης ενέργειας.
- Θα σχεδιαστεί και θα υλοποιηθεί μια Android εφαρμογή για την διαχείριση ενός smarthome, η οποία θα χρησιμοποιεί το REST API που θα σχεδιαστεί.

Η εφαρμογή σχεδιάζεται για Smartphones, που τρέχουν Android, και έχει τις παρακάτω λειτουργίες:

- Σύνδεση στον απομακρυσμένο Server με τα στοιχεία του χρήστη.
- Διαχείριση συστήματος συναγερμού:
    - Ενεργοποίηση/απενεργοποίηση συστήματος.
    - Ενεργοποίηση/απενεργοποίηση συναγερμού.
    - Έλεγχος της κατάστασης στις διάφορες ζώνες.
- Διαχείριση κεντρικής θέρμανσης:
    - Ενεργοποίηση/απενεργοποίηση θέρμανσης.
    - Ενεργοποίηση/απενεργοποίηση θέρμανσης με χρονοδιακόπτη (on/off timer).

- Διαχείριση θερμοσίφωνα:
    - Ενεργοποίηση/απενεργοποίηση θερμοσίφωνα.
    - Ενεργοποίηση/απενεργοποίηση θερμοσίφωνα με χρονοδιακόπτη (on/off timer).
- Διαχείριση των παραθύρων και της γκαραζόπορτας.
- Επισκόπηση συστημάτων (Εμφάνιση μηνυμάτων από τις διάφορες απομακρυσμένες συσκευές).

Η δομή της υπόλοιπης εργασίας, είναι η ακόλουθη:

- Στο δεύτερο κεφάλαιο, θα γίνει εισαγωγή στις έννοιες Internet of Things και Web of Things.
- Στο τρίτο κεφάλαιο, θα γίνει εισαγωγή στο Android και στα βασικά δομικά στοιχεία του.
- Στο τέταρτο κεφάλαιο, παρουσιάζεται ο σχεδιασμός του συστήματος και οι τεχνολογίες που θα χρησιμοποιηθούν.
- Στο πέμπτο κεφάλαιο, παρουσιάζεται η υλοποίηση του συστήματος (εφαρμογή Android και Gateway).
- Η εργασία ολοκληρώνεται στο έκτο κεφάλαιο με τον επίλογο.

2. Internet of Things και Web of Things

2.1. Internet of Things

Σε παγκόσμια κλίμακα, υπάρχουν πάνω από 2 δισεκατομμύρια συσκευές, που έχουν σύνδεση στο διαδίκτυο, και πάνω από 1 δισεκατομμύριο χρήστες. Ασύρματες, κινητές και δικτυακές τεχνολογίες έδωσαν, σε πρώτη φάση, τη δυνατότητα διάχυτης επικοινωνίας μεταξύ συστημάτων και εφαρμογών, δημιουργώντας πολύ σημαντικές οικονομικές ευκαιρίες. Ωστόσο, αυτό μπορεί να είναι μόνο η αρχή: εξελισσόμενες τεχνολογίες δικτύωσης, όπως η ταυτοποίηση μέσω ραδιοσυχνοτήτων (RFID tags), ενδεχομένως θα μπορούν να αναπτυχθούν σε εκατοντάδες δισεκατομμύρια, για μια σχεδόν απεριόριστη γκάμα εφαρμογών.

Μακροπρόθεσμα, υπάρχει η πεποίθηση στην ερευνητική κοινότητα ότι οι απλές ετικέτες (tags), του σήμερα, θα μετεξελιχθούν σε δικτυωμένα αντικείμενα με δυνατότητα αποθήκευσης, επεξεργασίας δεδομένων και προσθήκης αισθητήρων. Αυτό, με τη σειρά του, θα ανοίξει ένα εντελώς νέο φάσμα σε δικτυακές εφαρμογές, όπως τις M2M (”Man-to-Machine, Machine-to-Man, Machine-to-Mobile και Mobile-to-Machine”) επικοινωνίες.

Το «Διαδίκτυο των Πραγμάτων» είναι ένα δίκτυο δισεκατομμυρίων ή τρισεκατομμυρίων μηχανών, οι οποίες επικοινωνούν μεταξύ τους. Είναι πολύ σημαντικό και κυρίαρχο θέμα για την εξέλιξη των πληροφοριών και των επικοινωνιών τις επόμενες δεκαετίες, καθώς η απλούστερη μορφή αυτού είναι ήδη σε χρήση. Υπάρχουν 1,3 δισεκατομμύρια ταυτοποιήσεις, μέσω ραδιοσυχνοτήτων (RFID), και 2 δισεκατομμύρια χρήστες κινητών υπηρεσιών, σε όλο τον κόσμο.

Η ιδέα έχει διευρυνθεί, από προηγμένες έννοιες, τα τελευταία 20 χρόνια:

- Ubiquitous Communications
- Pervasive Computing
- Ambient Intelligence

Από μια γενική σκοπιά, μπορεί να συναχθεί το συμπέρασμα ότι η τάση για τον όλο και μεγαλύτερο αριθμό συνδεδεμένων ευφυών αντικειμένων είναι αμετάκλητη, διότι η οικονομική αξία ενός συστήματος αντικειμένων και συσκευών είναι άμεσα συνδεδεμένη με το γεγονός ότι είναι «δικτυωμένες».

Το «Διαδίκτυο των Πραγμάτων» θα επιτρέψει, μέσω της πληροφορικής, την υποστήριξη των οικονομικών, της υγείας, της κοινωνικότητας αλλά και της ιδιωτικότητας.

Παρακάτω, αναφέρονται μερικά απλά παραδείγματα:

- Διαχείριση και Ιστορικό συσκευών και ανταλλακτικών: Με τη χρήση των RFID tags, μπορεί να γνωρίζει κανείς από πού έχει προέλθει κάποιο αντικείμενο, τη χρήση του, τι βλάβες έχει παρουσιάσει κλπ.
- Διαχείριση περιουσιακών στοιχείων: Ηλεκτρονική σήμανση και παρακολούθηση της θέσης των αποσκευών σε έναν αερολιμένα ή των εμπορευμάτων, στο πλαίσιο μιας διαδικασίας παραγωγής, στο εργοστάσιο.
- Υγειονομική περίθαλψη: Αισθητήρες για αρτηριακή πίεση και καρδιακή συχνότητα, όπου τακτικές μετρήσεις από το σπίτι του ασθενούς θα μπορούν να στέλνονται σε κάποιο κέντρο παρακολούθησης. Ένας υπολογιστής θα εντοπίζει τυχόν ανεπιθύμητες κινήσεις ή σήματα και θα ειδοποιεί τον υπεύθυνο γιατρό να ελέγξει τον ασθενή.
- Παρακολούθηση περιβάλλοντος: Ένα δίκτυο αισθητήρων θα είναι σε θέση να παρακολουθεί περιβαλλοντικά στοιχεία, όπως τις βροχοπτώσεις, ώστε να μπορεί να επιτευχθεί πρόβλεψη των πλημμυρών ή άλλων καταστροφών.

Οι πιθανές εφαρμογές της διαδεδομένης δικτύωσης είναι απεριόριστες. Μερικές εφαρμογές φαίνονται απαραίτητες. Για παράδειγμα, ο εντοπισμός ανθρώπων σε ένα πιθανό σενάριο καταστροφής, όπως φωτιά σε κάποιο τούνελ. Άλλες εφαρμογές μπορεί εκ πρώτης όψεως να φαίνονται μη ρεαλιστικές. Για να θεωρηθεί ότι αυτή η τεχνολογία πραγματικά αξίζει τον κόπο, υπάρχει ανάγκη για αντιμετώπιση του μεγάλου όγκου των εφαρμογών προκειμένου να κατανοηθούν καλύτερα οι διάφορες απαιτήσεις (π.χ. πραγματικός χρόνος, η ποιότητα των υπηρεσιών), που θα οδηγήσουν τελικά στην απαραίτητη "γενική" τεχνολογική εξέλιξη.

Τα θέματα, που θεωρείται ότι παρουσιάζουν ενδιαφέρον, όσο αφορά το «Διαδίκτυο των Πραγμάτων», είναι τα εξής:

- Η προοπτική του συστήματος, για το πώς τα δικτυωμένα συστήματα είναι πιθανό να εξελιχθούν.
- Οι επιπτώσεις της διασυνδεσιμότητας της συσκευής, για τις μελλοντικές αρχιτεκτονικές δικτύου και των νέων τεχνολογιών.
- Συγκεκριμένη ασφάλεια και προστασία της ιδιωτικότητας.

- Η αρχιτεκτονική, που απαιτείται, για την υποστήριξη των αιτήσεων των τρισεκατομμυρίων συσκευών, που συνδέονται.
- Η σύνθεση των υπηρεσιών.
- Ο ρόλος των "ανοικτών" (λογισμικών) μοντέλων.
- Ο ρόλος των ασύρματων τεχνολογιών.

Αυτά καθιερώθηκαν σαν ερωτήσεις που πρέπει να απαντηθούν, προκειμένου να κατανοηθούν οι απαιτήσεις για το παγκόσμιο δίκτυο του αύριο, για το πώς μπορεί να οικοδομηθεί και για το πώς θα είναι.

2.1.1. Υπάρχουσες τεχνολογίες

Το διαδίκτυο έχει καταστεί η πλατφόρμα, ώστε να ξεκινήσουν οι InformationTechnology (IT) υπηρεσίες για όλα τα είδη των βιομηχανιών. Τα δίκτυα και οι υπηρεσίες λογισμικού συνδέονται μεταξύ τους, με κάθε τρόπο. Οι αρχιτεκτονικές αρχές, για τη δημιουργία προϊόντων λογισμικού και υπηρεσιών, έχουν εξελιχθεί. Αρχιτεκτονικές αρχές, όπως Representational State Transfer (REST), Event Driven Architecture (EDA) και Service Oriented Architecture (SOA), χρησιμοποιούνται για τη δημιουργία υπηρεσιών λογισμικού, που χρησιμοποιούνται από εκατομμύρια ανθρώπους και επιχειρήσεις σε όλο τον κόσμο. Για παράδειγμα, το REST έχει αποδείξει τη χρησιμότητά του ως μια επιτυχημένη αρχή σχεδιασμού, αφού όλο το World Wide Web (WWW) έχει σχεδιαστεί με βάση αυτό. Το SOA χρησιμοποιείται ευρέως για τη δημιουργία centric και web εφαρμογών επιχειρήσεων και υπηρεσιών. Έχει εξελιχθεί σε ένα νέο ώριμο επίπεδο, σε σύγκριση με άλλες τεχνολογίες που είναι διαθέσιμες στο παρελθόν για την παροχή γρήγορων υπηρεσιών. Όλες αυτές οι αρχές αρχιτεκτονικής συμπληρώνουν, επίσης, η μία την άλλη για το σχεδιασμό νέων υπηρεσιών.

Η παράγραφος, που ακολουθεί, παρέχει τεχνικές πληροφορίες σχετικά με τα συστήματα RFID, που είναι μία βασική τεχνολογία του «Διαδικτύου των Πραγμάτων».

Μία από τις πιθανές τεχνολογίες αναγνώρισης, που προτείνεται στην έκθεση της ITU, είναι η Radio Frequency IDentification (RFID). Η RFID είναι μια αυτόματη μέθοδος αναγνώρισης, που χρησιμοποιεί τα ραδιοκύματα για να εντοπίσει σηματοδοτημένα στοιχεία, όπως ανθρώπους, ζώα ή αντικείμενα. Είναι παρόμοιο με το barcode, αλλά χρησιμοποιεί ένα

μικροτσίπ μαζί με ραδιοκύματα. Μια ετικέτα RFID, επίσης γνωστή και ως "Electronic Label", ή "Transponder" ή "Codeplate", αποτελείται από ένα τσιπ RFID, που συνδέεται με μία κεραία. Οι ετικέτες μπορούν να διαβαστούν από πολλά μέτρα μακριά και χωρίς να χρειάζεται οπτική επαφή. Η πιο κοινή μέθοδος αναγνώρισης είναι να φυλάσσεται ο αύξοντας αριθμός, που προσδιορίζει κάποιο άτομο ή κάποιο αντικείμενο. Μια RFID ετικέτα μπορεί, επιπλέον, να αποθηκεύσει και άλλες πρόσθετες πληροφορίες. Τα συστήματα RFID χρησιμοποιούν ετικέτες RFID, που αποτελούν τη σημαντικότερη και αποδοτικότερη μέχρι στιγμής τεχνολογία για τις υπηρεσίες αναγνώρισης στοιχείου, όσον αφορά το IoT («Internet of Things»).

Τα συστήματα RFID αποτελούνται από τρία κύρια στοιχεία: ετικέτες RFID, αναγνώστες RFID, και RFID λογισμικό.

Μια ετικέτα RFID ή αναμεταδότης μεταφέρει τα στοιχεία ταυτότητας, που μπορεί να είναι μια μοναδική συμβολοσειρά ή ένας κωδικός. Επιπλέον, η ετικέτα μπορεί να αποθηκεύσει πληροφορίες ανάλογα με το μέγεθος της μνήμης της. Μια ετικέτα επισυνάπτεται, φυσικά, στο αντικείμενο που πρέπει να εντοπιστεί. Πρόκειται για μία ηλεκτρική συσκευή σχεδιασμένη να λαμβάνει ένα ειδικό σήμα και να στέλνει αυτόματα μια συγκεκριμένη απάντηση. Αποτελείται από ένα στοιχείο σύνδεσης (όπως ένα πηνίο ή μια κεραία μικροκυμάτων) και ένα ηλεκτρονικό μικροτσίπ (λιγότερο από το 1 / 3 του χιλιοστού σε μέγεθος). Μπορεί να είναι παθητική, ημι-παθητική ή ενεργητική, με βάση την πηγή τροφοδοσίας της και τον τρόπο χρησιμοποίησής της, και μπορεί να είναι μόνο ανάγνωσης, ανάγνωσης ή εγγραφής και ανάγνωσης / εγγραφής και αναδιατύπωσης, ανάλογα με το πώς τα δεδομένα είναι κωδικοποιημένα. Οι παθητικές ετικέτες RFID λαμβάνουν την ενέργεια, από το ηλεκτρομαγνητικό πεδίο που εκπέμπεται, από τους αναγνώστες. Οι ετικέτες χρησιμοποιούν συχνότητες εκπομπής, που κυμαίνονται στα kiloHertz, megaHertz και gigaHertz.

Ένας RFID αναγνώστης είναι μία συσκευή, που χρησιμοποιείται για να διαβάσει τα δεδομένα που στέλνονται από την ετικέτα. Για ανάγνωση παθητικών ετικετών RFID, ο αναγνώστης πρέπει να χρησιμοποιήσει επιπλέον ενέργεια σε ηλεκτρομαγνητική μορφή. Η συσκευή ανάγνωσης αποτελείται από μια διπολική κεραία και ένα μικροεπεξεργαστή. Η διπολική κεραία χρησιμοποιείται για τη μετάδοση των σημάτων και την τροφοδότηση της ετικέτας. Ο μικροεπεξεργαστής ελέγχει και διευθύνει όλες τις σχετικές διεργασίες της συσκευής ανάγνωσης. Ο αναγνώστης μπορεί να είναι είτε μια ειδική συσκευή χειρός, είτε να είναι ενσωματωμένος σε μια κινητή συσκευή ή μπορεί ακόμα να ενταχθεί σε ένα τοίχο

(σταθερές συσκευές ανάγνωσης). Σε σύγκριση με τις ετικέτες, οι συσκευές ανάγνωσης είναι μεγαλύτερες, πιο ακριβές, και καταναλώνουν περισσότερη ενέργεια ανάλογα με την ιδιότητά τους. Η συσκευή ανάγνωσης πρέπει να έχει, επίσης, τη δυνατότητα επικοινωνίας (ενσύρματη ή ασύρματη).

Το RFID λογισμικό είναι ένα λογισμικό, που τρέχει για τη συσκευή ανάγνωσης RFID. Υπάρχουν διάφορες υπηρεσίες, που περιλαμβάνονται σε αυτό. Υπηρεσίες όπως η ανάγνωση, το κλείδωμα, η εγγραφή και η διαγραφή μιας ετικέτας περιλαμβάνονται στο βασικό σύνολο λειτουργιών. Επιπλέον, το φιλτράρισμα, το άθροισμα, η αναζήτηση, η καταγραφή συμβάντων και οι υπηρεσίες ασφαλείας περιλαμβάνονται και αυτά στο λογισμικό. Το υλικολογισμικό, επίσης, χρειάζεται για να επικοινωνεί η συσκευή ανάγνωσης με το δίκτυο, η οποία θα πρέπει να επεξεργάζεται και τις εντολές που λαμβάνει από ένα απομακρυσμένο σύστημα ελέγχου.

Υπάρχουν δύο τύποι RFID: κοντινού πεδίου RFID και μακρινού πεδίου RFID. Με Near Field Communication (NFC) RFID είναι εξοπλισμένα πολλά τηλέφωνα, τα τελευταία χρόνια, και πολλές NFC υπηρεσίες είναι διαθέσιμες, ήδη, για NFC τηλέφωνα. Οι NFC συσκευές ανάγνωσης διαβάζουν την ετικέτα είτε με ένα άγγιγμα είτε εντός εμβέλειας 5 cm. Μακρινού πεδίου RFID καλύπτουν απόσταση μεταξύ 3 και 6 μέτρων, από σταθερές συσκευές ανάγνωσης.

2.1.2. Προοπτικές

Το «Διαδίκτυο των Πραγμάτων» είναι ένα δίκτυο επικοινωνίας συσκευών, με τέσσερις βαθμούς πολυπλοκότητας:

- Καθαρά παθητικές συσκευές (RFID), που δίνουν σταθερά δεδομένα μετά από ερώτηση εξ' αποστάσεως.
- Συσκευές με μέτρια επεξεργαστική ισχύ, οι οποίες διαμορφώνουν το μήνυμα που ίσως να μεταβάλλεται με το χρόνο και θέση.
- Τηλεπισκοπικές συσκευές, που παράγουν και στέλνουν πληροφορίες σχετικά με το περιβάλλον τους, όπως για παράδειγμα πίεση, θερμοκρασία, επίπεδο του φωτός, τοποθεσία.

- Συσκευές με μεγαλύτερη επεξεργαστική ισχύ, που μπορεί να αποφασίσουν, χωρίς ανθρώπινη παρέμβαση, να επικοινωνήσουν με κάποια άλλη συσκευή. Το τελικό αυτό στάδιο κάνει "Pro Active Computing", δηλαδή αυτές οι συσκευές, έχοντας επίγνωση του φυσικού τους περιβάλλοντος, απαντούν σε αυτό το πλαίσιο με κάποια μορφή αντίδρασης, που πιθανό να προκαλέσει την αλλαγή του πλαισίου.

Η δύναμη του «Διαδικτύου των Πραγμάτων» και των εφαρμογών, που υποστηρίζει, οφείλεται στο γεγονός ότι οι συσκευές συνδέονται μεταξύ τους. Οι συσκευές αυτές, ορισμένες φορές, αποκαλούνται "έξυπνες συσκευές", αν και ομάδες συσκευών δεν χρειάζεται να είναι το ίδιο έξυπνες. Μια gateway device μπορεί να απορρίπτει, να αθροίζει και να ελέγχει τη μορφή των δεδομένων, από μια ομάδα απλών αισθητήρων, πριν τα στείλει κάπου αλλού. Αυτό είναι ιδίως σημαντικό σε σχέση με την ασύρματη πρόσβαση και την επιλογή της κατάλληλης αρχιτεκτονικής, η οποία μπορεί να ποικίλλει ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής.

Το «Διαδίκτυο των Πραγμάτων», όπως φαίνεται, επιβάλλει μια νέα τάξη δικτύου, που παρουσιάστηκε ως Capillary Networks. Αυτά τα, μικρών-αποστάσεων, δίκτυα αιχμής είναι η επέκταση των υπαρχόντων δικτύων και υπηρεσιών σε όλες τις συσκευές, που είναι εξοπλισμένες με αισθητήρες και ενεργοποιητές.

Στα Capillary Networks ανήκουν τα BAN, PAN και, ενδεχομένως, τα LAN περιβάλλοντα, με Gateway Devices που παρέχουν διεπαφή μεταξύ των επιπέδων, όπου αυτή χρειάζεται.

Τα Capillary Networks είναι αυτόνομα και αυτό-οργανώνονται, λόγω της ευελιξίας τους και της οικονομίας τους. Τα μικρά αυτά δίκτυα δεν πρέπει να επιβαρύνονται με το κόστος διαμόρφωσης, διαχείρισης και συντήρησής τους, ως εκ τούτου το «Διαδίκτυο των Πραγμάτων» θα πρέπει να λειτουργεί όπως ακριβώς απαιτεί το "Plug and Play" με τους ενσύρματους υπολογιστές. Οι γειτονικές συσκευές θα πρέπει να ανταλλάσσουν μηνύματα για να ανακαλύψουν η μία την άλλη και να ρυθμίσουν οι ίδιες την καλύτερη τοπολογία για την ανταλλαγή και την αποστολή μηνυμάτων προς, από και διαμέσω της μιας και της άλλης.

Με τον τρόπο αυτό, το δίκτυο θα μπορεί να αντιδράσει, όταν μια συσκευή φθάνει, φεύγει ή κινείται και θα μπορεί να διατηρηθεί, όταν πρόκειται να υπάρξει παρέμβαση ή διατάραξη των διαύλων επικοινωνίας. Η αυτό-οργάνωση και η δυνατότητα δυναμικής δρομολόγησης αποτελούν βάση για τη βιωσιμότητα και την αξιοπιστία των δικτύων του «Διαδικτύου των Πραγμάτων».

