



Πανεπιστήμιο Δυτικής Αττικής

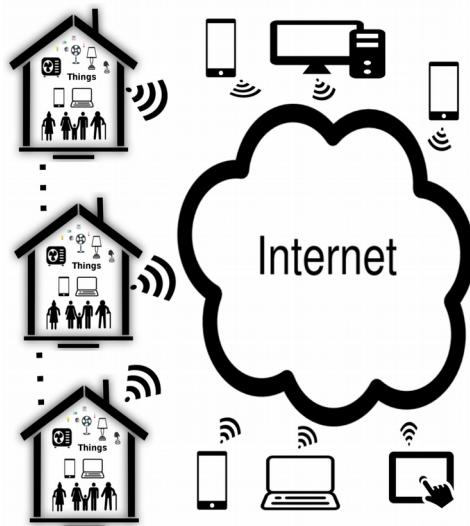
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Π.Μ.Σ «ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΟΛΟΓΙΣΜΟΥ ΚΑΙ ΔΙΚΤΥΩΝ»

Έξυπνα σπίτια: Παρελθόν, Παρόν και Μέλλον.

Σχεδίαση και πιλοτική ανάπτυξη ενός κατανεμημένου συστήματος για την απομακρυσμένη σε πραγματικό χρόνο διαχείριση μικρο-ελεγκτών, με σκοπό την υποβοήθηση υπερηλικων/ασθενών.



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΙΧΑΗΛ ΓΑΛΛΙΑΚΗ

Επιβλέπων : **Χρήστος Σκουρλάς**
Καθηγητής Πανεπιστημίου Δυτικής Αττικής

Αθήνα, Μάιος 2018



Πανεπιστήμιο Δυτικής Αττικής

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Π.Μ.Σ «ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΟΛΟΓΙΣΜΟΥ ΚΑΙ ΔΙΚΤΥΩΝ»

Έξυπνα σπίτια: Παρελθόν, Παρόν και Μέλλον.

Σχεδίαση και πιλοτική ανάπτυξη ενός κατανεμημένου συστήματος για την απομακρυσμένη σε πραγματικό χρόνο διαχείριση μικρο-ελεγκτών, με σκοπό την υποβοήθηση υπερηλίκων/ασθενών.



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΙΧΑΗΛ ΓΑΛΛΙΑΚΗ

Επιβλέπων : Χρήστος Σκουρλάς
Καθηγητής Πανεπιστημίου Δυτικής Αττικής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31^η Μαΐου 2018.

(Υπογραφή)

.....
Χρήστος Σκουρλάς
Καθηγητής Παν. Δυτ. Αττ.

(Υπογραφή)

.....
Ιωάννης Βογιατζής
Καθηγητής Παν. Δυτ. Αττ.

(Υπογραφή)

.....
Βασίλειος Μάμαλης
Καθηγητής Παν. Δυτ. Αττ.

Αθήνα, Μάιος 2018

(Υπογραφή)

ΜΙΧΑΗΛ ΓΑΛΛΙΑΚΗ

Copyright © Μιχαήλ Γαλλιάκη 2018.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εικφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Δυτικής Αττικής.

Περίληψη

Με σκοπό να αναπτυχθεί μια εφαρμογή για έξυπνα σπίτια, έγινε μελέτη της σχετικής βιβλιογραφίας και φάνηκε το μεγάλο ενδιαφέρον των ερευνητών για ευφυείς εφαρμογές, που στοχεύουν να καλύψουν τις ανάγκες των υπερήλικων/ασθενών στο σπίτι. Εστιάζοντας περισσότερο στην μελέτη τέτοιου είδους εργασιών και εφαρμογών, αποφασίστηκε να αναπτυχθεί ένα σύστημα που θα εκπληρώνει κάποια σενάρια χρήσης των ατόμων αυτών. Έπειτα, έγινε μια λεπτομερής ανάλυση απαιτήσεων και καθορίστηκαν οι ρόλοι των επί μέρους εφαρμογών ενός κατανεμημένου συστήματος που θα έπρεπε να αναπτυχθούν. Παράλληλα, έγινε αναζήτηση ώστε να βρεθούν οι πιο κατάλληλες τεχνολογίες, οι γλώσσες προγραμματισμού και τα προγραμματιστικά εργαλεία με τα οποία θα υλοποιούνταν το σύστημα. Μετέπειτα, σχεδιάστηκαν οι τεχνικές λεπτομέρειες και ακολούθησε η πιλοτική υλοποίηση του συστήματος. Τέλος, έγινε έλεγχος ώστε να αποδειχθεί ότι το σύστημα πραγματοποιεί όλα τα αρχικά σενάρια χρήσης, καταγράφηκαν κάποιες μελλοντικές επεκτάσεις του συστήματος και εκπορεύτηκαν κάποια συμπεράσματα από την εργασία.

Λέξεις Κλειδιά: Έξυπνα σπίτια/κτήρια. Σύστημα βοήθειας υπερηλίκων/ασθενών, Διαδίκτυο πραγμάτων, Μοντέλο Εξυπηρετητή/Πελάτη, Υπηρεσίες Ιστού, Κατανεμημένη εφαρμογή ιστού, Arduino, Java, Javascript, MEAN Stack, Database, Διαδικτυακή εφαρμογή, Cloud.

Abstract

In order to develop an application for smart homes, the related bibliography was studied from which it became clear that researchers were very interested in intelligent applications, which aim to cover the needs of the elderly/patients at home. Focusing more on studying such works and applications, it was decided that a system should be developed that would fulfill some scenarios that could be used by these individuals. Then, a detailed analysis of requirements was made and the roles of the individual applications of a distributed system were defined, which should be developed. At the same time, a search was conducted so as to find the most appropriate technologies, programming languages and programming tools to create the system. Subsequently, the technical details were designed, and the pilot implementation of the system followed. Finally, a check was made to show that the system performs all initial use case scenarios, some future system extensions were recorded, and some conclusions were drawn from the overall project.

Keywords: Smart homes/buildings, Assistance System for Elders/Patients, Internet of things, Server/Client, Web Services, Distributed Web Application, Arduino, Java, Javascript, MEAN Stack, Database, Web-based Application, Cloud.

