Ατομική Διπλωματική Εργασία

SMART HOME APPLICATION GUI

Μιχάλης Γιάλλουρος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Smart Home Application GUI

Μιχάλης Γιάλλουρος

Επιβλέπων Καθηγητής
Ανδρέας Πιτσιλλίδης

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2010

Ευχαριστίες

Θα ήθελα να ευχαριστήσω το κύριο Ανδρέα Πιτσιλλίδη για την ευκαιρία καταρχήν που μου έδωσε να ασχοληθώ με αυτό το πολύ ενδιαφέρον project καθώς και την υποστήριξη που είχα από αυτόν καθόλη την διάρκεια της διπλωματικής. Επίσης ένα πολύ μεγάλο ευχαριστώ στον υποψήφιο διδακτορικό φοιτητή του τμήματος μας τον Αντρέα Καμηλάρη για την άψογη συνεργασία που είχαμε τα δύο προηγούμενα εξάμηνα. Το project που ξεκίνησε έχει πάρα πολλές προοπτικές και πολλή καινοτομία και νιώθω προσωπικά ιδιαίτερη ικανοποίηση που πρόσθεσα και εγώ ένα κομμάτι σε αυτό.

Περίληψη

Η διπλωματική μου εργασία θα σχετίζεται με την χρήση αισθητήρων δικτύων, οι οποίοι θα χρησιμοποιηθούν για την επίλυση του προβλήματος του Smart Home. Στην εργασία αυτή δηλαδή εξετάζεται η εκμετάλλευση των αισθητήρων, σε μια εφαρμογή αυτοματοποίησης οικίας.

Χρησιμοποιήθηκε ένα υπάρχων application framework for wireless sensor networks. Με βάση αυτό έχω αναπτύξει, σε πιο ψηλό επίπεδο, μια πρωτότυπη εφαρμογή, μέσω της οποίας ο χρήστης, μέσα από ένα απλό και γραφικό περιβάλλον (GUI) θα μπορεί εύκολα μέσω του Ίντερνετ να παρακολουθεί συσκευές και να ενημερώνεται για διάφορες καταστάσεις στην οικία του.

Συγκεκριμένα ο χρήστης έχει την δυνατότητα μέσω της εφαρμογής δει σε μορφή λίστας τις συσκευές που είναι συνδεδεμένες με το σύστημα και τις υπηρεσίες που προσφέρουν έχοντας συνάμα την δυνατότητα να (απ)ενεργοποιήσει μια ηλεκτρική συσκευή. Επίσης θα μπορεί να ανακτήσει πληροφορίες σχετικές με την κατάσταση του "περιβάλλοντος" το οποίο επιθυμεί να αυτοματοποιήσει, μέσω των δεδομένων που θα παρέχονται από τους αισθητήρες .

Τέλος έχουν αναπτυχθεί δυο μηχανισμοί που βασιζόμενοι στην "κατάσταση του περιβάλλοντος" εκτελούν αυτόματα κάποιες λειτουργίες. Ο πρώτος είναι ένας μηχανισμός eventing μέσω του οποίου κάποιοι απομακρυσμένοι χρήστες γίνονται subscribe για να ενημερώνονται για κάποιες καταστάσεις μέσω email ή sms. Και ο δεύτερος δίνει τη δυνατότητα στο χρήστης να θέσει κάποιους πιο σύνθετους κανόνες (rules), με βάση τους οποίους μια κατάσταση (event) να προκαλεί την εκτέλεση κάποιων άλλων λειτουργιών.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Τι είναι το "έξυπνο σπίτι"

Σε ένα smart home[1] σύστημα στόχος είναι η αυτοματοποίηση μιας οικίας. Η βασική ιδέα πίσω από την αυτοματοποίηση, είναι η εγκαθίδρυση μηχανισμών που παρακολουθούν το περιβάλλον της οικίας και μπορούν να το μεταβάλουν ρυθμίζοντας το σύμφωνα με τις απαιτήσεις του χρήστη. Συνάμα προσφέρουν πληθώρα υπηρεσιών που κάνουν τη ζωή των ενοίκων ακόμα πιο εύκολη. Μια εφαρμογή smart Home μπορεί να εξοικονομήσει πολλά λεφτά στον ιδιοκτήτη μιας οικίας και συγχρόνως μπορεί να βοηθήσει και στο πρόβλημα της μόλυνσης του περιβάλλοντος αφού μπορεί να μειώσει την άσκοπη σπατάλη ενέργειας στην οικία.

Για την επίτευξη του στόχου αυτού εγκαθίσταται στην οικία ένα εσωτερικό δίκτυο αισθητήρων (ενσύρματο ή και ασύρματο) μέσω του οποίου το σύστημα είναι σε θέση να "παρατηρεί" συνεχώς την κατάσταση του περιβάλλοντος (τρέχουσα θερμοκρασία, υγρασία κτλ.). Μετά όλα αυτά τα δεδομένα τυγχάνουν επεξεργασίας από μια μονάδα ελέγχου-οργάνωσης (hardware–software controllers) και ανάλογα με τις ρυθμίσεις του

συστήματος ενεργοποιούνται οι μηχανισμοί/συσκευές (actuators) που θα εκτελέσουν την αυτοματοποίηση(πχ αυτόματη ενεργοποίηση ενός κλιματιστικού).

Μέσω μιας εφαρμογής smart home ο χρήστης έχει την δυνατότητα εκτός από την αυτοματοποίηση βασικών λειτουργιών της οικίας του να μπορεί να είναι ενήμερος και για την κατάσταση του σπιτιού του οποιαδήποτε ώρα, σε οποιοδήποτε σημείο στον κόσμο μέσο Ίντερνετ.

"The goal is to adapt the environment to user preferences without user intervention (Maximizing intelligence by minimizing explicit user interaction)" [2]

1.2 Background Theory

Πρώτου προχωρήσουμε στην ανάλυση των απαιτήσεων και στόχων της εφαρμογής έγινε μια μελέτη σειράς άρθρων σχετικές με το θέμα μας.

Το "έξυπνο περιβάλλον" που δημιουργείται στην ουσία μέσω από μια εφαρμογή smart home, βασίζεται στο Ubiquitous Computing [3][4][5]. Ο σχεδιασμός και το κτίσιμο μιας τέτοιας εφαρμογής απαιτεί γνώσεις και ιδέες από πολλές περιοχές της Πληροφορικής όπως sensor networks[6], pervasive communications[7][8][9][2], mobile computing καθώς και ένα ευρύ πεδίο γνώσεων για Web Technologies.

1.3 Ανάλυση απαιτήσεων-στόχων του application

Έχοντας υπόψη ότι η εφαρμογή θα είναι Διαδικτυακή θα πρέπει να σχεδιαστεί με τέτοιο τρόπο ώστε να είναι όσο πιο lightweighted γίνεται και να είναι απλή και κατανοητή ούτως ώστε να μπορεί να χρησιμοποιηθεί από το μέσο χρήστη.

1.3.1 Απαιτήσεις σε ψηλό επίπεδο

Πιο κάτω παρατίθενται οι γενικές απαιτήσεις στις οποίες πρέπει να ανταποκρίνεται η εφαρμογή σε ψηλό επίπεδο μετά την υλοποίηση της.

Ένας χρήστης μέσω της εφαρμογής θα πρέπει να έχει την δυνατότητα να δει σε μορφή λίστας τις συσκευές που είναι συνδεδεμένες με το σύστημα και τις διάφορες υπηρεσίες που προσφέρουν. Πιο συγκεκριμένα θα μπορεί μέσα από μια απλή διαδικασία να ενημερωθεί άμεσα για το ποίες συσκευές είναι διαθέσιμες τη δεδομένη χρονική στιγμή ενώ παράλληλα θα έχει πρόσβαση σε μια μικρή περιγραφή των προσφερόμενων υπηρεσιών. Αυτές τις πληροφορίες (devices description) το application που θα αναπτυχθεί θα πρέπει να τις λαμβάνει από το υπάρχων application framework.

Επίσης ο χρήστης αφού ενημερωθεί για τις available συσκευές θα πρέπει να έχει την δυνατότητα να ανακτήσει δεδομένα από αυτές και να ενημερώνεται δυναμικά για την κατάστασή του "περιβάλλοντος". Για παράδειγμα θα μπορεί να ζητήσει από τον sensor που μετρά τη θερμοκρασία να επιστρέψει την τρέχουσα θερμοκρασία. Ως επέκταση αυτής της λειτουργίας το σύστημα θα πρέπει να έχει την δυνατότητα να αυτοενημερώνεται για την κατάσταση του περιβάλλοντος. Αυτή η λειτουργία μπορεί και να επαναλαμβάνεται κάθε κάποιο χρονικό διάστημα ορισμένο από το χρήστη.

Ακόμη οι ένοικοι θα πρέπει να μπορούν να (απ)ενεργοποιήσούν μέσω του συστήματος μια ηλεκτρική συσκευή. Μέσω της συσκευής αυτής (που στην ουσία είναι ένας actuator) ο χρήστης θα μπορεί να αλλάξει την κατάσταση του περιβάλλοντος. Πχ να ενεργοποιήσει ένα κλιματιστικό και να μεταβάλει την θερμοκρασία..

Συνδυάζοντας τώρα τις δύο προηγούμενες προσφερόμενες λειτουργίες ο χρήστης θα έχει τη δυνατότητα να συνθέσει κάποιους κανόνες (rules), με βάση τους οποίους μια κατάσταση (event) να προκαλεί την εκτέλεση μιας λειτουργίας (πχ όταν η θερμοκρασία ανεβεί πάνω από 25 βαθμούς Κελσίου να ενεργοποιηθεί ο κλιματισμός ή και πιο πολύπλοκες λειτουργίες όπως αν το φως είναι ανοικτό σε ένα δωμάτιο και δεν παρατηρηθεί κίνηση στο συγκεκριμένο δωμάτιο για 30 λεπτά να κλείσουν πχ τα φώτα)

Τέλος θα πρέπει να αναπτυχθεί και ένας eventing μηχανισμός μέσω του οποίου κάποιοι απομακρυσμένοι χρήστες που γίνονται subscribe να ενημερώνονται για κάποιες καταστάσεις μέσω Emails και Sms. Ο χρήστης θα μπορεί ορίζει κάποιες καταστάσεις για τις οποίες θέλει να ενημερώνεται και αν με το τελευταίο update του συστήματος οι καταστάσεις πληρούνται τότε το σύστημα θα αποστέλλει τα ανάλογα notifications.

1.3.2 Απαιτήσεις πιο χαμηλών επίπεδών

Πιο κάτω αναφέρονται requirements που αναφέρονται σε πιο συγκεκριμένους τομείς της εφαρμογής και αφορούν πιο χαμηλά επίπεδα.

Η αλληλεπίδραση του χρήστη με την εφαρμογή θα πρέπει να γίνεται εύκολα και απλά (effortless and seamless)[10]. Δηλαδή η εφαρμογή πρέπει να γίνεται configured και να χρησιμοποιείται εύκολα. Ακόμη θα πρέπει να τοποθετηθεί ένας μηχανισμός authentication του χρήστη ούτως ώστε να μην μπορεί ο οποιοσδήποτε να έχει πρόσβαση στο σύστημα.

Σε ακόμα πιο χαμηλό επίπεδο η εφαρμογή θα πρέπει να υιοθετεί ένα robust μηχανισμό ο οποίος να της επιτρέπει να εντοπίσει γρήγορα άγνωστες devices/services οι οποίες μπορεί να μην είναι συνεχώς συνδεδεμένες ή available στο δίκτυο. Οι διάφορες συσκευές ή υπηρεσίες πιθανών να είναι ετερογενείς και το σύστημα θα πρέπει να είναι σε θέση να καταλάβει τις δυνατότητες τους με ένα ομοιόμορφο τρόπο και να μπορεί να αλληλεπιδράσει μαζί τους αποδοτικά [11][12].

Επίσης θα πρέπει η εφαρμογή να παρέχει και security service. Δηλαδή θα πρέπει να παρέχει ένα μηχανισμό που να κάνει user authentication ούτως ώστέ να μην μπορεί ο οποιοσδήποτε να έχει πρόσβαση στο σύστημα[5]

Τέλος πιθανών η εφαρμογή να περιέχει κάποιους μηχανισμούς caching για να άπαντα αμέσως σε κάποια Get requests χωρίς να χρειάζεται να επικοινωνεί συνεχώς με τον sensor οι οποίοι θα προσδίδουν καλύτερη απόδοση στο σύστημα[5].

1.4 Thesis overview(Επισκόπηση διπλωματικής εργασίας)

Έχοντας πριν ορίσει το "έξυπνο σπίτι" και έχοντας υπόψη τις απαιτήσεις του συστήματος θα προχωρούμε στο σχεδιασμό και την υλοποίηση της εφαρμογής.

Το GUI βασίζεται σε ένα υπάρχων Application Framework το οποίο αναλύεται και περιγράφεται στο Κεφάλαιο 2. Επειδή το υπάρχων Application Framework που χρησιμοποιείται είναι γραμμένο σε JAVA και αφού το GUI μας πρόκειται για Διαδικτυακή εφαρμογή στο Κεφαλαίο 3 γίνεται μια εκτενής έρευνα για Java Web Development για να αποφασιστεί ποια είναι η πιο ιδανική τεχνολογία που πρέπει να χρησιμοποιηθεί. Επίσης μελετάται πιο αρχιτεκτονικό στυλ θα πρέπει ακολουθηθεί ούτως ώστε να καθίσταται η εφαρμογή όσο πιο αποδοτική γίνεται. Τέλος στο ίδιο κεφάλαιο γίνεται μια έρευνα για database services αφού σύμφωνα με τις προδιαγραφές θα χρειαστεί μια βάση δεδομένων για να κρατά πληροφορίες σχετικές με τους χρήστες και τα αιτήματα τους.

Στην συνέχεια έχοντας όλα δεδομένα που χρειαζόμαστε από την έρευνα που προηγήθηκε, στο Κεφάλαιο 4 προχωρούμε στη σχεδίαση και ανάλυση της εφαρμογής. Με το τέλος της σχεδίασης/αναλυσης προχωρούμε στην υλοποίηση του GUI και στο Κεφάλαιο 5 παρουσιάζουμε τις διάφορες λειτουργίες της υλοποιημένης πλέον εφαρμογής.

Προχωρώντας στο Κεφάλαιο 6 γίνεται μια ανασκόπηση και αξιολόγηση όλης της εργασίας και παρατίθενται τα εξαγόμενα συμπεράσματα. Ενώ κλείνοντας στο Κεφάλαιο 7 αναφέρονται πιθανές εφαρμογές ενός τέτοιου συστήματος και προτείνονται μελλοντικές εργασίες που μπορούν να επεκτείνουν ακόμη περισσότερο την εφαρμογή.

Κεφάλαιο 2

Συναφείς εργασίες

2.1 Περιγραφή του υπάρχων Application Framework

Η διπλωματική μου εργασία βασίζεται σε ένα υπάρχων Application Framework (υλοποιημένο στο ETH Πανεπιστήμιο της Ζυρίχης από τον υποψήφιο διδακτορικό φοιτητή του τμήματος Πληροφορικής στο Πανεπιστήμιο Κύπρου Αντρέα Καμηλάρη). Βασικά το GUI αναπτύχθηκε ως επέκταση της προϋπάρχουσας εφαρμογής προσδίδοντας της την ιδιότητα πλέον μιας ολοκληρωμένης διαδικτυακής εφαρμογής.

Επειδή στην ουσία ένα σύστημα smart home είναι ένα embedded distributed system και εμείς θέλουμε να υλοποιηθεί μια Διαδικτυακή εφαρμογή, διαφαίνεται πλέον η ανάγκη για τη δημιουργία ενός είδους μεσολαβητή (πχ ενός gateway) που θα γεφυρώσει τα δύο.

Ο gateway[13] που υπάρχει παρέχει ένα Application Framework ικανό να διαχειριστεί και να αναπαραστήσει αποδοτικά τις embedded devices του συστήματος. Χρησιμοποιώντας κάποια standardized μεθοδολογία ο gateway μπορεί να αλληλεπιδράσει με αυτές, ούτως ώστε να μπορεί σε πρώτη φάση να τις ανακαλύψει, να ενημερωθεί για τις υπηρεσίες που προσφέρουν και τελικά να τις χρησιμοποιήσει.

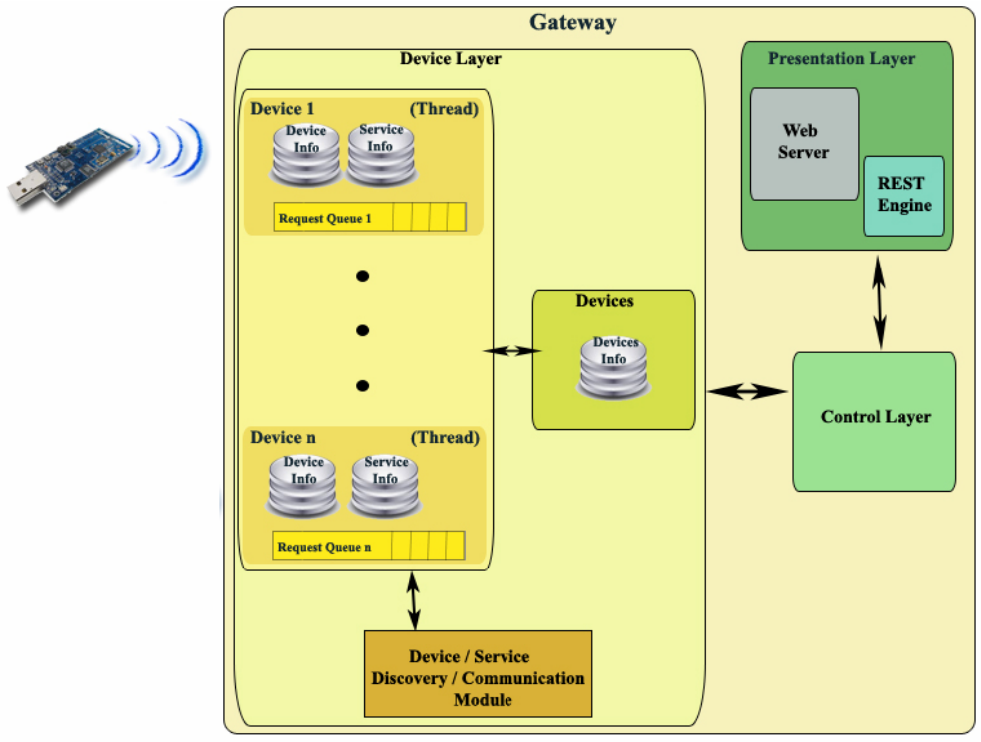
Έχοντας αυτές τις δυνατότητες, το Application Framework αν ενσωματωθεί (encapsulated) σε ένα πιο δυνατή συσκευή μέσα στο υφιστάμενο Network infrastructure όπως για παράδειγμα σε ένα δρομολογητή(router machine) ή και σε ένα

απλό PC θα αποτελέσει πλέον τη γέφυρα που χρειάζεται και θα δώσει πλήρη πρόσβαση από το διαδίκτυο σε αυτές τις embedded devices του Smart Home. Όποτε πλέον μέσω αυτού του gateway οποιαδήποτε Διαδικτυακή εφαρμογή θα μπορεί να τις διαχειριστεί και να τις χρησιμοποιήσει.