Η ανάγκη επικοινωνίας των συσκευών στο «Διαδίκτυο των Πραγμάτων» μεταξύ τους, οπουδήποτε και αν είναι αυτές εγκατεστημένες στον κόσμο, συνεπάγεται τη σπουδαιότητα της ονοματοδοσίας και της διευθυνσιοδότησης, οι οποίες θα προκύψουν μέσα από κατάλληλη έρευνα και επεξεργασία. Μέσω αυτών θα βρεθεί αν η επιθυμητή συσκευή υπάρχει, πού βρίσκεται, και ποιοι δίαυλοι επικοινωνίας απαιτούνται για τη μεταξύ των συσκευών επικοινωνία. Επιπλέον, οι πληροφορίες, που παρέχονται από τις συσκευές αυτές, μπορούν, τελικά, να συγκεντρώνονται και να χρησιμοποιούνται, μαζί με τις υπόλοιπες πληροφορίες, σε έναν κοινό Web Server. Το γεγονός αυτό σημαίνει ότι οι συσκευές μπορούν να συνδέονται με μια μοναδική ταυτότητα (μέσω της ονοματοδοσίας και της διευθυνσιοδότησης), η οποία δημιουργεί την επιτακτική ανάγκη για προστασία των προσωπικών δεδομένων και ασφαλή ανταλλαγή πληροφοριών.

Το «Διαδίκτυο των Πραγμάτων» πρέπει να είναι σύμφωνο με τα πρότυπα και τις πρότυπες μεθόδους λειτουργίας. Ενώ η κάθε μεμονωμένη εφαρμογή πρέπει να έχει την δυνατότητα να χρησιμοποιεί τις δικές της ειδικές συσκευές, τα δικά της πρωτόκολλα και τους δικούς της τρόπους λειτουργίας, είναι συμφέρον όταν είναι άμεσα διαθέσιμες συνήθεις διαλειτουργικές πλατφόρμες να χρησιμοποιούνται αυτές.

Πρέπει να εξελιχθούν οι υπηρεσίες πλατφορμών, πέρα από την τρέχουσα "στατική" διαμόρφωση υπηρεσιών, και να προχωρήσουν προς την κατεύθυνση της παροχής υπηρεσιών σε προσανατολισμένες αρχιτεκτονικές. Σήμερα, οι υπηρεσίες είναι προγραμματισμένες και δεν μπορούν να προσαρμοστούν δυναμικά στις μεταβαλλόμενες απαιτήσεις και στα πλαίσια εφαρμογής. Η διαλειτουργικότητα εξακολουθεί να απαιτεί από τους χρήστες να γνωρίζουν τις δυνατότητες που προσφέρονται από τους παρόχους και τις αντίστοιχες υπηρεσίες τους, από πριν. Αυτή η "Closed World" προσέγγιση συνεπάγεται ότι όλες οι Client υπηρεσίες, και τα χαρακτηριστικά αυτών, είναι γνωστά εκ των προτέρων. Από την άλλη, η σημασιολογική μοντελοποίηση θα πρέπει να καταστεί δυνατή για τους Service Requestors, ώστε να μπορούν να καταλάβουν ποια υπηρεσία και ποιοι πάροχοι μπορούν να προσφερθούν. Αυτό αποτελεί βασικό ζήτημα για την πορεία προς μια "Open World" προσέγγιση, όπου νέα, άγνωστα ή τροποποιημένα προϊόντα-υπηρεσίες μπορεί να εμφανιστούν σε οποιαδήποτε χρονική στιγμή. Αυτό έχει σαν συνέπεια να υπάρξουν ενδιάμεσες απαιτήσεις, όπως αυτές που χρειάζονται για τη διασύνδεση μεταξύ των συσκευών. Αυτό είναι ένα βασικό στοιχείο για την πρόοδο των δικτυακών συσκευών, που τις καθιστά ικανές να προσαρμόζονται δυναμικά στις αλλαγές του πλαισίου, που μπορεί να επιβληθούν από τα σενάρια εφαρμογής (π.χ. μετάβαση από την παρακολούθηση της

λειτουργίας σε κατάσταση συναγερμού με σκοπό την επαγρύπνηση, το οποίο μπορεί να συνεπάγεται διαφορετικές υπηρεσίες, καθώς και την εφαρμογή διαφορετικών συμπεριφορών).

2.1.3. Web of Things

Το «Web of Things» είναι εμπνευσμένο από το «Internet of Things», όπου αντικείμενα που περιέχουν μια ενσωματωμένη συσκευή, ή ένα υπολογιστή, συνδέονται πλήρως στο Web και είναι προσβάσιμα μέσω του πρωτοκόλλου HTTP. Παραδείγματα έξυπνων συσκευών και αντικειμένων είναι: τα ασύρματα δίκτυα αισθητήρων, οι συσκευές περιβάλλοντος, οι οικιακές συσκευές, κλπ.

Σε αντίθεση με τα συστήματα που υπάρχουν για το «Διαδίκτυο των Πραγμάτων», ο «Ιστός των Πραγμάτων» βασίζεται στα υπάρχοντα και ευρέως διαδεδομένα πρότυπα του Web για την διασύνδεση και την επέκταση του οικοσυστήματος των ενσωματωμένων συσκευών. Αποδεκτά και διαδεδομένα πρότυπα, όπως το URI, HTTP, REST, χρησιμοποιούνται για την πρόσβαση στη λειτουργικότητα των έξυπνων αντικειμένων.

Η πρόσβαση στις έξυπνες συσκευές βασίζεται, λοιπόν, στο κλασικό μοντέλο της client-server αρχιτεκτονικής. Για την επικοινωνία client-server, υπάρχουν διάφορες λύσεις:

- Τα Web Services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους, ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα Web Service είναι μια διεπαφή λογισμικού (Software Interface) που περιγράφει μια συλλογή από λειτουργίες, οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (Operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από Web Services, οι οποίες αλληλεπιδρούν μεταξύ τους, καθορίζει μια εφαρμογή Web Services. Η τεχνολογία αυτή χρησιμοποιείται συχνά για την επικοινωνία ανάμεσα σε απομακρυσμένα συστήματα, αλλά απαιτεί αρκετούς υπολογιστικούς πόρους και δεν προτείνεται για την χρήση σε κινητές συσκευές. Τα Web Services χρησιμοποιούν το HTTPS πρωτόκολλο μεταφοράς, ενώ η δομή των μηνυμάτων περιγράφεται από το πρωτόκολλο SOAP.

- Η αρχιτεκτονική REST κάνει, επίσης, χρήση του HTTP πρωτόκολλου. Η βασική του αρχή είναι η ύπαρξη πόρων (πηγών δεδομένων ή άλλης πληροφορίας), κάθε μια από τις οποίες αναφέρεται από ένα μοναδικό URI. Τα δεδομένα οργανώνονται σε συλλογές από URI (Collection of URI's) της μορφής: http://example.com/resources/. Για την επιστροφή των δεδομένων χρησιμοποιείται είτε κάποια δομή XML είτε, για μεγαλύτερη ταχύτητα και ευκολία, προτιμάται συνήθως το πρότυπο JSON.

Τα πλεονεκτήματα, του παραπάνω μοντέλου, είναι τα εξής:

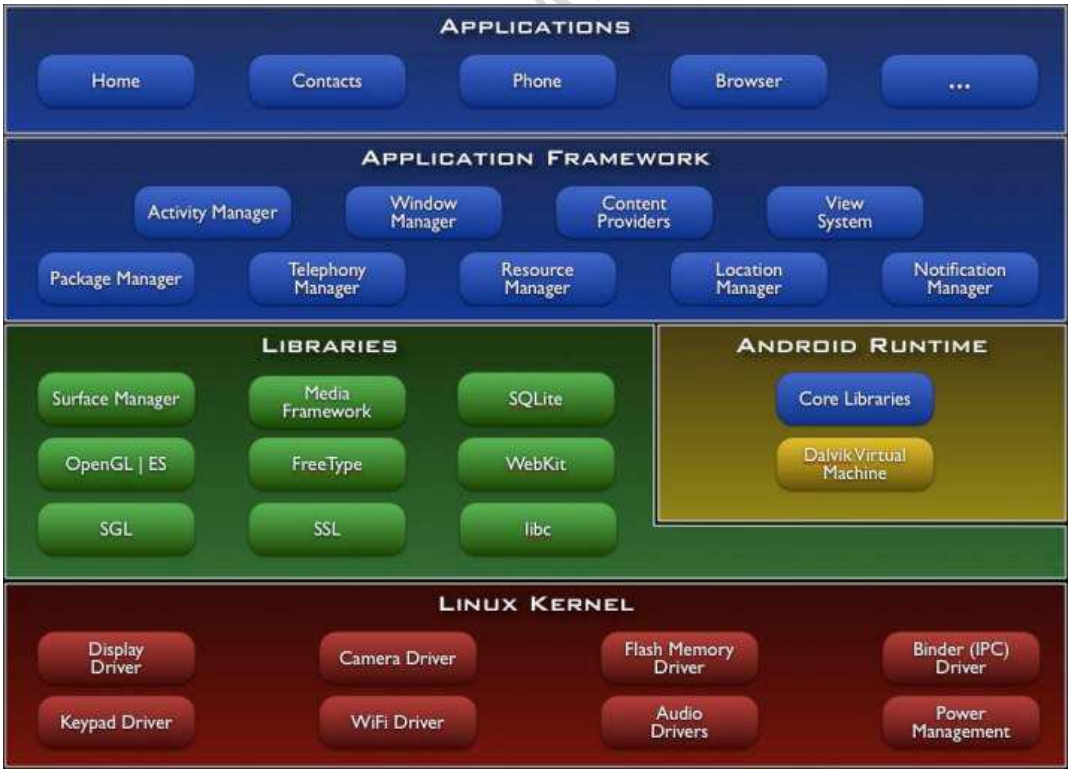
- Το HTTP είναι απλό και θεωρείται standard, ενώ υπάρχουν βιβλιοθήκες για τη χρήση του.
- Η αποθήκευση δεδομένων βρίσκεται σε κεντρικό εξυπηρετητή, συνεπώς ο χρήστης μπορεί να συνδεθεί από διάφορες συσκευές.
- Η αποθήκευση και ο χειρισμός των δεδομένων γίνεται από το σύστημα βάσης δεδομένων και, έτσι, εξασφαλίζεται η καλή κατάσταση τους.
- Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate).
- Το κατανεμημένο σύστημα είναι ανεξάρτητο πλατφόρμας και γλώσσας προγραμματισμού, χάρη στη χρήση του HTTP και του JSON.

3. Εισαγωγή στο Android

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

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

Η αρχιτεκτονική του Android φαίνεται στο, αμέσως, παρακάτω σχήμα :



Εικόνα 1 – Αρχιτεκτονική του Android

3.1. Δομικά στοιχεία αρχιτεκτονικής.

Linux Kernel

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

Android Runtime

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

Βιβλιοθήκες

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

- System C Libraries.
- Media Libraries: Οι βιβλιοθήκες αυτές υποστηρίζουν δυνατότητες εγγραφής και playback πολλών γνωστών τύπων ήχου, εικόνας και video, όπως: MPEG4, H.264, MP3, AAC, AMR, JPEG, PNG.
- Surface Manager: Βοηθά την πρόσβαση στα Subsystems της εφαρμογής.
- WebKit: Μηχανή για μοντέρνο Web Browser.
- SGL: Μηχανή για 2D γραφικά.
- 3D Libraries: Βιβλιοθήκες βασισμένες στα OpenGL ES 1.0 APIs.

- SQLite: Μια ευέλικτη και ισχυρή βάση δεδομένων, διαθέσιμη σε όλες τις εφαρμογές.

Application Framework

Σχεδόν όλες οι εφαρμογές του Android αποτελούνται από μια γκάμα με γραφικά και Services, τα οποία ανταποκρίνονται στις απαιτήσεις του χρήστη και είναι γραφικά εργαλεία, όπως: ActivityViews, Grids, Lists, TextViews, EditIntroductionTexts, Spinners, Buttons, έναν ενσωματωμένο Web Browser, MapView.

Οι Content Providers χρησιμοποιούνται για να επιτρέπουν στις εφαρμογές να έχουν πρόσβαση σε δεδομένα άλλων εφαρμογών (όπως π.χ. στις επαφές του κινητού τηλεφώνου) ή να μοιράζονται με άλλες εφαρμογές τα δικά τους δεδομένα.

Επιπλέον, οι Resource Manager, Activity Manager και Notification Manager διαχειρίζονται την κατάσταση των εφαρμογών και παρέχουν, κατά τη διάρκεια της λειτουργίας του κινητού, χρήσιμες πληροφορίες, όπως π.χ. ότι υπάρχει διαθέσιμο δίκτυο Wi-Fi εντός εμβέλειας, προειδοποίηση για κάποια πληροφορία ή ενός νέου E-mail στο λογαριασμό μας.

Applications

Μια σειρά από εφαρμογές βρίσκονται στο υψηλότερο επίπεδο της αρχιτεκτονικής του Android λογισμικού και περιλαμβάνουν: Email Client, SMS/MMS, Ημερολόγιο, Web Browser, Χάρτες, Επαφές (Contacts) κ.α. Όλες οι εφαρμογές είναι γραμμένες σε γλώσσα προγραμματισμού Java.

3.2. Ανατομία μιας Εφαρμογής Android.

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

AndroidManifest

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

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

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

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

<manifest>

    <uses-permission/>
    <permission/>
    <permission-tree/>
    <permission-group/>
    <instrumentation/>
    <uses-sdk/>
```

```
    <uses-configuration/>
    <uses-feature/>
    <supports-screens/>
    <compatible-screens/>
    <supports-gl-texture/>

    <application>

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

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

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

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

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

        <uses-library/>

    </application>

</manifest>
```

Ιδιαίτερο στοιχείο, το οποίο δηλώνεται στο αρχείο Manifest, είναι οι άδειες πρόσβασης της εφαρμογής. Στο Android, οι διάφοροι πόροι της συσκευής (δίκτυο, εξωτερική κάρτα μνήμης, GPS) προστατεύονται και για να μπορεί κάποια εφαρμογή να τους χρησιμοποιήσει πρέπει να έχει δηλωθεί η αντίστοιχη άδεια, στο αρχείο αυτό.

Για παράδειγμα, αν η εφαρμογή μας χρησιμοποιεί την κάμερα της συσκευής θα πρέπει να δηλώνεται και η αντίστοιχη άδεια. Αν θέλουμε να έχουμε πρόσβαση στο διαδίκτυο, από την εφαρμογή μας, θα πρέπει να δηλώνεται η αντίστοιχη άδεια. Όμοια ισχύουν για το αν θέλουμε να αποθηκεύσουμε αρχεία στην κάρτα SD, αν θέλουμε να στείλουμε μηνύματα

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

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

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

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

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

Activity

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

καινούριας Activity. Σε μερικές περιπτώσεις μια Activity μπορεί να επιστρέψει μια τιμή σε μια προηγούμενη Activity.

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

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

Υπάρχει, επίσης, μια σχετική κλάση που ονομάζεται IntentFilter. Αν το Intent είναι μία αίτηση για να γίνει κάτι, το IntentFilter περιγράφει ποια και πόσα τέτοια Intents μπορεί να χειριστεί μια Activity.

Intent Receiver

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

Services

Ένα Service είναι τμήμα κώδικα, που εκτελείται χωρίς κάποιο Interface χρήστη. Ένα καλό παράδειγμα Service είναι ο Media Player, που παίζει τραγούδια από μια λίστα. Σε μια εφαρμογή Media Player είναι λογικό να υπάρχουν διάφορες οθόνες, άρα και πολλές

Activities, όπου ο χρήστης θα μπορεί να επιλέξει τραγούδια και να τα ακούσει. Παρόλα αυτά το playback δε θα πρέπει να χειρίζεται από μια Activity, γιατί ο χρήστης περιμένει να μπορεί να περιηγείται στη λίστα τραγουδιών του χωρίς το τραγούδι, που ακούει εκείνη τη στιγμή, να σταματήσει. Σε αυτή την περίπτωση, η κύρια Activity του Media Player θα ξεκινήσει να εκτελεί ένα Service στο Background, οπότε ο χρήστης θα μπορεί να κάνει Pause, Rewind, κλπ.

Content Providers

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

Resources

Στα Resources, μιας εφαρμογής, ορίζεται: το Layout των Activities, οι διάφορες εικόνες και τα λεκτικά, που χρησιμοποιούνται στα Activities. Σε κάθε Activity αντιστοιχεί ένα Layout αρχείο, το οποίο περιγράφει τη θέση των διάφορων αντικειμένων στην οθόνη.

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

Στην ενότητα για την διεπαφή χρήστη αναφέρθηκε ότι η διάταξη των γραφικών στοιχείων δηλώνεται σε αρχεία XML. Συγκεκριμένα, στο Project της εφαρμογής μας, υπάρχει ο φάκελος res/layout/, στον οποίο τοποθετούμε όλα τα αρχεία XML που αφορούν το User Interface της εφαρμογής μας (το res προέρχεται από το resources).

Έστω ότι η εφαρμογή μας περιέχει τρεις διαφορετικές οθόνες (δηλαδή τρεις διαφορετικές Activities), με κάθε μία να χρησιμοποιεί το δικό της User Interface. Τοποθετούμε, λοιπόν, τα αρχεία myfirstscreen.xml, mysecondscreen.xml και mythirdscreen.xml στον παραπάνω

φάκελο. Όμως, η τρίτη οθόνη της εφαρμογής μας εμφανίζει τρεις εικόνες, όπως το παράδειγμα παραπάνω, άρα θέλουμε να χρησιμοποιήσουμε διαφορετική διάταξη για το Landscape Mode της τρίτης οθόνης. Όλα τα αρχεία XML, που αφορούν το User Interface και συγκεκριμένα την περίπτωση του Landscape Mode, τοποθετούνται στο φάκελο res/layout-land/ (δηλαδή layout-landscape). Άρα, λοιπόν, δημιουργούμε ένα ακόμα αρχείο XML, με το ίδιο ακριβώς όνομα mythirdscreen.xml, το οποίο όμως τοποθετούμε στο φάκελο res/layout-land/.

Η εντολή για να δηλωθεί το αρχείο XML, που θα χρησιμοποιηθεί σε μία Activity, είναι η setContentView(). Όταν θα εκτελεστεί η εφαρμογή μας, και συγκεκριμένα η εντολή setContentView(), το λειτουργικό σύστημα Android θα ελέγξει τον προσανατολισμό της συσκευής μας και θα επιλέξει το κατάλληλο αρχείο XML. Αν βρισκόμαστε στην πρώτη οθόνη, σε προσανατολισμό Portrait, και αλλάξουμε προσανατολισμό σε Landscape, τότε το Android θα ελέγξει αν υπάρχει το αρχείο myfirstscreen.xml στον φάκελο res/layout-land/. Δε θα το βρει, όμως, διότι δεν το έχουμε ορίσει, άρα θα χρησιμοποιήσει το προεπιλεγμένο αρχείο, το οποίο βρίσκεται στο φάκελο res/layout/, και τελικά η διάταξη των στοιχείων θα είναι κοινή και για τους δύο προσανατολισμούς.

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

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

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/andr
oid"
android:layout_width="match_parent"
android:layout_height="match_parent">


<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
```

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

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

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

Handlers

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

Υπάρχουν δύο κύριες χρήσεις, για ένα Handler:

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

Η δεύτερη περίπτωση έχει ιδιαίτερο ενδιαφέρον, καθώς για λόγους καλής λειτουργίας, χρονοβόρες διαδικασίες που πρέπει να τρέξουν σε κάποιο Activity, τοποθετούνται σε διαφορετικό Thread. Αν, για παράδειγμα, με το πάτημα ενός πλήκτρου πρέπει να σταλεί κάτι στο διαδίκτυο και η απάντηση να ενημερώσει κάποιο block στο Activity, τότε η

διαδικασία γίνεται μέσω ενός νέου νήματος. Προκειμένου, όμως, το νήμα να μπορεί να ενημερώσει το Activity, που το δημιούργησε, θα πρέπει να χρησιμοποιήσει ένα Handler.

Ακολουθεί ένα παράδειγμα χρήσης:

```
//handles messages send from other threads

public Handler _handler = new Handler() {

        @Override

        public void handleMessage(Message msg) {

        Bundle bundle = msg.getData();

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

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

        while(it.hasNext()){

                String key = it.next();

        Log.d(TAG, "Bundle key:" + key + " ,value:" +
bundle.getString(key));

        }

        super.handleMessage(msg);

        }

};
```

Αντίστοιχα, στο Thread του παρακάτω παραδείγματος, δημιουργείται ένα μήνυμα (Message):

```
Message msg = Message.obtain();

Bundle messageData = new Bundle();

msg.what = 1;
```

```
messageData.putString("text", "This is the content of the message");

msg.setData(messageData);

//send message to activity

activity._handler.sendMessage(msg);
```

3.3. Διεπαφή χρήστη.