Πίνακας περιεχομένων

1 Εισαγωγή	1
1.1 Έξυπνα σπίτια.....	1
1.2 Αντικείμενο διπλωματικής.....	1
1.2.1 Συνεισφορά.....	2
1.3 Οργάνωση κειμένου.....	4
2 Σχετικές εργασίες	5
3 Θεωρητικό υπόβαθρο	6
3.1 Κάποιες βασικές τεχνολογικές έννοιες.....	6
3.1.1 Monáδες.....	6
3.1.2 Socket.....	7
3.1.3 Arduino.....	7
3.1.4 Raspberry Pi.....	8
3.1.5 Web services & RESTful API.....	8
3.2 Τεχνολογικά εργαλεία και γλώσσες προγραμματισμού.....	9
3.2.1 Java & Javascript Programming languages.....	9
3.2.2 MEAN Stack.....	9
3.2.3 Maven & Gradle.....	9
3.2.4 Docker (Container).....	10
3.2.5 JSON.....	10
3.2.6 Git.....	10
4 Ανάλυση Απαιτήσεων Συστήματος	11
4.1 Αρχιτεκτονική.....	11
4.2 Περιγραφή κατανεμημένου συστήματος.....	13
4.2.1 Arduino.....	13
4.2.2 Device Client.....	14
4.2.3 User Clients (Desktop & Android).....	14
4.2.4 Server.....	15
4.2.5 Web App.....	15
4.2.6 Web Service.....	16
4.3 Μοντέλο Οντοτήτων Συσχετίσεων.....	16
5 Σχεδίαση Συστήματος	18
5.1 Αρχιτεκτονική.....	18
5.2 Σχεδιασμός των δυνατοτήτων του συστήματος.....	22
5.3 Περιγραφή των επιμέρους λογισμικών του συστήματος.....	25
5.3.1 Library.....	25

5.3.2 Device Client.....	26
5.3.3 Simulator Device Client.....	26
5.3.4 Server.....	27
5.3.5 Desktop User Client.....	27
5.3.6 Android User Client.....	28
5.3.7 Web Service (Back-end).....	28
5.3.8 Angular (Front-end).....	29
5.3.9 Arduino template sketch.....	30
5.4 Βάση δεδομένων.....	30

6 Υλοποίηση 36

6.1 Ενδεικτικά κάποιες λεπτομέρειες υλοποίησης (software).....	36
6.1.1 Μήνυμα πρωτοκόλλον.....	36
6.1.2 Άλλαγή τιμής μιας μονάδας από απομακρυσμένο χρήστη.....	37
6.1.3 Διαδικασία ειδοποίησης των User Clients όταν συμβεί κάποιο event.....	38
6.1.4 Κάποιες λειτουργίες του Web Service (Back-end).....	41
6.1.5 Κάποιες λειτουργίες του Angular project (Front-end).....	43
6.2 Υλοποίηση κατασκευών (hardware).....	45
6.3 Deploy του συστήματος στο Cloud.....	46
6.4 Πλατφόρμες και προγραμματιστικά εργαλεία.....	46

7 Έλεγχος 47

8 Επίλογος 61

8.1 Σύνοψη και συμπεράσματα.....	61
8.2 Μελλοντικές επεκτάσεις.....	62

9 Βιβλιογραφία 64

1

Εισαγωγή

1.1 Εξυπνα σπίτια

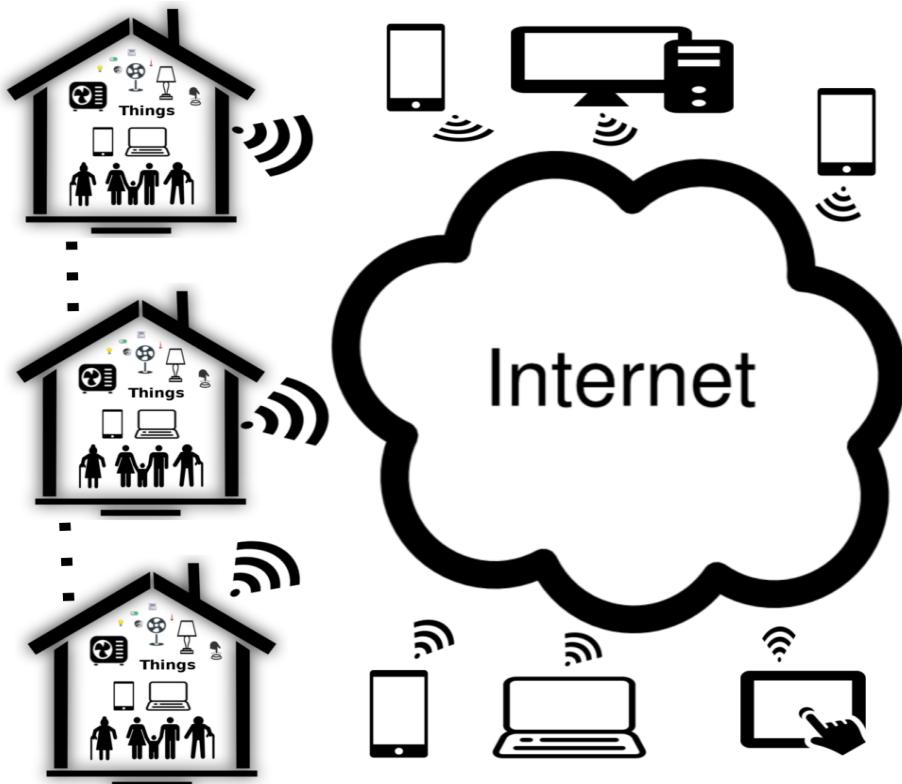
Έξυπνα σπίτια συνηθίζονται να λέγονται οι χώροι εκείνοι που με την βοήθεια της τεχνολογίας έχουν κάποιουν είδους “ευφυΐα”. Για παράδειγμα, μια περίπτωση “ευφυίας” είναι όταν το σπίτι είναι σε θέση να ανοίγει τον κλιματισμό αυτόματα όταν αλλάζει η θερμοκρασία του χώρου, είτε με βάση την θερμοκρασία που επικρατεί (πχ αν έχει κρύο να ζεσταίνει και αν έχει ζέστη να ψύχει), είτε όταν έρθει κάποια συγκεκριμένη χρονική στιγμή (πχ στις 14:00 επειδή στις 15:00 θα έχουν επιστρέψει όλοι οι ένοικοι). Μια άλλη περίπτωση “ευφυίας” είναι όταν το σπίτι αντιλαμβάνεται κάποια γεγονότα που συμβαίνουν (πχ πυρκαγιά, παραβίαση κάποιας πόρτας) και ειδοποιεί κατάλληλα κάποια φυσικά πρόσωπα. Ένα άλλο χαρακτηριστικό των έξυπνων σπιτιών είναι η δυνατότητα να τα εποπτεύονται και να τα διαχειρίζονται απομακρυσμένα οι ένοικοι τους (πχ να ανοίγει κάποιος τον θερμοσίφωνα με το κινητό του, ώστε όταν φτάσει στο σπίτι του να έχει ζεστό νερό). Αναφέρονται τα παραπάνω, ώστε με απλά καθημερινά παραδείγματα να φανερωθεί το “πνεύμα” των έξυπνων σπιτιών και κάποιες από τις ιδιαιτερότητες/χαρακτηριστικά τους.