2.1.1 Περιγραφή δομής του Gateway

Όπως αναφέρθηκε πιο πάνω ο στόχος του Gateway είναι η αναπαράσταση των διάφορων συσκευών(αισθητήρων) αποδοτικά και αξιόπιστα. Σε αυτό το σημείο γίνεται μια περιγραφή της δομής και του τρόπου λειτουργίας του Gateway σε ψηλό επίπεδο.

Το gateway ακολουθεί αρχιτεκτονική στρωμάτων (layered architecture) και αποτελείται από τρία βασικά στρώματα: το Control Layer, το Presentation Layer και το Device Layer (Σχήμα 2.1).



Σχήμα 2.1 Αρχιτεκτονική του Gateway

Το Control Layer λειτουργεί σαν την "κεντρική μονάδα επεξεργασίας του συστήματος". Τρέχει συνεχώς στο background και εκεί είναι που διενεργούνται όλα τα initializations και όλα τα απαιτούμενα routines κατά την διάρκεια της εκτέλεσης του gateway διασφαλίζοντας συνάμα την ομαλή του λειτουργία σύμφωνα πάντα με τις προδιαγραφές του συστήματος.

Το Presentation Layer έχει την ευθύνη της αναπαράστασης των διαφόρων συσκευών (και των υπηρεσιών τους) στο Διαδίκτυο μέσω ενός web Server Interface. Βασικά υλοποιείται ένας Web Server που δίνει την πρόσβαση στους χρήστες του Διαδικτύου ούτως ώστε οι προηγούμενοι να μπορούν να διαχειριστούν τις συσκευές. Επίσης στο Web Server παρέχεται REST support μέσω του REST Engine οπότε οι χρήστες μπορούν να χρησιμοποιήσουν το REST Architectural Style(Βλέπε Υποκεφ. 2.1.2) για να αλληλεπιδράσουν εμμέσως με τις συσκευές μέσω του Web Server.

Τέλος το Device Layer είναι υπεύθυνο για την επικοινωνία, διαχείριση και γενικά με ότι έχει να κάνει με τις πραγματικές συσκευές. Κάθε device αναπαριστάται ως ένας ξεχωριστός πόρος για τον οποίο γνωρίζουμε πληροφορίες τόσο για αυτόν όσο και για τις υπηρεσίες που προσφέρει. Επίσης κάθε device έχει τη δική του Request queue στην οποία αποθηκεύονται τυχών αιτήματα που αφορούν την συσκευή. Επίσης περιλαμβάνει το devices Module(Βλέπε Υποκεφ. 2.1.2).

2.1.2 Αρχιτεκτονικό Στυλ και συσχέτιση του GUI

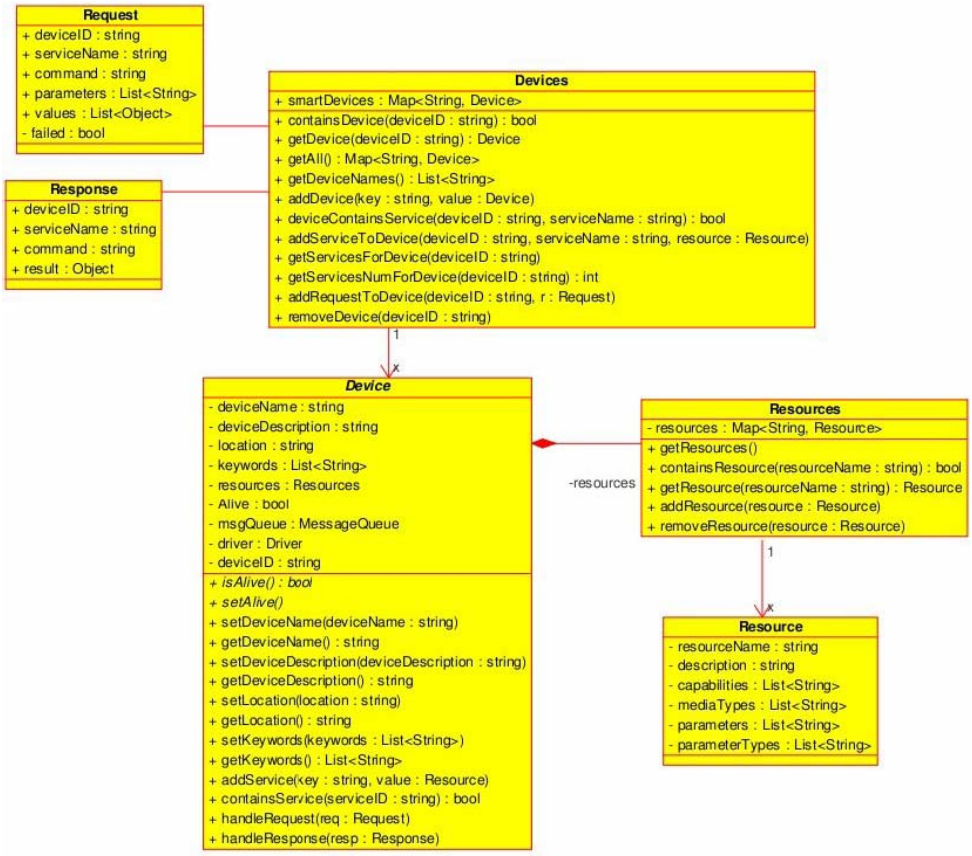
Ένα από τα κύρια κομμάτια του home automation application είναι το Wireless Sensor Network το οποίο περιλαμβάνει πολλές μονάδες (sensors-actuators) οι οποίες δρουν ανεξάρτητα η μια από την άλλη και έχοντας παράλληλα διαφορετικές δυνατότητες.
Ο στόχος είναι να βρεθεί κάποιος αποδοτικός τρόπος όπου μέσω του Internet να μπορούμε να ανακαλύψουμε αυτές τις απομακρυσμένες μονάδες και να επικοινωνήσουμε μαζί τους. Σε επόμενο στάδιο αφού ενημερωθούμε για τις υπηρεσίες που προσφέρουν να μπορούμε να αλληλεπιδράσουμε τελικά μαζί τους. Οπότε πρέπει να βρεθεί μια κοινή "γλώσσα" -τρόπος επικοινωνίας μεταξύ αυτών των μονάδων και του διαδικτύου.

Έχοντας υπόψη μας ότι το κύριο πρωτόκολλο που τρέχει στο διαδίκτυο είναι το HTTP στο οποίο οι διάφοροι πόροι είναι addressable μέσω URIs τότε ένα καλό αρχιτεκτονικό στυλ για το προγραμματισμό της εφαρμογής είναι το REST (Representational State Transfer) architectural style[14].

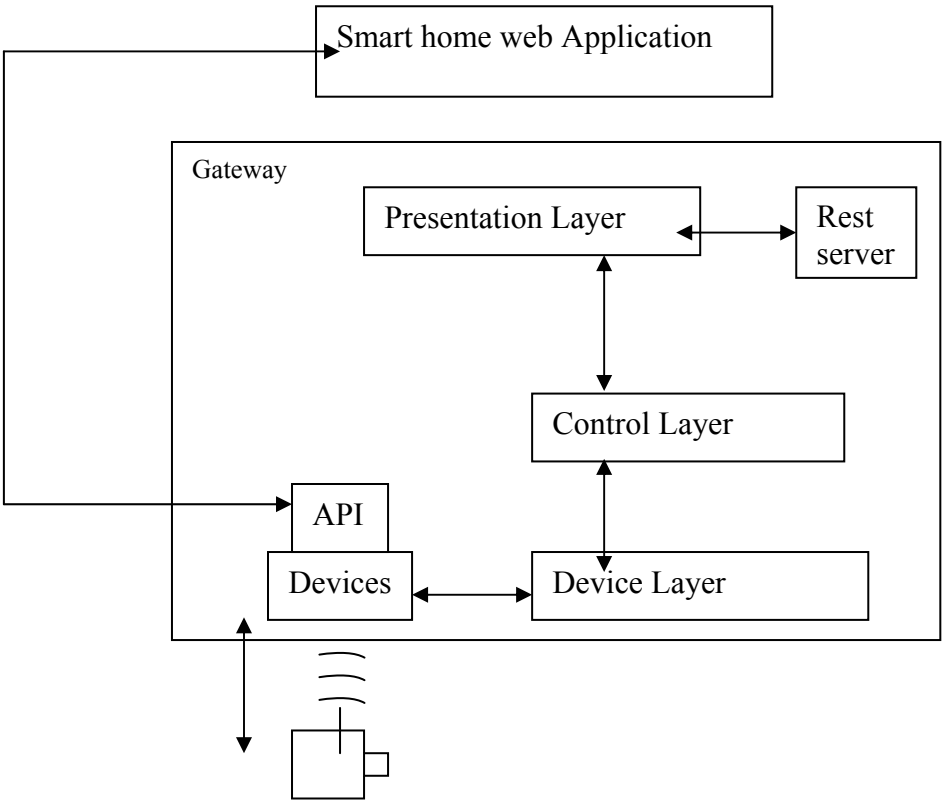
Το REST architectural style βασικά μας παρέχει τη δυνατότητα να χρησιμοποιήσουμε το HTTP σαν ένα application protocol αφού προσφέρει services οι οποίες είναι βασισμένες στο HTTP. Έτσι θα έχουμε την δυνατότητα να εκμεταλλευόμαστε τις δυνατότητες των πόρων μέσω της σειράς μεθόδων που ορίζονται από τα HTTP standards.
Μέσω του REST οι διάφορες μονάδες – resources/devices που εμπεριέχονται στην εφαρμογή μας καθίστανται πλέων addressable μέσω URIs και μπορούμε να έχουμε πρόσβαση σε αυτές μέσω του HTTP χρησιμοποιώντας μεθόδους όπως GET,POST, PUT ,DELETE. Εκείνο που απομένει είναι μια διπροσωπία(API) η οποία θα επιστρέφει τις πληροφορίες στην Διαδικτυακή εφαρμογή ούτως ώστε αύτη να τις παρουσιάζει.

Αυτό το API είναι ακριβώς το τι προσφέρεται από το Devices Module που περιέχεται στο device Layer στο Σχήμα 2.1.Το Devices Module είναι υπεύθυνο για την αλληλεπίδραση ανάμεσα στους "πόρους Devices" και σε πιο ψηλά layers. Εκεί διατηρείται μια λίστα με όλες τις πληροφορίες σχετικά με τις διαθέσιμες συσκευές και υπηρεσίες και παρέχει πρόσβαση σε αυτές μέσω μιας απλής διπροσωπίας(Σχήμα 2.2).
Στο σχήμα 2.3 βλέπουμε ένα γραφικό που περιγράφει ακριβώς την συσχέτιση του GUI application που θα δημιουργηθεί με το υπάρχων application framework.

Με την εγκαθίδρυση του REST ο gateway πλέων δρα όπως ένας κοινός router. Δηλαδή απλά προωθεί πακέτα – request στους πόρους χωρίς να ασχολείται ιδιαίτερα με το περιεχόμενο που αποστέλλεται πίσω στον αιτητή(βλέπε υποκ.4.2.2).



Σχήμα 2.2 API του gateway για επικοινωνία με πιο ψηλά στρώματα



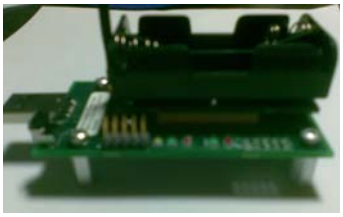
Σχήμα 2.3 Συσχέτιση GUI με το υπάρχων application framework

2.2 Ανάλυση του sensor network και των devices

Ένα sensor network αποτελείται από sensors, actuators και hardware ή software controllers. Για τους σκοπούς της εφαρμογής χρησιμοποιήθηκαν οι Crossbow Telos Sensors(Σχήμα 2.3) και οι MICA(Σχήμα 2.4). Οι οποίοι έχουν αισθητήρες για Θερμοκρασία, Υγρασία, Φωτισμό (Illumination) και σαν actuators παρέχουν λαμπτήρες LED.

Αυτά τα motes δεν περιλαμβάνουν αρχικά IP networking functionality απλά μόνο κάποιους μηχανισμούς ανταλλαγής μηνυμάτων μέσω του λειτουργικού συστήματος του WSN δηλαδή του tinyOs[24]. Όποτε για να είναι δυνατή αποδοτική επικοινωνία από το διαδίκτυο με το WSN έπρεπε να χρησιμοποιηθεί και εδώ το REST architectural style(tinyREST [15][6][16]). Επίσης ακολουθήθηκε centralized αρχιτεκτονική στο δίκτυο αφού ορίστηκε ένας από τους sensors σαν base station ο οποίος λαμβάνει τις διάφορες πληροφορίες από τους άλλους, γίνεται η επεξεργασία στο σύστημα και ο base station πάλι δίνει τις ανάλογες εντολές στους actuators.

Σχήμα 2.3 CrossBow Telos Sensor



Σχήμα 2.4 MICA Sensor

Κεφάλαιο 3

Ανάπτυξη Λογισμικού

3.1 Ανάλυση προδιαγραφών του λογισμικού

Λαμβάνοντας υπόψη το υπόβαθρο, τις διάφορες απαιτήσεις και τους στόχους που πρέπει να πληρεί η εφαρμογή προχωρούμε σε αυτό το κεφαλαίο στην ανάλυση των προδιαγραφών του λογισμικού.

Καταρχήν πρέπει να καθοριστεί ποια τεχνολογία θα χρησιμοποιηθεί για την ανάπτυξη της εφαρμογής. Το GUI που θα αναπτύξουμε πρόκειται για μια Διαδικτυακή εφαρμογή και το υπάρχων Application Framework που χρησιμοποιείται είναι γραμμένο σε JAVA (για λόγους portability and versatility) άρα θα αναπτύξουμε την εφαρμογή με Java Web Development.

Μετά πρέπει να καθοριστεί πιο αρχιτεκτονικό στυλ θα ακολουθηθεί στο λογισμικό. Λαμβάνοντας υπόψη το υφιστάμενο Application Framework συμπεραίνουμε ότι θα πρέπει να διατηρηθεί το REST architectural style για την απλότητα του και για τα πλεονεκτήματα που αναφέρονται στο υποκεφ. 3.3 .

Επίσης από τις προδιαγραφές κρίνεται αναγκαία η εισαγωγή μιας βάσης δεδομένων. Η βάση αυτή θα διατηρεί πληροφορίες αναγκαίες για την ομαλή λειτουργία του συστήματος.

Τέλος επισημαίνεται ότι η εφαρμογή θα πρέπει να ακολουθεί το client-server μοντέλο στο οποίο όπως αναλύεται στο Κεφάλαιο 4 όλο το functionality/Intelligence της εφαρμογής, (δηλαδή σε μας ο gateway) πρέπει θα ανήκει στην πλευρά του server ενώ το user Interface στην πλευρά του client.

3.2 Έρευνα για JAVA web development

Σε αυτό το σημείο προχωρούμε σε μια εκτενή έρευνα για Java Web Development για να αποφασιστεί ποια είναι η πιο ιδανική τεχνολογία που πρέπει να χρησιμοποιηθεί ώστε να έχουμε μια όσο πιο αποδοτική[16] εφαρμογή γίνεται .

3.2.1 Java Server Pages[31]

Η τεχνολογία JSP επιτρέπει την ανάμειξη στατικού HTML με δυναμικού. Τα JSP επιτρέπουν στους σχεδιαστές την ταχεία ανάπτυξη Web-based applications οι οποίες είναι platform independent. Πολύ σημαντικό στοιχείο σε αυτή την τεχνολογία αποτελεί το γεγονός ότι τα JSP διαχωρίζουν το User Interface του Application σε σχέση με το πως λειτουργεί το application για να παράξει την τελική πληροφορία. Έτσι με αυτό επιτρέπεται στο οποιονδήποτε σχεδιαστή να μπορεί να αλλάξει το page layout χωρίς οποιαδήποτε αλλαγή στο τρόπο που λειτουργεί το application και παράγει δυναμικά πληροφορίες. Τα JSP στην ουσία μεταφράζονται σε servlets(3.2.2) τα οποία μετά μπορούν να μεταγλωττιστούν χρησιμοποιώντας ένα standard java compiler. Έτσι διατηρούν με αυτό τον τρόπο όλες τις δυνατότητες που παρέχονται από τις βιβλιοθήκες της Java. Η JSP γλώσσα είναι εύκολα επεκτάσιμη με βιβλιοθήκες (tag libraries) που επιτρέπουν τον εύκολο σχεδιασμό πολύπλοκων Web Application. Τέλος η συγγραφή και συντήρηση σελίδων είναι εύκολη αφού η γλώσσα JavaServer Pages Standard Tag Library (JSTL) (η οποία είναι ακόμη μια συλλογή από tag

libraries) υποστηρίζεται από τη JSP και παρέχει πληθώρα από λειτουργίες ικανές να υποστηρίξουν general-purpose functionality για Web Applications.

3.2.2 **Java Servlets**[31]

Είναι κάποια platform-independent, server-side modules που μπορούν να επεκτείνουν τις δυνατότητες κάποιου Web Server ούτως ώστε να μπορεί να επεξεργαστεί κάποια requests(GET,POST) και να δημιουργήσει responses δυναμικά μετά από επεξεργασία δεδομένων εισόδου από κάποιο προκαθορισμένο κώδικα.

Όταν ο πελάτης (client) καλεί ένα servlet τότε αυτό λαμβάνει δύο αντικείμενα (objects): Ένα ServletRequest, που εξασφαλίζει την επικοινωνία από τον πελάτη προς τον server και ένα ServletResponse, που εξασφαλίζει την επικοινωνία από το servlet πίσω στον πελάτη.

Το servlet framework περιέχει ένα Java-based API εξειδικευμένο για προγραμματισμό διαδικτυακών εφαρμογών. Το βασικό πακέτο των servlet χρησιμοποιεί "αντικείμενα" Java για να αναπαραστήσει τόσο τα request και τα Response αλλά και τα configuration του servlet μαζί με το κώδικα που θα κάνει την επεξεργασία των δεδομένων.

3.2.3 **AJAX**

Η τεχνολογία AJAX (Asynchronous JavaScript and XML) είναι ένα σύνολο από τεχνικές για Web Development από πλευράς πελάτη(client-side) και χρησιμοποιούνται στη δημιουργία interactive Web Applications. Μέσω του Ajax τα διάφορα web applications έχουν την δυνατότητα να ανακτούν δεδομένα από τους servers ασύγχρονα στο background χωρίς επηρεάζουν το παρουσιαστικό και συμπεριφορά τους. Δηλαδή δεν "παγώνουν " περιμένοντας να ολοκληρωθεί η ανάκτηση των δεδομένων.