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

3.3.1. Layout.

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

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

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

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

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

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

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

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

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

Ακολουθούν δύο παραδείγματα διάταξης, των γραφικών στοιχείων της οθόνης:

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

```
<LinearLayout

android:layout_width="fill_parent"

android:layout_height="fill_parent"

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

<TextView

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

/>

<Button

android:layout_width=" 100px"

android:layout_height="wrap_content"

android:text="Button"

/>

</LinearLayout>
```

Εικόνα 2 – Παράδειγμα κατασκευής ενός LinearLayout

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

<TableLayout
```

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

<TableRow>

<TextView

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

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

</TableRow>

<TableRow>

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

<EditText

android:id="@+id/txtPassword"

android:password="true"

/>

</TableRow>

<TableRow>

<TextView />

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

/>

</TableRow>

<TableRow>

<Button

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

</TableRow>

</TableLayout>
```

Εικόνα 3 – Παράδειγμα κατασκευής ενός TableLayout

3.3.2. Τα μενού.

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

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

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

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

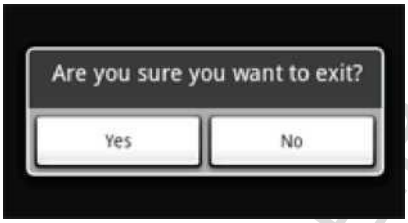
3.3.3. Dialogs.

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

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

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

. setCancelable(false)

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

```
}

})

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

}

});

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



Εικόνα 4 – Παράδειγμα κατασκευής ενός AlertDialog

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

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

3.3.4. Ειδοποιήσεις (Notifications).

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

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

ToastNotification

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

```
Context context - getApplicationContext();

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

int duration = Toast.LENGTH_SHORT;

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

StatusBarNotification

Η Status Bar Notification, όπως φανερώνει και το όνομά της, είναι μία ειδοποίηση η οποία εμφανίζεται στη Status Bar του κινητού τηλεφώνου μας, και την οποία μπορούμε να

ανοίξουμε είτε βρισκόμαστε στο κεντρικό μενού του τηλεφώνου μας, είτε σε κάποια εφαρμογή. Αντίθετα με την Toast, η Status Bar Notification μπορεί να επιλεχθεί και να ξεκινήσει κάποια λειτουργία ανάλογα με τις ενέργειες, που έχουμε ορίσει, στον κώδικα της εφαρμογής. Για παράδειγμα, όταν κατεβάζουμε ένα αρχείο από το διαδίκτυο και η λήψη του έχει ολοκληρωθεί, θα θέλαμε να επιλέξουμε την ειδοποίηση αυτή και, με τον τρόπο αυτό, είτε να ανοίξουμε το φάκελο που βρίσκεται το αρχείο, είτε να το τρέξουμε (Εικόνα 3.10). Τις περισσότερες φορές οι Toast Notifications ενεργοποιούνται από Activities, ενώ οι Status Bar Notifications από Services.

Εικόνα 5 – Παράδειγμα StatusBarNotification

3.3.5. Application Resources & Συμβατότητα.

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

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

3.3.6. Προσανατολισμός.

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

Αντιλαμβάνεται κανείς ότι τα γραφικά στοιχεία της οθόνης θα θέλαμε, κάποιες φορές, να εμφανίζονται με διαφορετικό τρόπο, ανάλογα τον προσανατολισμό της συσκευής. Το λειτουργικό σύστημα Android έχει τη δυνατότητα να επανασχεδιάζει τα γραφικά αυτά στοιχεία, όταν αλλάζουμε προσανατολισμό, και να τα εμφανίζει με τον τρόπο που έχουμε ορίσει εμείς. Ο προεπιλεγμένος τρόπος είναι να παρουσιάζεται το User Interface, με την ίδια διάταξη, και στους δύο προσανατολισμούς. Έστω, λοιπόν, ότι θέλουμε να εμφανίσουμε

τρεις εικόνες σε κάποια οθόνη της εφαρμογής μας, οι οποίες να καλύπτουν όλο το χώρο της οθόνης.

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

3.3.7. Μέγεθος Οθόνης.

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

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

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

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

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

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

3.3.8. Γλώσσα.

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

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

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

Έστω, λοιπόν, ότι θέλουμε προεπιλεγμένη γλώσσα της εφαρμογής μας να είναι η Αγγλική, άρα στο αρχείο res/values/strings.xml δηλώνουμε τα Strings με το όνομά τους και την τιμή τους στην Αγγλική γλώσσα. Για λόγους συμβατότητας, θέλουμε οι Έλληνες χρήστες να χρησιμοποιούν την εφαρμογή μας στην Ελληνική γλώσσα. Δημιουργούμε, λοιπόν, νέο φάκελο res/values-el/ και μέσα στο φάκελο αυτόν δημιουργούμε το αρχείο strings.xml. Χρησιμοποιούμε ακριβώς το ίδιο όνομα για κάθε String, αντίστοιχα με το προηγούμενο αρχείο strings.xml, αλλά τώρα με διαφορετική τιμή, δηλαδή στην ελληνική γλώσσα. Όταν ο χρήστης εγκαταστήσει και εκτελέσει την εφαρμογή, το Android θα εντοπίσει αυτόματα τη γλώσσα που χρησιμοποιεί ο χρήστης στη συσκευή του. Αν, λοιπόν, ο χρήστης χρησιμοποιεί

την Ελληνική γλώσσα τότε το Android, για κάθε String που γίνεται αναφορά στον κώδικα, θα χρησιμοποιεί αυτό που βρίσκεται στο αρχείο strings.xml στο φάκελο res/values- el/. Με τον ίδιο τρόπο, μπορούμε να δημιουργήσουμε φακέλους και για άλλες γλώσσες, όπως: res/values-de/, res/values-fr/ για γερμανική και γαλλική γλώσσα. Σε περίπτωση που δε βρεθεί φάκελος για τη γλώσσα που χρησιμοποιεί ο χρήστης, τότε χρησιμοποιούνται οι τιμές που έχουμε στον προεπιλεγμένο φάκελο res/values/.

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

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

- Σε περίπτωση που η συσκευή χρησιμοποιεί οποιαδήποτε γλώσσα εκτός της Ελληνικής (χρησιμοποιείται το προεπιλεγμένο αρχείο res/values/strings.xml), το οπτικό αποτέλεσμα, της εφαρμογής, θα είναι το εξής:

Εικόνα 6 – Παράδειγμα χρησιμοποίησης Αγγλικής γλώσσας

- Σε περίπτωση που η συσκευή χρησιμοποιεί την Ελληνική γλώσσα (χρησιμοποιείται το αρχείο res/values-el/strings.xml), το οπτικό αποτέλεσμα, της εφαρμογής, θα είναι το εξής:

Εικόνα 7 – Παράδειγμα χρησιμοποίησης Ελληνικής γλώσσας

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

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

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

4. Σχεδιασμός της εφαρμογής

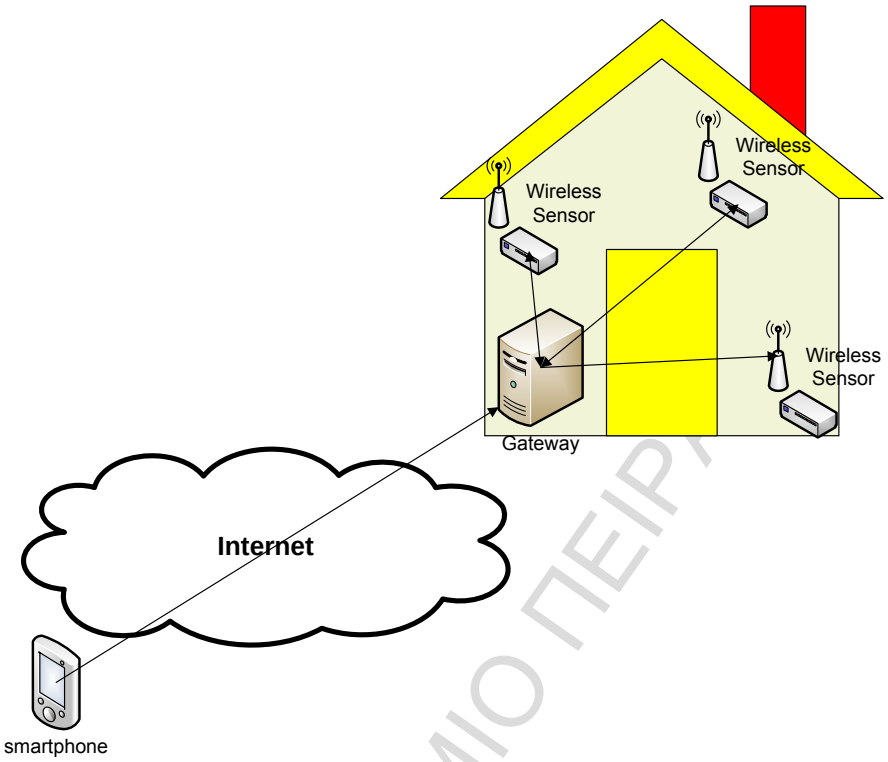
4.1. Λειτουργικές απαιτήσεις

Η εφαρμογή σχεδιάζεται για Smartphones, που τρέχουν Android, και έχει τις παρακάτω λειτουργίες:

- Σύνδεση στον απομακρυσμένο Server με τα στοιχεία του χρήστη.
- Διαχείριση συστήματος συναγερμού:
    - Ενεργοποίηση/απενεργοποίηση συστήματος.
    - Ενεργοποίηση/απενεργοποίηση συναγερμού.
    - Έλεγχος της κατάστασης στις διάφορες ζώνες.
- Διαχείριση κεντρικής θέρμανσης:
    - Ενεργοποίηση/απενεργοποίηση θέρμανσης.
    - Ενεργοποίηση/απενεργοποίηση θέρμανσης με χρονοδιακόπτη (on/off timer).
- Διαχείριση θερμοσίφωνα:
    - Ενεργοποίηση/απενεργοποίηση θερμοσίφωνα.
    - Ενεργοποίηση/απενεργοποίηση θερμοσίφωνα με χρονοδιακόπτη (on/off timer).
- Διαχείριση των παραθύρων και της γκαραζόπορτας.
- Επισκόπηση συστημάτων (Εμφάνιση μηνυμάτων από τις διάφορες απομακρυσμένες συσκευές).

4.2. Αρχιτεκτονική της εφαρμογής

Το έξυπνο σπίτι αποτελείται από διάφορες έξυπνες συσκευές και σένσορες, που συνδέονται ενσύρματα ή ασύρματα σε ένα κεντρικό εξυπηρετητή, που λειτουργεί ως Gateway. Ο χρήστης συνδέεται, μέσω της εφαρμογής για κινητά τηλέφωνα, στο Gateway και μπορεί να διαχειριστεί τις διάφορες έξυπνες συσκευές του σπιτιού.

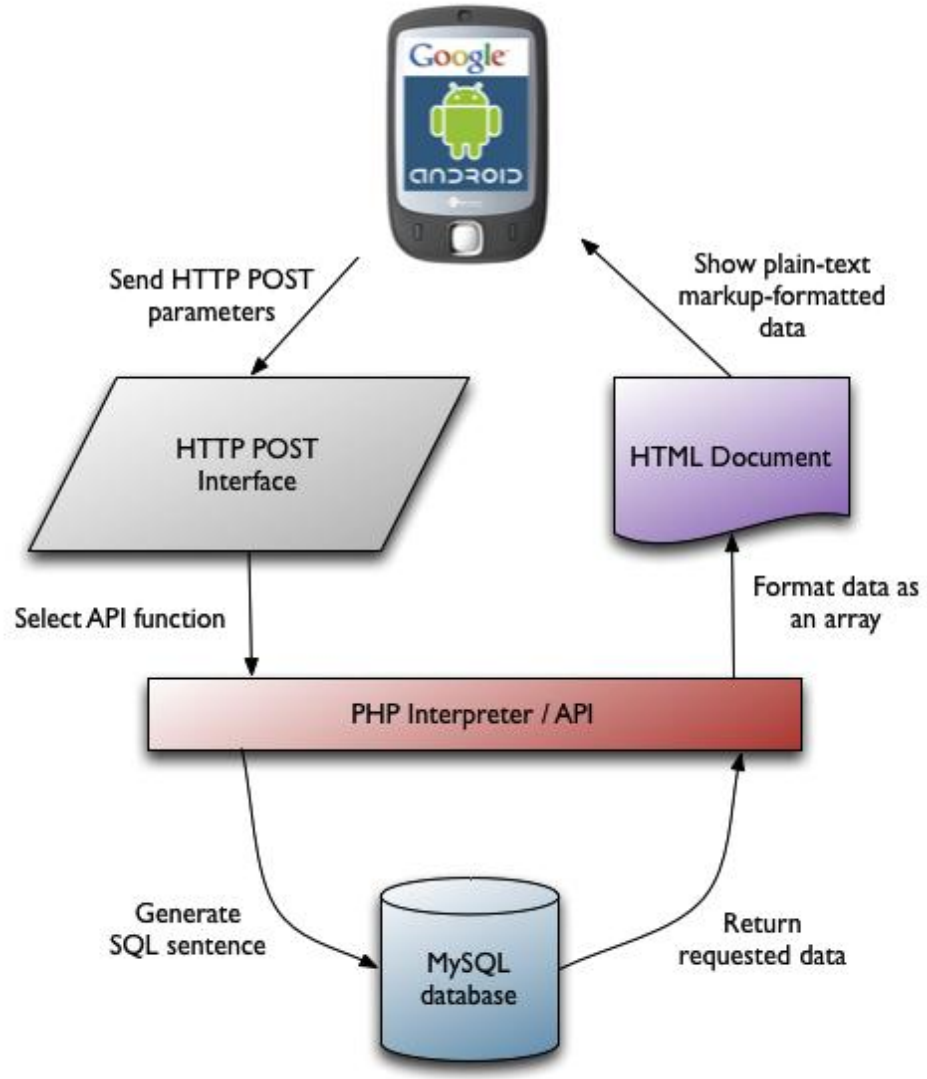


Εικόνα 8 – Αρχιτεκτονική του συστήματος

Πρόκειται για ένα κλασικό μοντέλο, Client-Server, αρχιτεκτονικής. Η αρχιτεκτονική, που επιλέχτηκε, βασίζεται στη χρήση:

- Του πρωτοκόλλου HTTP (REST).
- Του JSON.

Στο παρακάτω σχήμα, φαίνεται το μοντέλο της αρχιτεκτονικής, που χρησιμοποιείται:



Εικόνα 9 – Μοντέλο της αρχιτεκτονικής

Τα βήματα επικοινωνίας, έχουν ως εξής:

- Η συσκευή στέλνει κάποιο αίτημα, μέσω ενός HTTP POST Request. Για λόγους ασφαλείας και ιδιωτικότητας των δεδομένων, μπορεί να χρησιμοποιηθεί το HTTPS. Το HTTPS προσφέρει κρυπτογράφηση, από άκρο σε άκρο, ενώ με τη χρήση πιστοποιητικού στη πλευρά του εξυπηρετητή εξακριβώνεται η ταυτότητά του από τον πελάτη.
- Η εφαρμογή στον εξυπηρετητή είναι μια δυναμική ιστοσελίδα, η οποία λαμβάνει το HTTP POST Request, και με βάση τις παραμέτρους που έχει θέσει ο πελάτης, επικοινωνεί με την τοπική βάση δεδομένων και αποθηκεύει ή λαμβάνει δεδομένα από τη βάση.
- Τα δεδομένα επιστρέφονται, σε ένα HTTP Response.

Τα πλεονεκτήματα, του παραπάνω μοντέλου, είναι τα εξής:

- Το HTTP είναι απλό και θεωρείται standard, ενώ υπάρχουν βιβλιοθήκες για τη χρήση του.
- Η αποθήκευση δεδομένων βρίσκεται σε κεντρικό εξυπηρετητή, συνεπώς ο χρήστης μπορεί να συνδεθεί από διάφορες συσκευές.
- Η αποθήκευση και ο χειρισμός των δεδομένων γίνεται από το σύστημα βάσης δεδομένων και, έτσι, εξασφαλίζεται η καλή κατάσταση τους.
- Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate).
- Το κατανεμημένο σύστημα είναι ανεξάρτητο πλατφόρμας και γλώσσας προγραμματισμού, χάρη στη χρήση του HTTP και του JSON.

4.3. Σχεδιασμός διεπαφής της εφαρμογής

Η εφαρμογή θα αποτελείται από 6 βασικές οθόνες (Android Activities):

- Login (Launcher Activity).
- Main (Εμφανίζει την προεπισκόπηση του συστήματος και έχει το μενού, για την πλοήγηση στα διάφορα συστήματα).
- Alarm System Activity.
- Central Heating Activity.
- Water Heater Activity.
- Windows Activity.

Ακολουθεί, για τα παραπάνω Activities, ο κώδικας των Layout, σε XML, και η προεπισκόπηση του καθενός:

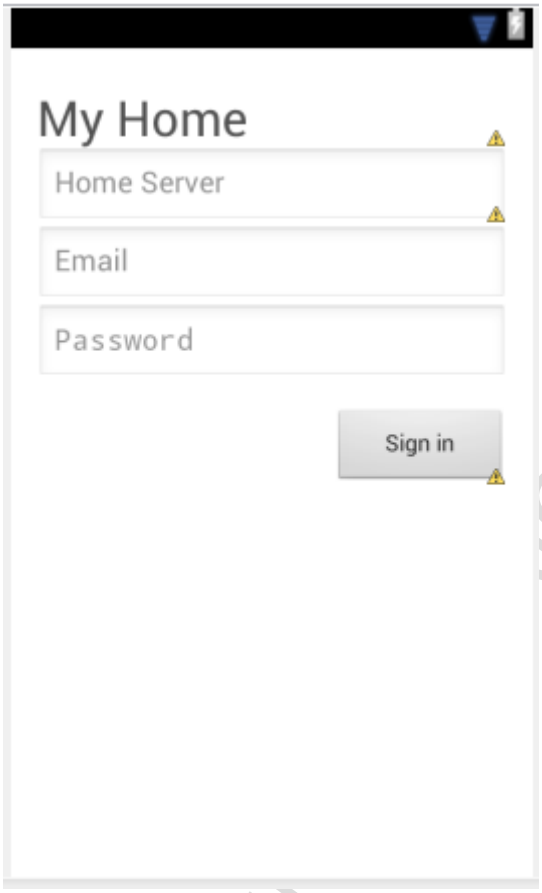
4.3.1. Login Activity

```
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".Login" >

    <!-- Login progress -->

    <LinearLayout
        android:id="@+id/login_status"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:visibility="gone" >

        <ProgressBar
            style="?android:attr/progressBarStyleLarge"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp" />

        <TextView
            android:id="@+id/login_status_message"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            android:fontFamily="sans-serif-light"
            android:text="@string/login_progress_signing_in"

android:textAppearance="?android:attr/textAppearanceMedium" />
    </LinearLayout>

    <!-- Login form -->

    <ScrollView
        android:id="@+id/login_form"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <LinearLayout
            style="@style/LoginFormContainer"
            android:orientation="vertical" >


            <TextView
                  android:layout_centerHorizontal="true"
                  android:layout_marginTop="10dp"
                android:text="My Home"
                android:textSize="30dp"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                />

            <EditText
                android:id="@+id/host"
                android:layout_width="match_parent"
```

```
                android:layout_height="wrap_content"
                android:hint="Home Server"
                android:inputType="text"
                android:maxLines="1"
                android:singleLine="true" />

            <EditText
                android:id="@+id/email"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="@string/prompt_email"
                android:inputType="textEmailAddress"
                android:maxLines="1"
                android:singleLine="true" />

            <EditText
                android:id="@+id/password"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="@string/prompt_password"
                android:imeActionId="@+id/login"
                android:imeActionLabel="@string/action_sign_in_short"
                android:imeOptions="actionUnspecified"
                android:inputType="textPassword"
                android:maxLines="1"
                android:singleLine="true" />

            <Button
                android:id="@+id/sign_in_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="right"
                android:layout_marginTop="16dp"
                android:paddingLeft="32dp"
                android:paddingRight="32dp"
                android:text="Sign in" />
        </LinearLayout>
    </ScrollView>

</merge>
```

Αποτελείται από:

- 3 πεδία, τύπου EditText, για την εισαγωγή της διεύθυνσης (hostname) του εξυπηρετητή και των στοιχείων πρόσβασης του χρήστη (Email, Password).
- 1 πλήκτρο, τύπου Button, για την “μεταφορά” του χρήστη στο μενού πλοήγησης, των διάφορων συστημάτων, της εφαρμογής.