1.2 Αντικείμενο διπλωματικής

Στα πλαίσια της διπλωματικής εργασίας, αναπτύχθηκε πιλοτικά ένα κατανευμημένο σύστημα ώστε να γίνεται, μέσω δικτύου και σε πραγματικό χρόνο, η διαχείριση μικρο-ελεγκτών από εγγεγραμμένους χρήστες. Το σύστημα αυτό μπορεί να εφαρμοστεί για να καλύψει τις ανάγκες απομακρυσμένης εποπτείας και διαχείρισης έξυπνων χώρων/σπιτιών. Συγκεκριμένα, δόθηκε έμφαση ώστε μέσω του συστήματος να μπορούν να υποβοηθηθούν κάποιες ευπαθείς ομάδες της κοινωνίας και ειδικότερα οι υπερήλικες και οι ασθενείς.

Να διευκρινιστεί, ότι η διπλωματική εργασία ασχολείται κυρίως με τη μελέτη και την ερεύνα που χρειάζεται να γίνει ώστε να υλοποιηθεί και να λειτουργήσει στη πράξη ένα κατανεμημένο σύστημα διαχείρισης μικρο-ελεγκτών. Έτσι, γίνεται μια σύντομη αναφορά της μελέτης από την σχετική βιβλιογραφία που χρησιμοποιήθηκε και γίνεται εστίαση κυρίως στο τεχνικό κομμάτι που αφορά την ανάλυση, σχεδίαση και ανάπτυξη εμπράκτως του συστήματος και όχι μόνο σε θεωρητικό επίπεδο.

1.2.1 Συνεισφορά



Σχήμα 1.2.1 Μια γενική εικόνα του συστήματος.

Μέσα από έρευνα στην βιβλιογραφία και την μελέτη κάποιων εφαρμογών για έξυπνα σπίτια, καθορίστηκε ο **σκοπός** και δημιουργήθηκε ένα σύστημα με ανοικτές τεχνολογίες αιχμής, που συνοπτικά παρέχει στους χρήστες του τις εξής δυνατότητες:

- ✓ Να εποπτεύουν σε πραγματικό χρόνο ένα ή περισσότερα έξυπνα σπίτια. Με τον όρο εποπτεία εννοείται να μπορεί ο χρήστης να βλέπει μέσω μιας διεπαφής (είτε μέσα από το σπίτι, είτε απομακρυσμένα) την κατάσταση των διαφόρων συσκευών/λαμπών/διακοπών του κάθε σπιτιού.
- ✓ Να αλλάζουν την κατάσταση των συσκευών/λαμπών/διακοπών, δηλαδή πχ μια λάμπα από ον να γίνεται off ή ακόμη και να παίρνει κάποια διακριτή τιμή που να καθορίζει την φωτεινότητα της.
- ✓ Να δέχονται ειδοποιήσεις από τα έξυπνα σπίτια τους (ακόμη και όταν δεν εποπτεύουν) όταν συμβεί κάποιο σημαντικό γεγονός (πχ να υπάρξει πυρκαγιά).

- ✓ Να χρονο-προγραμματίζουν το πότε να γίνονται διάφορες αλλαγές στις καταστάσεις των συσκευών/λαμπών/διακοπτών, ώστε να μην χρειάζεται να τις αλλάζει κάποιος χρήστης την πραγματική ώρα που θα θέλει να συμβούν, μέσα από εποπτεία.

Οι παραπάνω ιδιότητες του συστήματος καθορίστηκαν έτσι, με απότερο σκοπό να καλυφθούν και κάποιες περιπτώσεις υποβοήθησης υπερηλίκων ή ασθενών κατ' οίκον. Πιο συγκεκριμένα, το σύστημα δίνει την δυνατότητα να πραγματοποιηθούν τα παρακάτω **σενάρια χρήσης**:

- Να γίνεται υπενθύμιση για τη λήψη κάποιου φαρμάκου. Συγκεκριμένα, στα πλαίσια της εργασίας αναπτύχθηκε πιλοτικά μια ηλεκτρονική συσκευή που υπενθυμίζει σε κάποιον να πάρει τα φάρμακα του τη κατάλληλη ώρα. Η συσκευή αυτή συνεργάζεται/επικοινωνεί με το όλο σύστημα, ώστε να μπορεί κάποιος απ' οπουδήποτε, να προγραμματίσει/ενημερώσει μια agenda ειδοποιήσεων για λήψη των φαρμάκων την κατάλληλη χρονική στιγμή. (*Γίνεται χρήση της δυνατότητας του προγραμματισμού αλλαγής κατάστασης των μονάδων* σε άλλη χρονική στιγμή...)*)
- Όταν συμβεί κάποιο σημαντικό γεγονός στο σπίτι ενός υπερήλικα/ασθενή, τα οικεία του πρόσωπα, ο φροντιστής ή και κάποιο κέντρο επειγόντων περιστατικών, λαμβάνουν άμεσα **ειδοποιήσεις** στις ηλεκτρονικές συσκευές τους (πχ στα smartphones τους). Σημαντικά γεγονότα μπορεί να είναι για παράδειγμα:
 - Το πάτημα κάποιου emergency button, που μπορεί να είναι ένα πραγματικό κουμπί ή κάποιο ηλεκτρονικό μέσα στο smartphone/tablet του υπερήλικα/ασθενή.
 - Η ανίχνευση καπνού ή φωτιάς, με χρήση αντίστοιχου αισθητήρα.
 - Η ανίχνευση εισβολής, κίνησης ή ήχου μέσα στο χώρο, με τη χρήση διαφόρων αισθητήρων όπως sonar, φωτοκύτταρου ή αισθητήρα ήχου.