Πλεονεκτήματα είναι η καλύτερη ποιότητα web services λόγω του ασύγχρονου mode. Επίσης το AJAX επιτρέπει την δημιουργία πιο interactive και πιο δυναμικών διαπροσωπιών για τις σελίδες αφού ουσιαστικά κάνουν τον browser του χρήστη να συμπεριφέρεται περισσότερο ως ένα desktop Application που μπορεί να αλληλεπιδρά με το server πολύ πιο αποδοτικά.

3.2.4 Google Web Toolkit-GWT

Το GWT[25] είναι ένα open source software development framework (σύνολο από εργαλεία) που χρησιμοποιούνται στη δημιουργία complex JavaScript front-end applications σε Java. Χρησιμοποιείται στο user interface programming καθώς επίσης στην υλοποίηση λειτουργιών client-side JavaScript. Επίσης ο κώδικας Java που γράφεται μπορεί να μεταφραστεί εύκολα σε AJAX. Στο GWT ο κώδικας της εφαρμογής χωρίζεται σε δύο κατηγορίες τον server side code και τον client side code.

Ο server side code που όπως καταλαβαίνουμε και από το όνομα της κατηγορίας είναι κώδικας Java που τρέχει στην πλευρά του server. Ο server side code μεταγλωττίζεται κανονικά σε Java Bytecode και τρέχει σε οποιοδήποτε σύστημα που έχει JVM. Εκεί συνήθως βρίσκεται όλο το Functionality της εφαρμογής.

Ο client side code συνήθως είναι ο κώδικας που αφορά την παρουσίαση των αποτελεσμάτων της εφαρμογής(user interface). Ο client side code μετατρέπεται από το compiler του GWT σε browser-compliant HTML και JavaScript κώδικα που τρέχει locally στην πλευρά του χρήστη.

Ο client code μπορεί να επικοινωνήσει με το server side και να λάβει τα δεδομένα που πρέπει να παρουσιάσει μέσω RPC (Υποκεφ. 4.2.1)

3.2.5 Σύγκριση τεχνολογιών συμπεράσματα

Από την πιο πάνω έρευνα και με βάση την εφαρμογή που θέλουμε να αναπτύξουμε η πιο ιδανική τεχνολογία είναι σαφώς το GWT γιατί συνδυάζει πλεονεκτήματα από όλες τις άλλες τεχνολογίες.

Καταρχήν αφού η εφαρμογή μας ακολουθεί το client server μοντέλο τότε τα JSPs δεν μπορούν να χρησιμοποιηθούν καθώς αποτελούν client side modules. Επίσης επειδή στην εφαρμογή μας δεν θα έχουμε πάντα σταθερό response time από τον gateway δεν μπορούμε να χρησιμοποιήσουμε τα Java Servlets αφού με αυτή την τεχνολογία η εφαρμογή δεν έχει την δυνατότητα να ανακτά δεδομένα από τους servers ασύγχρονα στο background όπως εξηγήθηκε πιο πάνω.

Τώρα συγκρίνοντας την τεχνολογία AJAX με το GWT το πιο ιδανικό για μας είναι σίγουρα το GWT. Καταρχήν η τεχνολογία AJAX έχει ένα σοβαρό μειονέκτημα. Υπάρχουν προβλήματα compatibility ανάμεσα στους διάφορους browsers και operating systems λόγο του ότι οι πρώτοι υλοποιούν τα JavaScript διαφορετικά. Αυτή η κατάσταση επιλύεται στο GWT καθώς σε αυτό το περιβάλλον εξακολουθούμε να έχουμε το πλεονέκτημα των AJAX αλλά εδώ παρέχεται internationalization and browser portability. Επίσης το GWT έχει υψηλή αποδοτικότητα και επίσης παρέχονται libraries που βοηθούν στην δημιουργία πολύπλοκών διαδραστικών user interfaces. Ακόμη η δομή και ο τρόπος λειτουργίας είναι ιδανικός για την εφαρμογή μας(βλέπε Υποκεφ. 4.1.1). Τέλος σημαντικό πλεονέκτημα είναι ότι υπάρχει plug in του GWT για το IDE eclipse γεγονός που καθιστά την ανάπτυξη του application ακόμα πιο εύκολη

3.3 Αρχιτεκτονικό Στυλ.

Όπως βλέπουμε από τις προδιαγραφές που θέσαμε το αρχιτεκτονικό στυλ που θα ακολουθεί είναι το REST architectural style[17]. Πιο κάτω γίνεται μια ανάλυση αυτού του αρχιτεκτονικού στυλ και επισημαίνονται τα πλεονεκτήματα που προσδίδει στην εφαρμογή μας.

3.3.1 Rest architectural Style- Πλεονεκτήματα

Το αρχιτεκτονικό στυλ Rest θεωρείται ένα πολύ πετυχημένο μοντέλο για network Applications (όπως τη δική μας) στο Διαδίκτυο. Στο Rest θεωρούμε τα πάντα στο δίκτυο (δεδομένα, συσκευές, υπηρεσίες) σαν πόρους και εγκαθιδρύονται ανάλογοι μηχανισμοί (σύνολο μεθόδων με καθορισμένη σημασιολογία) μέσω τους οποίους μπορούμε να διαχειριστούμε τους πόρους αυτούς.

Όπως αναφέρθηκε και στο υποκεφάλαιο 2.1.2 μέσω του REST οι διάφορες μονάδες – devices (πλέον πόροι) που εμπεριέχονται στην εφαρμογή μας καθίστανται addressable μέσω URIs και μπορούμε πλέον να έχουμε πρόσβαση σε αυτές από το διαδίκτυο μέσω HTTP verbs.

Το GET αποτελεί την πιο σημαντική " μέθοδο " του REST καθώς αν το καλέσουμε με URI κάποιου πόρου τότε παίρνουμε πίσω μια πλήρη αναπαράσταση του πόρου αυτού. Μέσα σε αυτή περιλαμβάνονται μεταξύ άλλων γενικές πληροφορίες για τον πόρο, όλες τις μεθόδους που υποστηρίζονται (μαζί με πληροφορίες για τις σχετικές παραμέτρους αν αυτές χρειάζονται) καθώς και τυχών πληροφορίες για ενσωματωμένους πόρους.

Άλλες μέθοδοι είναι το PUT που χρησιμοποιείται για την εισαγωγή ενός νέου πόρου ή για update, το POST μέσω του οποίου μπορούμε να μεταβάλουμε την κατάσταση ενός πόρου και τέλος το delete που χρησιμοποιείται για την διαγραφή κάποιου πόρου.

Το κύριο πλεονέκτημα του Rest αποτελεί το γεγονός ότι βρισκόμαστε πλέον σε θέση να διαχειριστούμε πολλαπλά devices όχι απαραίτητα ομοιογενή και να τα κάνουμε να αλληλεπιδράσουν μεταξύ τους (interoperability).

Άλλα πλεονεκτήματα είναι το χαμηλό bandwidth που χρησιμοποιείται και τέλος η ταχύτητα και ευελιξία που προσφέρει μια τέτοια αρχιτεκτονική στην εφαρμογή.

3.4 Έρευνα για lightweight Database service

Σε αυτό το σημείο γίνεται μια έρευνα σχετικά με Database services για να βρεθεί η πιο κατάλληλη που να πληρεί τις προδιαγραφές που θέσαμε.

Η εφαρμογή πρέπει να είναι όσο πιο lightweighted γίνεται αφού όπως αναφέρθηκε πιο πριν θα πρέπει να μπορεί να ενσωματωθεί σε άλλες συσκευές του δικτύου. Από πριν γνωρίζουμε ότι στην εφαρμογή τρέχει ήδη ένα web server service καθώς επίσης και η REST engine οπότε θα πρέπει να αποφύγουμε την επιβάρυνση του συστήματος και με ακόμα ένα server service για βάση Δεδομένων.

Στην αρχή μελετήθηκε η postgreSQL γιατί είναι σχετικά ένα light version της Sql αλλά διαπιστώθηκε ότι δεν ήταν κατάλληλη γιατί για να λειτουργήσει προϋπόθετε την ύπαρξη DB server service και εκτός αυτού δεν μπορούσαμε εύκολα να αλληλεπιδράσουμε με τη βάση μέσω της Java.

Τελικά η πιο ιδανική Database service για την εφαρμογή μας ήταν η SQLite[27]. Η SQLite αποτελεί βασικά μια in-process βιβλιοθήκη που υλοποιεί μια serverless transactional SQL database engine που είναι ακριβώς αυτό που χρειαζόμαστε.

Η SQLite είναι self-contained δηλαδή, έχει ελάχιστες απαιτήσεις από το λειτουργικό σύστημα ή από external libraries. Αυτό την καθιστά ιδανική για embedded συσκευές που δεν έχουν την υποδομή που προσφέρεται πχ από ένα desktop computer, καθώς και σε εφαρμογές που πρέπει να τρέχουν σε διάφορα υπολογιστικά συστήματα χωρίς πολλές αλλαγές.

Επίσης δεν χρειάζεται κάποιο setup για να χρησιμοποιηθεί αφού δεν υπάρχει όπως είπαμε κάποιο server process που χρειάζεται να ξεκινήσει ή να ρυθμιστεί.

Ακόμη η SQLite γράφει και διαβάζει σε συνηθισμένα αρχεία στο δίσκο παρέχοντας όλες της δυνατότητες μιας σχεσιακής βάσης δεδομένων
Για να μπορούμε να έχουμε πρόσβαση στη βάση μέσω της Java υπάρχει το SQLiteJDBC[28] που απλά ενσωματώνουμε στη εφαρμογή και έχουμε πλέον την δυνατότητα ν αλληλεπιδράσουμε με τη βάση μέσω μιας σειράς απλών μεθόδων.

Κεφάλαιο 4

Ανάλυση – Σχεδίαση του GUI

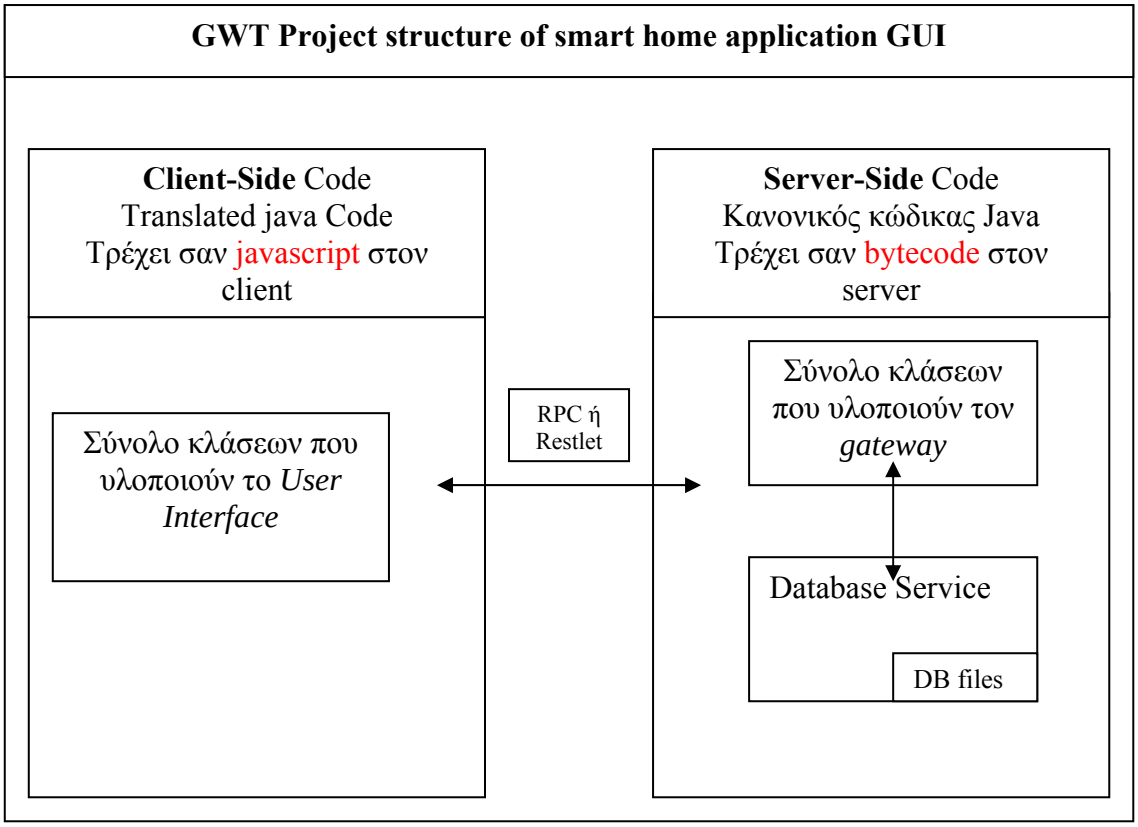
4.1 Γενικές Πληροφορίες και Περιγραφή δομής

Σύμφωνα με τις απαιτήσεις ο οποιοσδήποτε χρήστης θα πρέπει να έχει πρόσβαση στο σύστημα μέσω οποιασδήποτε συσκευής που του παρέχει πρόσβαση στο Ίντερνετ. Αυτό σημαίνει ότι η εφαρμογή πρέπει να κτιστεί έτσι ώστε να μην προϋποθέτει την εγκατάσταση συγκεκριμένου λογισμικού στην πλευρά του πελάτη. Θεωρούμε ότι ο πελάτης μας το μόνο "λογισμικό" πού έχει στην κατοχή του είναι ένας απλός Web browser. Συνεπώς προκύπτει ότι για τη δομή της εφαρμογή θα πρέπει να ακολουθηθεί το client – server μοντέλο όπου το client part πρέπει να τρέχει locally στον πελάτη προσφέροντας του μια διαπροσωπία μέσω της οποίας να μπορεί ο χρήστης να διαχειριστεί το σύστημα.

Στην περίπτωση μας η λειτουργικότητα πίσω από τον client (User Interface) θα είναι να επιτρέπει στο χρήστη να αλληλεπιδράσει με το σύστημα δηλαδή να του επιτρέπει να κάνει τα request που επιθυμεί και μετά να του παρουσιάζει τα ανάλογα αποτελέσματα. Σημαντικό είναι το γεγονός ότι οι διάφοροι υπολογισμού που απαιτούνται για την εξαγωγή των αποτελεσμάτων δεν θα πρέπει να γίνονται τοπικά άλλα απομακρυσμένα στο server side. Αντιθέτως ο server δεν θα έχει καμία σχέση με την παρουσίαση των

αποτελεσμάτων ή με το user Interface οπότε έχουμε καλύτερή επίδοση του συστήματος, χαμηλότερο bandwidth και κυρίως μειωμένο φόρτο εργασίας για το server. Άπλα θα δέχεται τα διάφορα Requests από το client side παράγει τα ανάλογα responses(αποτελέσματα) και τα αποστέλλει πίσω.

Όπως αναλύθηκε πιο πάνω, ο κώδικας του client πρέπει να τρέχει locally στο browser και αυτό συνεπάγεται ότι θα πρέπει στην ουσία να αποτελεί ένα JavaScript. Από τη μελέτη που έγινε στο κεφαλαίο 3 εξάγεται ότι μέσα από το GWT client side μπορούμε να πετύχουμε σε πρώτη φάση αυτό το στόχο. Όποτε εκεί θα μπουν όλες οι αναγκαίες κλάσεις της Java που θα υλοποιούν το UserInterface της εφαρμογής μας. Στο server-side code τώρα του GWT project θα μπουν βασικά όλες οι κλάσεις που υλοποιούν το functionality και στην περίπτωση μας αυτές δεν είναι άλλες από το σύνολο των κλάσεων που υλοποιούν το gateway. Επίσης στο server side θα εγκατασταθεί και το database service. Για την πρόσβαση στη βάση έχουν αναπτυχθεί μια σειρά κλάσεων μέσα στον gateway παρέχουν την δυνατότητα διαχείρισης της. Στο σχήμα 4.1 βλέπουμε μια γραφική αναπαράσταση της δομής αυτής .



Σχήμα. 4.1 Ανάλυση της δομής της εφαρμογής βάση του GWT project structure

4.1.1 GWT project structure

Προχωρώντας τώρα με τη σχεδίαση της εφαρμογής σε αυτό το σημείο γίνεται μια εκτενής περιγραφή του user Interface part και περιγράφονται τα δομικά στοιχεία που το αποτελούν.
Στο GWT τα δομικά στοιχεία τα οποία αποτελούν ένα User Interface ονομάζονται widgets. Τα widgets προσφέρονται για τη δημιουργία διαδραστικών διαπροσωπιών και χρησιμοποιούνται εκτενώς στη δημιουργία GUIs. Στο περιβάλλον που χρησιμοποιούμε έχουμε διαθέσιμη μια μεγάλη βιβλιοθήκη από τα widgets. Στο widget list περιλαμβάνονται μεταξύ άλλων buttons, labels, textboxes, radio buttons, listBoxes, Panels και FlexTables. Τα Panels είναι βασικά containers και μπορούν να ενθυλακώσουν μέσα τους άλλα widgets ή ακόμη και άλλα Panels.