Εικόνα 10 – Αρχική οθόνη (Login)

4.3.2. Main activity

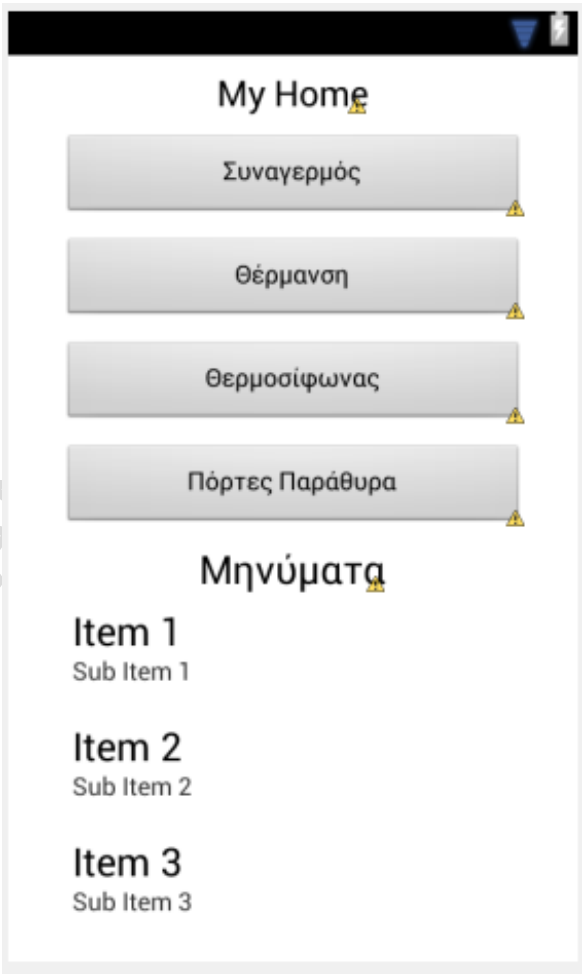
```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    tools:context=".MainActivity" >


     <TextView
           android:id="@+id/title"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_centerHorizontal="true"
           android:layout_marginTop="10dp"
           android:textSize="20dp"
            android:text="My Home"
           android:textAppearance="?android:attr/textAppearanceLarge"
/>
    <Button
```

```
       android:id="@+id/alarmButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"

        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp"
        android:layout_marginTop="10dp"
        android:layout_below="@+id/title"
        android:text="Συναγερμός" />

    <Button
       android:id="@+id/centralHeatingbutton"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_alignParentLeft="true"

       android:layout_marginLeft="30dp"
       android:layout_marginRight="30dp"
       android:layout_marginTop="10dp"
       android:layout_below="@+id/alarmButton"
       android:text="Θέρμανση" />

      <Button
       android:id="@+id/waterHeaterButton"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_alignParentLeft="true"

       android:layout_marginLeft="30dp"
       android:layout_marginRight="30dp"
       android:layout_marginTop="10dp"
       android:layout_below="@+id/centralHeatingbutton"
       android:text="Θερμοσίφωνας" />

      <Button
       android:id="@+id/WindowsButton"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_alignParentLeft="true"

       android:layout_marginLeft="30dp"
       android:layout_marginRight="30dp"
       android:layout_marginTop="10dp"
       android:layout_below="@+id/waterHeaterButton"
       android:text="Πόρτες Παράθυρα" />

       <TextView
          android:id="@+id/messagesLabel"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_below="@+id/WindowsButton"
          android:layout_centerHorizontal="true"
          android:layout_marginTop="10dp"
          android:text="Μηνύματα"
          android:textAppearance="?android:attr/textAppearanceLarge"
/>

      <ListView
          android:id="@+id/messagesListView"
          android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_below="@+id/messagesLabel"
            android:layout_centerHorizontal="true"

            android:layout_marginLeft="30dp"
            android:layout_marginRight="30dp" >

        </ListView>

    </RelativeLayout>
```

Αποτελείται από:

- 4 πλήκτρα, τύπου Button, που οδηγούν σε Activity για τη διαχείριση συγκεκριμένου συστήματος.
- 1 πεδίο, τύπου ListView, στο οποίο εμφανίζονται τα μηνύματα από τα διάφορα συστήματα.



Εικόνα 11 – Κεντρική οθόνη

4.3.3. Διαχείριση συναγερμού

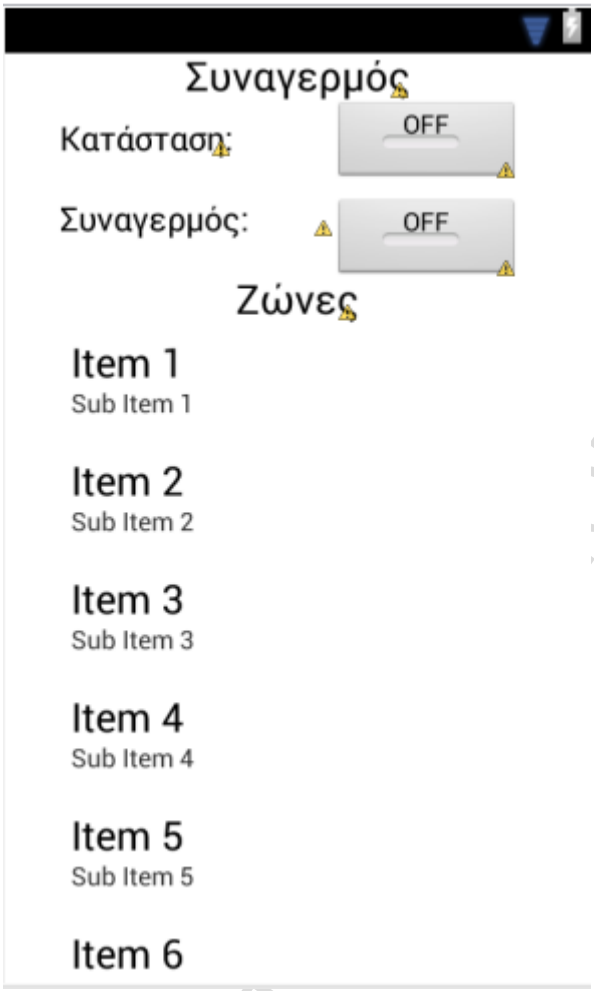
```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AlarmActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Συναγερμός"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/StatustextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="10dp"
        android:text="Κατάσταση"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

    <TextView
        android:id="@+id/zonesTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/AlarmToggleButton"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:text="Ζώνες"
        android:textAppearance="?android:attr/textAppearanceLarge" />

     <ListView
           android:id="@+id/ZonesListView"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:layout_below="@+id/zonesTextView"
           android:layout_centerHorizontal="true"
            android:layout_marginRight="30dp"
           android:layout_marginLeft="30dp"
      >
      </ListView>

      <TextView
        android:id="@+id/AlarmStatustextView"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/toggleButton"
        android:layout_marginLeft="30dp"
```

```
        android:layout_marginTop="10dp"
        android:text="Συναγερμός:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

     <ToggleButton
         android:id="@+id/AlarmToggleButton"
         android:layout_width="100dp"
         android:layout_height="wrap_content"
         android:layout_alignBaseline="@+id/AlarmStatustextView"
         android:layout_alignBottom="@+id/AlarmStatustextView"
         android:layout_toRightOf="@+id/AlarmStatustextView"
         android:text="ToggleButton" />

     <ToggleButton
         android:id="@+id/toggleButton"
         android:layout_width="100dp"
         android:layout_height="wrap_content"
         android:layout_alignRight="@+id/AlarmToggleButton"
         android:layout_below="@+id/textView1"
         android:text="ToggleButton" />

</RelativeLayout>
```

Αποτελείται από:

- 2 πλήκτρα, τύπου toggle (ON/OFF) Button, για την ενεργοποίηση:
    - Του συστήματος συναγερμού.
    - Του συναγερμού (σειρήνα).
- 1 πεδίο, τύπου ListView, που παρουσιάζει την κατάσταση των διάφορων ζωνών του συστήματος.

Μια ζώνη είναι μια ομάδα από έναν ή περισσότερους αισθητήρες. Μπορεί να περιέχει ασύρματους και ενσύρματους αισθητήρες κίνησης, παγίδες για τις πόρτες και τα παράθυρα, αισθητήρες καπνού και άλλα. Επιλέγοντας κάποιο αντικείμενο στο ListView, εμφανίζεται διάλογος με πληροφορίες για τη ζώνη και την κατάστασή της.



Εικόνα 12 – Σχεδιασμός οθόνης διαχείρισης συναγερμού

4.3.4. Διαχείριση κεντρικής θέρμανσης

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".CentralHeating" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Κεντρική θέρμανση"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
```

```
        android:id="@+id/StatustextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="10dp"
        android:text="Κατάσταση:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

    <TextView
        android:id="@+id/TimerStatustextView"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/toggleButton"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="10dp"
        android:text="Χρονοδιακόπτης:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

    <ToggleButton
        android:id="@+id/TimerToggleButton"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/TimerStatustextView"
        android:layout_alignBottom="@+id/TimerStatustextView"
        android:layout_toRightOf="@+id/TimerStatustextView"
        android:text="ToggleButton" />

    <ToggleButton
        android:id="@+id/toggleButton"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/TimerToggleButton"
        android:layout_below="@+id/textView1"
        android:text="ToggleButton" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/TimerStatustextView"
        android:layout_below="@+id/TimerToggleButton"
        android:layout_marginTop="90dp"
        android:text="ON:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

    <TimePicker
        android:id="@+id/timePicker1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/TimerToggleButton"
        android:layout_below="@+id/TimerToggleButton"
        android:layout_marginTop="31dp" />

    <TimePicker
        android:id="@+id/timePicker2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/timePicker1"
        android:layout_below="@+id/timePicker1"
        android:layout_marginTop="33dp" />

    <TextView
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/timePicker2"
        android:layout_alignRight="@+id/textView2"
        android:layout_marginBottom="54dp"
        android:text="OFF:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

</RelativeLayout>
```

Αποτελείται από:

- 2 πλήκτρα, τύπου toggle (ON/OFF) Button, για την ενεργοποίηση:
    - Του συστήματος της κεντρικής θέρμανσης.
    - Του χρονοδιακόπτη.

Αν ο χρονοδιακόπτης είναι ενεργός, τότε εμφανίζονται επιλογές για τη ρύθμιση έναρξης και τερματισμού του συστήματος της κεντρικής θέρμανσης (Timepicker).

Εικόνα 13 – Σχεδιασμός οθόνης διαχείρισης κεντρικής θέρμανσης

4.3.5. Διαχείριση θερμοσίφωνα

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".WaterHeaterActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Θερμοσίφωνας"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
```

```
        android:id="@+id/StatustextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="10dp"
        android:text="Κατάσταση:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

      <TextView
        android:id="@+id/TimerStatustextView"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/toggleButton"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="10dp"
        android:text="Χρονοδιακόπτης:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

     <ToggleButton
         android:id="@+id/TimerToggleButton"
         android:layout_width="100dp"
         android:layout_height="wrap_content"
         android:layout_alignBaseline="@+id/TimerStatustextView"
         android:layout_alignBottom="@+id/TimerStatustextView"
         android:layout_toRightOf="@+id/TimerStatustextView"
         android:text="ToggleButton" />

     <ToggleButton
         android:id="@+id/toggleButton"
         android:layout_width="100dp"
         android:layout_height="wrap_content"
         android:layout_alignRight="@+id/TimerToggleButton"
         android:layout_below="@+id/textView1"
         android:text="ToggleButton" />

     <TimePicker
         android:id="@+id/timePicker2"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/timePicker1"
          android:layout_below="@+id/timePicker1"
         android:layout_alignParentBottom="true" />

     <TextView
         android:id="@+id/TextView01"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignTop="@+id/timePicker2"
         android:layout_marginTop="60dp"
         android:layout_toLeftOf="@+id/timePicker2"
         android:text="OFF:"
         android:textAppearance="?android:attr/textAppearanceMedium"
/>

     <TimePicker
         android:id="@+id/timePicker1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/TimerToggleButton"
        android:layout_marginTop="20dp"
        android:layout_toRightOf="@+id/textView2" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/timePicker1"
        android:layout_alignLeft="@+id/TimerStatustextView"
        android:layout_marginBottom="46dp"
        android:text="ON:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>

</RelativeLayout>
```

Έχει αντίστοιχη λειτουργικότητα με την θέρμανση. Αποτελείται από:

- 2 πλήκτρα, τύπου toggle (ON/OFF) Button, για την ενεργοποίηση:
    - Του συστήματος του θερμοσίφωνα.
    - Του χρονοδιακόπτη.

Αν ο χρονοδιακόπτης είναι ενεργός, τότε εμφανίζονται επιλογές για τη ρύθμιση, έναρξης και τερματισμού, του συστήματος του θερμοσίφωνα (Timepicker).

Εικόνα 14 – Σχεδιασμός οθόνης διαχείρισης θερμοσίφωνα

4.3.6. Διαχείριση παραθύρων/πορτών

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AlarmActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Πόρτες / Παράθυρα"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
```

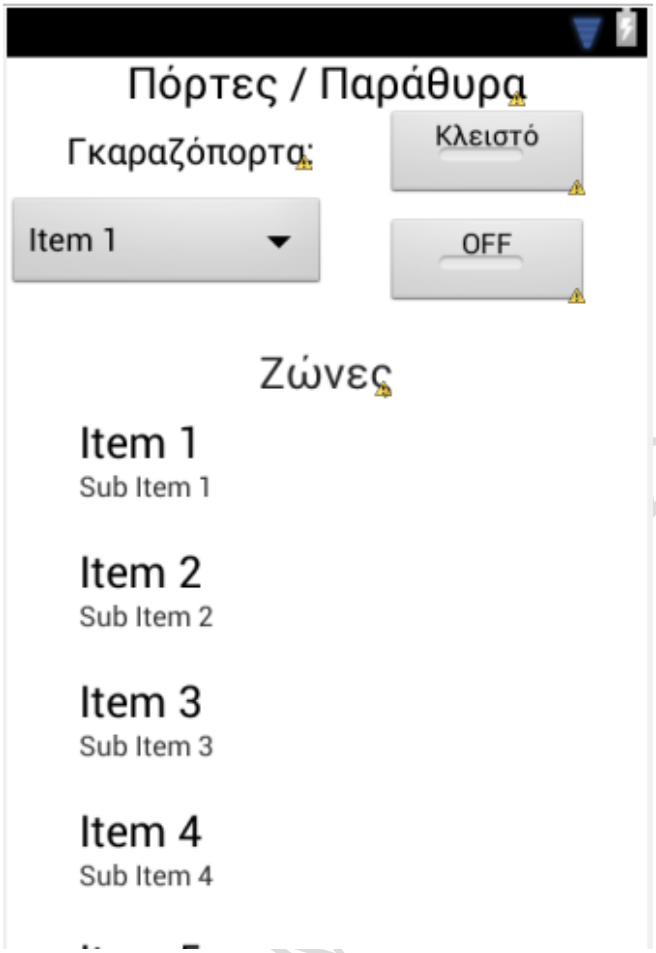
```
        android:id="@+id/ParkingtextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="10dp"
        android:text="Γκαραζόπορτα:"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>


     <ToggleButton
        android:id="@+id/parkingToggleButton"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
        android:layout_marginRight="30dp"
            android:layout_below="@+id/textView1"
        android:text="ToggleButton"
        android:textOff="@string/toggle_closed"
        android:textOn="@string/toggle_open" />


      <ToggleButton
        android:id="@+id/BlindsToggleButton"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_marginRight="30dp"
        android:layout_marginTop="10dp"
        android:layout_below="@+id/parkingToggleButton"
        android:text="ToggleButton" />


       <Spinner
        android:id="@+id/blindsSpinner"
        android:layout_below="@+id/ParkingtextView"
        android:layout_marginTop="10dp"
        android:layout_marginRight="30dp"
        android:layout_toLeftOf="@+id/BlindsToggleButton"
        android:layout_height="50dp"
        android:layout_width="200dp"
       >

    </Spinner>

    <TextView
        android:id="@+id/zonesTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:layout_below="@+id/BlindsToggleButton"
        android:text="Ζώνες"
        android:textAppearance="?android:attr/textAppearanceLarge" />


    <ListView
        android:id="@+id/ZonesListView"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@+id/zonesTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp" >
    </ListView>