Χρησιμοποιώντας συνδυαστικά και κάποιο σύστημα online μετάδοσης εικόνας/βίντεο, θα μπορεί να γίνεται άμεσα προληπτικός έλεγχος των υπερηλίκων/ασθενών ώστε να διαπιστώνεται πχ αν έχει υπάρξει απώλεια αισθήσεων. Υπάρχουν διάφορα τέτοια συστήματα στην αγορά, με μικρό κόστος και αρκετά ποιοτικά χαρακτηριστικά.

Ένα παράδειγμα ενός συστήματος με 4 κάμερες HD & ένα καταγραφικό (~80), ένα σκληρό δίσκο 1TB (~45), καλώδια (~15) και σπιράλ (~50) κοστίζει συνολικά γύρω στα 200 ευρό σημερα. Οι κάμερες του εν λόγω συστήματος συνδέονται ασύρματα στο καταγραφικό και αυτό με τη σειρά του συνδέεται στο router του σπιτιού (Wi-Fi), ώστε να μπορεί να γίνει online μετάδοση βίντεο (P2P Remote Access) μέσω ενός δωρεάν app (πχ το IP Pro). Οι κάμερες του παραδείγματος έχουν λήψη ημέρας και νύχτας, μπορούν να κάνουν εστίαση (zoom) και αναγνωρίζουν την ανθρώπινη παρουσία στο χώρο (εμφανίζεται στη λήψη μια ένδειξη με ένα ανθρωπάκι).

- Με τη χρήση μιας ηλεκτρονικής συσκευής (όπως πχ ένα smartphone, tablet, laptop ή pc) δύναται ο ίδιος ο υπερήλικας/ασθενής ή κάποιο άλλο πρόσωπο (πχ ένας φροντιστής, το παιδί του υπερήλικα, ο γονέας του ασθενή ή οποιοδήποτε οικείο άτομο), μέσα από το σπίτι ή από κάποιο άλλο μέρος του κόσμου, άμεσα σε πραγματικό χρόνο (ή και με την δυνατότητα προγραμματισμού εκτέλεσης σε άλλη χρονική στιγμή) να **ανοίγει/κλείνει** οποιαδήποτε **ηλεκτρική συσκευή** του σπιτιού. Κάποια συγκεκριμένα παραδείγματα είναι:
 - Να καθορίζει πολύ εύκολα τον κατάλληλο φωτισμό σε κάθε χώρο του σπιτιού.
 - Να ανοίγει και να κλείνει ανεμιστήρες μέσα στο σπίτι.
 - Να ανοίγει και να κλείνει τη τηλεόραση ή το ραδιόφωνο.
 - Να ξεκινάει ή να σταματάει τη λειτουργία κάποιου απορροφητήρα.
 - Να καθορίζει την έναρξη λειτουργίας ή το κλείσιμο συσκευών θέρμανσης-ψύξης όπως πχ μιας ηλεκτρικής σόμπας, ενός ανεμιστήρα ή κλιματιστικού (aircondition).
 - Να καθορίζει την έναρξη λειτουργίας ή το κλείσιμο ενός υγραντήρα ή αφυγραντήρα.

1.3 Οργάνωση κειμένου

Στο 1^ο κεφάλαιο γίνεται μια εισαγωγική σύντομη αναφορά για τα έξυπνα σπίτια. Κατόπιν, περιγράφεται το αντικείμενο και ο σκοπός της εργασίας, όπως επίσης και η συνεισφορά που προσφέρεται στην ερευνητική και εκπαιδευτική κοινότητα. Στην συνέχεια στο 2^ο κεφάλαιο, παρουσιάζονται κάποιες σχετικές εργασίες από την βιβλιογραφία που μελετήθηκαν και στο 3^ο κεφάλαιο αναφέρονται κάποιες βασικές τεχνολογικές έννοιες μαζί με τα τεχνολογικά εργαλεία και τις γλώσσες προγραμματισμού που χρησιμοποιήθηκαν (για την ανάπτυξη του συστήματος της εργασίας). Έπειτα στο 4^ο κεφάλαιο, γίνεται η ανάλυση απαιτήσεων του συστήματος και περιγράφεται η αρχιτεκτονική και τα επιμέρους λογισμικά που αποτελείται. Στο 5^ο κεφάλαιο, μέσα από μια τεχνολογική σκοπιά, ακολουθεί η σχεδίαση της αρχιτεκτονικής και των δυνατοτήτων του συστήματος και η περιγραφή των επιμέρους εφαρμογών και της βάσης δεδομένων. Στην συνέχεια στο 6^ο κεφάλαιο, αναφέρονται ενδεικτικά μερικές λεπτομέρειες υλοποίησης, ο τρόπος εκτέλεσης σε πραγματικές συνθήκες και οι πλατφόρμες μαζί με τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν. Έπειτα στο 7^ο κεφάλαιο, περιγράφεται ένα πραγματικό σενάριο χρήσης του συστήματος. Στο 8^ο κεφάλαιο, γίνεται μια σύνοψη της εργασίας με τα σημαντικότερα συμπεράσματα που προκύπτουν και αναφέρονται κάποιες μελλοντικές επεκτάσεις του συστήματος. Στο τέλος, αναγράφεται η βιβλιογραφία που χρησιμοποιήθηκε.