Με την δημιουργία ενός νέου project στο GWT δημιουργούνται κάποια αρχεία τα οποία αναφέρονται επιγραμματικά σε αυτό το σημείο καθώς θα γίνεται μετά αναφορά σε αυτά. Αν υποθέσουμε ότι το νέο project ονομάζεται smartHome τότε θα παραχθούν κάποια αρχεία από τα οποία τα πιο κύρια είναι τα εξής:

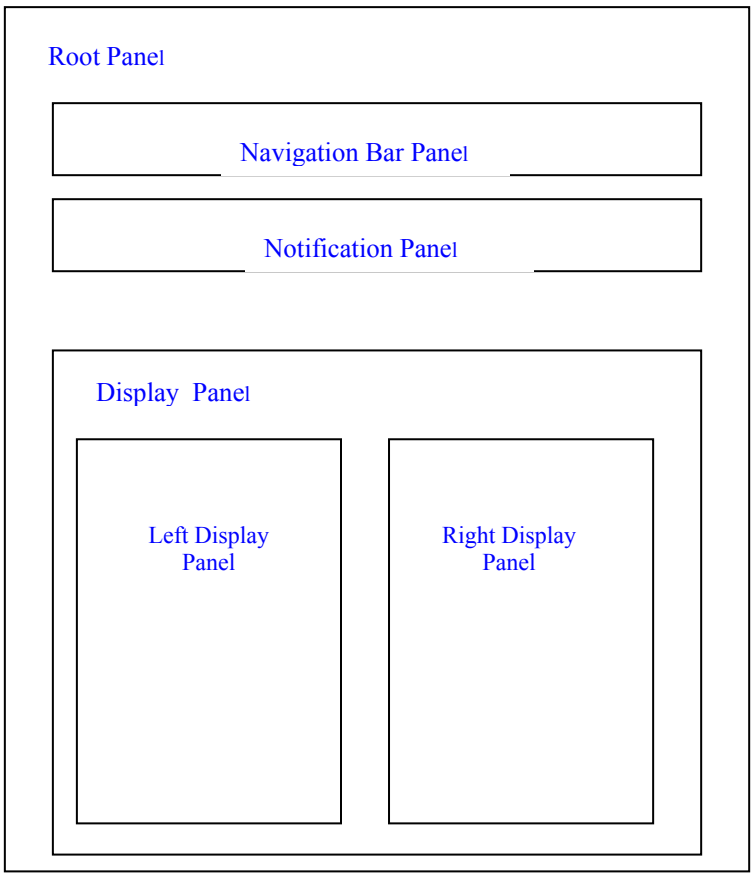
Το smartHome.html που αποτελεί την ιστοσελίδα εκείνη με την οποία θα γίνει bind το URL στο οποίο θα τρέχει η εφαρμογή. Επίσης στο GWT μπορούμε να καθορίσουμε την μορφή των widgets με css styles τα οποία ορίζονται στο αρχείο smartHome.css το οποίο γίνεται επίσης bind με την ιστοσελίδα. Επίσης δημιουργούνται τα πακέτα που θα περιλαμβάνουν το client-side code και τον server –side code(βλέπε υποκ.3.2.4).

Στο client –side package δημιουργείται το smartHome.java που αποτελεί το GWT entry point class η οποία κάνει implement την GWT interface EntryPoint. Σε αυτή την κλάση περιέχεται η μέθοδος onModuleLoad η οποία καλείται αμέσως μόλις γίνει launched η εφαρμογή καθώς η κλάση smartHome.java ορίστηκε ώς entry point. Μέσα στην onModuleLoad ορίζεται ένα container panel (RootPanel) που περιέχει όλα τα άλλα widget της εφαρμογής και αυτό το panel συνδέεται με την ιστοσελίδα όπου στην ουσία περικλείει όλο το body του html αρχείου και έτσι πλέον τα δεδομένα που τροφοδοτούν την σελίδα δεν θα είναι στατικά αλλά δυναμικά και θα παράγονται από το GWT βάση της εφαρμογής μας .

4.1.2 Client side: αλληλεπίδραση με το χρήστη

Γνωρίζοντας τη δομή πλέον του GWT project προχωρούμε πλέον σε αυτό το σημείο στη σχεδίαση του δικού μας User Interface. Μετά αναφέρονται επιγραμματικά οι διάφορες λειτουργίες που θα είναι διαθέσιμες προς τον χρήστη.

Μετά από μελέτη των απαιτήσεων και έχοντας πάντα υπόψη ότι το GUI πρέπει να είναι απλό και εύχρηστο και να μην προαπαιτείται από το χρήστη να έχει εξειδικευμένες γνώσεις προτείνεται για το σχεδιασμό το template που φαίνεται στο σχήμα 4.2



Σχήμα 4.2 Panel Template of User Interface

Στο navigation Panel θα μπει μια σειρά από buttons το καθένα από τα οποία θα εκτελεί και από μια λειτουργία. Στο notification Panel θα δίνονται συνεχώς οδηγίες χρήσης του συστήματος στο χρήστη. Ενώ στο display Panel που θα διαθέτει δύο οθόνες όπου θα παρουσιάζονται τα αποτελέσματα κάθε φορά, ανάλογα με button που θα πιεστεί.

Τώρα προχωρούμε με τις λειτουργίες του User Interface. Καταρχήν η πρώτη οθόνη που να παρουσιάζεται στο χρήστη θα είναι μια Log in screen όπου στην ουσία μπορούμε να εισάξουμε ένα νέο χρήστη στο σύστημα ή αν ο χρήστης προϋπάρχει γίνεται το authentication του. Μετά από επιτυχές authentication εμφανίζεται στο χρήστη το κυρίως μενού της εφαρμογής όπου από εκεί θα μπορεί πατώντας τα ανάλογα buttons θα έχει πρόσβαση στις διάφορες υπηρεσίες οι οποίες είναι: γραφική αναπαράσταση διαθέσιμων πόρων, ανάκτηση πληροφοριών από το σύστημα, αλληλεπίδραση με κάποιο actuator, request builder service, μηχανισμοί eventing, mashUps και ανάκτηση γενικών πληροφοριών του gateway(εκτενής περιγραφή των υπηρεσιών γίνεται στο Κεφάλαιο 5).

4.1.3 Server side: Πόροι και προσφερόμενες λειτουργίες

Όταν στην εφαρμογή μας η οποία όπως είδαμε πιο πριν τρέχει στο browser του χρήστη παρουσιαστεί η ανάγκη για λήψη κάποιων δεδομένων που πρέπει να παρουσιαστούν στο χρήστη, είτε για αποθήκευση κάποιας πληροφορίας τότε πρέπει να επικοινωνήσει - αλληλεπιδράσει με τον server. Για το σκοπό αυτό δημιουργείται ένα HTTP request και αποστέλλεται στο server side χρησιμοποιώντας το restlet Framework υποκεφ. 4.2.

Το κύριο κομμάτι του server side code της εφαρμογής μας είναι ο gateway που περιγράφεται στο υποκεφ. 2.1 στον οποίον αφού ακολουθείται το REST τα πάντα θεωρούνται σαν πόροι[11]. Έτσι το κάθε HTTP request αφορά ένα συγκεκριμένο πόρο και εξυπηρετείται ανάλογα από το gateway.

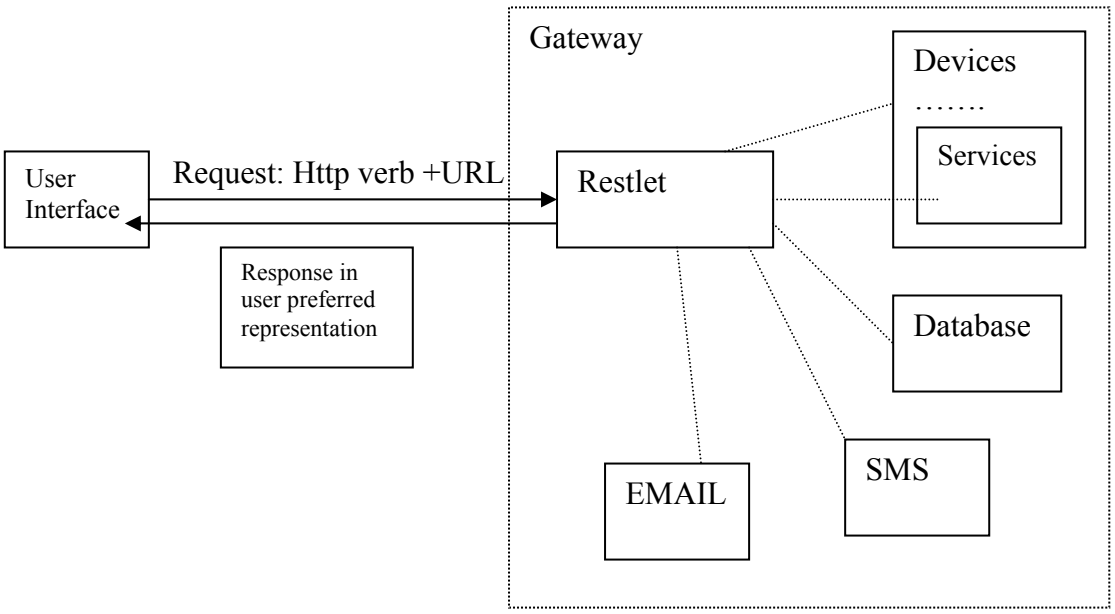
Οι κύριοι πόροι του gateway είναι τα devices. Το κάθε device περιέχει τις προσφερόμενες υπηρεσίες που προσφέρει. Ενώ η κάθε υπηρεσία περιέχει τις παραμέτρους που χρειάζεται και τον τύπο της.

Έτσι ο gateway είναι σε θέση να δεχθεί ένα HTTP request και με βάση το restlet url να βρει το πόρο στον οποίο αναφέρεται το request να τον κάνει invoke και να εξυπηρετήσει το αίτημα (Σχήμα 4.3).

Για παράδειγμα ένα request του τύπου *GET …/devices/2* στον gateway έχει ως αποτέλεσμα να επιστραφούν τα πληροφορίες και τα services σχετικά με το device 2.
Επίσης ένα request του τύπου *GET …/devices/2/services/temperature* επιστρέφει πληροφορίες σχετικές με το service temperature του device 2.

Όποτε ο gateway μπορεί να εξυπηρετήσει με αυτό τον τρόπο από πολύ απλές λειτουργίες όπως τα πιο πάνω request ή πολύ πιο πολύπλοκές χρησιμοποιώντας συνδυασμούς υπηρεσιών όπως θα δούμε στο Κεφάλαιο 5.

Άλλος πόρος του Server side είναι φυσικά η βάση δεδομένων (που οι λειτουργίες της περιγράφονται στο υποκεφ. 4.1.4).
Επίσης πόροι του συστήματος μπορούν να θεωρηθούν το Email και Sms sending capability που υπάρχει στο σύστημα των οποίων η λειτουργία περιγράφεται στο μηχανισμό eventing (υποκεφ. 5.7)



Σχήμα 4.3 Χρήση πόρων από το User Interface

4.1.4 Βάση δεδομένων- λειτουργικότητα

Η βάση δεδομένων είναι ένας από τους πόρους που γίνονται συχνά invoke όχι μόνο από τη πλευρά του client αλλά και από την πλευρά του server απευθείας.
Η βάση δεδομένων χρησιμοποιείται καταρχήν για την αποθήκευση τον διάφορων πληροφοριών και έλεγχο κατά την εισαγωγή ενός νέου χρήστη στο σύστημα.
Άλλη χρήση της είναι φυσικά στην φάση του authentication του χρήστη αφού είναι από εκεί που λαμβάνονται τα δεδομένα για να εξακριβωθεί το validity του (username-password).
Άλλο σημείο που χρειάζεται η βάση είναι στην υλοποίηση του μηχανισμού eventing(υποκεφ. 5.7) όπου καταρχήν χρειάζεται στην αποθήκευση των καταστάσεων για τις οποίες θέλει να ενημερώνεται ο χρήστης καθώς επίσης και στη εύρεση της ηλεκτρονικής διεύθυνσης και του αριθμού τηλεφώνου του για την αποστολή του notification.

Τέλος η βάση είναι αναγκαία στο κτίσιμο των physical MashUps(Υποκεφ 5.8) όπου και πάλι αποθηκεύονται οι διάφορες καταστάσεις του περιβάλλοντος καθώς και οι πράξεις που πρέπει να γίνουν εκ μέρους των actuator σε περίπτωση ικανοποίησης των προηγουμένων.

4.2 Επικοινωνία μεταξύ client και server

Σε αυτό σημείο αφού έχουμε να κάνουμε με ένα client/sever application γίνεται μια σύγκριση δύο μηχανισμών που παρέχουν την επικοινωνία μεταξύ του client side και του server side για να διαπιστωθεί ποιος είναι ο πιο κατάλληλος για να χρησιμοποιηθεί.

Σχετικά με την επικοινωνία πρέπει να τονιστεί το γεγονός ότι η επικοινωνία με το server στο GWT γίνεται ασύγχρονα σαν ένα AJAX application. Αντιθέτως με τις παραδοσιακές HTML web applications, οι AJAX όντας περισσότερο applications που τρέχουν στο browser δεν χρειάζεται κάθε φορά που θα γίνει ένα update στο user Interface να ζητούν και νέα html σελίδα από τον εξυπηρετητή. Όμως εξακολουθεί να υπάρχει η ανάγκη για επικοινωνία με το server όπως εξηγήθηκε στο προηγούμενο υποκεφάλαιο. Έτσι ο μηχανισμός που υπάρχει στο GWT για την αλληλεπίδραση με τον server μέσω του δικτύου είναι τα RPCs

4.2.1 RPC

Τα RPC (Remote Procedure Calls) αποτελούν βασικά ένα μηχανισμό του GWT που επιτρέπει στον client και το server να ανταλλάσσουν Java objects μέσω HTTP. Στα RPC ο κώδικας της κλάσης που ανήκει στο server side και τον οποίο θέλουμε να καλέσουμε από τον client side ονομάζεται service (στην ορολογία του GWT)

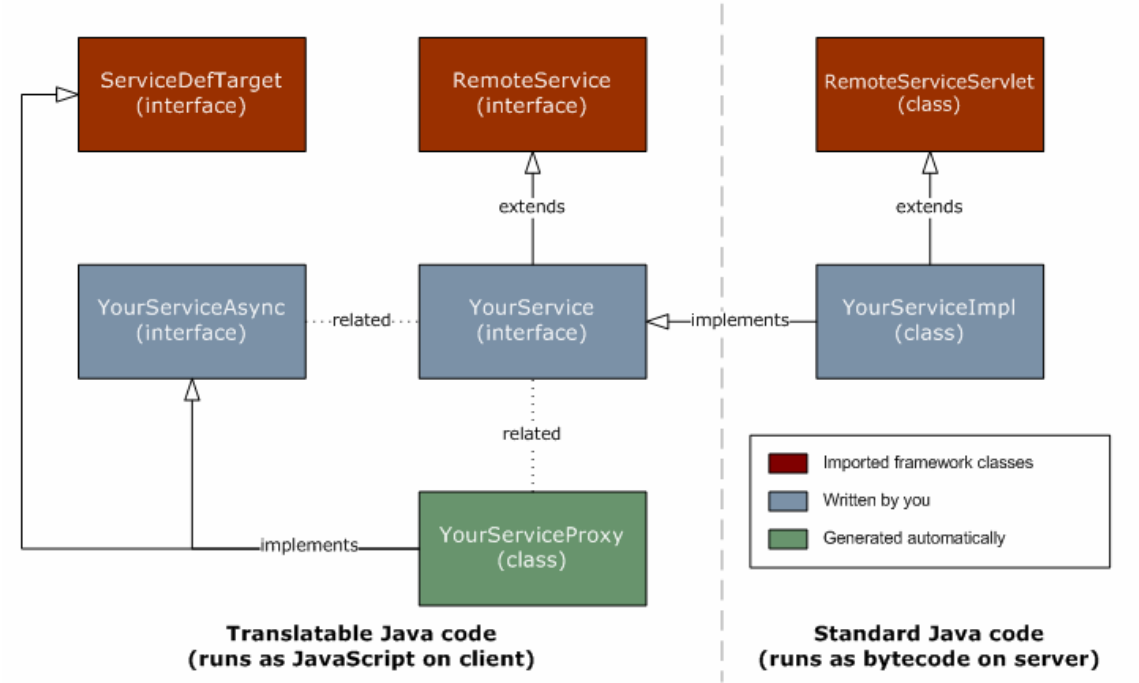
Προτού να είμαστε σε θέση να καλέσουμε ένα GWT service μέσω RPC υπάρχει μια διαδικασία που πρέπει να γίνει αφού θα πρέπει να οριστούν 3 συγκεκριμένες κλάσεις.

Η πρώτη κλάση που θα πρέπει να οριστεί είναι ένα interface του GWT-service στην πλευρά του client που να κάνει extend τη RemoteService και μέσα να περιέχει τα ονόματα των μεθόδων της κλάσης του server που θα θέλαμε να γίνονται invoke μέσω RPC.
Η δεύτερη κλάση ορίζεται επίσης στην πλευρά του client και αποτελεί ένα ασύγχρονο αυτή τη φορά interface του service και βασίζεται στο πρώτο interface και ορίζει το callback object. Επειδή η επικοινωνία με τον server γίνεται ασύγχρονα και αφού δεν

γνωρίζουμε πότε ακριβώς θα επιστρέψει το response ο server, η κλάση που κάνει το request αποστέλλει και το λεγόμενο call back object που είναι εκεί που θα αποθηκευτεί το response που παράγεται μετά την εξυπηρέτηση του request από το server. Έτσι τελικά η επικοινωνία από την πλευρά του server προς τον client γίνεται μέσω αυτού του αντικειμένου.

Τρίτη κλάση είναι το Implementation του service που ορίζεται στην πλευρά του server. Δηλαδή είναι ο κώδικας ο οποίος υλοποιεί το service στο οποίο κάνουμε το RPC. Η κλάση αυτή κάνει extend τη RemoteServiceServlet και υλοποιεί στην ουσία το interface που ορίσαμε στη 1η κλάση καθώς και το σύνολο των απαραίτητων μεθόδων.

Στο σχήμα 4.4 μπορούμε να δούμε μια γραφική αναπαράσταση της διαδικασίας που περιγράψαμε πιο πάνω για να ορίσουμε ένα GWT service



Σχήμα 4.4 Δημιουργία GWT service

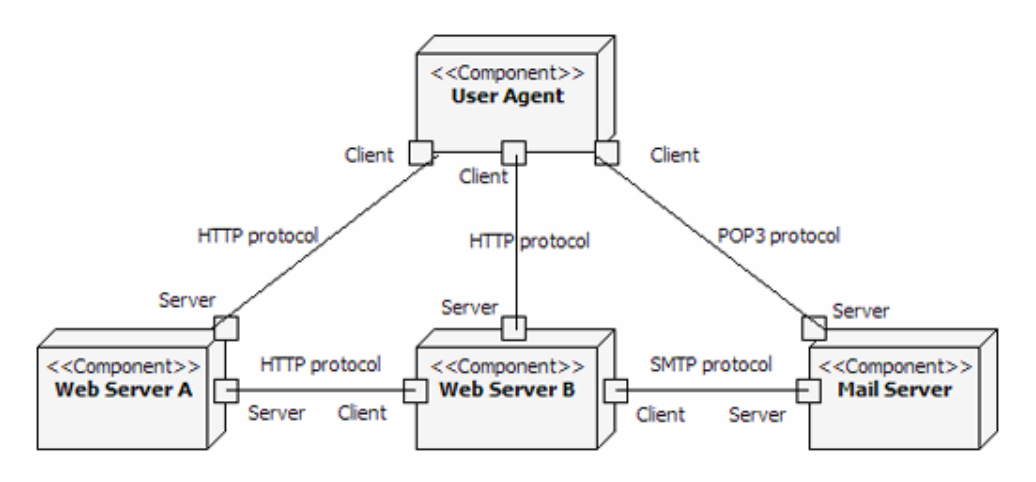
4.2.2 Restlet Framework

Ο επόμενος μηχανισμός για την επικοινωνία μεταξύ client και server που μελετήθηκε είναι το Restlet framework[26].