         -->

</RelativeLayout>
```

Στην οθόνη αυτή, μας ενδιαφέρει να έχουμε πρόσβαση στην γκαραζόπορτα, αλλά και στα ρολά των παραθύρων.

Αποτελείται από:

- 1 πλήκτρο, τύπου toggle (ON/OFF) Button, για το άνοιγμα ή το κλείσιμο της πόρτας του γκαράζ.
- 1 λίστα πολλαπλών επιλογών, τύπου Spinner, για την επιλογή του παραθύρου, στο οποίο ο χρήστης θέλει να ανοίξει ή να κλείσει το ρολό.
- 1 λίστα με τις ζώνες, τύπου ListView, που αντιστοιχούν σε ρολά, ώστε να ξέρει ο χρήστης ποια ρολά είναι ανοιχτά.



Εικόνα 15 – Οθόνη διαχείρισης γκαραζόπορτας και ρολών παραθύρων

4.4. Τεχνολογίες υλοποίησης

Οι τεχνολογίες, που χρησιμοποιήθηκαν για την ανάπτυξη του κατανεμημένου συστήματος, είναι οι ακόλουθες:

- Η εφαρμογή, του κινητού, βασίζεται στο Android SDK, ενώ η ανάπτυξη έγινε σε περιβάλλον Eclipse. Η εφαρμογή δοκιμάστηκε στον Android Emulator, που προσφέρει το Android SDK.
- Η ανάπτυξη έγινε με Android Minimum SDK Version = "8" και Android Target SDK Version = "16".
- Για την αποστολή και λήψη του HTTP, χρησιμοποιήθηκε η βιβλιοθήκη HTTP APACHE Components. Η επικοινωνία, ανάμεσα στην εφαρμογή Android και τον Smart Home

Gateway, γίνεται πάνω από το HTTP. Προαιρετικά, για λόγους ασφάλειας, μπορεί να χρησιμοποιηθεί το HTTPS (HTTP over SSL).

- Για την ανάλυση των κειμένων, που είναι κωδικοποιημένα σε μορφή JSON από την εφαρμογή κινητού, χρησιμοποιήθηκε το επίσημο API για Java.
- Για την εφαρμογή του εξυπηρετητή χρησιμοποιήθηκε η γλώσσα PHP, η οποία υποστηρίζεται από την πλειοψηφία των παρόχων δικτυακών τόπων και είναι εξαιρετικά διαδεδομένη.
- Για το σύστημα αποθήκευσης δεδομένων χρησιμοποιήθηκε η MySQL, το πιο διαδεδομένο ελεύθερο σύστημα διαχείρισης βάσεων δεδομένων.

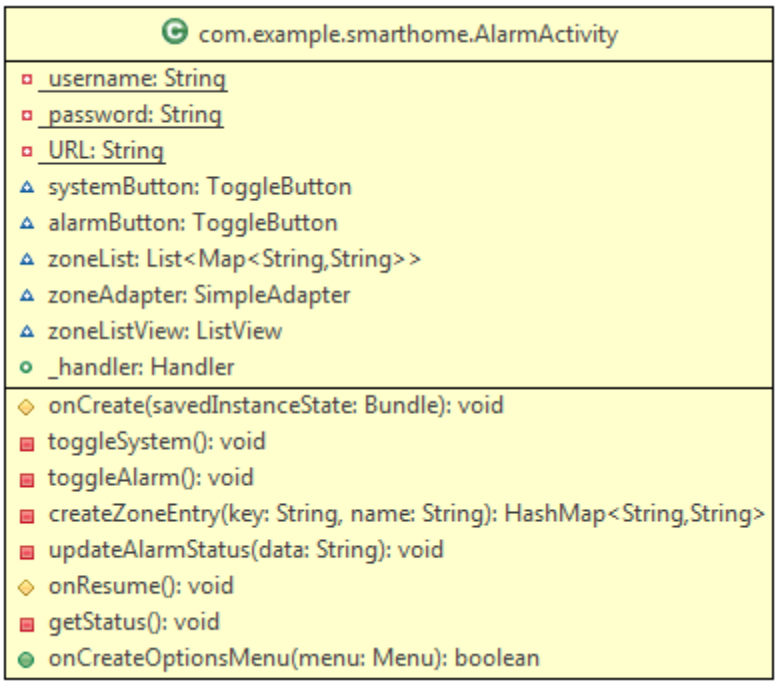
5. Υλοποίηση του Συστήματος

5.1. Υλοποίηση της εφαρμογής Android

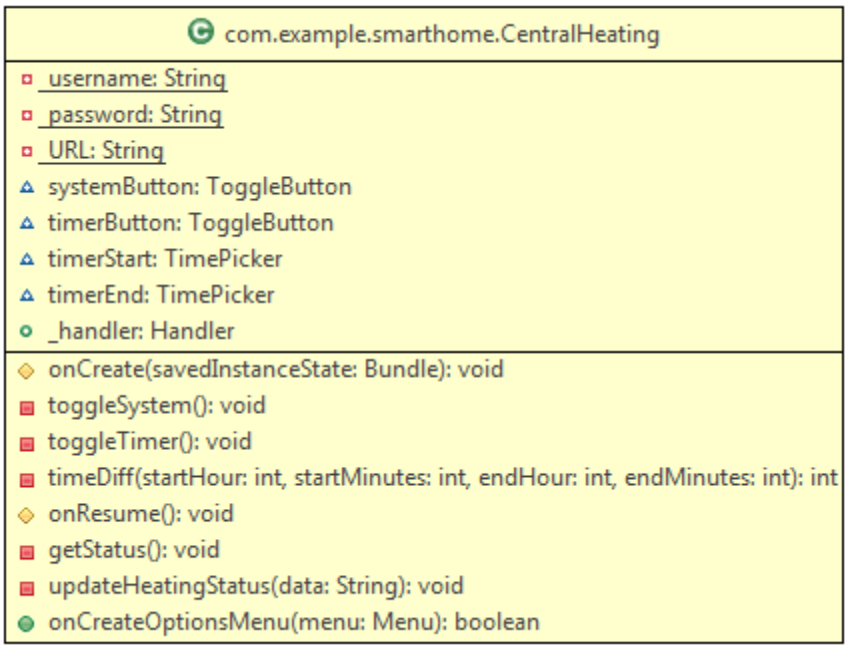
Η εφαρμογή Android αποτελείται από τις, παρακάτω, κλάσεις:

- Login: Αρχική οθόνη, για τη σύνδεση του χρήστη στο Smart Home Gateway.
- MainActivity: Κεντρική οθόνη, που εμφανίζει μια λίστα με μηνύματα και χρησιμεύει ως μενού, για τη σύνδεση στις επιμέρους υπηρεσίες.
- AlarmActivity: Χειρισμός του συναγερμού.
- CentralHeating: Χειρισμός της κεντρικής θέρμανσης.
- WaterHeaterActivity: Χειρισμός του θερμοσίφωνα.
- Windows: Χειρισμός των ρολών και της γκαραζόπορτας.
- HttpPostThread: Κλάση/Thread για την επικοινωνία με τον Gateway.

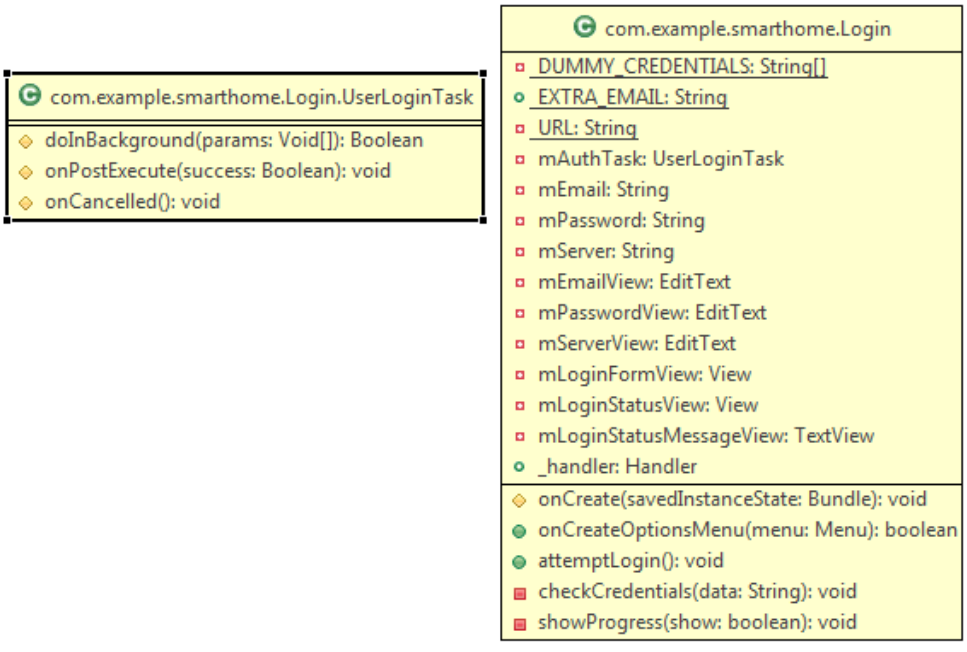
Ακολουθεί η δομή, των παραπάνω κλάσεων, σε διαγράμματα UML:



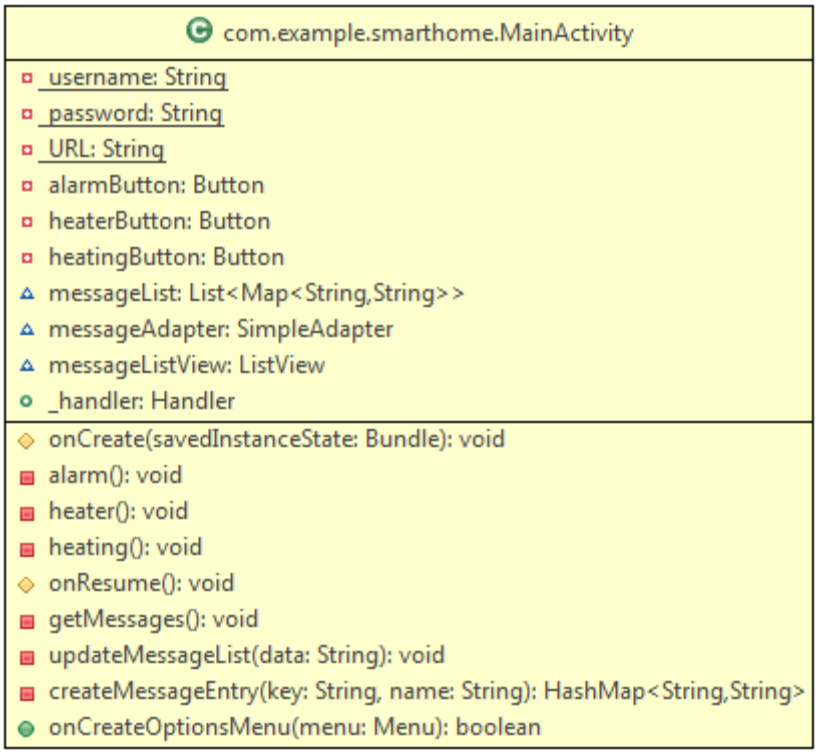
Εικόνα 16 – Διάγραμμα UML: AlarmActivity



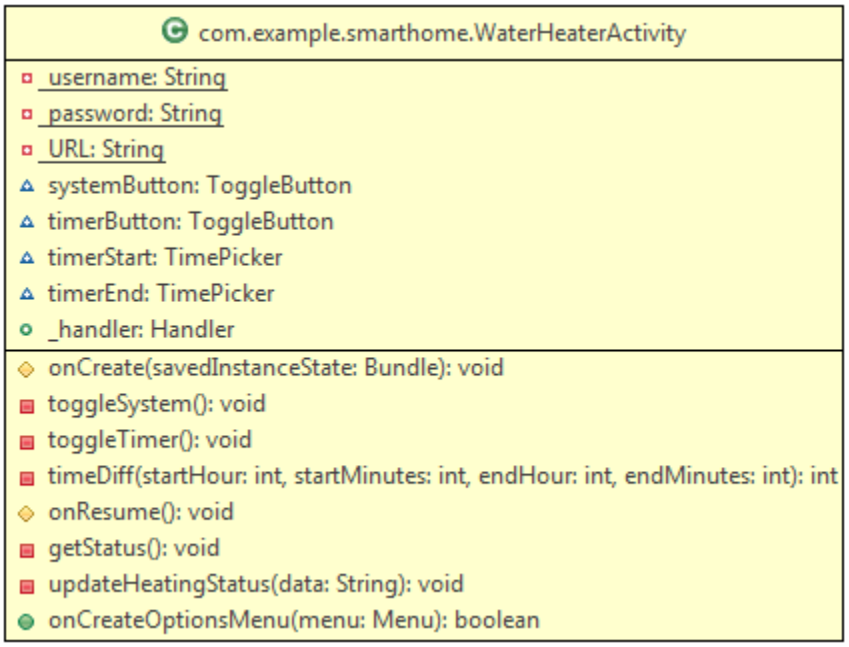
Εικόνα 17 – Διάγραμμα UML: CentralHeating



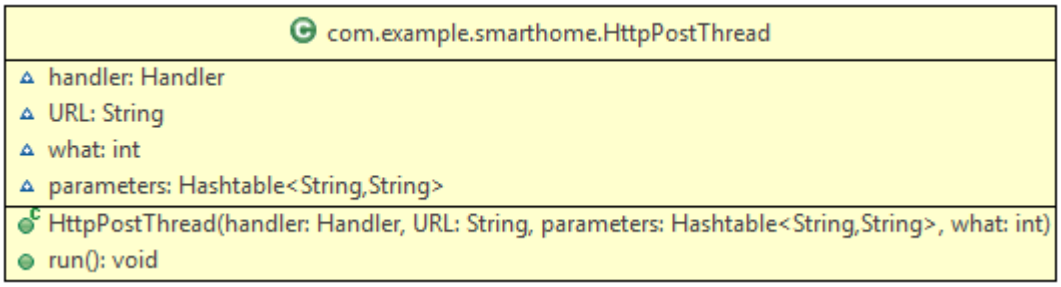
Εικόνα 18 – Διάγραμμα UML: Login



Εικόνα 19 – Διάγραμμα UML: MainActivity



Εικόνα 20 – Διάγραμμα UML: WaterHeaterActivity



Εικόνα 21 – Διάγραμμα UML: HttpPostThread

Χρησιμοποιείται, επίσης, η βιβλιοθήκη json.org, για την αποκωδικοποίηση των μηνυμάτων από τον Gateway.

5.2. Παραμετροποίηση εφαρμογής

Η παραμετροποίηση της εφαρμογής, βρίσκεται στο αρχείο AndroidManifest.xml. Γίνεται χρήση permission, για την πρόσβαση στο δίκτυο.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myhome"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
         android:theme="@android:style/Theme.Light.NoTitleBar"
         >
        <activity
            android:name="com.example.myhome.Login"
            android:label="@string/app_name"
            android:windowSoftInputMode="adjustResize|stateVisible" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```
        <activity
            android:name="com.example.myhome.MainActivity"
            android:label="@string/title_activity_main" >
        </activity>
        <activity
            android:name="com.example.myhome.AlarmActivity"
            android:label="@string/title_activity_alarm" >
        </activity>
        <activity
            android:name="com.example.myhome.WaterHeaterActivity"
            android:label="@string/title_activity_water_heater" >
        </activity>
        <activity
            android:name="com.example.myhome.CentralHeating"
            android:label="@string/title_activity_central_heating" >
        </activity>
        <activity
            android:name="com.example.myhome.Windows"
            android:label="@string/title_activity_windows" >

        </activity>
    </application>

</manifest>
```

5.3. Επικοινωνία με τον εξυπηρετητή

Στη συνέχεια, με ένα παράδειγμα, περιγράφουμε τον τρόπο επικοινωνίας με τον εξυπηρετητή. Θα εξετάσουμε την περίπτωση ενεργοποίησης της κεντρικής θέρμανσης.

Τα βήματα, είναι τα εξής:

- Ο χρήστης επιλέγει το πλήκτρο ενεργοποίησης/απενεργοποίησης, με όνομα systemButton.
- Ενεργοποιείται ο OnClickListener, που έχει οριστεί, και καλείται η μέθοδος toggleSystem().
- Η toggleSystem δημιουργεί ένα νέο HTTP POST THREAD, το οποίο τρέχει στο παρασκήνιο, και κάνει ένα HTTP POST REQUEST, στο Gateway. Περνάνε, ως παράμετροι: το username/password, για την ταυτοποίηση του χρήστη, και η παράμετρος type="toggleHeating".
- Ο Gateway επιστρέφει ένα HTTP RESPONSE, με ένα κείμενο, σε κωδικοποίηση JSON. Το κείμενο αυτό επιστρέφεται στο Activity μέσω του Handler, που έχει οριστεί στην CentralHeatingActivity.

- Ο Handler ανοίγει το μήνυμα. Στη μεταβλητή data, έχει αποθηκευτεί το κείμενο σε μορφή JSON. Καλείται η μέθοδος updateHeatingStatus, για την αποκωδικοποίηση του μηνύματος.
- Η updateHeatingStatus αποκωδικοποιεί το JSON, το οποίο περιέχει τη κατάσταση της κεντρικής θέρμανσης. Ενημερώνεται η κατάσταση των πλήκτρων ενεργοποίησης της θέρμανσης και του Timer, με βάση τα περιεχόμενα του κειμένου σε JSON.

5.4. Υλοποίηση Gateway

Η υλοποίηση του εξυπηρετητή είναι σχετικά απλή και βασίζεται, μονάχα, σε δυο αρχεία:

- service.php
- database.php

Το αρχείο database.php περιέχει τη σύνδεση με τη βάση δεδομένων, καθώς και συναρτήσεις για την τροποποίηση των δεδομένων. Πρακτικά, η βάση δεδομένων χρησιμοποιείται για να κρατά την κατάσταση των διάφορων συσκευών. Η ενημέρωση της βάσης δεδομένων, από τις συσκευές, δεν αποτελεί μέρος αυτής της διπλωματικής.

Στη συνέχεια, ακολουθεί ένα παράδειγμα μιας συνάρτησης για την επικοινωνία της με τη βάση δεδομένων. Η συνάρτηση φορτώνει τις εγγραφές, από τη βάση δεδομένων, σε μια PHP array και τις μετατρέπει σε κωδικοποίηση JSON, μέσω της συνάρτησης json_encode.

```
function getHeaterStatus() {

   $row = array();

$query = "SELECT * from heater ";
$res = mysql_query($query);
```

```
    // iterate over every row
if ($row = mysql_fetch_assoc($res)) {
        // for every field in the result..
for ($i=0; $i < mysql_num_fields($res); $i++) {
            $info = mysql_fetch_field($res, $i);
            $type = $info->type;
        // cast for real
if ($type == 'real')
                $row[$info->name] = doubleval($row[$info->name]);
            // cast for int
if ($type == 'int')
                $row[$info->name] = intval($row[$info->name]);
        }
    }
    // JSON-ify all rows together as one big array
echojson_encode($row, JSON_UNESCAPED_UNICODE);

}
```

Η δομή του service.php, περιγράφεται στη συνέχεια. Φορτώνεται το αρχείο database.php, που δημιουργεί τη σύνδεση με τη βάση δεδομένων και έχει τις διάφορες συναρτήσεις.

```
<?php
include("database.php");
```

Στη συνέχεια, λαμβάνονται, από τις POST παραμέτρους, το username και το password.

```
$username="";
$password="";
if(isset($_POST["username"]))
```

```
        $username= $_POST["username"];


if(isset($_POST["password"]))
        $password = $_POST["password"];
```

Έπειτα, εξετάζονται οι παρακάτω περιπτώσεις:

- Αν υπάρχει παράμετρος email, τότε γίνεται κλήση για σύνδεση στο σύστημα (από το LoginActivity) της εφαρμογής του κινητού.

```
if(isset($_POST["email"])){
        signin($_POST["email"],$password);
}
```

- Διαφορετικά, αν, δηλαδή, το username και password είναι αποδεκτά.

```
else if(login($username,$password)){
```

Κατόπιν, ελέγχεται η παράμετρος type για τις εξής περιπτώσεις:

- messages: Εμφάνιση των δέκα πιο πρόσφατων μηνυμάτων, προς τον χρήστη.

```
        $type = $_POST["type"];
                if(isset($type) && $type=='messages' ){
                getMessages();
        }
```

- alarm_status: Έλεγχος της κατάστασης του συναγερμού.

```
        else if(isset($type) && $type=='alarm_status' ){
```

```
        getAlarmStatus();
    }
```

- heating_status: Έλεγχος της κατάστασης της κεντρικής θέρμανσης.

```
    else if(isset($type) && $type=='heating_status' ){
        getHeatingStatus();
    }
```

- heater_status: Έλεγχος της κατάστασης του θερμοσίφωνα.

```
    else if(isset($type) && $type=='heater_status' ){
        getHeaterStatus();
    }
```

- toggleAlarm: Ενεργοποίηση/Απενεργοποίηση του συναγερμού (σειρήνα).

```
    else if(isset($type) && $type=='toggleAlarm' ){
        toggleAlarmStatus();
    }
```

- 'toggleAlarmSystem': Ενεργοποίηση/Απενεργοποίηση του συστήματος συναγερμού.

```
    else if(isset($type) && $type=='toggleAlarmSystem' ){
        toggleAlarmSystemStatus();
    }
```

- 'toggleHeating': Ενεργοποίηση/Απενεργοποίηση της θέρμανσης.

```
        else if(isset($type) && $type=='toggleHeating' ){

                toggleHeatingStatus();

        }
```

- 'toggleHeatingTimer': Ενεργοποίηση/Απενεργοποίηση του timer της θέρμανσης.

```
        else if(isset($type) && $type=='toggleHeatingTimer' ){

        toggleHeatingTimerStatus($_POST['startHour'],$_POST['startMin
utes'],$_POST['endHour'],$_POST['endMinutes']);

        }
```

- 'toggleHeater': Ενεργοποίηση/Απενεργοποίηση του θερμοσίφωνα.

```
        else if(isset($type) && $type=='toggleHeater' ){

                toggleHeaterStatus();

        }
```

- 'toggleHeaterTimer': Ενεργοποίηση/Απενεργοποίηση του Timer του θερμοσίφωνα.

```
        else if(isset($type) && $type=='toggleHeaterTimer' ){


        toggleHeaterTimerStatus($_POST['startHour'],$_POST['startMinu
tes'],$_POST['endHour'],$_POST['endMinutes']);

        }
```

- 'windows_status ': Η κατάσταση της γκαραζόπορτας και των ρολών.

```
        else if(isset($type) && $type=='windows_status' ){

                getWindowsStatus();}
```

- 'toggleParking': Άνοιγμα/Κλείσιμο της γκαραζόπορτας.

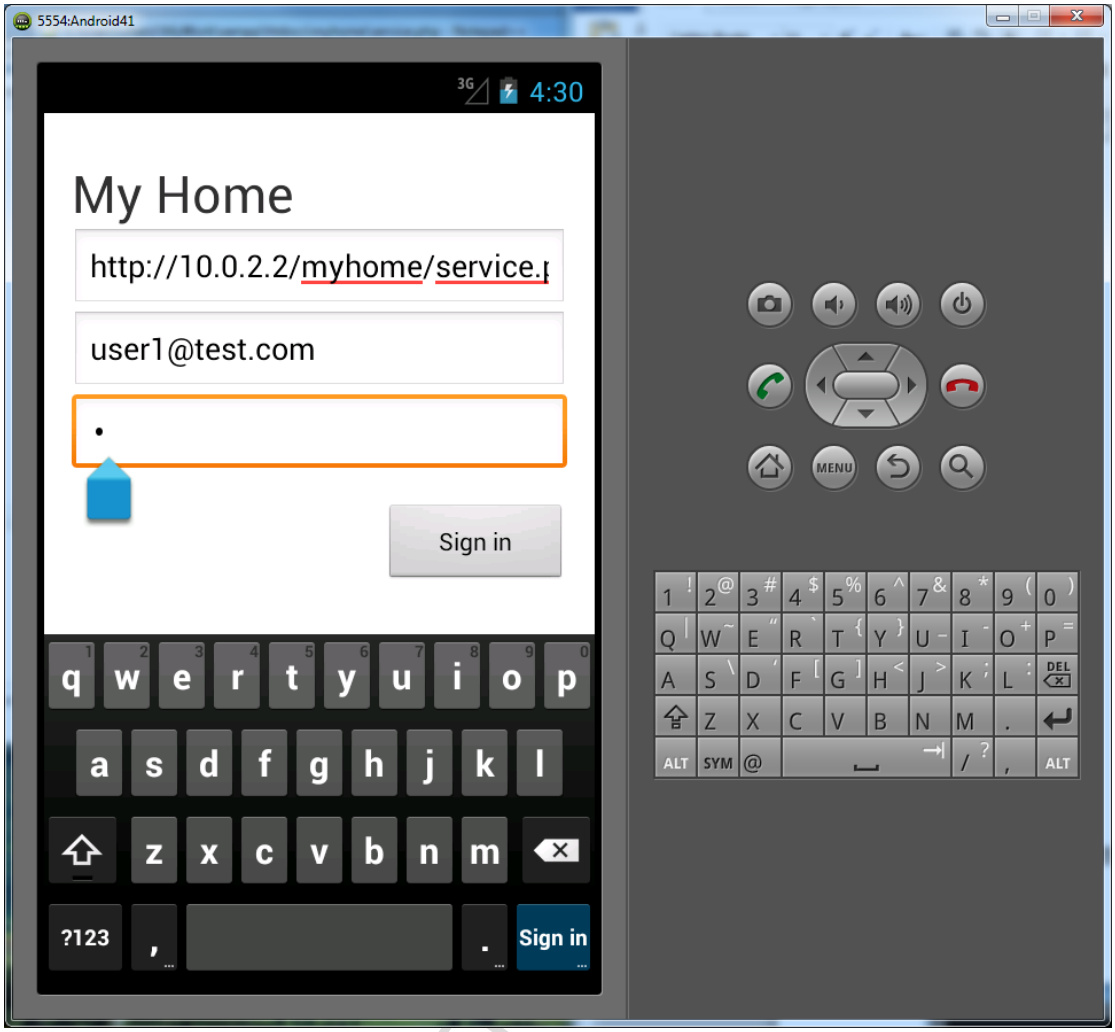
```
else if(isset($type) && $type=='toggleParking' ){
        toggleParkingStatus();
}
```

- 'toggleBlinds': Άνοιγμα/Κλείσιμο του ρολού με αναγνωριστικό ίσο με zoneID.