2

Σχετικές εργασίες

Από την αναζήτηση στην βιβλιογραφία, βρέθηκαν πολλές εφαρμογές για έξυπνα σπίτια σχετικές με υπερήλικες/ασθενείς. Οι περισσότερες έκαναν χρήση διαφόρων αισθητήρων για την φροντίδα αυτών των ατόμων. Οι αισθητήρες αυτοί είτε βρίσκονται πάνω στο σώμα (βιοαισθητήρες) για μετρήσεις όπως αρτηριακή πίεση, καρδιακοί παλμοί, οξυγόνο κ.ά, είτε γενικά βρίσκονται μέσα στο χώρο, όπως πχ κάποιος αισθητήρας βάρους, για την αναγνώριση του χρόνου που βρίσκεται κάποιος υπερήλικας στο κρεβάτι του. Επίσης, πολλές εφαρμογές περιλάμβαναν συστήματα παρακολούθησης των ατόμων αυτών. Ακόμη, για την χρήση των συστημάτων έχουν αναπτυχθεί ειδικές προσαρμοσμένες διεπαφές (User Interface) που λειτουργούν μέσα από διαδραστικές ηλεκτρικές συσκευές, όπως smartphones, tables, pc και game consoles. Επιπλέον, υπάρχουν εφαρμογές αναγνώρισης διαφόρων γεγονότων (Event Recognition) όπως για παράδειγμα της αναγνώρισης κάποιας πτώσης ενός υπερήλικα, με χρήση αισθητήρων. Επίσης, έχουν φτιαχτεί εφαρμογές υπενθύμισης πραγμάτων σε υπερήλικες. Μια τέτοια εφαρμογή υπενθύμισης, είναι ένα έξυπνο κουτί χαπιών που θυμίζει στους υπερήλικες πότε να παίρνουν τα χάπια τους. Έχουν σχεδιαστεί και συστήματα διαχείρισης έξυπνων σπιτιών, όπου οι υπερήλικες χρησιμοποιούν την φωνή τους ή κάποιες χειρονομίες για να ελέγχουν τον χώρο (πχ να κλείσουν-ανοίξουν κάποιο φως). Σε μια άλλη σχετική εργασία, είχε σχεδιαστεί ένα ρομποτ-βιοηθός με ένα έξυπνο σύστημα έκτακτης ανάγκης για την ανεξάρτητη διαβίωση των υπερηλίκων. Άλλες εφαρμογές ασχολήθηκαν με την ανάλυση της κατάστασης των υπερηλίκων και τον εντοπισμό των κινδύνων που σχετίζονται με τις καθημερινές δραστηριότητες τους. Να αναφερθεί, ότι γίνεται κυρίως χρήση ασύρματων τεχνολογιών για την επικοινωνία και ότι προσφέρονται λύσεις στο Cloud για τέτοιου είδους εφαρμογές.

3

Θεωρητικό υπόβαθρο

Σε αυτή την ενότητα ακολουθεί μια πολύ σύντομη περιγραφή κάποιων βασικών τεχνολογικών εννοιών, εργαλείων και γλωσσών προγραμματισμού, που χρησιμοποιήθηκαν στην εργασία.

3.1 Κάποιες βασικές τεχνολογικές έννοιες

3.1.1 Μονάδες

Η λέξη “**Μονάδα**” είναι ένας όρος που δημιουργήθηκε σε αυτήν εδώ την εργασία, με σκοπό να εκφράσει την έννοια ενός αντικειμένου που μπορεί να συνδεθεί με κάποιον μικρο-ελεγκτή arduino. Ένα τέτοιο αντικείμενο μπορεί να είναι κάποιος αισθητήρας, ένας διακόπτης, ένα led λαμπάκι, μια ηλεκτρική συσκευή ή οτιδήποτε άλλο που έχει μία υπόσταση και θέλουμε να το χειριστούμε σαν ένα πράγμα. Κάθε **μονάδα** έχει κάποιες συγκεκριμένες ιδιότητες όπως όνομα, τύπο, τιμή κ.ά.

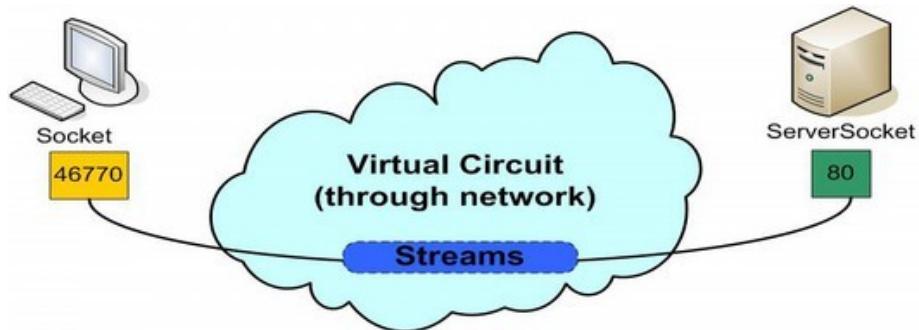
Παραδείγματα μονάδων μπορεί να είναι τα ακόλουθα:



Με την σειρά: Θερμόμετρο, sonar, λάμπα υψηλής τάσης, κουμπί, αισθητήρας φωτιάς, led λαμπάκι.

3.1.2 *Socket*

Είναι μια τεχνολογία που επιτρέπει να συνδεθούν 2 “σημεία” μεταξύ τους (πχ 2 υπολογιστές-προγράμματα) και να επιτευχθεί αμφίδρομη επικοινωνία σε πραγματικό χρόνο μέσα σε δίκτυο. Ουσιαστικά δηλαδή, υπάρχει η δυνατότητα με χρήση των sockets να στηθεί ένας δίαυλος επικοινωνίας μεταξύ 2 διεργασιών (που μπορεί το κάθε ένα να τρέχει σε οποιοδήποτε υπολογιστή αρκεί να “βλέπει” το ένα το δίκτυο του άλλου) ώστε να μεταφέρονται bytes με διάφορες πληροφορίες εκατέρωθεν. Στην περίπτωση της εργασίας αυτής μεταφέρονται JSON συμβολοσειρές. Μια κοινώς γνωστή χρήση των socket είναι στα chat προγράμματα.



3.1.3 *Arduino*

To Arduino είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++).