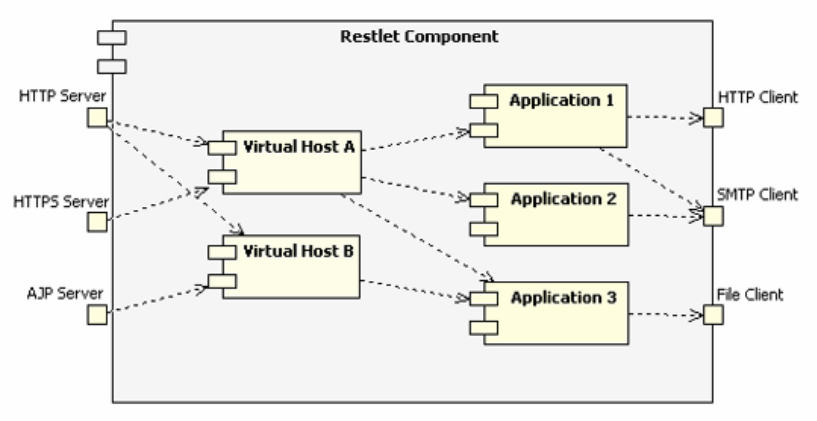
Αποτελεί βασικά ένα RESTful web framework για Java μέσω του οποίου μπορούν να αναπτυχθούν διαδικτυακές εφαρμογές βασισμένες στο αρχιτεκτονικό στυλ REST παρέχοντας συνάμα όλα τα οφέλη που απορρέουν από αυτό όπως την απλότητα του, τη επεκτασιμότητα της εφαρμογής κ.ο.κ. Υποστηρίζει όλες τις έννοιες πίσω από το REST όπως Resource, Representation, Connector , Component.

Το Restlet Framework παρέχει γνωρίσματα του πρωτοκόλλου HTTP όπως content negotiation, caching and conditional processing καθώς επίσης και του SMTP. Όμως το μεγάλο πλεονέκτημα αποτελεί η δυνατότητα που παρέχεται στο χρήστη κατά την οποία για τον ίδιο πόρο(που είναι addressable με ένα URI) μπορούν επιστρέφουν διάφορα είδη representation (XML, JSON, WADL). Ο χρήστης το μόνο που έχει να κάνει είναι να ορίσει το preferred resource representation που θέλει κατά τον ορισμό του request.

Το Restlet framework αποτελεί και client και server framework. Δηλαδή μπορούμε από το server side να κάνουμε request σε ένα πόρο που βρίσκεται τοπικά απευθείας μέσω του uri πόρου. Την ίδια στιγμή μπορούμε να πράξουμε το ίδιο και από τον client αλλά επειδή σε αυτή την περίπτωση ο πόρος είναι απομακρυσμένος ορίζεται ένα HTTP client connector. Ο connector στην ουσία αποτελεί ένα software element που προσφέρει την επικοινωνία μεταξύ components. Τα components αποτελούν μονάδες που περιέχουν κάποιο κώδικα που εκτελεί μια διεργασία. Στο project μας αφαιρετικά μπορεί να θεωρηθεί ότι component αναπαριστά ένα από τους πόρους. Στο σχήμα 4.5 μπορούμε να δούμε μια αναπαράσταση της αρχιτεκτονικής όπως αυτή περιγράφτηκε σε αυτή την παράγραφο ενώ στο σχήμα 4.6 μπορούμε να δούμε την δομή ενός component.



Σχημα 4.5 Περιγραφή της αρχιτεκτονικής Rest που ακλουθείτε στοRestlet framework. Στο σχήμα διαφαίνονται κάποια components όπου τα ports που περιέχουν στην ουσία αποτελούν τους connectors

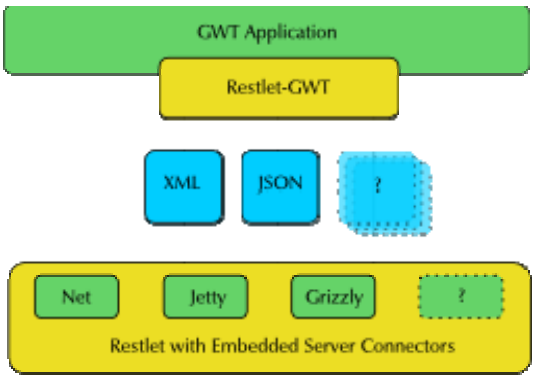


Σχήμα 4.6 Αναπαράσταση ενός restlet component

Τέλος επισημαίνεται ότι υπάρχει το ανάλογο edition του Restlet framework σε GWT(Google Web Toolkit) που είναι βασικά το περιβάλλον το οποίο θα εργαστούμε.
Η λογική πίσω από το σχεδιασμό της εφαρμογής μας αναπαριστάται στο σχήμα 4.7.
Βασικά ενσωματώνεται στο GWT Application η πιο πάνω έκδοση και έτσι δημιουργείται το Restlet περιβάλλον και έτσι η λειτουργικότητα των RPC μπορεί να αντικατασταθεί πλέων με server connectors όπως που αναλύθηκε πιο πάνω.

Στο appendix A-1 μπορούμε να δούμε τη συσχέτιση που γίνεται ανάμεσα στους πόρους και τα URLS. Ο gateway πλέων δρα όπως ένας κοινός router. Δηλαδή απλά προωθεί

πακέτα – request προς τους πόρους χωρίς να ασχολείται ιδιαίτερα με το περιεχόμενο που αποστέλλεται πίσω στον αιτητή.



Σχήμα 4.7 Restlet στο GWT και server connector functionality

4.2.3 Σύγκριση Συμπεράσματα

Βασικά η κύρια ιδέα πίσω την εργασία είναι η υλοποίηση ενός GUI (Graphical User Interface) για μια web –based εφαρμογή του Smart-Home.

Για να πληρείται αυτή η προδιαγραφή πρέπει η εφαρμογή να χρησιμοποιεί εξ ολοκλήρου web- technologies .Τα GWT-RPC καταρχήν δεν είναι web-based και σε τελική ανάλυση διαφαίνεται ότι αναγκάζουν σε tight-coupling μεταξύ του client και του server και έτσι μειώνεται πολύ το performance ενώ αυξάνεται κατά πολύ το complexity όπως φάνηκε από το κεφάλαιο 4.2.1.

Το Web Technology που μας δίνει Web-based functionality είναι όπως διαφαίνεται και από το Κεφάλαιο 3.3 το REST. Τα GWT-RPC είναι πιο κοντά στο SOAP παρά στο REST οπότε υιοθετήθηκε τελικά το Restlet framework και εφαρμόστηκε στο Application μέσω του Restlet Java library που προσφέρεται.

Ακόμη με το να έχουμε Restful end to end interaction απλοποιεί σε πολύ μεγάλο βαθμό την διαδικασία του implementation αφού παύει πλέον η ανάγκη για τον ορισμό μεγάλου

αριθμού κλάσεων μόνο και μόνο για την δημιουργία πρόσβασης σε μια κλάση του server.

Τέλος στα RPC υπήρχαν πάρα πολλοί περιορισμοί στο είδος των αντικειμένων που επιστρέφονταν από το server. Βασικά μόνο Java primitive types επιτρέπονταν να διακινηθούν ανάμεσα στο server και τον client. Οποιοσδήποτε άλλος τύπος έπρεπε να περάσει από τη διαδικασία του serialization και αυτό εκτός του ότι προσθετέ περισσότερο overhead στην εφαρμογή καθιστούσε το implementation ακόμα πιο δύσκολο.
Με το restlet ο χρήστης μπορεί να καθορίσει τι representation του πόρου θέλει να αποστέλλεται ως response στα request. Τα representations μπορεί να είναι μεταξύ άλλων simple text, json ,xml. και γίνονται parsed στο client πολύ εύκολα μέσω των προσφερομένων μεθόδων.

Παρόλο που στην αρχή το project ξεκίνησε με RPC, για τους πιο πάνω λόγους στην συνέχεια εγκαταλείψαμε αυτό τον μηχανισμό (τα RPC calls) και υιοθετήθηκε το restlet framework. Μέσω αυτού πλέων υπήρχε η δυνατότητα να καλέσουμε URI από το server χωρίς την ανάγκη για ύπαρξης επιπρόσθετών (client) servers. Έτσι μετά την υιοθέτηση του μηχανισμού ο server πολύ απλά σε ένα request πχ του στυλ http://localhost:8080/devices/2/ επιστρέφει πλέον σε JSON ή XML μορφή τα descriotion data του device 2.

Στη σελίδα A-3 στο Appendix μπορούμε να δούμε μια σύγκριση της πολυπλοκότητας του κώδικα των δύο μηχανισμών.

Κεφάλαιο 5

Υλοποίηση και παρουσίαση λειτουργιών του GUI

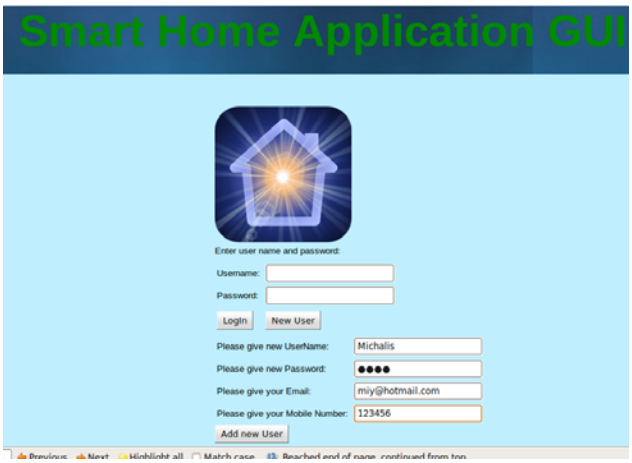
5.1 Υλοποίηση του GUI

Μετά από την φάση της ανάλυσης και σχεδίασης του λογισμικού και ακολουθώντας πάντα τον κύκλο ανάπτυξης λογισμικού προχωρούμε στην φάση της υλοποίησης. Για την υλοποίηση απαιτητό η εγκατάσταση του tiny-Os[24] σε προσωπικό υπολογιστή. Μετά από αυτό για την συγγραφή του κώδικα χρησιμοποιήθηκε το IDE Eclipse στο οποίο εγκαταστάθηκε το GWT plug –In που αναφέραμε στο υποκεφ. 3.2.5.

Πιο κάτω παρουσιάζονται και αναλύονται οι τελικές λειτουργίες που προσφέρει πλέον το GUI μετά την υλοποίηση του.

5.2 Εισαγωγή νέων χρηστών και user authentication

Η πρώτη οθόνη που εμφανίζεται στο χρήστη είναι αυτή που βλέπουμε στο πιο κάτω σχήμα 5.1.



Σχήμα 5.1 Log In Screen

Μέσω αυτής της οθόνης μπορούμε να εισάγουμε νέους χρήστες στο σύστημα ή αν ο χρήστης προϋπάρχει γίνεται το authentication του.

Για να γίνει επιτυχώς η εισαγωγή ενός νέου χρήστη πρέπει ο προηγούμενος να γεμίσει όλα τα απαιτούμενα στοιχεία στην φόρμα καθώς όλα είναι απαραίτητα για την ομαλή λειτουργία του συστήματος. Επίσης το username που θα δοθεί πρέπει να μην προϋπάρχει στη βάση. Αν συμβεί κάτι από τα προηγούμενα το σύστημα θα εμφανίσει σύστημα λάθους στο χρηστή και αφού αρχικοποιηθούν τα πεδία θα προτρέψει τον χρήστη να επαναλάβει την διαδικασία.

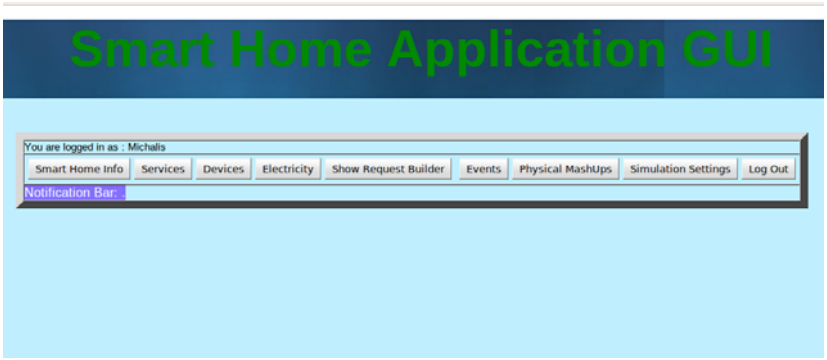
Τόσο ο έλεγχος του username-password όσο και η εισαγωγή ενός νέου χρήστη γίνονται με HTTP request στον “πόρο database” .

Φυσικά μέσα στην εφαρμογή υπάρχει και το ανάλογο πλήκτρο Log out που όταν πιεστεί, η εφαρμογή αρχικοποιείται και επιστρέφει σε αυτή την οθόνη.

5.3 Λειτουργίες κυρίως μενού

Μετά από επιτυχημένο user authentication ο χρήστης μεταφέρεται στο κυρίως μενού.

Εκεί πλέων έχει πρόσβαση σε όλες τις λειτουργίες και μπορεί πλέον να διαχειριστεί την οικία του. Όπως βλέπουμε υπάρχει ένα navigation bar που βοηθά στην πλοήγηση του χρήστη στο σύστημα. Το κυρίως μενού ακολουθεί το panel structure που προέκυψε από την φάση της ανάλυσης και έχει τη μορφή που φαίνεται στο σχήμα 5.2.

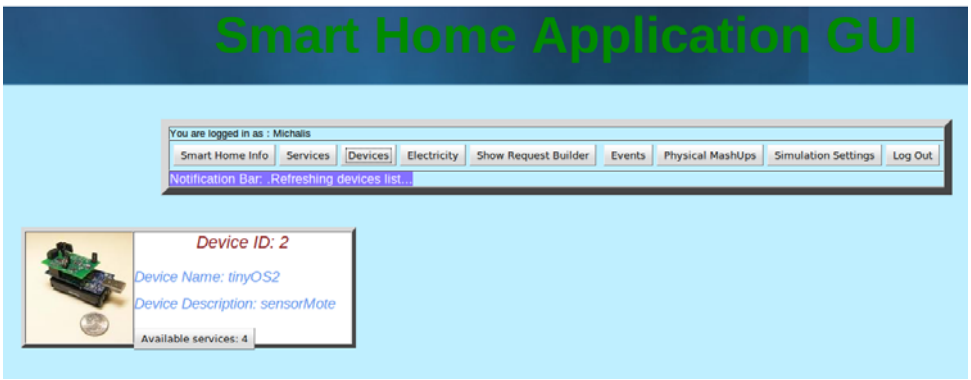


Σχήμα 5.2 Κυρίως μενού- Navigation bar

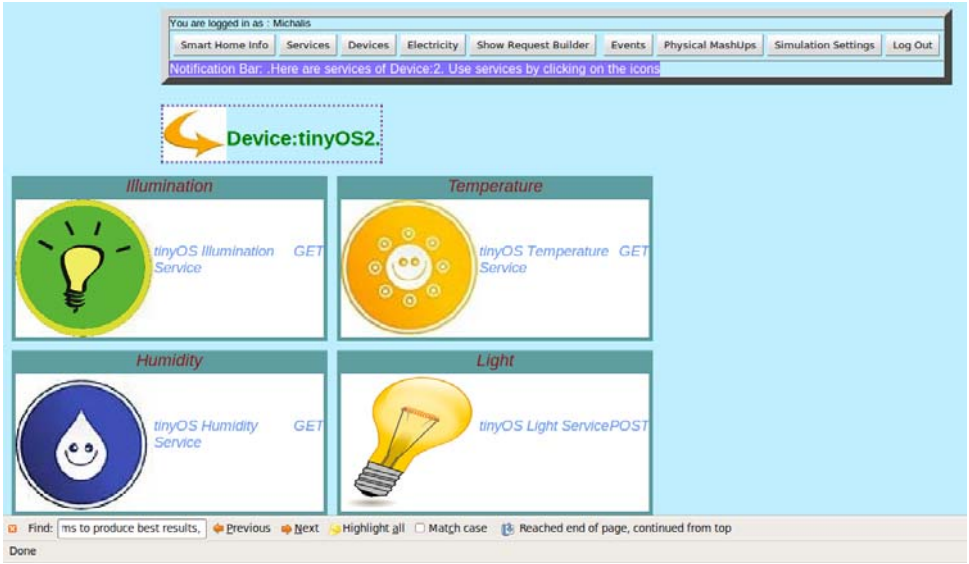
5.3.1 Διαθέσιμες Συσκευές-Υπηρεσίες

Πιέζοντας το Devices, το σύστημα παρουσιάζει στο χρήστη όλες τις συσκευές που είναι διαθέσιμες στο σύστημα. Ο χρήστης ενημερώνεται σε πρώτη φάση για το όνομα και το id της συσκευής καθώς και για τον αριθμό των διαθέσιμων υπηρεσιών που προσφέρονται από τη συσκευή(Σχήμα 5.3). Μετέπειτα ο χρήστης μπορεί να επιλέξει μια συσκευή και να δει ποιες είναι οι διαθέσιμες υπηρεσίες καθώς και γενικές πληροφορίες που αφορούν την κάθε υπηρεσία (Σχήμα 5.4).

Για την λήψη των απαιτούμενων δεδομένων ο αλγόριθμος εκτελεί GET request στους ανάλογους πόρους με τον τρόπο που εξηγήθηκε σε προηγούμενα κεφάλαια.



Σχήμα 5.3 Διαθέσιμες Συσκευές

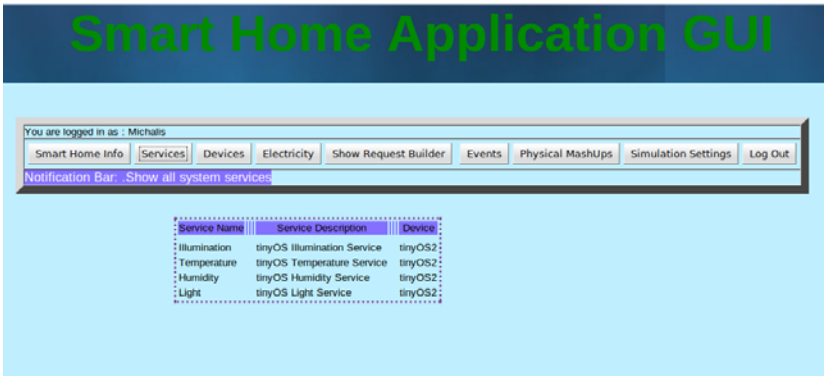


Σχήμα 5.4 Προσφερόμενες Υπηρεσίες συσκευής

5.3.2 Διαθέσιμες Υπηρεσίες συστήματος

Μια άλλη λειτουργία που παρέχεται από το σύστημα είναι η συγκεντρωτική παρουσίασή όλων των υπηρεσιών που προσφέρονται από όλες τις συσκευές στο σύστημα. Αυτό είναι το functionality πίσω από το κουμπί Services (Σχήμα 5.5).