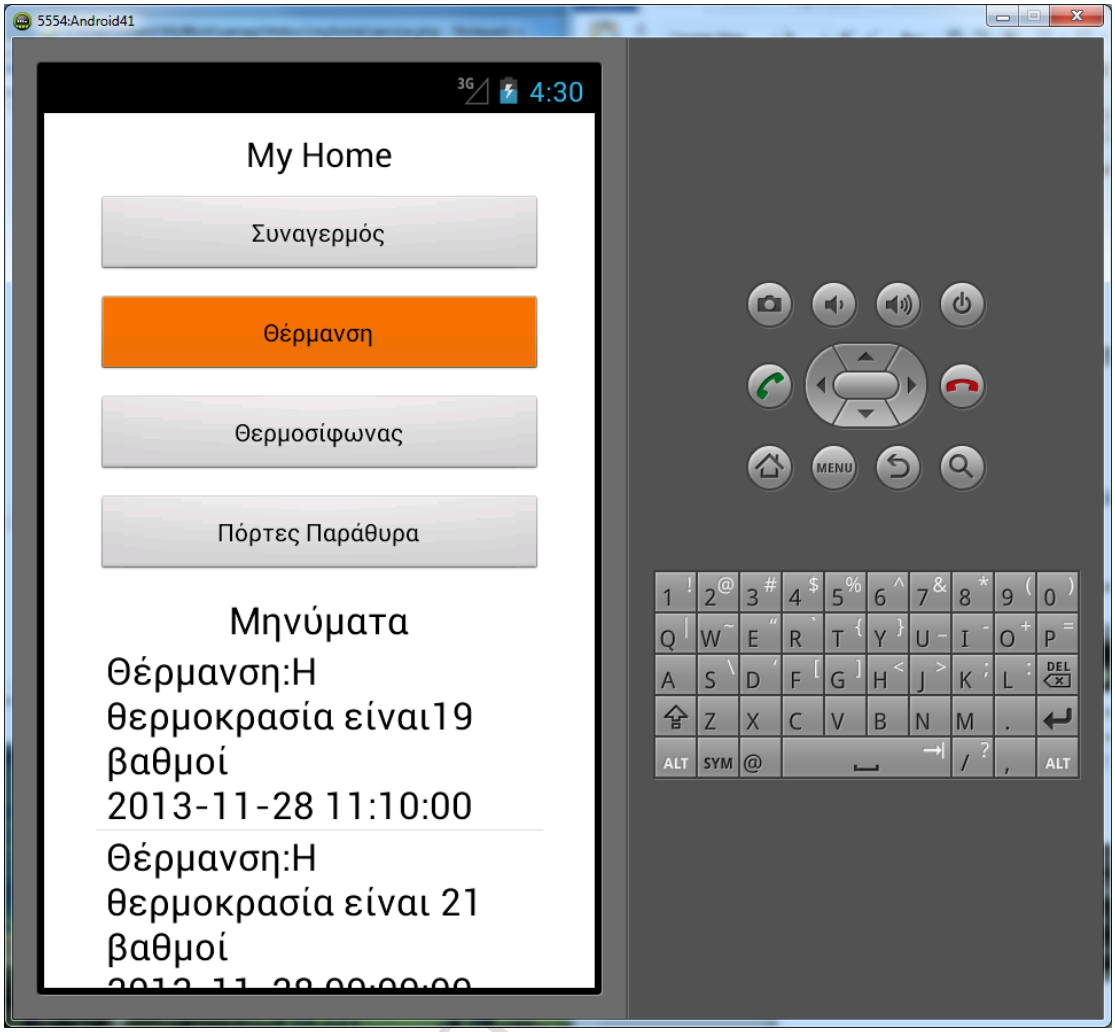
```
else if(isset($type) && $type=='toggleBlinds' ){
        toggleBlindsStatus($_POST['zoneID']);
}
```

5.5. Παραδείγματα Λειτουργίας

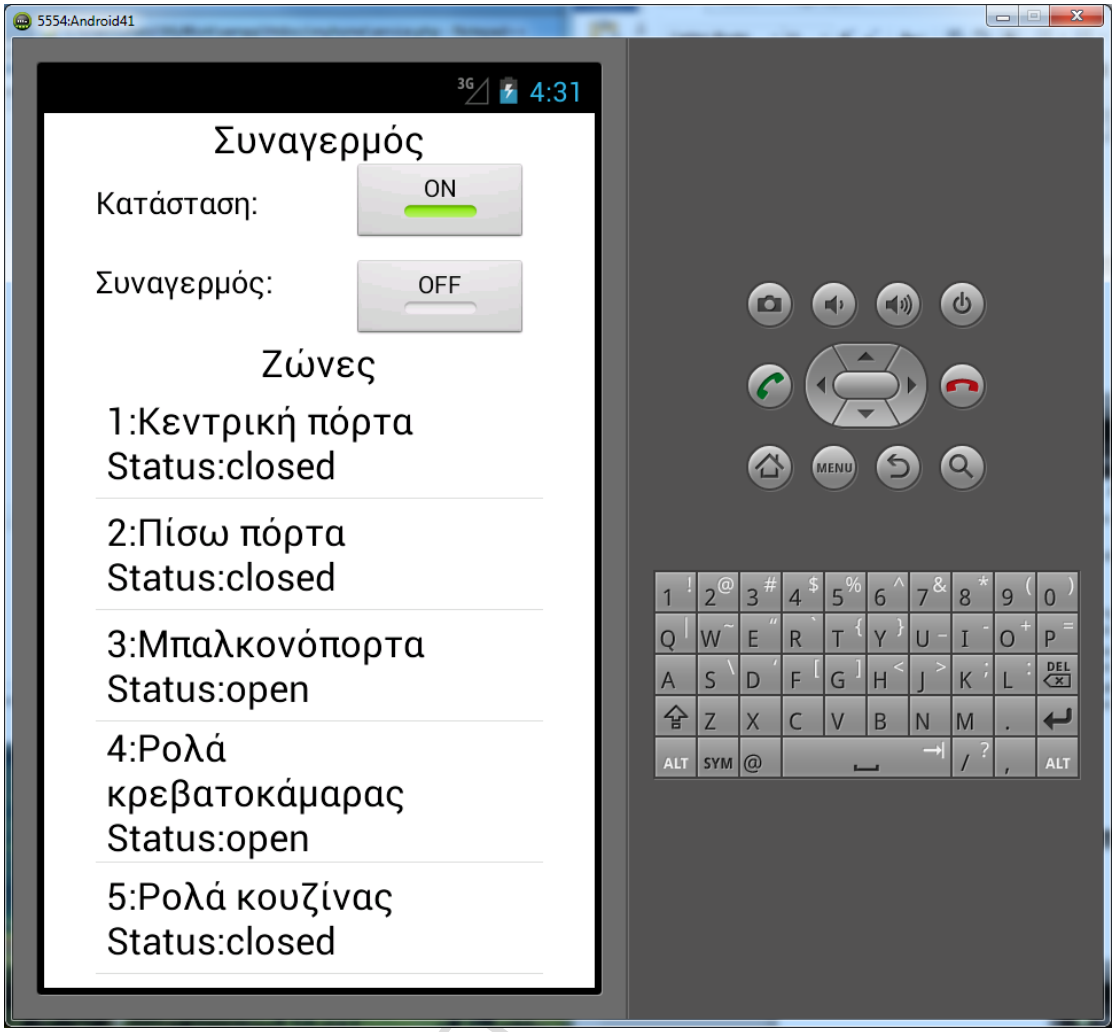
Στην ενότητα αυτή, φαίνονται τα στιγμιότυπα λειτουργίας της εφαρμογής Android.



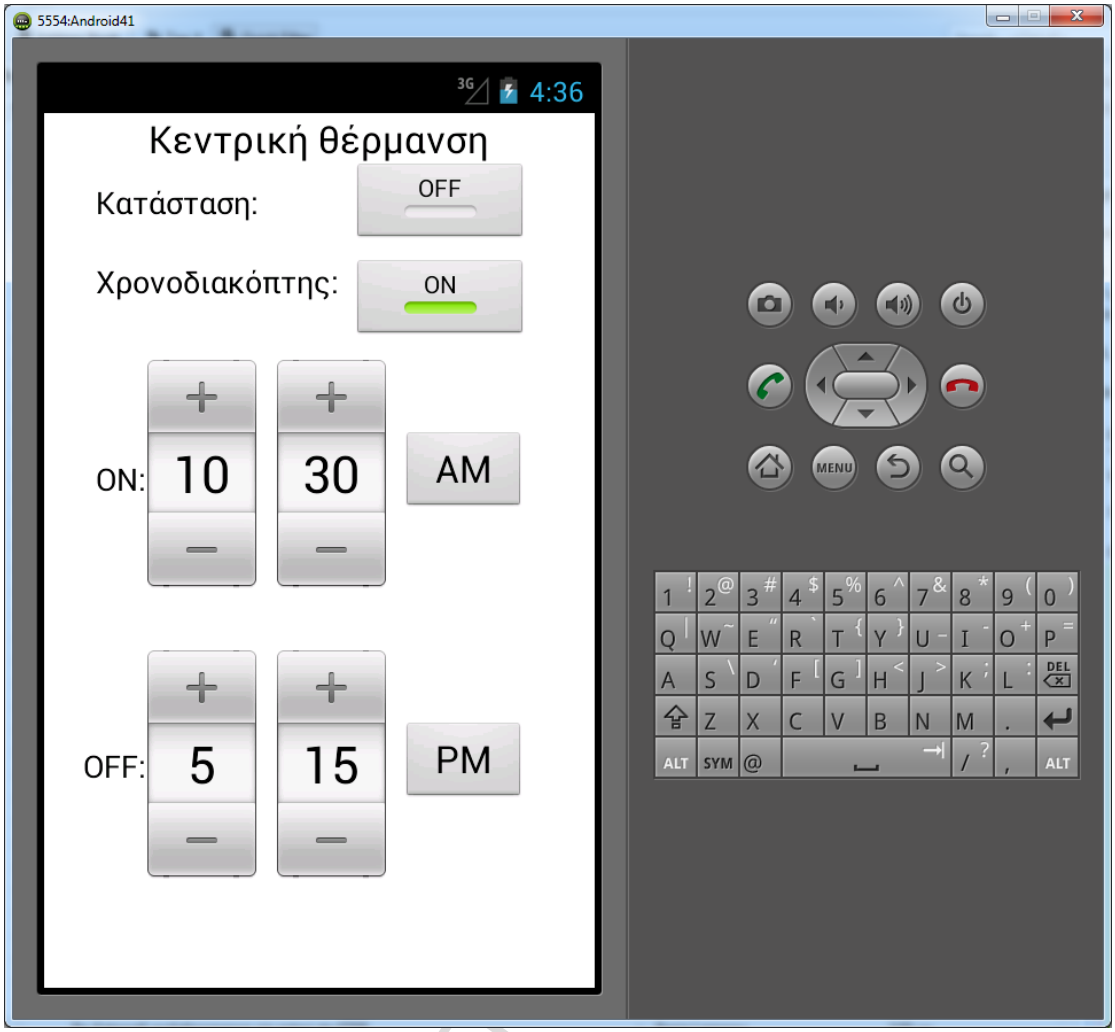
Εικόνα 22 – Οθόνη Login



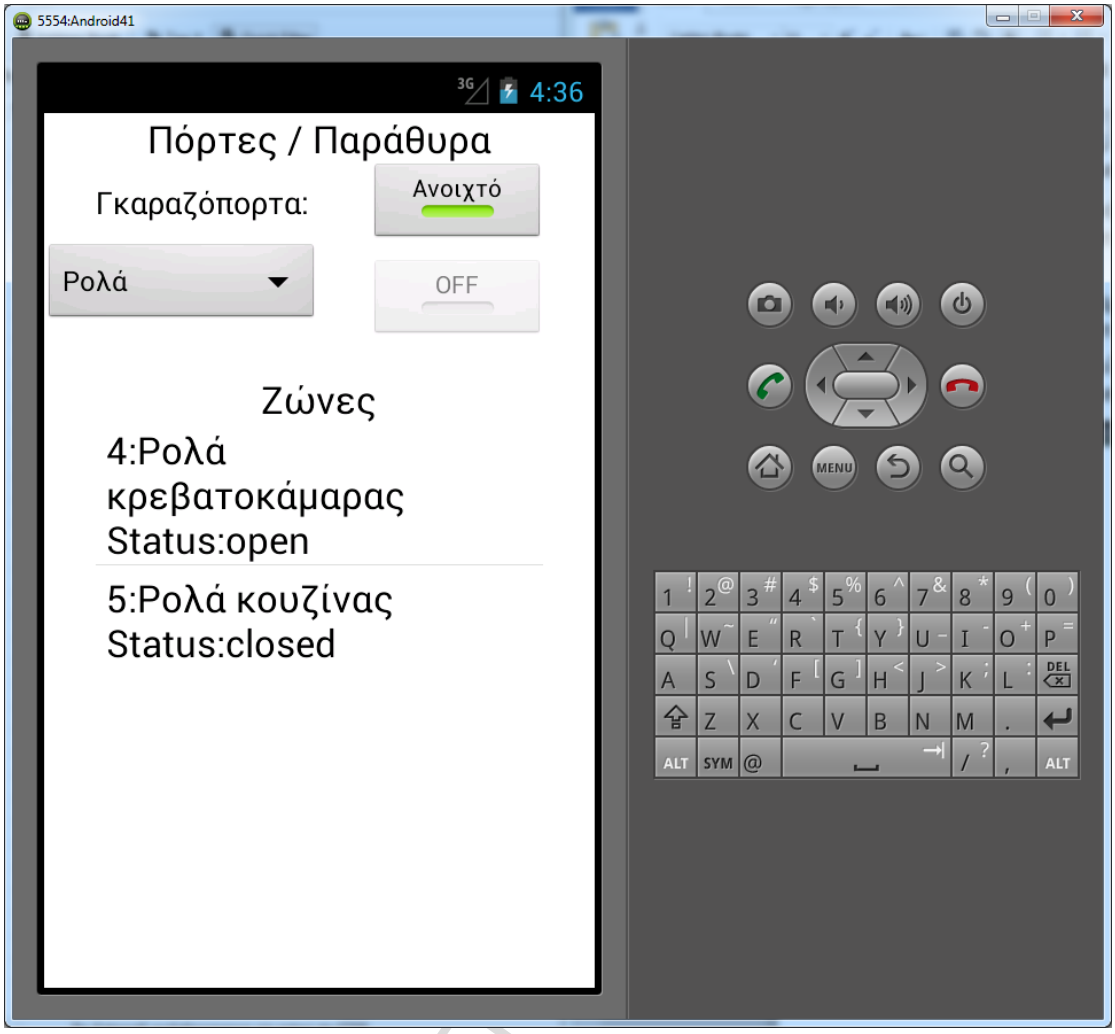
Εικόνα 23 – Κεντρική οθόνη



Εικόνα 24 – Οθόνη διαχείρισης συναγερμού



Εικόνα 25 – Οθόνη διαχείρισης κεντρικής θέρμανσης



Εικόνα 26 – Οθόνη διαχείρισης ρολών/γκαραζόπορτας

6. Επίλογος

Τα έξυπνα συστήματα μπορούν να ελέγχουν, εκτός από τις ηλεκτρολογικές εγκαταστάσεις, τις μηχανολογικές εγκαταστάσεις αλλά και τις οικιακές συσκευές και τις συσκευές πολυμέσων (Multimedia), δημιουργώντας ένα ενοποιημένο σύστημα. Στις τελευταίες εντάσσονται οι συσκευές τηλεπικοινωνιών, τα ηχοσυστήματα αλλά και οι τηλεοράσεις του σπιτιού. Συνδυάζοντας όλες αυτές τις ανεξάρτητες, αρχικά, εγκαταστάσεις σε μία κοινή βάση αποκτάται πλήρης έλεγχος της οικίας, ο οποίος μπορεί να διεξαχθεί ακόμα και από μακριά.

Πρόκειται για μια εφαρμογή του «Web of Things», όπου κάθε αντικείμενο είναι προσβάσιμο μέσω του πρωτοκόλλου HTTP και ταυτοποιείται μοναδικά, μέσω ενός URI. Βασισμένο σε αρχιτεκτονικές αρχές, όπως το REpresentational State Transfer (REST) και το Service Oriented Architecture (SOA), χρησιμοποιεί ως βάση το Web για τη δημιουργία υπηρεσιών λογισμικού, για την απομακρυσμένη διαχείριση του σπιτιού.

Στα πλαίσια της εργασίας αυτής, δημιουργήθηκε μια εφαρμογή Android για τη διαχείριση ενός έξυπνου σπιτιού. Από το "έξυπνο σπίτι" υλοποιήθηκε μέρος του Gateway, ο οποίος λειτουργεί ως διεπαφή ανάμεσα στην εφαρμογή και τις συσκευές προς διαχείριση.

Η εφαρμογή σχεδιάστηκε για Smartphones, που τρέχουν Android, και έχει τις παρακάτω λειτουργίες:

- Σύνδεση στον απομακρυσμένο Server με τα στοιχεία του χρήστη.
- Διαχείριση συστήματος συναγερμού:
    - Ενεργοποίηση/απενεργοποίηση συστήματος.
    - Ενεργοποίηση/απενεργοποίηση συναγερμού.
    - Έλεγχος της κατάστασης στις διάφορες ζώνες.
- Διαχείριση κεντρικής θέρμανσης:
    - Ενεργοποίηση/απενεργοποίηση θέρμανσης.
    - Ενεργοποίηση/απενεργοποίηση θέρμανσης με χρονοδιακόπτη (on/off timer).
- Διαχείριση θερμοσίφωνα:
    - Ενεργοποίηση/απενεργοποίηση θερμοσίφωνα.
    - Ενεργοποίηση/απενεργοποίηση θερμοσίφωνα με χρονοδιακόπτη (on/off timer).

- Διαχείριση της γκαραζόπορτας και των ρολών των παραθύρων.
- Επισκόπηση συστημάτων (Εμφάνιση μηνυμάτων από τις διάφορες απομακρυσμένες συσκευές).

Με τη χρήση του συστήματος, ο ένοικος ωφελείται σε τρεις διαφορετικούς τομείς:

- Εξοικονόμηση ενέργειας: Η κατανάλωση ενέργειας μειώνεται, με τον αυτόματο έλεγχο των θερμαντικών σωμάτων και του θερμοσίφωνα. Μπορεί να γίνει απομακρυσμένος χειρισμός ή ορισμός χρονοδιακόπτη.
- Ποιότητα ζωής: Ο ένοικος, μέσω του κινητού του τηλεφώνου, μπορεί να χειριστεί τις κύριες λειτουργίες της κατοικίας, κατά τη διάρκεια απουσίας του. Έτσι, έχει τη δυνατότητα να ανάψει το θερμοσίφωνα, λίγο πριν φτάσει σπίτι του, και να ρυθμίσει τη θερμοκρασία του σπιτιού του.
- Ασφάλεια: Το σύστημα προσφέρει δυνατότητα παρακολούθησης της κατοικίας. Έτσι, ο ιδιοκτήτης έχει τη δυνατότητα να ενημερώνεται για τη κατάσταση των διαφόρων ζωνών ασφαλείας και να ενεργοποιεί ή να απενεργοποιεί το συναγερμό, από μακριά.

Αναφορές

[1] Kamilaris A. Enabling Smart Homes using Web Technologies. PhD Thesis, University of Cyprus, Nicosia, Cyprus, December, 2012.

[2] Spicer, Dag (August 2000). "If You Can't Stand the Coding, Stay Out of the Kitchen". Dr. Dobb's Journal. Retrieved 2010-09-02.

[3] Griffiths, Melanie (March 2008). "Smart Home Security". Homebuilding & Renovating. Retrieved 27 February 2012.

[4] European Commission, Information Society and Media Directorate-General: From RFID to the Internet of Things, Pervasive networked systems, www.iot-visitthefuture.eu,2006.

[5] Communication from the commission to the European parliament, the council ,the European economic and social committee and the committee of the regions, early Early Challenges regarding the “Internet of Things”,{COM(2008) aaa},{SEC(2008) bbb}.

[6] Maarten Botterman for the European Commission, Information Society and Media Directorate General: Internet of Things: an early reality of the Future Internet, Prague, http://ec.europa.eu/information_society/tl/policy/, 2009.

[7] Guinard, Dominique; Vlad Trifa, Erik Wilde (2010-11). "A Resource Oriented Architecture for the Web of Things". Proc. of IoT 2010 (IEEE International Conference on the Internet of Things). Tokyo, Japan.