3.1.4 Raspberry Pi

Το Raspberry Pi είναι ένας μικρός υπολογιστής σε μέγεθος μιας πιστωτικής κάρτας. Έχει δυνατότητες και χαρακτηριστικά όπως ενός smartphone. Ακόμη, να αναφερθεί ότι γίνεται να συνδεθεί με οθόνη και μέσω USB να έχει πληκτρολόγιο, ποντίκι κ.ά.



Να διευκρινιστεί, ότι στην περίπτωση της διπλωματικής εργασίας, χρειάζεται από την πλευρά κάθε “έξυπνου” χώρου ένας υπολογιστής που να έχει απαραίτητα επεξεργαστή, κύρια και βοηθητική μνήμη, θύρες USB ή bluetooth adapter, κάρτα δικτύου (WiFi ή Ethernet) και λειτουργικό σύστημα. Θα μπορούσε δηλαδή να χρησιμοποιηθεί ένα παλαιό (ή και καινούριο) pc, ένα laptop ή για πρακτικούς λόγους ένας υπολογιστής “τσέπης” όπως ένα raspberry pi. Δεν είναι όμως απαραίτητη η χρήση κάποιου Raspberry Pi.

3.1.5 Web services & RESTful API

Τα web services αποτελούν μία αρχιτεκτονική κατανεμημένων συστημάτων κατασκευασμένη από πολλά διαφορετικά υπολογιστικά συστήματα τα οποία επικοινωνούν μέσω του δικτύου ώστε να δημιουργήσουν ένα σύστημα. Αποτελούνται από ένα σύνολο από πρότυπα τα οποία επιτρέπουν στους υπεύθυνους για την ανάπτυξη (προγραμματιστές - developers) να υλοποιήσουν κατανεμημένες εφαρμογές (χρησιμοποιώντας διαφορετικά εργαλεία από διαφορετικούς προμηθευτές) ώστε να κατασκευάσουν εφαρμογές που χρησιμοποιούν ένα συνδυασμό από ενότητες λογισμικού (software modules) οι οποίες καλούνται από συστήματα που ανήκουν σε διαφορετικά τμήματα ενός οργανισμού ή σε διαφορετικούς οργανισμούς.

Η βασική αρχή σχεδίασης του REST είναι η ένα-προς-ένα αντιστοίχιση μεταξύ λειτουργιών CRUD (create, read, update, delete) και HTTP μεθόδων. Σύμφωνα με αυτή την αντιστοίχιση:

- Για τη δημιουργία ενός πόρου στον server, χρησιμοποιούμε την μέθοδο POST.
- Για την ανάσυρση ενός πόρου, χρησιμοποιούμε την GET.
- Για την αλλαγή της κατάστασης ενός πόρου ή την ενημέρωσή του, χρησιμοποιούμε την PUT.
- Για την απομάκρυνση ή διαγραφή ενός πόρου, χρησιμοποιούμε την DELETE.

Με βάση το REST το URI δεν χρησιμοποιείται πια για την περιγραφή της ενέργειας που θέλουμε να εκτελέσουμε αλλά μόνο τον εντοπισμό του πόρου επί του οποίου θα ασκηθεί η ενέργεια, και τα δεδομένα δεν μεταφέρονται ως παράμετροι στο URI ενός GET αιτήματος αλλά ως XML ή JSON-formatted δεδομένα στα περιεχόμενα μιας POST ή PUT μεθόδου. Με άλλα λόγια σε μια υπηρεσία REST, ένα URI εκφράζει ένα αντικείμενο στο οποίο παρέχει πρόσβαση η υπηρεσία μέσω ενός HTTP αιτήματος. Το

είδος του αιτήματος καθορίζει την ενέργεια που θέλουμε να εφαρμόσουμε στο αντικείμενο αυτό και το περιεχόμενο του αιτήματος περιέχει διάφορες εξειδικεύσεις της ενέργειας.

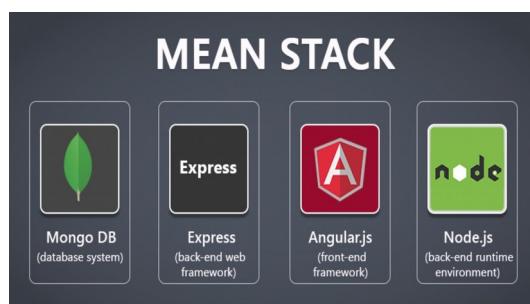
3.2 Τεχνολογικά εργαλεία και γλώσσες προγραμματισμού

3.2.1 Java & Javascript Programming languages

Η Java είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού και ένα από τα βασικά πλεονεκτήματα της έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουν ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές.

3.2.2 MEAN Stack

- MongoDB: Είναι μια NoSQL βάση δεδομένων.
- Express: Είναι ένα μικρό και ευέλικτο web application framework του node.js. Έχει ένα σύνολο από χαρακτηριστικά με σκοπό την ανάπτυξη Web εφαρμογών.
- AngularJS: Επιτρέπει την επέκταση του λεξιλογίου HTML με σκοπό να υπάρχει δυναμικό περιεχόμενο σε Web εφαρμογές. Το περιβάλλον που προκύπτει είναι εξαιρετικά εκφραστικό, ευανάγνωστο και γρήγορο.
- NodeJS: Είναι μια πλατφόρμα για εύκολη δημιουργία γρήγορων και κλιμακούμενων εφαρμογών δικτύου.



3.2.3 Maven & Gradle

Το Maven και το Gradle είναι εργαλεία για να γίνεται αυτοματοποιημένα το build εφαρμογών.

3.2.4 Docker (Container)

Το Docker (Ντόκερ) είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα που υλοποιεί Εικονικοποίηση (Virtualization) σε επίπεδο Λειτουργικού Συστήματος. Ουσιαστικά το Docker προσφέρει αυτοματοποιημένες διαδικασίες για την ανάπτυξη εφαρμογών σε απομονωμένες περιοχές χρήστη (User Spaces) που ονομάζονται Software Containers. Το λογισμικό χρησιμοποιεί τεχνολογίες του πυρήνα του Linux όπως τα cgroups και οι χώροι ονομάτων πυρήνα (kernel namespaces), για να επιτρέπει σε ανεξάρτητα software containers να εκτελούνται στο ίδιο λειτουργικό σύστημα. Έτσι αποφεύγεται η χρήση επιπλέον υπολογιστικών πόρων που θα απαιτούσε μια εικονική μηχανή (virtual machine).