Για τους σκοπούς υλοποίησης αυτής της λειτουργίες γίνεται αρχικά ένα GET request για την λήψη όλων των συσκευών(devices Ids). Μετά αφού γνωρίζουμε τις συσκευές εκτελείται μια ρουτίνα στην οποία διαδοχικά για το κάθε device αποστέλλεται πολύ απλά άλλο GET request (GET localhost/devices/deviceId) για να ληφθούν οι υπηρεσίες του. Ειδικά σε αυτό το σημείο διαφαίνεται καθαρά πόσο πιο απλοποιείται η υλοποίηση και πόσο πιο αποδοτικός γίνεται ο αλγόριθμος αν ακολουθηθεί το REST.



Σχήμα 5.5 Διαθέσιμες Υπηρεσίες Συστήματος

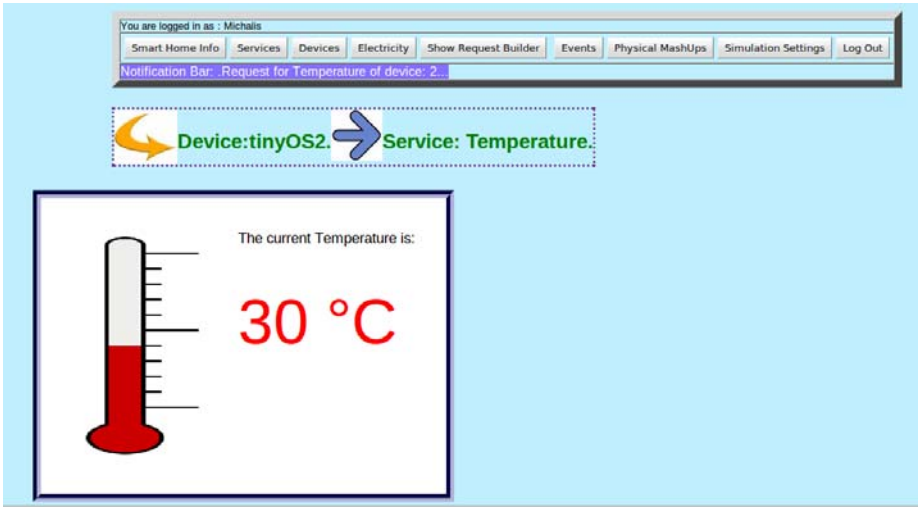
5.4 Ανάκτηση Πληροφοριών από το σύστημα

Φυσικά μετά την παρουσίαση των υπηρεσιών μιας συσκευής(Σχήμα 5.4) ο χρήστης μπορεί να επιλέξει την υπηρεσία που θέλει να καλέσει και να του παρουσιασθούν οι ανάλογες πληροφορίες. Αυτό μπορεί απλά να το κάνει πατώντας στο εικονίδιο της κάθε υπηρεσίας. Στην λειτουργία αυτή ακολουθείται το request/response μοντέλο.

Σε αυτήν την υποενότητα παρουσιάζονται οι υπηρεσίες που επιστρέφουν κάποιες πληροφορίες στο χρήστη (GET Services). Δηλαδή γίνονται invoke με GET requests.Τα requests είναι της μορφής GET "localhost/deviceId/serviceName". Πιο κάτω θεωρούμε ότι το device id της συσκευής στην οποία ανήκουν τα services είναι το 2.

5.4.1 Τρέχουσα Θερμοκρασία

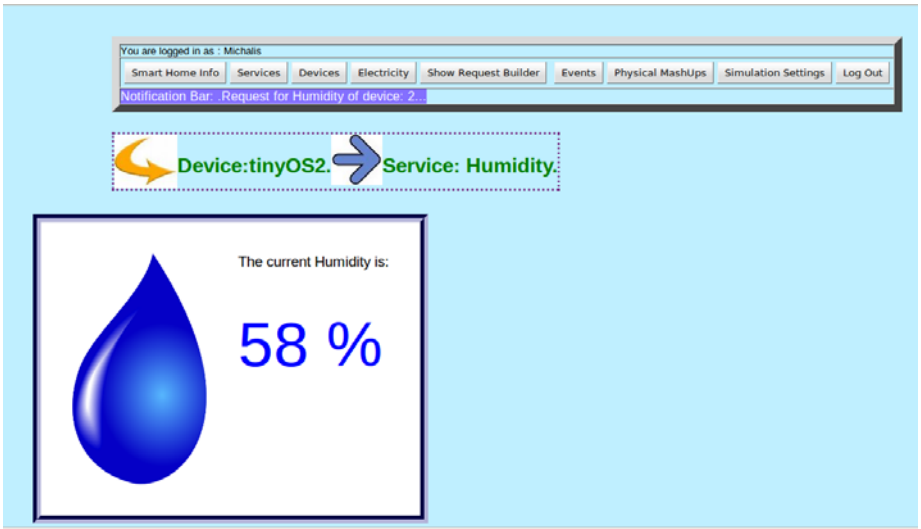
Πιέζοντας το εικονίδιο της θερμοκρασίας εμφανίζεται στο χρήστη ένα γραφικό(Σχήμα 5.6) που παρουσιάζει την τρέχουσα τιμή της θερμοκρασίας που μετρήθηκε από το συγκεκριμένο αισθητήρα(device). Όντας GET service το url το οποίο χρησιμοποιείται είναι το GET localhost/2/temperature.



Σχήμα 5.6 Τρέχουσα θερμοκρασία

5.4.2 Τρέχουσα Υγρασία

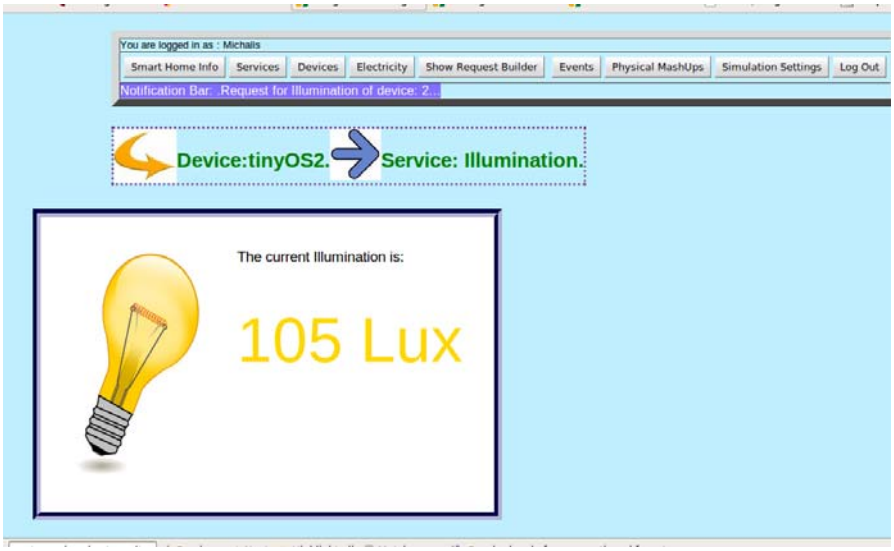
Πιέζοντας το εικονίδιο της υγρασίας εμφανίζεται στο χρήστη ένα γραφικό(Σχήμα 5.7) που παρουσιάζει την τρέχουσα τιμή της υγρασίας που μετρήθηκε από το συγκεκριμένο αισθητήρα(device). Όντας GET service το url το οποίο χρησιμοποιείται είναι το GET localhost/2/humidity.



Σχήμα 5.7 Τρέχουσα Υγρασία

5.4.3.Τρεχων Φωτισμός (Illumination)

Πιέζοντας το εικονίδιο του φωτισμού εμφανίζεται στο χρήστη ένα γραφικό(Σχήμα 5.8) που παρουσιάζει την τρέχουσα τιμή του φωτισμού που μετρήθηκε από το συγκεκριμένο αισθητήρα(device). Όντας GET service το url το οποίο χρησιμοποιείται είναι το GET localhost/2/illumination.

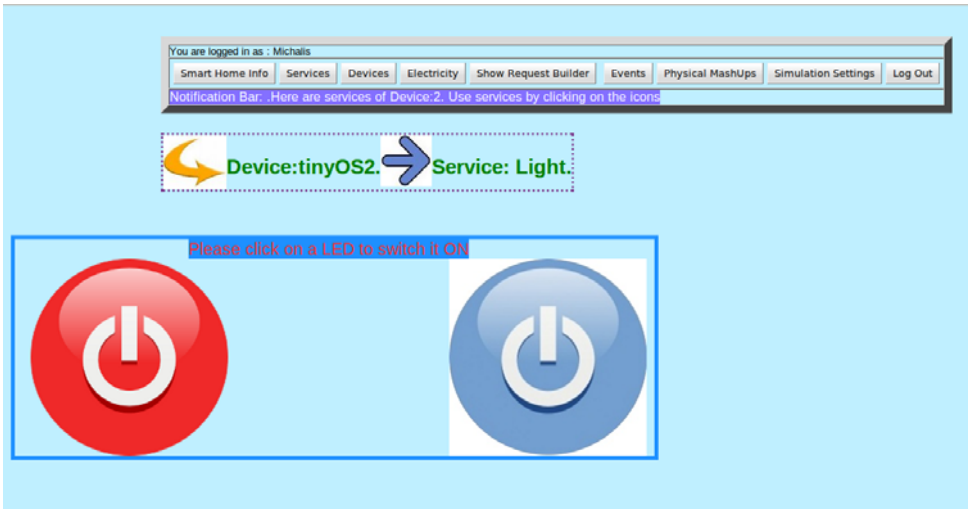


Σχήμα 5.8 Τρέχών φωτισμός δωματίου

5.5 Αλληλεπίδραση με κάποιο actuator

Σε αυτή την υποενότητα περιγράφεται η λειτουργικότητα που χαρακτηρίζει τις υπηρεσίες που ανήκουν στην κατηγορία (Post services). Οι υπηρεσίες αυτές δεν χρησιμοποιούνται για λάβουν δεδομένα αλλά συνήθως η κύρια λειτουργικότητα τους είναι η αλληλεπίδραση με κάποιο actuator ο οποίος εκτελεί κάποιο action.

Στην εφαρμογή μας η μόνη υπηρεσία που ανήκει σε αυτή την κατηγορία είναι η light(Σχήμα 5.9). Με αυτή την υπηρεσία ο χρήστης έχει τη δυνατότητα να ενεργοποιήσει ή να απενεργοποιήσει τους δύο λαμπτήρες της συσκευής(κόκκινο και μπλε).
Το URL για να χρησιμοποιηθεί η υπηρεσία είναι της μορφής
Post localhost/devices/deviceId/light και αποστέλλοντας παράλληλα ως παράμετρο το χρώμα.



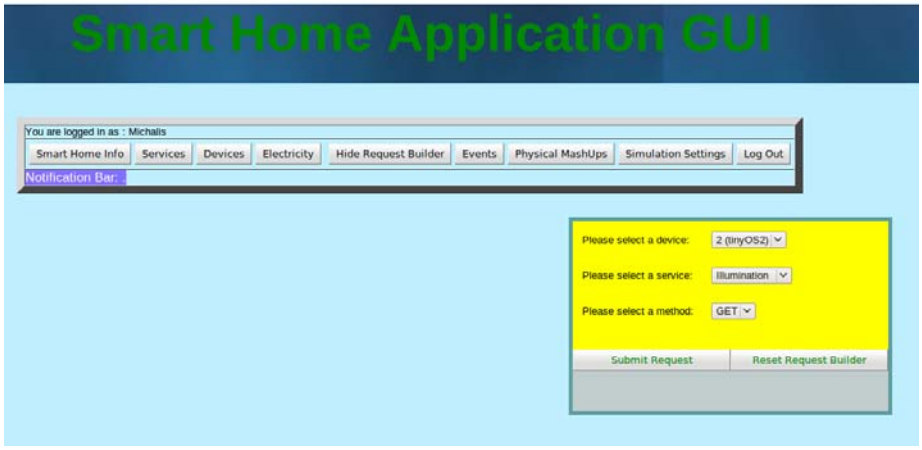
Σχήμα 5.9 Light service

Όμως επισημαίνεται ότι ένας actuator μπορεί να είναι και μια οποιαδήποτε ηλεκτρική συσκευή που ελέγχεται από sensor ploggs(Future work). Το GUI σχεδιαστικέ με τέτοιο τρόπο ώστε να είναι όσο πιο αφαιρετικό γίνεται και να μπορεί να παρουσιάσει οποιαδήποτε συσκεύη/υπηρεσία είναι ανιχνεύσιμη από τον Gateway.

5.6 Request Builder Functionality

Χρησιμοποιώντας την ίδια λογική που περιγράφεται στα υποκεφ. 5.4 και 5.5 το GUI μας, προσφέρει την δυνατότητα στο χρήστη να κτίσει manual ένα request και να το αποστείλει στο server για να εξυπηρετηθεί.

Ο request builder είναι ένα περιβάλλον που βοηθά τον χρήστη στη δημιουργία του request. Όπως φαίνεται και στο σχήμα 5.10 ο χρήστης πρέπει να διαλέξει την συσκευή την υπηρεσία, το verb που θέλει και αν απαιτούνται παράμετροι να τις δώσει. Ο χρήστης επιλέγει αυτά τα στοιχεία από drop down menus και το ένα σημαντικό σημείο εδώ είναι το γεγονός ότι τα στοιχεία των menus τροφοδοτούνται δυναμικά ανάλογα με τις επιλογές που προηγήθηκαν. Δηλαδή για μια συγκεκριμένη συσκευή παρουσιάζονται οι ανάλογες υπηρεσίες. Για κάθε υπηρεσία τα ανάλογα Http verbs που υποστηρίζει και τέλος ζητούνται αν απαιτούνται οι ανάλογες παράμετροι.



Σχήμα 5.10 Request Builder

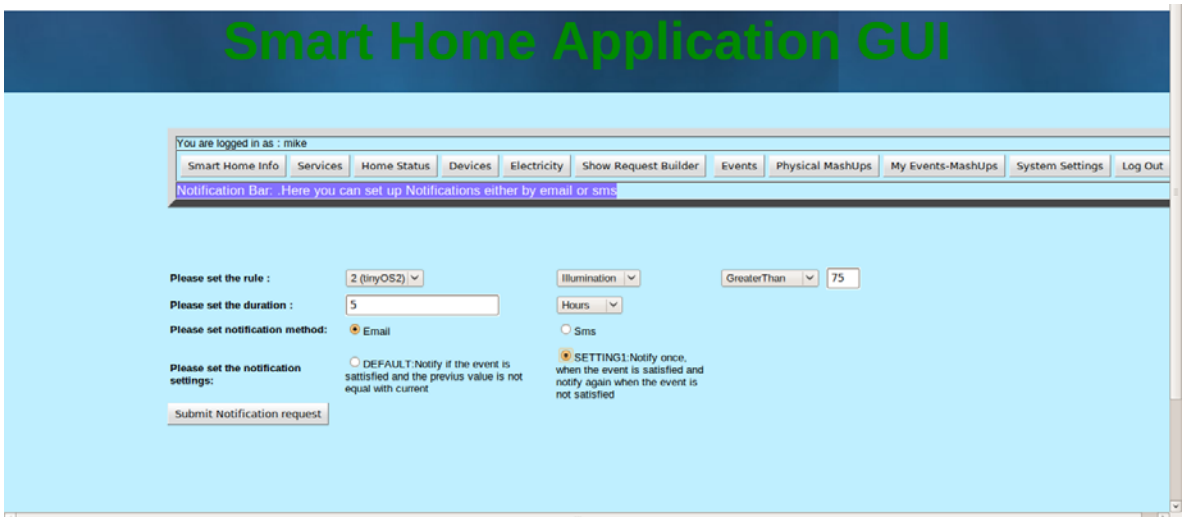
5.7.Μηχανισμός eventing

Όπως περιγράφεται από τις απαιτήσεις ο χρήστης θα πρέπει να έχει την δυνατότητα να ορίσει κάποιες καταστάσεις για τις οποίες θέλει να ενημερώνεται όταν συμβούν. Συνήθως αυτές οι καταστάσεις ορίζουν κάτι μη συνηθισμένο για το οποίο ο χρήστης θέλει να ενημερώνεται άμεσα. Αυτό ακριβώς κάνει ο eventing μηχανισμός.Το eventing ακλουθεί το publish/subscribe μοντέλο στο όποίο μόλις παρουσιαστεί μια κατάσταση (γίνει published) τότε ενεργοποιείται ο κατάλληλος μηχανισμός για να την διαχειριστεί[9][18]. Αυτή ακριβώς η λειτουργικότητα βρίσκεται πίσω από το κουμπί eventing.

Όπως βλέπουμε από το σχήμα 5.11 ο χρήστης μπορεί να ορίσει την "κατάσταση" μέσω των διαθέσιμών μενού και πεδίών που υπάρχουν. Επίσης βλέπουμε ότι μπορεί να επιλέξει και ανάμεσα σε δυο τρόπους μέσω τους οποίους θα του αποσταλεί το notification σε περίπτωση που προκύψει η κατάσταση που όρισε. Αυτοί είναι το email και το sms οι οποίοι περιγράφονται στα υποκεφ. 5.7.1και 5.7.2 αντίστοιχα.
Τέλος ο χρήστης μπορεί να ορίσει και το χρόνο για τον οποίο το κάθε entry στο μηχανισμό να είναι έγκυρο.

Επίσης επισημαίνεται ότι ο χρήστης μπορεί να ορίσει πότε να του αποστέλλεται ένα νέο notification. Για παράδειγμα μπορεί να ορίσει πχ ότι κάθε 5 updates της υπηρεσίας να του αποστέλλεται notification (monitoring). Ή ακόμη να ορίσει να του αποστέλλεται ένα notification όταν ικανοποιηθεί η συνθήκη που όρισε και ακόμα ένα μετά από ένα μελλοντικό update οπού στο οποίο η τιμή της υπηρεσίας δεν θα ικανοποιεί πλέον την συνθήκη.
Η default ρύθμιση είναι αν η τιμή της υπηρεσίας από το αμέσως προηγούμενου update για το οποίο στάλθηκε ήδη notification είναι η ίδια με την τρέχουσα τιμή. είναι να μην αποστέλλεται άλλο notification.

Από το σχήμα συμπεραίνουμε ότι ο χρήστης Mike θέλει αν κάποια στιγμή μέσα στις επόμενές πέντε ώρες το illumination ανεβεί πάνω από 75 Lux να ενημερωθεί άμεσα με email και να του αποσταλεί η τρέχουσα τιμή του service που επέλεξε. Επίσης όπως βλέπουμε επιλέγει το setting 1 όπου με αυτή την ρύθμιση λέει στο σύστημα ότι θέλει να του αποσταλεί ένα email όταν ικανοποιηθεί η συνθήκη και ένα όταν θα πάψει να ικανοποιείται.