[8] The Internet of Things Council: http://www.theinternetofthings.eu/what-is-the-internet-of-things

[9] European Research Cluster on the Internet of Things: http://www.internet-of-things-research.eu/pdf/IERC_Cluster_Book_2012_WEB.pdf

[10] Android developers: http://developer.android.com/index.html

[11]Android SDK: http://developer.android.com/sdk/index.html

[12]Eclipse IDE: http://www.eclipse.org/

[13]Web Services: http://en.wikipedia.org/wiki/Web_service

[14]Rest Services: http://en.wikipedia.org/wiki/Rest

[15]JSON: http://www.json.org/json-el.html

[16]PHP: http://php.net/

[17]MySQL: http://www.mysql.com/

[18]Apache Http Client: http://hc.apache.org/

Παράρτημα – Κώδικες

Login.java

```
package com.example.myhome;

import java.util.Hashtable;

import org.json.JSONException;
import org.json.JSONObject;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.widget.EditText;
import android.widget.TextView;

/**
 * Activity which displays a login screen to the user, offering
registration as
 * well.
 */
public class Login extends Activity {

      public static final String EXTRA_EMAIL =
"com.example.android.authenticatordemo.extra.EMAIL";

      private static String URL="http://10.0.2.2/myhome/service.php";

      /**
       * Keep track of the login task to ensure we can cancel it if
requested.
       */
      private UserLoginTask mAuthTask = null;

      // Values for email and password at the time of the login
attempt.
      private String mEmail;
      private String mPassword;
      private String mServer;

      // UI references.
      private EditText mEmailView;
      private EditText mPasswordView;
      private EditText mServerView;
      private View mLoginFormView;
      private View mLoginStatusView;
```

```
      private TextView mLoginStatusMessageView;


      //handles messages send from other threads
        public Handler _handler = new Handler() {
              @Override
              public void handleMessage(Message msg) {

                    Bundle bundle = msg.getData();
                    String data = bundle.getString("data");

                    if(msg.what==3){
                         checkCredentials(data);
                    }

              super.handleMessage(msg);

              }

        };

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            setContentView(R.layout.activity_login);

            // Set up the login form.
            mEmail = getIntent().getStringExtra(EXTRA_EMAIL);
            mEmailView = (EditText) findViewById(R.id.email);
            //mEmailView.setText(mEmail);
            mEmailView.setText("user1@test.com");

            mServerView = (EditText) findViewById(R.id.host);
            mServerView.setText(URL);

            mPasswordView = (EditText) findViewById(R.id.password);
            mPasswordView
                        .setOnEditorActionListener(new
TextView.OnEditorActionListener() {

                              public boolean onEditorAction(TextView
textView, int id,KeyEvent keyEvent) {
                                    if (id == R.id.login || id ==
EditorInfo.IME_NULL) {
                                          attemptLogin();
                                          return true;
                                    }
                                    return false;
                              }
                        });

            mLoginFormView = findViewById(R.id.login_form);
            mLoginStatusView = findViewById(R.id.login_status);
            mLoginStatusMessageView = (TextView)
findViewById(R.id.login_status_message);

            findViewById(R.id.sign_in_button).setOnClickListener(
                        new View.OnClickListener() {

                              public void onClick(View view) {
```

```
                                    attemptLogin();
                                }
                            });
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            super.onCreateOptionsMenu(menu);
            getMenuInflater().inflate(R.menu.activity_login, menu);
            return true;
      }

      /**
       * Attempts to sign in or register the account specified by the
login form.
       * If there are form errors (invalid email, missing fields,
etc.), the
       * errors are presented and no actual login attempt is made.
       */
      public void attemptLogin() {
            if (mAuthTask != null) {
                  return;
            }

            // Reset errors.
            mEmailView.setError(null);
            mPasswordView.setError(null);
            mServerView.setError(null);

            // Store values at the time of the login attempt.
            mEmail = mEmailView.getText().toString();
            mPassword = mPasswordView.getText().toString();
            mServer = mServerView.getText().toString();

            boolean cancel = false;
            View focusView = null;


            // Check for a valid server.
            if (TextUtils.isEmpty(mServer)) {

      mServerView.setError(getString(R.string.error_field_required));
                  focusView = mPasswordView;
                  cancel = true;
            }
            else if (!mServer.startsWith("http://")) {

      mServerView.setError(getString(R.string.error_invalid_host));
                  focusView = mServerView;
                  cancel = true;
            }

            // Check for a valid password.
            if (TextUtils.isEmpty(mPassword)) {

      mPasswordView.setError(getString(R.string.error_field_required)
);
                  focusView = mPasswordView;
                  cancel = true;
            }
```

```
            // Check for a valid email address.
            if (TextUtils.isEmpty(mEmail)) {

      mEmailView.setError(getString(R.string.error_field_required));
                  focusView = mEmailView;
                  cancel = true;
            } else if (!mEmail.contains("@")) {

      mEmailView.setError(getString(R.string.error_invalid_email));
                  focusView = mEmailView;
                  cancel = true;
            }

            if (cancel) {
                  // There was an error; don't attempt login and
focus the first
                  // form field with an error.
                  focusView.requestFocus();
            } else {
                  // Show a progress spinner, and kick off a
background task to
                  // perform the user login attempt.

      mLoginStatusMessageView.setText(R.string.login_progress_signing
_in);
                  showProgress(true);
                  mAuthTask = new UserLoginTask();
                  mAuthTask.execute((Void) null);
            }
      }


      private void checkCredentials(String data) {

            try {

                  JSONObject myjson = new JSONObject(data);
            String username = myjson.getString("username");
            if(username.equalsIgnoreCase("null")){
                  showProgress(false);

      mPasswordView.setError(getString(R.string.error_incorrect_passw
ord));
                  mPasswordView.requestFocus();
            }
            else {
                  Intent mainActivity = new Intent(this,
MainActivity.class);
                  mainActivity.putExtra("username",username);
                  mainActivity.putExtra("password",this.mPassword);
                  mainActivity.putExtra("URL",this.mServer);
                  startActivity(mainActivity);
                  showProgress(false);
            }


        } catch (JSONException e) {
            e.printStackTrace();
            showProgress(false);
```

```
      mPasswordView.setError(getString(R.string.error_incorrect_passw
ord));
            mPasswordView.requestFocus();
        }

      }


      /**
       * Shows the progress UI and hides the login form.
       */
      @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
      private void showProgress(final boolean show) {
            // On Honeycomb MR2 we have the ViewPropertyAnimator
APIs, which allow
            // for very easy animations. If available, use these APIs
to fade-in
            // the progress spinner.
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB_MR2) {
                  int shortAnimTime = getResources().getInteger(

      android.R.integer.config_shortAnimTime);

                  mLoginStatusView.setVisibility(View.VISIBLE);

      mLoginStatusView.animate().setDuration(shortAnimTime)
                              .alpha(show ? 1 : 0)
                              .setListener(new
AnimatorListenerAdapter() {
                                    @Override
                                    public void
onAnimationEnd(Animator animation) {

      mLoginStatusView.setVisibility(show ? View.VISIBLE
                                                      : View.GONE);
                                    }
                              });

                  mLoginFormView.setVisibility(View.VISIBLE);
                  mLoginFormView.animate().setDuration(shortAnimTime)
                              .alpha(show ? 0 : 1)
                              .setListener(new
AnimatorListenerAdapter() {
                                    @Override
                                    public void
onAnimationEnd(Animator animation) {

      mLoginFormView.setVisibility(show ? View.GONE
                                                      :
View.VISIBLE);
                                    }
                              });
            } else {
                  // The ViewPropertyAnimator APIs are not available,
so simply show
                  // and hide the relevant UI components.
                  mLoginStatusView.setVisibility(show ? View.VISIBLE
: View.GONE);
```

```
                  mLoginFormView.setVisibility(show ? View.GONE :
View.VISIBLE);
            }
      }

      public class UserLoginTask extends AsyncTask<Void, Void,
Boolean> {
            @Override
            protected Boolean doInBackground(Void... params) {
                  //attempt authentication against a network service.
                  HttpPostThread t;
                  Hashtable<String,String> parameters = new
Hashtable<String,String>();
                  parameters.put("email", mEmail);
                  parameters.put("password", mPassword);
                  parameters.put("type", "login");
                  t = new
HttpPostThread(_handler,mServer,parameters,3);
                  t.start();


                  // TODO: register the new account here.
                  return true;
            }


            @Override
            protected void onPostExecute(final Boolean success) {
                  mAuthTask = null;

            }

            @Override
            protected void onCancelled() {
                  mAuthTask = null;
                  showProgress(false);
            }
      }
}
```

MainActivity.java

```
package com.example.myhome;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.List;
import java.util.Map;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.ListView;
import android.widget.SimpleAdapter;

public class MainActivity extends Activity {


      private static String username;
      private static String password;
      private static String URL;

      private Button alarmButton;
      private Button heaterButton;
      private Button heatingButton;
      private Button windowsButton;

      List<Map<String, String>> messageList = new
ArrayList<Map<String,String>>();
      SimpleAdapter messageAdapter;
      ListView messageListView;


      //handles messages send from other threads
        public Handler _handler = new Handler() {

              public void handleMessage(Message msg) {
                    Bundle bundle = msg.getData();
                    String data = bundle.getString("data");
                    if(msg.what==1){ //get messages
                          updateMessageList(data);
                    }
              super.handleMessage(msg);
              }
        };

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            messageListView = (ListView)
findViewById(R.id.messagesListView);


            alarmButton = (Button) findViewById(R.id.alarmButton);
            alarmButton.setOnClickListener(new View.OnClickListener()
{
                  public void onClick(View arg0) {
                        alarm();
                  }
            });


            heaterButton = (Button)
findViewById(R.id.waterHeaterButton);
            heaterButton.setOnClickListener(new
View.OnClickListener() {
                  public void onClick(View arg0) {
```

```
                        heater();
                  }
            });

            heatingButton = (Button)
findViewById(R.id.centralHeatingbutton);
            heatingButton.setOnClickListener(new
View.OnClickListener() {
                  public void onClick(View arg0) {
                        heating();
                  }
            });

            windowsButton = (Button)
findViewById(R.id.WindowsButton);
            windowsButton.setOnClickListener(new
View.OnClickListener() {
                  public void onClick(View arg0) {
                        windows();
                  }

            });

      }

      private void alarm() {
            Intent alarmActivity = new Intent(this,
AlarmActivity.class);
            alarmActivity.putExtra("username",username);
            alarmActivity.putExtra("password",password);
            alarmActivity.putExtra("URL",URL);
      startActivity(alarmActivity);

      }

      private void heater() {
            Intent heaterActivity = new Intent(this,
WaterHeaterActivity.class);
            heaterActivity.putExtra("username",username);
            heaterActivity.putExtra("password",password);
            heaterActivity.putExtra("URL",URL);
      startActivity(heaterActivity);

      }

      private void heating() {
            Intent heatingActivity = new Intent(this,
CentralHeating.class);
            heatingActivity.putExtra("username",username);
            heatingActivity.putExtra("password",password);
            heatingActivity.putExtra("URL",URL);
      startActivity(heatingActivity);

      }


      private void windows() {
      Intent wActivity = new Intent(this, Windows.class);
      wActivity.putExtra("username",username);
      wActivity.putExtra("password",password);
      wActivity.putExtra("URL",URL);
```

```
      startActivity(wActivity);
      }

      protected void onResume(){
            super.onResume();

            if( this.getIntent().getStringExtra("username")!=null)
                  username =
this.getIntent().getStringExtra("username");
            if( this.getIntent().getStringExtra("password")!=null)
                  password =
this.getIntent().getStringExtra("password");
            if( this.getIntent().getStringExtra("URL")!=null)
                  URL = this.getIntent().getStringExtra("URL");

            getMessages();

      }

      private void getMessages() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "messages");

            t = new HttpPostThread(_handler,URL,parameters,1);
            t.start();
      }

      private void updateMessageList(String data) {
            this.messageList.clear();


            JSONObject myjson;
            try {
                  myjson = new JSONObject(data);
                  JSONArray  messages =
myjson.getJSONArray("messages");

                  for(int i=0;i<messages.length();i++)
                  {
                  JSONObject station = (JSONObject) messages.get(i);

                  String text = station.getString("text");
                  String date = station.getString("date");
                  String source = station.getString("source");

                  messageList.add(createMessageEntry("message",
source + ":" + text + "\n" + date));


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

```
            messageAdapter = new SimpleAdapter(this, messageList,
android.R.layout.simple_list_item_1, new String[] {"message"}, new
int[] {android.R.id.text1});
            this.messageListView.setAdapter(messageAdapter);
      }

      private HashMap<String, String> createMessageEntry(String key,
String name) {
          HashMap<String, String> message = new HashMap<String,
String>();
          message.put(key, name);
          return message;
      }

}
```

HttpPostThread.java

```
package com.example.myhome;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.List;
import java.util.Map;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

import android.os.Bundle;
import android.os.Handler;
import android.os.Message;

public class HttpPostThread extends Thread {

      Handler handler;
      String URL="";
      int what;
      Hashtable<String, String> parameters;


      public HttpPostThread(Handler handler,String
URL,Hashtable<String, String> parameters, int what){
            this.handler=handler;
            this.URL=URL;
            this.what=what;
            this.parameters=parameters;
```

```
      }

      public void run(){

            HttpClient httpclient = new DefaultHttpClient();
            try {
                  HttpPost post = new HttpPost(URL);
                  List<NameValuePair> params = new
ArrayList<NameValuePair>();
                  for (Map.Entry<String, String> entry :
parameters.entrySet()) {
                        params.add(new
BasicNameValuePair(entry.getKey(), entry.getValue()));
                  }

                  post.setEntity(new UrlEncodedFormEntity(params,
"UTF-8"));

                  //Execute and get the response.
                  HttpResponse response = httpclient.execute(post);
                  HttpEntity entity = response.getEntity();
                  String responseBody="";
                  if (entity != null) {
                      InputStream instream = entity.getContent();
                      BufferedReader br= new BufferedReader(new
InputStreamReader(instream));
                      StringBuilder sb = new StringBuilder();
                      String line;
                      while ((line = br.readLine()) != null) {
                        sb.append(line);
                      }


                  responseBody = sb.toString();
                  br.close();

                        System.out.println("----------------------
----------------");
                        System.out.println(responseBody);
                        System.out.println("----------------------
----------------");

                        Message msg = Message.obtain();
                        Bundle messageData = new Bundle();
                        msg.what = what;
                        messageData.putString("data", responseBody);
                        msg.setData(messageData);

                        //send message to  activity
                        handler.sendMessage(msg);

                  }

            } catch (ClientProtocolException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
             } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
             } finally {
                // When HttpClient instance is no longer needed,
```

```
                // shut down the connection manager to ensure
                // immediate deallocation of all system resources
                httpclient.getConnectionManager().shutdown();
            }

      }
}
```

AlarmActivity.java

```
package com.example.myhome;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.List;
import java.util.Map;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.ToggleButton;


public class AlarmActivity extends Activity {

      private static String username;
      private static String password;
      private static String URL;


      ToggleButton systemButton;
      ToggleButton alarmButton;

      List<Map<String, String>> zoneList = new
ArrayList<Map<String,String>>();
      SimpleAdapter zoneAdapter;
      ListView zoneListView;

      //handles messages send from other threads
        public Handler _handler = new Handler() {

              public void handleMessage(Message msg) {
                    Bundle bundle = msg.getData();
                    String data = bundle.getString("data");
                    if(msg.what==2){ //get alarm status
                          updateAlarmStatus(data);
                    }

              super.handleMessage(msg);
```

```
            }


        };


      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_alarm);


            zoneListView = (ListView)
findViewById(R.id.ZonesListView);

            systemButton = (ToggleButton)
findViewById(R.id.toggleButton);
            systemButton.setOnClickListener(new
View.OnClickListener() {
                  public void onClick(View arg0) {
                        toggleSystem();
                  }

            });

            alarmButton = (ToggleButton)
findViewById(R.id.AlarmToggleButton);
            alarmButton.setOnClickListener(new View.OnClickListener()
{
                  public void onClick(View arg0) {
                        toggleAlarm();
                  }

            });

      }


      private void toggleSystem() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "toggleAlarmSystem");

            t = new HttpPostThread(_handler,URL,parameters,2);
            t.start();

      }

      private void toggleAlarm() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "toggleAlarm");

            t = new HttpPostThread(_handler,URL,parameters,2);
            t.start();
```

```
	}


	private HashMap<String, String> createZoneEntry(String key,
String name) {
		HashMap<String, String> zone = new HashMap<String,
String>();
		zone.put(key, name);
		return zone;
	}



	private void updateAlarmStatus(String data) {
		this.zoneList.clear();


		JSONObject myjson;
		try {
			myjson = new JSONObject(data);

			String alarm_status =
myjson.getString("alarm_status");
			String system_status =
myjson.getString("system_status");

			if(system_status.equals("off"))
					systemButton.setChecked(false);
			else
				systemButton.setChecked(true);

			if(alarm_status.equals("off"))
				alarmButton.setChecked(false);
			else
				alarmButton.setChecked(true);


			JSONArray  messages =
myjson.getJSONArray("zones");

			for(int i=0;i<messages.length();i++)
			{
			JSONObject station = (JSONObject) messages.get(i);

			String description =
station.getString("description");
			String ID = station.getString("ID");
			String status = station.getString("status");

			zoneList.add(createZoneEntry("zone",  ID + ":" +
description + "\nStatus:" +  status));


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

```
            zoneAdapter = new SimpleAdapter(this, zoneList,
android.R.layout.simple_list_item_1, new String[] {"zone"}, new int[]
{android.R.id.text1});
            this.zoneListView.setAdapter(zoneAdapter);

      }


      protected void onResume(){
            super.onResume();

            if( this.getIntent().getStringExtra("username")!=null)
                  username =
this.getIntent().getStringExtra("username");
            if( this.getIntent().getStringExtra("password")!=null)
                  password =
this.getIntent().getStringExtra("password");
            if( this.getIntent().getStringExtra("URL")!=null)
                  URL = this.getIntent().getStringExtra("URL");

            getStatus();

      }

      private void getStatus() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "alarm_status");

            t = new HttpPostThread(_handler,URL,parameters,2);
            t.start();

      }

}
```

CentralHeating.java

```
package com.example.myhome;

import java.text.ParseException;
import java.util.Hashtable;

import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.TimePicker;
import android.widget.Toast;
import android.widget.ToggleButton;
```

```
public class CentralHeating extends Activity {


      private static String username;
      private static String password;
      private static String URL;


      ToggleButton systemButton;
      ToggleButton timerButton;
      TimePicker timerStart;
      TimePicker timerEnd;

      //handles messages send from other threads
        public Handler _handler = new Handler() {
              @Override
              public void handleMessage(Message msg) {
                    Bundle bundle = msg.getData();
                    String data = bundle.getString("data");
                    if(msg.what==4){ //get alarm status
                          updateHeatingStatus(data);
                    }

              super.handleMessage(msg);
              }

        };

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_central_heating);

            systemButton = (ToggleButton)
findViewById(R.id.toggleButton);
            systemButton.setOnClickListener(new
View.OnClickListener() {
                  public void onClick(View arg0) {
                        toggleSystem();
                  }
            });

            timerButton = (ToggleButton)
findViewById(R.id.TimerToggleButton);
            timerButton.setOnClickListener(new View.OnClickListener()
{
                  public void onClick(View arg0) {
                        toggleTimer();
                  }
            });

            timerStart = (TimePicker) findViewById(R.id.timePicker1);
            timerEnd = (TimePicker) findViewById(R.id.timePicker2);

      }

      private void toggleSystem() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
```

```
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "toggleHeating");

            t = new HttpPostThread(_handler,URL,parameters,4);
            t.start();

      }

      private void toggleTimer() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "toggleHeatingTimer");

            Integer startHour = timerStart.getCurrentHour();
            Integer startMinutes =timerStart.getCurrentMinute();
            Integer endHour = timerEnd.getCurrentHour();
            Integer endMinutes = timerEnd.getCurrentMinute();

            int operatingTime =
timeDiff(startHour,startMinutes,endHour,endMinutes) ;
            if( operatingTime > 0 ){
                  parameters.put("startHour",startHour.toString() );
                  parameters.put("startMinutes",
startMinutes.toString());
                  parameters.put("endHour", endHour.toString());
                  parameters.put("endMinutes",endMinutes.toString());


                  t = new HttpPostThread(_handler,URL,parameters,4);
                  t.start();
                  if(timerButton.isChecked())
                        Toast.makeText(getApplicationContext(), "Η
θέρμανση θα λειτουργήσει " + operatingTime + " λεπτά",
Toast.LENGTH_LONG).show();

            }
            else {
                  Toast.makeText(getApplicationContext(), "Ο χρόνος
έναρξης πρέπει να είναι νωρίτερος από τον χρόνο τερματισμού!",
Toast.LENGTH_LONG).show();
                  timerButton.setChecked(false);
            }

      }

      private int timeDiff(int startHour, int startMinutes, int
endHour,    int endMinutes) {

            java.text.DateFormat df = new
java.text.SimpleDateFormat("hh:mm:ss");

            try {
                  java.util.Date date1 = df.parse("" +startHour +":"
+ startMinutes +":00" );
                  java.util.Date date2 = df.parse("" +endHour +":" +
endMinutes +":00" );
```

```
                  int diff = (int) (date2.getTime() -
date1.getTime())/60000;
                  return diff;
            } catch (ParseException e) {
                  return -1;
            }


      }

      protected void onResume(){
            super.onResume();

            if( this.getIntent().getStringExtra("username")!=null)
                  username =
this.getIntent().getStringExtra("username");
            if( this.getIntent().getStringExtra("password")!=null)
                  password =
this.getIntent().getStringExtra("password");
            if( this.getIntent().getStringExtra("URL")!=null)
                  URL = this.getIntent().getStringExtra("URL");

            getStatus();

      }


      private void getStatus() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "heating_status");

            t = new HttpPostThread(_handler,URL,parameters,4);
            t.start();

      }

      private void updateHeatingStatus(String data) {
            JSONObject myjson;
            try {
                  myjson = new JSONObject(data);

                  String timer_status = myjson.getString("timer");
                  String system_status = myjson.getString("status");
                  int startHour = myjson.getInt("startHour");
                  int startMinutes = myjson.getInt("startMinutes");
                  int endHour = myjson.getInt("endHour");
                  int endMinutes = myjson.getInt("endMinutes");

                  timerStart.setCurrentHour(startHour);
                  timerStart.setCurrentMinute(startMinutes);
                  timerEnd.setCurrentHour(endHour);
                  timerEnd.setCurrentMinute(endMinutes);

                  if(system_status.equals("off"))
                        systemButton.setChecked(false);
                  else
                        systemButton.setChecked(true);
```

```
                  if(timer_status.equals("off"))
                        timerButton.setChecked(false);
                  else
                        timerButton.setChecked(true);

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


      }

}
```