3.2.5 JSON

Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράψουν. Επίσης, είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, Standard ECMA-262 Έκδοση 3η - Δεκέμβριος 1999. Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού.

3.2.6 Git

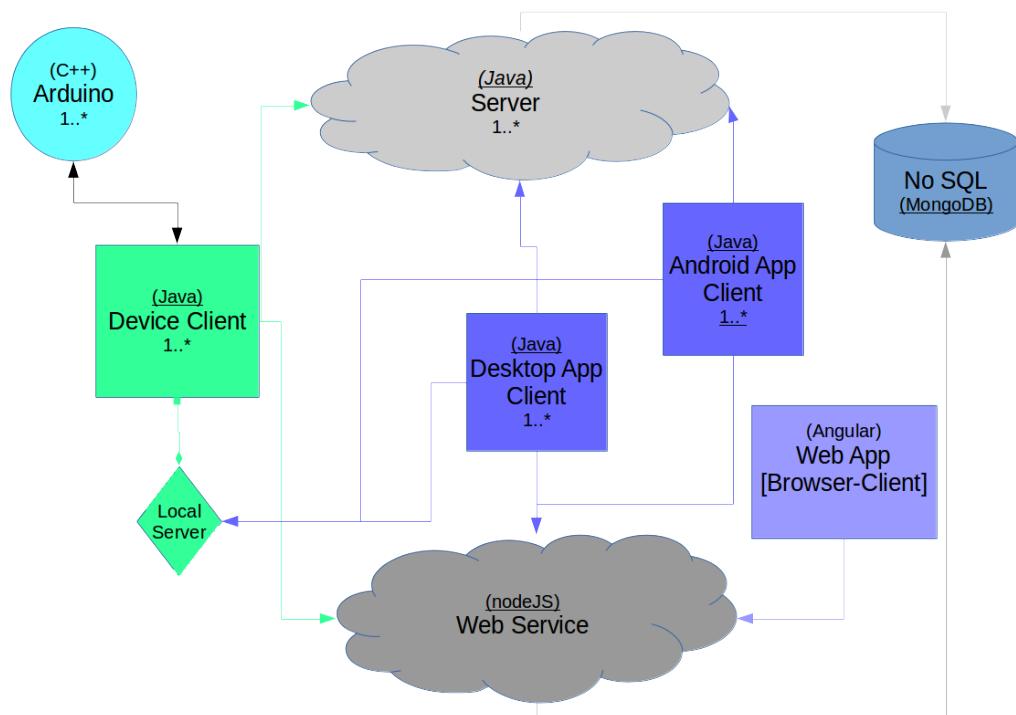
Το Git είναι το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων για ανάπτυξη λογισμικού.

4

Ανάλυση Απαιτήσεων Συστήματος

Σε αυτή την ενότητα περιγράφεται η αρχιτεκτονική, γίνεται ανάλυση των λειτουργιών του συστήματος και τέλος παρουσιάζεται το μοντέλο οντοτήτων συσχετίσεων.

4.1 Αρχιτεκτονική



Σχήμα 4.1.1: Αρχιτεκτονική κατανεμημένου συστήματος.

Για να επιτευχθεί ο σκοπός και να μπορούν να πραγματοποιηθούν τα διάφορα σενάρια χρήσης που καθορίστηκαν ως προδιαγραφές της εφαρμογής, σχεδιάστηκε η αρχιτεκτονική ενός κατανεμημένου συστήματος (Σχήμα 4.1.1), όπου αποτελείται συνοπτικά από τα εξής μέρη:

Λογισμικό από την πλευρά του έξυπνου χώρου/σπιτιού

- 1. Arduino:** Πρότυπο sketch ώστε να μπορεί εύκολα να προσαρμοστεί ανάλογα τις εκάστοτε μονάδες* που θέλουμε να έχει ο κάθε μικρό ελεγκτής. (Επικοινωνεί με τον Device Client)
- 2. Device Client:** Επικοινωνεί με ένα ή περισσότερα arduino σειριακά (ενσύρματα [πχ USB καλώδιο] ή ασύρματα [πχ Bluetooth]) και με τον Server, ώστε να μπορεί να γίνει Real Time εποπτεία. Επίσης, επικοινωνεί με το Web Service ώστε να μπορεί να γίνεται καταγραφή διαφόρων events (πχ πυρκαγιά) και παράλληλα να διαβάζονται από το Web Service διάφορες υπενθυμίσεις που γίνονται αυτόματα από το σύστημα (πχ να κλείσουν κάποια φώτα, να παρθεί κάποιο χάπι κ.ά).

Λογισμικό/διεπαφές των χρηστών (χρήση τοπικά μέσα στο έξυπνο σπίτι ή απομακρυσμένα)

- 3. Desktop App & Android App:** Επικοινωνούν με τον server για να πραγματοποιείται Real Time εποπτεία διαφόρων έξυπνων σπιτιών. Επίσης, δύναται να συνδεθούν τοπικά απευθείας με κάποιο Device Client για εποπτεία. Ακόμη, επικοινωνούν με το web service για να διαβάζουν τα διάφορα events που προκύπτουν στους έξυπνους χώρους του εκάστοτε χρήστη που τα χρησιμοποιεί.
- 4. Web App:** Μεταξύ άλλων, δίνει τη δυνατότητα στους χρήστες να κάνουν διαχείριση λογαριασμών (Χρηστών και Συσκευών [Devices]) και καταχώριση υπενθυμίσεων (πχ πότε να γίνεται ειδοποίηση για να παρθεί κάποιο φάρμακο, πότε να κλείσουν/ανοίξουν κάποια φώτα κ.ά).