Σχήμα 5.11 Μηχανισμός eventing

Κάθε entry αυτής της μορφής αποθηκεύεται στην βάση δεδομένων. Όταν έρθει update για την τιμή μιας υπηρεσίας από το σύστημα ελέγχεται η βάση για να διαπιστωθεί ποια από τα έγκυρα entries του eventing σχετίζονται με την υπηρεσία και αν ικανοποιούνται τότε γίνεται invoke ο ανάλογος πόρος (email ή sms) για να αποσταλεί το notification.

5.7.1 Email notifications

Τα Emails θεωρούνται από το σύστημα ως ένας ακόμη πόρος. Συγκεκριμένα όπως ενσωματώθηκαν στην εφαρμογή μπορούμε να αποστείλουμε ένα email πολύ απλά με το να αποστείλουμε ένα Post request στον πόρο email με παραμέτρους τη διεύθυνση του παραλήπτη και το μήνυμα που θέλουμε να αποστείλουμε.

Για την συγγραφή του κώδικα που διενεργεί την αποστολή των emails χρησιμοποιήθηκε το JavaMail API[29] που ενσωματώθηκε στο project και παρείχε όλες τις απαιτούμενες βιβλιοθήκες.

Για το email του αποστολέα (της εφαρμογής) αρχικά χρησιμοποιήθηκε ένα email account της Cytanet άλλα ο μηχανισμός λειτουργούσε μόνο αν ο ISP σου ήταν η Cyta. Έτσι δημιουργήθηκε ένα GMAIL account αφού το gmail προσφέρει ένα portable SMTP server και το πρόβλημα επιλύθηκε.

5.7.2 Sms notifications

Τα Sms παρομοίως με τα Emails ενσωματώθηκαν και αυτά ως πόροι του συστήματος. Για να αποστείλουμε ένα sms επίσης πολύ απλά κάνουμε ένα Post request στον πόρο sms με παραμέτρους το κινητό του παραλήπτη και το μήνυμα που θέλουμε να αποστείλουμε.

Για τους σκοπούς της εφαρμογής χρησιμοποιήθηκε ένας sms provider[30] που παρέχει την δυνατότητα αποστολής sms μέσω του διαδικτύου. Η υπηρεσία προσφέρεται επίσης μέσα από ένα ανάλογο API που μπορεί να ενσωματωθεί σε ένα οποιοδήποτε java project.

5.8.Physical MashUps functinonality

Ο πιο πολύπλοκος μηχανισμός που υπάρχει στο σύστημα είναι τα physical mashUps (βλέπε υποκεφ. 5.8.1).
Βάση των προδιαγραφών ο χρήστης της εφαρμογής θα πρέπει να μπορεί να ορίσει κάποιους κανόνες που θα ισχύουν πάντα. Οι κανόνες θα έχουν την εξής μορφή: Αν πληρείται ένα σύνολο προϋποθέσεων να γίνούν triggered ένα σύνολο από actions.

Στην δική μας εφαρμογή υποθέτουμε ότι το σύνολο από τις προϋποθέσεις θα αποτελείται μόνο από καταστάσεις υπηρεσιών της κατηγορίας GET (δηλαδή υπηρεσίες αισθητήρων που επιστρέφουν κάποια μέτρηση). Από την άλλη πλευρά το σύνολο των actions θα αποτελείται μόνο από υπηρεσίες κατηγορίας POST(δηλαδή actuators όπως το light). Το σύνολο των προϋποθέσεων μπορεί να αποτελείται από περισσότερες από

μια καταστάσεις που θα σχετίζονται με τους τελεστές AND και OR. Το σύνολο των actions επίσης μπορεί να αποτελείται πάνω από ένα αλλά αν ικανοποιηθούν οι προϋποθέσεις πάντα εκτελούνται όλα τα actions.

Ο χρήστης μέσω από το μενού που προσφέρεται θα μπορεί να ορίζει ένα physical mushUp και να το κάνει submit στο σύστημα. Ένα παράδειγμα φαίνετε στο σχήμα 5.12.

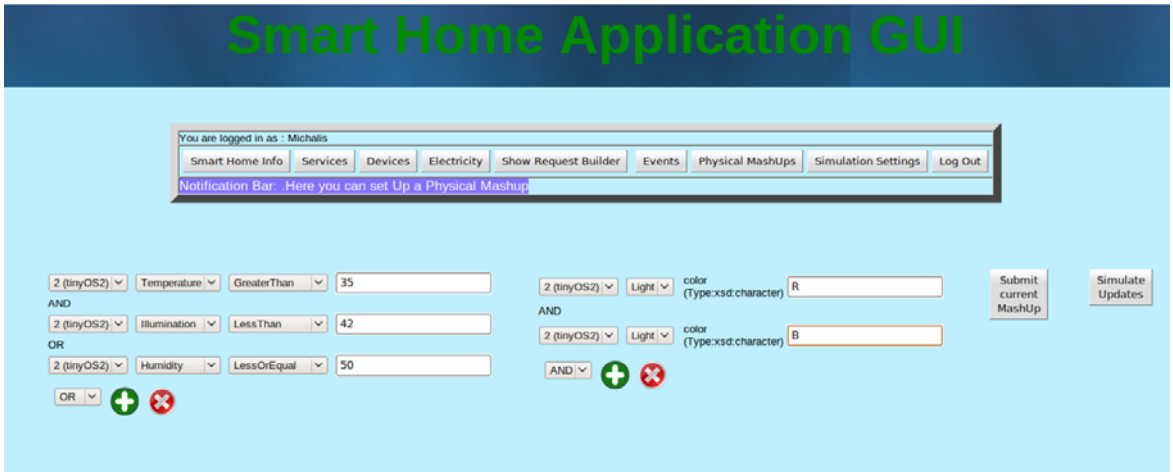
Ο χρηστής Michalis ορίζει ως

Προυποθέσεις
Αν η θερμοκρασία είναι πάνω από 45^{o}C και το Illumination είναι λιγότερο από 42lux ή η τιμή της υγρασίας είναι μικρότερη ή ίση του 50 %.

Actions
Άναψε τον κόκκινο και τον μπλε λαμπτήρα.

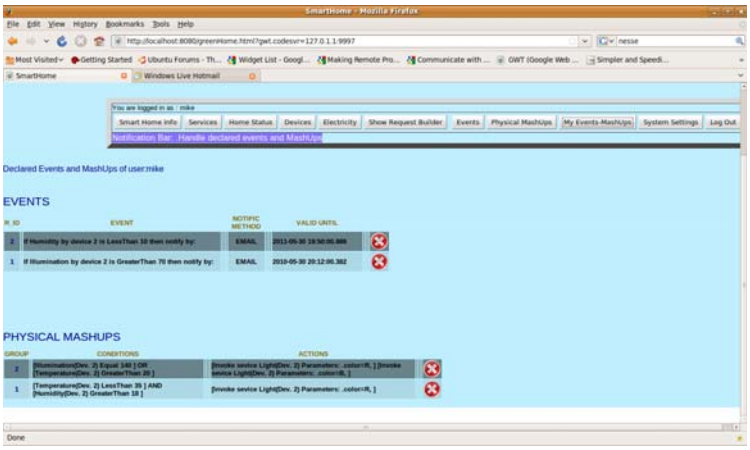
Η αλγοριθμική αναπαράσταση είναι η εξής

if((temperature>45)AND((Illumination <42)OR(Humidiy<=50) then]
{
switchOn (Red Light)
switchOn (Blue Light)
}



Σχήμα 5.12 Physical Mashups

Επίσης ο χρήστης μπορεί να δει ποια events και ποια physical mashups δημιούργησε ο ίδιος στο σύστημα μέχρι στιγμής. Επίσης μέσω αυτής της οθόνης παρέχεται παράλληλα η δυνατότητα διαγραφής από το σύστημα των διαφόρων events και mashups (Σχήμα 5.13)



Σχήμα 5.13 Handle events and Mashups

5.8.1. Mashups overview

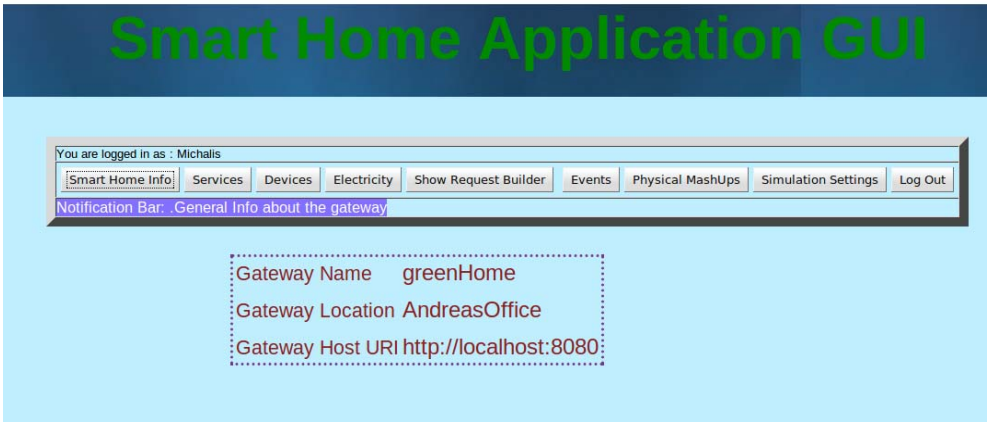
Τα mashups[19][20][21] είναι γενικά web based πόροι. Δημιουργούνται συνδυάζοντας τη λειτουργικότητα αριθμού υφιστάμενων πόρων. Έτσι η επαναχρησιμοποίηση και ο συνδυασμός των πόρων αυτών οδηγεί στη σύνθεση ενός νέου πόρου, του mashup. Δηλαδή το mashup μπορεί να διενεργεί ένα πολύπλοκο functionality μέσω της χρήσης πιο απλών functionalities υφιστάμενων πόρων.

Υπάρχουν πολλοί τύποι mashups. Δυο από τους πιο κύριους είναι τα web mashups και τα physical mashups. Τα web mashups χρησιμοποιούν τα APIs από διάφορα websites και παράγουν ένα νέο web application που τα συνδυάζει. Για παράδειγμα sites που διαφημίζουν διάφορες εκδηλώσεις μπορεί να παίρνουν πληροφορίες από το site του Google Maps για να δείξουν που θα είναι η εκδήλωση ή να συνδυάσουν και το youtube για να δείξουν σχετικά videos.

Τα physical mashups όπως αυτό που δημιουργήθηκε στην εφαρμογή συνδυάζούν πόρους όπως για παράδειγμα αισθητήρες οι οποίοι είναι πραγματικές solid συσκευές

5.9.Ανάκτηση γενικών πληροφοριών του Gateway

Η τελευταία λειτουργία που παρέχεται είναι η παρουσίαση γενικών πληροφοριών που αφορούν τον gateway(Σχήμα 5.13). Ο χρήστης πιέζοντας το πλήκτρο gateway Info βλέπει σχετικές πληροφορίες για τον gateway όπως το όνομα του, το location του το καθώς και το URL -Port στο οποίο τρέχει.



Σχήμα 5.13 Γενικές πληροφορίες για τον gateway

5.10 Settings

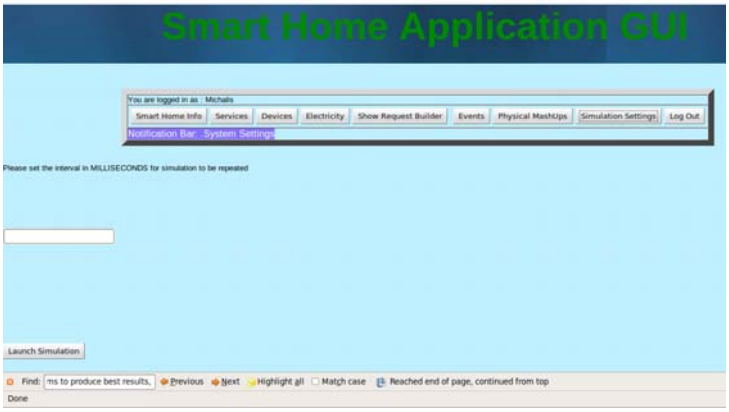
Κλείνοντας επισημαίνουμε ότι ο χρήστης μέσω του πλήκτρου settings μπαίνει σε ένα menu που περιλαμβάνει γενικές ρυθμίσεις της εφαρμογής (Σχήμα 5.14).

Μια από τις ρυθμίσεις είναι η ενεργοποίηση ενός μηχανισμού που εκτελεί αυτόματα queries στους sensors για όλες τις υπηρεσίες ανά τακτά χρονικά διαστήματα που καθορίζονται από το χρήστη. Δηλαδή με απλά λόγια το σύστημα κάνει αυτόματα update τις τιμές που γνωρίζει για τις διάφορες υπηρεσίες αυτόματα. Κάθε φορά που έρχεται ένα νέο update για μια υπηρεσία ελέγχονται όλα τα events και όλα τα mashups που υπάρχουν αποθηκευμένα στο σύστημα και περιέχουν την υπηρεσία την οποία αφορά το τελευταίο update για να διαπιστωθεί ποια από αυτά γίνονται validate true και χρήζουν περαιτέρω διαχειρίσεις οπού εκεί θα γίνουν και τα απαιτούμενα actions .

Ένα event για να γίνει validate το μόνο που χρειάζεται είναι η τρέχουσα τιμή της υπηρεσίας η οποία παρέχεται από το τελευταίο update.

Ένα mashup όμως πιθανών να περιλαμβάνει και άλλες υπηρεσίες εκτός από αυτή που αφορά το τελευταίο update. Οπότε για να γίνει validate την τιμή για τις άλλες υπηρεσίες την παίρνουμε από το cache του συστήματος (πιο πρόσφατη τιμή). Στο σύστημα υπάρχει cache για την κάθε υπηρεσία ξεχωριστά όπου εκεί αποθηκεύονται οι τιμές που προκύπτουν από όλα τα updates που αφορούν την συγκεκριμένη υπηρεσία.

Μόλις ξεκινήσει το σύστημα το cache όλων των υπηρεσιών είναι κενό και αν σε ένα mashup μια υπηρεσία χρειάζεται τιμή από το cache της και το cache της είναι κενό τότε το mashup αγνοείται.



Σχήμα 5.14 Settings του συστήματος

Κεφάλαιο 6

Συμπεράσματα

6.1 Ανασκόπηση της εφαρμογής

Σε αυτή την διπλωματική εργασία πετύχαμε την υλοποίηση ενός αποδοτικού GUI για μια εφαρμογή smart home μέσω του οποίου ο χρήστης πλέον θα μπορεί να διαχειρίζεται την το "έξυπνο σπίτι" από το Ίντερνετ.

Ξεκινήσαμε με μια γενική μελέτη γύρω από το θέμα του home automation και από εκεί εξάγαμε τις απαιτήσεις που πρέπει να ικανοποιούνται από την εφαρμογή.
Μετά μελετήθηκε σε βάθος ένα application framework για home automation πάνω στο οποίο βασίστηκε σε τελική ανάλυση το GUI.
Έχοντας τα προηγούμενα ως υπόψη προχωρήσαμε σε μια εκτενή έρευνα γύρω από τις διάφορες διαθέσιμες τεχνολογίες που θα χρειαζόμαστε για να προχωρήσουμε στην ανάπτυξη του λογισμικού. Και μετά από ανάλυση τους καταλήξαμε στις πιο ιδανικές.

Προχωρόντας στην φάση της σχεδίασης έγινε μια εις βάθος διερεύνηση των τελικών προδιαγραφών που απαιτούντο από το σύστημα και εξάγαμε την δομή την οποία θα ακολουθούσε η εφαρμογή.
Γνωρίζοντας πλέον τη δομή προχωρήσαμε στην τελική φάση της σχεδίασης καθορίζοντας τις διάφορες λειτουργίες από τις οποίες θα απαρτίζετο η εφαρμογή ψάχνοντας συνεχώς για τους πιο αποδοτικούς τρόπους υλοποίησης τους. Τέλος προχωρήσαμε στην υλοποίηση της εφαρμογής.

6.2 Αξιολόγηση-Συμπεράσματα

Το GUI που υλοποιήθηκε αποτελεί μια σημαντική επέκταση του application framework και οδηγεί σιγά σιγά πλέον στη δημιουργία ενός ολοκληρωμένου συστήματος Home Automation. Όπως είδαμε κτίσθηκε σε πιο ψηλό επίπεδο πάνω στο υπάρχων application framework χρησιμοποιώντας το API όπως είδαμε που παρέχεται.

Όσο αφορά το δίκτυο ακολουθήθηκε centralized αρχιτεκτονική αφού για μια τέτοια εφαρμογή είναι η πιο ιδανική.

Σχετικά με την αρχιτεκτονική της εφαρμογής είδαμε πως με το REST επιλύθηκαν πολλά προβλήματα που σχετίζονται με την ανομοιομορφία μεταξύ των πόρων καθώς και το interoperability μεταξύ τους. Άρα ο στόχος που τέθηκε για την δημιουργία μιας efficient και επεκτάσιμης εφαρμογής ικανοποιήθηκε.

Επίσης μέσω αυτής της εργασίας μελετήθηκαν οι βασικές λειτουργίες μιας εφαρμογής αυτοματοποίησης οικίας οι οποίες ορίσθηκαν στις προδιαγραφές του συστήματος μας και υλοποιήθηκαν με επιτυχία. Επίσης η απλότητα και οι συνεχείς οδηγίες που παρέχονται από το σύστημα καθιστούν την εφαρμογή εύχρηστη αφού δεν προαπαιτεί εξειδικευμένες γνώσεις για το χειρισμό της.

Τέλος επισημαίνεται ότι οι προοπτικές που έχει μια τέτοια εφαρμογή είναι αμέτρητες όπως μπορούμε να δούμε και στο υποκεφ 7.1

Κεφάλαιο 7

Discussion and future work

7.1 Προοπτικές-εφαρμογές

Η εγκατάσταση ενός συστήματος home automation σε μια οικία προσφέρει αμέτρητα οφέλη και προνομία στους ένοικους. Όπως αναφέρθηκε και στην αρχή σε ένα έξυπνο σπίτι ο άνθρωπος έχει τον απόλυτο έλεγχο του περιβάλλοντος στο οποίο κατοικεί και μπορεί να το προσαρμόσει σύμφωνα με τις δικές του ανάγκες ή απαιτήσεις.