WaterHeaterActivity.java

```
package com.example.myhome;

import java.text.ParseException;
import java.util.Hashtable;

import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.TimePicker;
import android.widget.Toast;
import android.widget.ToggleButton;

public class WaterHeaterActivity extends Activity {

      private static String username;
      private static String password;
      private static String URL;


      ToggleButton systemButton;
      ToggleButton timerButton;
      TimePicker timerStart;
      TimePicker timerEnd;



      //handles messages send from other threads
        public Handler _handler = new Handler() {
              @Override
              public void handleMessage(Message msg) {
                    Bundle bundle = msg.getData();
                    String data = bundle.getString("data");
                    if(msg.what==5){ //get alarm status
                          updateHeatingStatus(data);
                    }
```

```
              super.handleMessage(msg);
              }

        };

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_water_heater);

            systemButton = (ToggleButton)
findViewById(R.id.toggleButton);
            systemButton.setOnClickListener(new
View.OnClickListener() {
                  public void onClick(View arg0) {
                        toggleSystem();
                  }
            });

            timerButton = (ToggleButton)
findViewById(R.id.TimerToggleButton);
            timerButton.setOnClickListener(new View.OnClickListener()
{
                  public void onClick(View arg0) {
                        toggleTimer();
                  }
            });

            timerStart = (TimePicker) findViewById(R.id.timePicker1);
            timerEnd = (TimePicker) findViewById(R.id.timePicker2);

      }

      private void toggleSystem() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "toggleHeater");

            t = new HttpPostThread(_handler,URL,parameters,5);
            t.start();

      }

      private void toggleTimer() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "toggleHeaterTimer");

            Integer startHour = timerStart.getCurrentHour();
            Integer startMinutes =timerStart.getCurrentMinute();
            Integer endHour = timerEnd.getCurrentHour();
            Integer endMinutes = timerEnd.getCurrentMinute();
```

```
            int operatingTime =
timeDiff(startHour,startMinutes,endHour,endMinutes) ;
            if( operatingTime > 0 ){
                  parameters.put("startHour",startHour.toString() );
                  parameters.put("startMinutes",
startMinutes.toString());
                  parameters.put("endHour", endHour.toString());
                  parameters.put("endMinutes",endMinutes.toString());


                  t = new HttpPostThread(_handler,URL,parameters,5);
                  t.start();
                  if(timerButton.isChecked())
                        Toast.makeText(getApplicationContext(), "Ο
θερμοσίφωνας θα λειτουργήσει για" + operatingTime + " λεπτά",
Toast.LENGTH_LONG).show();

            }
            else {
                  Toast.makeText(getApplicationContext(), "Ο χρόνος
έναρξης πρέπει να είναι νωρίτερος από τον χρόνο τερματισμού!",
Toast.LENGTH_LONG).show();
                  timerButton.setChecked(false);
            }




      }

      private int timeDiff(int startHour, int startMinutes, int
endHour,    int endMinutes) {

            java.text.DateFormat df = new
java.text.SimpleDateFormat("hh:mm:ss");

            try {
                  java.util.Date date1 = df.parse("" +startHour +":"
+ startMinutes +":00" );
                  java.util.Date date2 = df.parse("" +endHour +":" +
endMinutes +":00" );
                  int diff = (int) (date2.getTime() -
date1.getTime())/60000;
                  return diff;
            } catch (ParseException e) {
                  return -1;
            }


      }

      protected void onResume(){
            super.onResume();

            if( this.getIntent().getStringExtra("username")!=null)
                  username =
this.getIntent().getStringExtra("username");
            if( this.getIntent().getStringExtra("password")!=null)
                  password =
this.getIntent().getStringExtra("password");
            if( this.getIntent().getStringExtra("URL")!=null)
```

```
                  URL = this.getIntent().getStringExtra("URL");

            getStatus();

      }

      private void getStatus() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "heater_status");

            t = new HttpPostThread(_handler,URL,parameters,5);
            t.start();

      }

      private void updateHeatingStatus(String data) {
            JSONObject myjson;
            try {
                  myjson = new JSONObject(data);

                  String timer_status = myjson.getString("timer");
                  String system_status = myjson.getString("status");
                  int startHour = myjson.getInt("startHour");
                  int startMinutes = myjson.getInt("startMinutes");
                  int endHour = myjson.getInt("endHour");
                  int endMinutes = myjson.getInt("endMinutes");

                  timerStart.setCurrentHour(startHour);
                  timerStart.setCurrentMinute(startMinutes);
                  timerEnd.setCurrentHour(endHour);
                  timerEnd.setCurrentMinute(endMinutes);

                  if(system_status.equals("off"))
                        systemButton.setChecked(false);
                  else
                        systemButton.setChecked(true);

                  if(timer_status.equals("off"))
                        timerButton.setChecked(false);
                  else
                        timerButton.setChecked(true);

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


      }
}
```

Windows.java

```
package com.example.myhome;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.List;
import java.util.Map;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Spinner;
import android.widget.ToggleButton;

public class Windows extends Activity {

      private static String username;
      private static String password;
      private static String URL;


      ToggleButton blindsButton;
      ToggleButton parkingButton;

      List<String> spinnerList = new ArrayList<String>();
      ArrayAdapter<String> spinnerAdapter;
      Spinner spinner;

      List<Map<String, String>> zoneList = new
ArrayList<Map<String,String>>();
      SimpleAdapter zoneAdapter;
      ListView zoneListView;


      //handles messages send from other threads
        public Handler _handler = new Handler() {
              @Override
              public void handleMessage(Message msg) {
                    Bundle bundle = msg.getData();
                    String data = bundle.getString("data");
                    if(msg.what==6){ //get alarm status
                          updateZoneStatus(data);
                    }

              super.handleMessage(msg);
              }


        };

    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_windows);

        spinner = (Spinner) findViewById(R.id.blindsSpinner);
        spinner.setOnItemSelectedListener(new
OnItemSelectedListener(){

                  public void onItemSelected(AdapterView<?> arg0,
View arg1,int arg2, long arg3) {
                        if(spinner.getSelectedItemPosition()>0){
                              blindsButton.setEnabled(true);
                              String zone  =
(String)spinner.getSelectedItem();
                              String zoneID = zone.substring(0,
zone.indexOf(':'));
                              for(int i=0;i<zoneList.size();i++){
                                    Map<String,String> zonemap =
zoneList.get(i);

      if(zonemap.get("zone").startsWith(zoneID)){ //get zone status

      if(zonemap.get("zone").endsWith("open"))

      blindsButton.setChecked(true);
                                          else

      blindsButton.setChecked(false);

                                    }
                              }

                        }
                        else{
                              blindsButton.setEnabled(false);
                              blindsButton.setChecked(false);
                        }

                  }

                  public void onNothingSelected(AdapterView<?> arg0)
{


                  }


       });

        zoneListView = (ListView) findViewById(R.id.ZonesListView);

        blindsButton = (ToggleButton)
findViewById(R.id.BlindsToggleButton);
        blindsButton.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View arg0) {
                        toggleBlinds();
                  }

            });
```

```
        parkingButton = (ToggleButton)
findViewById(R.id.parkingToggleButton);
        parkingButton.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View arg0) {
                        toggleParking();
                  }

            });
    }

    private void toggleParking() {
      HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "toggleParking");

            t = new HttpPostThread(_handler,URL,parameters,6);
            t.start();

      }


      private void toggleBlinds() {
            if(spinner.getSelectedItemPosition()>0){
                  String zone  = (String)spinner.getSelectedItem();
                  String zoneID = zone.substring(0,
zone.indexOf(':'));


                  HttpPostThread t;
                  Hashtable<String,String> parameters = new
Hashtable<String,String>();
                  parameters.put("username", username);
                  parameters.put("password", password);
                  parameters.put("type", "toggleBlinds");
                  parameters.put("zoneID", zoneID);

                  t = new HttpPostThread(_handler,URL,parameters,6);
                  t.start();
            }
      }

      private HashMap<String, String> createZoneEntry(String key,
String name) {
          HashMap<String, String> zone = new HashMap<String,
String>();
          zone.put(key, name);
          return zone;
      }

      private void updateZoneStatus(String data) {
            this.zoneList.clear();
            this.spinnerList.clear();
            spinnerList.add("Ρολά");

            blindsButton.setEnabled(false);
            blindsButton.setChecked(false);

            JSONObject myjson;
```

```
            try {
                  myjson = new JSONObject(data);

                  String parking_status =
myjson.getString("parking_status");


                  if(parking_status.equals("closed"))
                              parkingButton.setChecked(false);
                  else
                        parkingButton.setChecked(true);


                  JSONArray  messages =
myjson.getJSONArray("blinds");

                  for(int i=0;i<messages.length();i++)
                  {
                  JSONObject station = (JSONObject) messages.get(i);

                  String description =
station.getString("description");
                  String ID = station.getString("ID");
                  String status = station.getString("status");


                  //add items to zones list view
                  zoneList.add(createZoneEntry("zone",  ID + ":" +
description + "\nStatus:" +  status));

                  //add items to spinner
                  spinnerList.add(ID + ":" + description);

                  }



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

            spinnerAdapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,
spinnerList);

      spinnerAdapter.setDropDownViewResource(android.R.layout.simple_
spinner_dropdown_item);
            spinner.setAdapter(spinnerAdapter);

            zoneAdapter = new SimpleAdapter(this, zoneList,
android.R.layout.simple_list_item_1, new String[] {"zone"}, new int[]
{android.R.id.text1});
            this.zoneListView.setAdapter(zoneAdapter);

      }

      protected void onResume(){
            super.onResume();

            if( this.getIntent().getStringExtra("username")!=null)
```

```
                  username =
this.getIntent().getStringExtra("username");
            if( this.getIntent().getStringExtra("password")!=null)
                  password =
this.getIntent().getStringExtra("password");
            if( this.getIntent().getStringExtra("URL")!=null)
                  URL = this.getIntent().getStringExtra("URL");

            getStatus();

      }

      private void getStatus() {
            HttpPostThread t;
            Hashtable<String,String> parameters = new
Hashtable<String,String>();
            parameters.put("username", username);
            parameters.put("password", password);
            parameters.put("type", "windows_status");

            t = new HttpPostThread(_handler,URL,parameters,6);
            t.start();

      }
}
```

database.php

```
<?php

define('DBHOST','localhost');

define('DBUSER','root');

define('DBPASS','root');

define('DBNAME','myhome');

$conn = @mysql_connect (DBHOST, DBUSER, DBPASS);

$conn = @mysql_select_db (DBNAME);

if(!$conn){

      die( "Sorry! There seems to be a problem connecting to our
database.");

}

mysql_query("SET NAMES UTF8");

function signin($email, $pass){
```

```
   $email = strip_tags(mysql_real_escape_string($email));

   $pass = strip_tags(mysql_real_escape_string($pass));

   $sql = "SELECT username FROM users WHERE email = '$email' AND
password = '$pass'";

   $res = mysql_query($sql) or die('Query failed. ' . mysql_error());

   if($res){

            $row = mysql_fetch_assoc($res);

            $arr = array('username' => $row['username']);

             echo json_encode($arr);

      }

}

function login($user, $pass){

   $user = strip_tags(mysql_real_escape_string($user));

   $pass = strip_tags(mysql_real_escape_string($pass));

   $sql = "SELECT * FROM users WHERE username = '$user' AND password
= '$pass'";

   $result = mysql_query($sql) or die('Query failed. ' .
mysql_error());

      if (mysql_num_rows($result) == 1)

            return true;

      else

            return false;

}

function getMessages() {

      $query = "select * from messages order by date DESC LIMIT 10";

      $res = mysql_query($query);

    // iterate over every row
```

```
    while ($row = mysql_fetch_assoc($res)) {

        // for every field in the result..

        for ($i=0; $i < mysql_num_fields($res); $i++) {

            $info = mysql_fetch_field($res, $i);

            $type = $info->type;

                  //var_dump($type);

            // cast for real

            if ($type == 'real')

                $row[$info->name] = doubleval($row[$info->name]);

            // cast for int

            if ($type == 'int')

                $row[$info->name] = intval($row[$info->name]);

        }

        $rows[] = $row;

    }

      $arr = array('messages' => $rows);

    // JSON-ify all rows together as one big array

    echo json_encode($arr, JSON_UNESCAPED_UNICODE);

}


function getAlarmStatus() {

   $arr = array();

      $query = "SELECT * from alarm ";

    $res = mysql_query($query);

      if($res){

            $row = mysql_fetch_assoc($res);
```

```
            $arr['system_status'] = $row['system_status'];
            $arr['alarm_status'] = $row['alarm_status'];
      }
      $rows = array();
      $query = "SELECT * from zones ";
   $res = mysql_query($query);
   // iterate over every row
   while ($row = mysql_fetch_assoc($res)) {
       // for every field in the result..
       for ($i=0; $i < mysql_num_fields($res); $i++) {
           $info = mysql_fetch_field($res, $i);
           $type = $info->type;
                 //var_dump($type);
           // cast for real
           if ($type == 'real')
               $row[$info->name] = doubleval($row[$info->name]);
           // cast for int
           if ($type == 'int')
               $row[$info->name] = intval($row[$info->name]);
       }
       $rows[] = $row;
   }
     $arr['zones'] = $rows;

   // JSON-ify all rows together as one big array
   echo json_encode($arr, JSON_UNESCAPED_UNICODE);
```

```
}

function toggleAlarmStatus() {

      $query = "UPDATE alarm SET `alarm_status`=(SELECT CASE
alarm_status WHEN 'on' THEN 'off' ELSE 'on' END)";

      $res = mysql_query($query);

      getAlarmStatus();

}

function toggleAlarmSystemStatus() {

      $query = "UPDATE alarm SET `system_status`=(SELECT CASE
system_status WHEN 'on' THEN 'off' ELSE 'on' END)";

      $res = mysql_query($query);

      getAlarmStatus();

}

function getHeatingStatus() {

   $row = array();

      $query = "SELECT * from heating ";

    $res = mysql_query($query);

    // iterate over every row

    if ($row = mysql_fetch_assoc($res)) {

        // for every field in the result..

        for ($i=0; $i < mysql_num_fields($res); $i++) {

            $info = mysql_fetch_field($res, $i);

            $type = $info->type;

                  //var_dump($type);

            // cast for real

            if ($type == 'real')

                $row[$info->name] = doubleval($row[$info->name]);
```

```
            // cast for int
            if ($type == 'int')
                $row[$info->name] = intval($row[$info->name]);
        }
    }

    // JSON-ify all rows together as one big array
    echo json_encode($row, JSON_UNESCAPED_UNICODE);
}
function toggleHeatingStatus() {
      $query = "UPDATE heating SET `status`=(SELECT CASE status WHEN
'on' THEN 'off' ELSE 'on' END)";
      $res = mysql_query($query);
      getHeatingStatus();
}

function
toggleHeatingTimerStatus($startHour,$startMinutes,$endHour,$endMinute
s) {
      $query = "UPDATE heating SET `timer`=(SELECT CASE timer WHEN
'on' THEN 'off' ELSE 'on' END), startHour='$startHour',
            endHour='$endHour',         startMinutes='$startMinutes',
endMinutes='$endMinutes' ";
      $res = mysql_query($query);
      getHeatingStatus();
}
function getHeaterStatus() {
   $row = array();
```

```
      $query = "SELECT * from heater ";

    $res = mysql_query($query);

    // iterate over every row

    if ($row = mysql_fetch_assoc($res)) {

        // for every field in the result..

        for ($i=0; $i < mysql_num_fields($res); $i++) {

            $info = mysql_fetch_field($res, $i);

            $type = $info->type;

                  //var_dump($type);

            // cast for real

            if ($type == 'real')

                $row[$info->name] = doubleval($row[$info->name]);

            // cast for int

            if ($type == 'int')

                $row[$info->name] = intval($row[$info->name]);

        }

    }


    // JSON-ify all rows together as one big array

    echo json_encode($row, JSON_UNESCAPED_UNICODE);


}

function toggleHeaterStatus() {

      $query = "UPDATE heater SET `status`=(SELECT CASE status WHEN
'on' THEN 'off' ELSE 'on' END)";

      $res = mysql_query($query);
```

```
      getHeaterStatus();

}

function
toggleHeaterTimerStatus($startHour,$startMinutes,$endHour,$endMinutes
) {

      $query = "UPDATE heater SET `timer`=(SELECT CASE timer WHEN
'on' THEN 'off' ELSE 'on' END), startHour='$startHour',

            endHour='$endHour',         startMinutes='$startMinutes',
endMinutes='$endMinutes' ";

      $res = mysql_query($query);

      getHeaterStatus();

}

function getWindowsStatus() {

   $arr = array();

      $query = "SELECT status from zones where type='parkingdoor' ";

    $res = mysql_query($query);

      if($res){

            $row = mysql_fetch_assoc($res);

            $arr['parking_status'] = $row['status'];

      }

      $rows = array();

      $query = "SELECT * from zones where type='blinds' ";

    $res = mysql_query($query);

    // iterate over every row

    while ($row = mysql_fetch_assoc($res)) {

        // for every field in the result..

        for ($i=0; $i < mysql_num_fields($res); $i++) {
```

```
            $info = mysql_fetch_field($res, $i);
            $type = $info->type;
                  //var_dump($type);
            // cast for real
            if ($type == 'real')
                $row[$info->name] = doubleval($row[$info->name]);
            // cast for int
            if ($type == 'int')
                $row[$info->name] = intval($row[$info->name]);
        }
        $rows[] = $row;
    }

      $arr['blinds'] = $rows;

    // JSON-ify all rows together as one big array
    echo json_encode($arr, JSON_UNESCAPED_UNICODE);

}

function toggleParkingStatus() {
      $query = "UPDATE zones SET `status`=(SELECT CASE status WHEN
'open' THEN 'closed' ELSE 'open' END) where type='parkingdoor'";
      $res = mysql_query($query);
      getWindowsStatus();
}
```

```
function toggleBlindsStatus($zoneID) {

      $query = "UPDATE zones SET `status`=(SELECT CASE status WHEN
'open' THEN 'closed' ELSE 'open' END) where ID='$zoneID'";

      $res = mysql_query($query);

      getWindowsStatus();

}

?>
```

myhome.sql

```
-- phpMyAdmin SQL Dump
-- version 3.5.2.2
-- http://www.phpmyadmin.net
--
-- Φιλοξενητής: 127.0.0.1
-- Χρόνος δημιουργίας: 11 Δεκ 2013 στις 15:50:33
-- Έκδοση διακομιστή: 5.5.27
-- Έκδοση PHP: 5.4.7

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Βάση: `myhome`
--

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `alarm`
--

CREATE TABLE IF NOT EXISTS `alarm` (
  `ID` int(11) NOT NULL,
  `alarm_status` enum('on','off') COLLATE utf8_unicode_ci NOT NULL,
  `system_status` enum('on','off') COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Άδειασμα δεδομένων του πίνακα `alarm`
--

INSERT INTO `alarm` (`ID`, `alarm_status`, `system_status`) VALUES
```

```
(1, 'off', 'on');

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `heater`
--

CREATE TABLE IF NOT EXISTS `heater` (
  `ID` int(11) NOT NULL,
  `status` enum('on','off') COLLATE utf8_unicode_ci NOT NULL,
  `timer` enum('on','off') COLLATE utf8_unicode_ci NOT NULL,
  `startHour` int(11) NOT NULL,
  `startMinutes` int(11) NOT NULL,
  `endHour` int(11) NOT NULL,
  `endMinutes` int(11) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Άδειασμα δεδομένων του πίνακα `heater`
--

INSERT INTO `heater` (`ID`, `status`, `timer`, `startHour`,
`startMinutes`, `endHour`, `endMinutes`) VALUES
(1, 'off', 'off', 5, 0, 7, 15);

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `heating`
--

CREATE TABLE IF NOT EXISTS `heating` (
  `ID` int(11) NOT NULL,
  `status` enum('on','off') COLLATE utf8_unicode_ci NOT NULL,
  `timer` enum('on','off') COLLATE utf8_unicode_ci NOT NULL,
  `startHour` int(11) NOT NULL,
  `startMinutes` int(11) NOT NULL,
  `endHour` int(11) NOT NULL,
  `endMinutes` int(11) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Άδειασμα δεδομένων του πίνακα `heating`
--

INSERT INTO `heating` (`ID`, `status`, `timer`, `startHour`,
`startMinutes`, `endHour`, `endMinutes`) VALUES
(1, 'off', 'on', 10, 30, 17, 15);

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `messages`
--

CREATE TABLE IF NOT EXISTS `messages` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `text` varchar(1000) COLLATE utf8_unicode_ci NOT NULL,
```

```
  `date` datetime NOT NULL,
  `source` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=4 ;

--
-- Άδειασμα δεδομένων του πίνακα `messages`
--

INSERT INTO `messages` (`ID`, `text`, `date`, `source`) VALUES
(1, 'Η θερμοκρασία είναι 21 βαθμοί', '2013-11-28 00:00:00',
'Θέρμανση'),
(2, 'The alarm system has been turned off', '2013-11-28 00:00:00',
'alarm'),
(3, 'Η θερμοκρασία είναι19 βαθμοί', '2013-11-28 11:10:00',
'Θέρμανση');

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `users`
--

CREATE TABLE IF NOT EXISTS `users` (
  `username` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Άδειασμα δεδομένων του πίνακα `users`
--

INSERT INTO `users` (`username`, `email`, `password`) VALUES
('user1', 'user1@test.com', 'u');

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `zones`
--

CREATE TABLE IF NOT EXISTS `zones` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `description` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `status` enum('open','closed') COLLATE utf8_unicode_ci NOT NULL,
  `alarmID` int(11) NOT NULL,
  `type` enum('window','blinds','door','parkingdoor','radar','smoke')
COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `alarmID` (`alarmID`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=7 ;

--
-- Άδειασμα δεδομένων του πίνακα `zones`
--
```

```
INSERT INTO `zones` (`ID`, `description`, `status`, `alarmID`,
`type`) VALUES
(1, 'Κεντρική πόρτα', 'closed', 1, 'door'),
(2, 'Πίσω πόρτα', 'closed', 1, 'door'),
(3, 'Μπαλκονόπορτα', 'open', 1, 'window'),
(4, 'Ρολά κρεβατοκάμαρας', 'open', 1, 'blinds'),
(5, 'Ρολά κουζίνας', 'closed', 1, 'blinds'),
(6, 'Γκαραζόπορτα', 'open', 1, 'parkingdoor');

--
-- Περιορισμοί για άχρηστους πίνακες
--

--
-- Περιορισμοί για πίνακα `zones`
--
ALTER TABLE `zones`
  ADD CONSTRAINT `zones_ibfk_1` FOREIGN KEY (`alarmID`) REFERENCES
`alarm` (`ID`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```