Οι προοπτικές που απορρέουν από ένα τέτοιο σύστημα είναι πάρα πολλές. Καταρχήν μπορεί να χρησιμοποιηθεί για την εξοικονόμηση άσκοπης ενέργειας που σπαταλείται σε μια οικία. Για παράδειγμα να κλείνει τα φώτα ενός δωματίου αν δεν παρατηρείται κίνηση για ένα χρονικό διάστημα, ή μέσω της ρύθμισης των κλιματιστικών. Η εξοικονόμηση ενέργειας σε μια οικία είναι πολύ σημαντικός παράγοντας αφού όχι μόνο εξοικονομούνται λεφτά αλλά συνάμα συμβάλλει στις προσπάθειες που γίνονται για την προστασία του περιβάλλοντός. Επίσης το έξυπνο σπίτι μπορεί να συμβάλει σε σημαντικό βαθμό στην προστασία της οικίας από τυχών παραβίαση. Μια σχετική λειτουργία θα ήταν το άναμμα των εξωτερικών φώτων αν παρουσιαστεί κίνηση. Επίσης το σύστημα μπορεί να προστατέψει την οικία από φωτιές αφού αν πχ αντιληφθεί το σύστημα θερμοκρασία πάνω από λογικά επίπεδα να ειδοποιά αυτόματα πχ τον ένοικο και την πυροσβεστική αποστέλλοντας για παράδειγμα τις συντεταγμένες της οικίας. Επίσης άλλο παράδειγμα αν το σύστημα εφοδιαστεί με τους κατάλληλους αισθητήρες να εντοπίζει τυχών διαρροές γκαζιού και να κλείνει αυτόματα την παροχή.

Ακόμη το σύστημα μπορεί να προσαρμοστεί ανάλογα ούτως ώστε να ευκολύνει τη ζωή ανθρώπων ηλικιωμένων ή με αναπηρίες πχ αυτόματο άναμμα φώτων ή άνοιγμα των πόρτων των δωματίων.
Τέλος μια εφαρμογή του που είδη αναπτύσσεται είναι η καλυτέρευση της ζωής και παρακολούθηση ασθενών που πιθανών να μην μπορούν να βγουν από την οικία.

7.2 Future Work

Το GUI που υλοποιήθηκε κάλυψε τις βασικές απαιτήσεις ενός smart home application. Η εφαρμογή επιδέχεται επεκτάσεις οι οποίες παρουσιάζονται σε αυτό το σημείο για μελλοντική μελέτη.
Καταρχήν επισημαίνεται το γεγονός ότι ήδη στο μέλλον προγραμματίζονται να γίνουν κάποιες αλλαγές και optimizations στην όλη εφαρμογή σχετικές με energy awareness σε ένα web-based smart home όπου θα επιτυγχάνεται εξοικονόμηση ενέργειας μέσω της διαχείρισης των διαφόρων συσκευών από ploggs smart meters. Και με την ολοκλήρωση τους θα γίνει ένα ολοκληρωμένο evaluation της εφαρμογής.

Όπως είναι υλοποιημένη η εφαρμογή μπορεί ο οποιοσδήποτε να έχει πρόσβαση στο σύστημα αφού μπορεί πολύ απλά να δημιουργήσει ένα νέο χρήστη. Σε μια μελλοντική εργασία το functionality της εισαγωγή νέου χρήστη και του authorization να γίνεται σε ένα απομακρυσμένο server όπου μόνο οι ένοικοι της οικίας να μπορούν να έχουν πρόσβαση, ή απλά να παρέχεται περισσότερος στη δημιουργία νέου account.
Ακόμη μια άλλη επέκταση είναι το task scheduling όπου ο χρήστης θα μπορεί να ορίσει κάποια actions που θα εκτελούνται σε μελλοντικό χρόνο. Πχ κάθε μέρα η ώρα 05.00 να ξεκινά το σύστημα άρδευσης.
Επίσης μια άλλη ιδέα που μπορεί να υλοποιηθεί σαν μια μελλοντική προέκταση είναι η δημιουργία profile για τον κάθε χρήστη και ο χρήστης θα έχει πρόσβαση στις διάφορες υπηρεσίες ανάλογα με το profile τους. Δηλαδή ο καθένας θα έχει διαφορετικά προνόμια. [5]
Επίσης μπορεί να εφαρμοστεί το και η ιδέα του device location awareness[22][23] όπου πλέον κάθε device θα γνωρίζει το location του και αυτό ανοίγει άλλες προοπτικές στην εφαρμογή.

Βιβλιογραφία

[1] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, “The Gator Tech Smart House: a programmable pervasive space”, IEEE Computer Magazine, pp. 50–60, March 2005

[2] Juan Ignacio Vazquez, Diego López de Ipiña,and Iñigo Sedano, “SoaM: An environmentadaptation model for the Pervasive Semantic Web”, In Proceedings of the 2nd UbiquitousWeb Systems and Intelligence Workshop(UWSI 2006), colocated with ICCSA 2006, Lecture Notes in Computer Science - LNCS, volume 3983, pages 108-117, May 2006.

[3] Anand Ranganathan, Shiva Chetan, Roy Campbell, "Mobile Polymorphic Applications in Ubiquitous Computing Environments", In Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous’04), pp. 402-412., Boston, Massachusetts, USA, August 2004

[4] N. Drossos, C. Goumopoulos and A. Kameas, “A Conceptual Model and The Supporting Middleware For Composing Ubiquitous Computing Applications”, Special Issue in the Journal of Ubiquitous Computing and Intelligence (JUCI), entitled "Ubiquitous Intelligence in Real Worlds", American Scientific Publishers (ASP), Vol. 1, No. 2, pp. 1-13, 2006.

[5] Shiva Chetan, Jalal Al-Muhtadi, Roy Campbell, M. Dennis Mickunas “A Middleware for Enabling Personal Ubiquitous Spaces”, Proceedings of UbiSys: System Support for Ubiquitous Computing Workshop at Sixth Annual Conference on Ubiquitous Computing, Nottingham, England, 2004

[6] Byoung-Kug Kim, Sung-Kwa Hong, Young-Sik Jeong, Doo-Seop Eom, "The Study of Applying Sensor Networks to a Smart Home," 2008 Fourth International Conference on Networked Computing and Advanced Information Management, ncm, vol. 1, pp.676-681, 2008

[7] Jalal al Muhtadi, Annand Ranganathan, Shiva Chetan and Roy Campbell “Super Spaces: A Middleware for large-Scale Pervasive Computing Enviroments”, , PerWare2004: Middleware Support for Pervasive Computing Workshop at the 2'nd IEEE International Conference on Pervasive Computing and Communications (PerCom 2004),Orlando, FL, March 14, 2004.

[8] Remi Emonet, Dominique Vaufreydaz Patrick Reigniet Julien Letessier “O3MiSCID, a Middleware for Pervasive Environments”, PRIMA-INRIA Rhone-Alpes, 1st IEEE International Workshop on services Integration in Pervasive Environments, Lyon, France, 2006

[9] Hector A. Duran-Limon, Gordon S. Blair, Adrian Friday, Paul Grace, George Samartzidis, Thirunavukkarasu Sivaharan , Maomao WU “Context-Aware Middleware for Pervasive and Ad Hoc Environments” , Technical Report Lancaster university, 2004

[10] C. Prehofer, J. van Gurp, C. di Flora “Towards the Web as a Platform for Ubiquitous Applications in Smart Spaces”, Second Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI), at UBICOMB Interfaces (DCCI) 1.0, W3C Candidate Recommendation, Dec, 2007

[11] No author given “Towards a resource-Oriented Architecture for an open Web of Things”, No Institute given

[12] Vlad Stirbu, "Towards a RESTful Plug and Play Experience in the Web of Things," icsc, 2008 IEEE International Conference on Semantic Computing, pp.512-517, Aug, 2008

[13] Peter Schramm Edwin Naroska Peter Resch Jorg Platte and Holger Linde "Integration of Limited Servers into Pervasive Computing Enviroments Using Dynamic Gateway Services", Computer Engineering Institute, University Dortmund, Germany Techical Report 0202 , 2002

[14] Erik Wilde. "Putting things to REST", Technical Report UCB iSchool Report 2007-015, School of Information, UC Berkeley, November 2007.

[15] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos and K. Kim, "TinyRest - a protocol for integrating sensor networks into the internet" in Proc. of REALWSN, Stockholm, Sweden, 2005.

[16] D. Yazar, A. Dunkels "Efficient Application Integration in IP-based Sensor Networks", In Proceedings of ACM BuildSys 2009, the First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys), at SenSys09Berkeley, CA, USA, November 2009

[17] W. Drytkiewicz, I.Radusch, S. Arbanowski, R. Zeletin, "pREST: A REST-based Protocol for Pervasive Systems", 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Oct. 24-27 2004, Ft. Lauderdale, USA, pp 340-348, Oct. 2004.

[18] A.Stanford-Clark and H.L Truong "MQTT for Sensor Networks(MQTT-S) Protocol Specification", version 1.1 August. 2008

[19] D. Guinard, V. Trifa, T. Pham, O. Liechti, "Towards Physical Mashups in the Web of Things", In Proceedings of the 6th International Conference on Networked Sensing Systems (INSS), 2009

[20] D. Guinard and V. Trifa., "Towards the web of things: Web mashups for embedded devices", In Proceedings of WWW (International World Wide Web Conferences), Madrid, Spain, 2009.

[21] B. Hartmann, S. Doorley, and S.R. Klemmer, "Hacking, Mashing, Gluing: Understanding Opportunistic Design", *IEEE Pervasive Computing,* vol. 7, no. 3, pp. 46–54. July-September 2008

[22] Vlad Trifa, Dominique Guinard, Philipp Bolliger, Samuel Wieland, “Design of a Web-based Distributed Location-Aware Infrastructure for Mobile Devices”, Proc. of the First IEEE International Workshop on the Web of Things (WOT2010). Mannheim, Germany, March 2010

[23] Barry Pearn B. Bus “Location Awareness in Wireless Networks”, Coursework Master thesis, University of Tasmania. , Nov. 2004

[24] *Tinyos*: An open-source operating system designed for wireless embedded sensor networks. Available online at URL: "www.tinyos.net/".

[25] *GWT*: development toolkit for building and optimizing complex browser-based applications. Available online at URL: http://code.google.com/webtoolkit/

[26] *Restlet*: Restfull Web Framework for Java
Available online at URL: http://www.restlet.org/ and
http://wiki.restlet.org/docs_2.0/13-restlet/275-restlet/144-restlet.html

[27] *SQLite*: A library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine Available online at URL: http://www.sqlite.org/about.html

[28] SQLiteJDBC: Available online at URL: http://www.zentus.com/sqlitejdbc/

[29] JavaMail API:
Available online at URL: http://forums.sun.com/thread.jspa?threadID=668779

[30] SMS API : Internet based SMS messaging services.

Available online at URL: http://www.bulksms.com/

[31] Anders Moller and Michael Schwarttzbach , “An Intoduction to xml and Web Technologies”, Addison Wesley, 2006

Παράρτημα Α

Αποσπάσματα κώδικα – Software setup

Restlet Router και πρωώθιση requests στους πόρους

Σε αυτό το απόσπασμα κώδικα βλέπουμε πώς με το Restlet ορίζεται ένας rooter ο οποίος θα προωθεί τα διάφορα http requests στους ανάλογους πόρους (βάση του URL) για να εξυπηρετηθούν

```
/**
* Creates a root Restlet that will receive all incoming calls.
*/
@Override
public synchronized Restlet createInboundRoot() {

initializeServer();

// Create a router Restlet that routes each call
Router router = new Router(getContext());

.............

// attach the delete facility
router.attach(Constants.MODIFY_DEVICE + "/{device}",
org.greenHome.server.presentationLayer.rest.devices.ModifyDeviceResource.class
);

// attach the create facility
router.attach(Constants.MODIFY_DEVICE,org.greenHome.server.presentationLayer.
rest.devices.ModifyDeviceResource.class);

// attach the devices facility
router.attach("/devices",org.greenHome.server.presentationLayer.
rest.devices.DevicesResource.class);

// attach the (general) services facility
router.attach("/services",org.greenHome.server.presentationLayer.
rest.devices.ServicesResource.class);

// attach the device link
router.attach("/devices/{device}",org.greenHome.server.presentationLayer.
rest.devices.InvokeDeviceResource.class);

// attach the service link
router.attach("/devices/{device}/{service}",org.greenHome.server.
presentationLayer.rest.devices.InvokeServiceResource.class);

// attach the service link
router.attach("/devices/{device}/{service}/{history}",
org.greenHome.server.presentationLayer.rest.devices.
InvokeServiceResourceHistory.class);
```

```
// attach users authentication functionality from database
router.attach("/users/{user}",
org.greenHome.server.presentationLayer.rest.database.UsersResource.class);

// attach users authentication functionality from database
router.attach("/notification/{user}",org.greenHome.server.presentationLayer.
rest.database.NotificationResource.class);

// attach physical MashUp functionality from database
router.attach("/mashUps",org.greenHome.server.presentationLayer.rest.database.
PhMashUpsResource.class);



// attach email sending functionality to database
router.attach("/emails",org.greenHome.server.presentationLayer.rest.database.
EmailResource.class);

// attach sms sending functionality to database
router.attach("/sms",
org.greenHome.server.presentationLayer.rest.database.SmsResource.class);


// attach maintenance functionality to database
router.attach("/tablesdrop",org.greenHome.server.presentationLayer.rest.
database.DbMaintenanceResource.class);

// attachgateway infos functionality
router.attach("/gatewayInfos",org.greenHome.server.presentationLayer.rest.
devices.GatewayInfoResource.class);

// attachgateway infos functionality
router.attach("/simulation/{interval}",org.greenHome.server.presentationLayer.
rest.devices.SimulationResource.class);


// attach the general WADL link
router.attach(Constants.WADL,
org.greenHome.server.presentationLayer.rest.wadl.WADL.class);

// attach the WADL Service Description Data link for tinyOS
router.attach(Constants.WADL_ROUTE_TINYOS,
org.greenHome.server.presentationLayer.rest.wadl.TinyOS.class);

// attach the WADL Service Description Data link for Contiki
router.attach(Constants.WADL_ROUTE_CONTIKI,
org.greenHome.server.presentationLayer.rest.wadl.Contiki.class);

// attach the WADL Service Description Data link for Ploggs
router.attach(Constants.WADL_ROUTE_PLOGG,
org.greenHome.server.presentationLayer.rest.wadl.Plogg.class);

router.attach("/", new Directory(getContext(), "war:///"));

return router;
}
```

Σύγκριση πολυπλοκότητας κώδικα RPC και Restlet

Εδώ μπορούμε να δούμε αποσπάσματα κώδικα για μια απλή κλήση (invoke) ενός service από τον client στον server πρώτα με RPC και μετά μέσω του Restlet framework. Συγκεκριμένα είναι ο κώδικας που ζητά από τον gateway να του επιστρέψει τις διαθέσιμες συσκευές στο σύστημα

RPC

```
private GatewayServiceAsync gtwSvc = GWT.create(GatewayService.class);


// code to execute
if (gtwSvc == null) {
System.out.println("No GWT SERVICE");

}
// Set up the callback object.
AsyncCallback<Map<String, String[]>> callback = new
AsyncCallback<Map<String, String[]>>() {

public void onFailure(Throwable caught) {
// TODO: Do something with errors.
notifications.setText("Error");

}

public void onSuccess(Map<String, String[]> result) {

→ Το result είνα το callback object που ορίσαμε
showDevices(result);
}

};

// / Make the call to get devices.

gtwSvc.getDevs(callback);

}
});
```

Restlet framework

```
// get Device Information
ClientResource r = new ClientResource("/devices");

// Set the callback object invoked when the response is received.
r.setOnReceived(new Uniform() {

public void handle(Request request, Response response) {
// Get the representation as an JsonRepresentation
JsonRepresentation rep = new JsonRepresentation(response.getEntity());
```

→ Ο κώδικας που ακολουθει είναι το parsing του JSON representation

```
// Displays the properties and values.
try {
JSONArray object = rep.getValue().isArray();
if (object != null) {
for (int i=0; i < object.size(); i++){
JSONObject obj = (JSONObject) object.get(i);
JSONValue id         = obj.get(JSON_DEVICE_ID);
JSONArray idValues = id.isArray();
JSONValue name         = obj.get(JSON_DEVICE_NAME);
JSONArray nameValues = name.isArray();
deviceslb.clear();
for (int j=0; j < idValues.size(); j++){
String deviceID   = idValues.get(j).toString().replaceAll("\"", "");
String deviceName = nameValues.get(j).toString().replaceAll("\"", "");
deviceslb.addItem(deviceID + " (" + deviceName + ")");
}
}
```

Συγκρίνοντας παρατηρούμε φυσικά ότι με το restlet έχουμε πολύ πιο απλό και αποδοτικό κώδικα για την ίδια λειτουργία.

Εγκατάσταση απαραίτητου λογισμικού. Συσκευές- System setup

Σε αυτό το σημείο γίνεται μια παρουσίαση των βασικών βημάτων που πρέπει να ακολουθήσει κάποιος για την εγκατάσταση του απαραίτητου λογισμικού στα sensor devices και στο IDE eclipse και το απαραίτητο setup που πρέπει να γίνει για να λειτουργήσει το όλο σύστημα.

TinyOs

Καταρχήν το πρώτο βήμα που πρέπει να γίνει είναι η εγκατάσταση του tinyOs στον υπολογιστή. Το tinyOs και όλες οι απαραίτητες βιβλιοθήκες και οδηγίες παρέχονται από τη ιστοσελίδα στο *http://www.tinyos.net/tinyos-2.x/doc/* καθώς επίσης και στην σελίδα *http://www.5secondfuse.com/tinyos/install.html*

Προγραμματισμός των sensor devices

Για τον προγραμματισμό του basestation χρησιμοποιείται ο κώδικας που επισυνάπτεται στον φάκελο Smart Devices.

Και χρησιμοποιούνται οι πιο κάτω εντολές όπου x είναι το USB port στο οποίο μπαίνουν οι συσκευές

make tmote install.1 bsl,/dev/ttyUSBx για tmote sensors και
make micaz install.1 mib520,/dev/ttyUSBx για micaz sensors

Για τον προγραμματισμό του client χρησιμοποιείται επίσης κώδικας που επισυνάπτεται στον φάκελο Smart Devices.

Και χρησιμοποιούνται οι πιο κάτω εντολές όπου x είναι το USB port στο οποίο μπαίνουν οι συσκευές

make tmote install.2 bsl,/dev/ttyUSBx για tmote sensors και
make micaz install.2 mib520,/dev/ttyUSBx για micaz sensors

Για να βρούμε το USB port του κάθε device γράφουμε την εντολή motelist.

Setup eclipse project

Αφού εγκατασταθεί το tinyOs και προγραμματιστούν και τα device. Προχωρούμε στην εγκατάσταση του GWT PlugIn στο eclipse. Πλήρης οδηγός υπάρχει στο http://code.google.com/webtoolkit/usingeclipse.html.

Μετά πρέπει να γίνει import το "greenHome"project στον eclipse και να οριστεί η environmental variable MOTECOM με τιμή serial@/dev/tttyUSBx:platform όπου x το USB port που βάλαμε τον basestation στο πιο πάνω βήμα, κατά τον προγραμματισμό των sensors και platform το ειδος τον αισθητήρων(tmote, micaz).