Λογισμικό για τον πυρήνα του συστήματος

- 5. Server:** Είναι ενδιάμεσος των Device Clients (Smart homes) και των Supervision Clients (Desktop app & Android app) ώστε να πραγματοποιείται Real Time επικοινωνία μεταξύ τους.
- 6. Web Service:** API ώστε να επικοινωνούν οι clients (Device, Android, Desktop, Angular/Browser) με τον πυρήνα του συστήματος (Ελεγχόμενη πρόσβαση στην βάση δεδομένων).
- 7. No SQL Database:** Απαραίτητη βάση δεδομένων για την λειτουργία του συστήματος.

4.2 Περιγραφή κατανεμημένου συστήματος

4.2.1 Arduino

Για να μπορεί κάποιο arduino να επικοινωνεί με το σύστημα της εργασίας, αναπτύχθηκε ένα πρότυπο arduino sketch με σκοπό να μοντελοποιηθεί η διαδικασία και να μπορεί κάποιος εύκολα να το προσαρμόζει στις εκάστοτε απαιτήσεις ενός έξυπνου σπιτιού. Δηλαδή, ανάλογα τις μονάδες* που θέλει να έχει ένας συγκεκριμένος μικρο-ελεγκτής, προσαρμόζεται ανάλογα ο κώδικας του arduino sketch.

Το πρότυπο έχει έτοιμες τις λειτουργίες ώστε να επικοινωνήσει αμφίδρομα με έναν υπολογιστή (και συγκεκριμένα με τον Device Client). Πιο συγκεκριμένα είναι σε θέση να διαβάζει εντολές από το Device Client και να πράττει αναλόγως (όπως για παράδειγμα διάφορες εντολές συγχρονισμού και αλλαγής mode-value κάποιων μονάδων). Επίσης, ένα arduino κάνει μετρήσεις από το πραγματικό κόσμο (μέσω διαφόρων αισθητήρων) και παρατηρεί αν έχει αλλάξει κατάσταση πχ κάποιος διακόπτης ή λάμπα (από κάποιο πιθανόν φυσικό πρόσωπο) και στέλνει τις νέες (διαφορετικές) τιμές των μονάδων κάθε φορά στον DeviceClient.

Κάθε μονάδα έχει τα εξής 10 χαρακτηριστικά:

- (1)Όνομα ελεγκτή και (2) όνομα μονάδας. (Ο συνδυασμός των ονομάτων του Ελεγκτή & Μονάδας πρέπει να είναι μοναδικός).
- (3)Τύπος μονάδας (Μέχρι στιγμής υπάρχουν 16 από 0-15)
 - 0: "No Category Dimming", 1: "No Category Switch", 2:"Lamp", 3: "Brightness", 4: "Motion",
 - 5: "Distance", 6: "Sound", 7: "Vibration", 8: "Smoke", 9: "Temperature", 10: "Humidity",
 - 11: "switch", 12: "pillBlue", 13: "pillGreen", 14: "pillRed", 15: "pillYellow".
- (4)Mode μονάδας (υπάρχουν 5 από 0-4).
 - 0 : Auto - Η μονάδα παίρνει αυτόματα τιμή και δεν μπορεί αλλαγεί απομακρυσμένα.
 - 1 : Remote – Η μονάδα αλλάζει μόνο απομακρυσμένα .
 - 2 : Both – Η μονάδα μπορεί να αλλάζει ταυτόχρονα είτε αυτόματα είτε απομακρυσμένα.
 - 3 : Auto – Το ίδιο με το 0 αλλά μπορεί να το ρυθμίσει ο χρήστης και να το κάνει Remote.
 - 4 : Remote - Το ίδιο με το 1 αλλά μπορεί να το ρυθμίσει ο χρήστης και να το κάνει Auto.
- (5)Value μονάδας (Η τιμή της κάθε μονάδας).
- (6)Max value μονάδας (Η μέγιστη τιμή που μπορεί να πάρει μια μονάδα).
- (7)Min value μονάδας (Η μικρότερη τιμή που μπορεί να πάρει μια μονάδα).
- (8)Limit (Υπάρχουν 3 περιπτώσεις)
 - Να μην έχει Limit (NL)
 - Να έχει έναν αριθμό σαν όριο και στο τέλος τα γράμματα TB -trackbar.
 - Να έχει έναν αριθμό σαν όριο και στο τέλος τα γράμματα SW -switch.
- Χρησιμοποιείτε για να ξέρουν οι User Clients πως να φερθούν στην συγκεκριμένη μονάδα. Αν η μονάδα έχει 2 ειδών εικόνες (Πχ ανοιχτή και κλειστή λάμπα) τότε αν το value περάσει το όριο δείχνει ο User Client ανοιχτή την λάμπα και το ανάποδο. Το “TB” σημαίνει ότι αν επιχειρήσει ένας χρήστης να αλλάξει τη τιμή της μονάδας θα έχει την επιλογή μιας trackbar αλλιώς αν είναι “SW” θα έχει 2 καταστάσεις η μονάδα και από τη μία κατάσταση θα γίνεται η άλλη πχ κλειστό-ανοιχτό...
- (9)Tag Μονάδας (Κάποια ετικέτα για τη μονάδα. Πχ Αν γραφτεί η λέξη “Event” τότε η συγκεκριμένη μονάδα προκαλεί event όταν το value της περάσει το limit).
- (10)isAnalog (true αν είναι αναλογική η μονάδα και false αν είναι ψηφιακή).

4.2.2 *Device Client*

- Το Device Client αναπτύχθηκε ώστε να αναγνωρίζει, να συντονίζει και να επικοινωνεί με όλους τους μικροελεγκτές (arduino) που είναι συνδεδεμένοι στον υπολογιστή που “τρέχει”.
 - Σκοπός του είναι να συνδέεται με τον Server (και αφού κάνει login με κάποιο λογαριασμό συσκευής) να του στείλει όλες τις πληροφορίες από τα arduino και τις μονάδες τους, που ήδη έχουν αναγνωριστεί πάνω στον υπολογιστή.
 - Στην συνέχεια μόλις έχει γίνει απόλυτα η αρχικοποίηση και ο πλήρης συντονισμός με το server και τα arduino, αναλαμβάνει να ενημερώνει σε πραγματικό χρόνο το Server με τις όποιες αλλαγές προκύπτουν στις μονάδες των arduino.
Παράλληλα, ο “DeviceClient” είναι έτοιμος να δεχθεί και να αλλάξει τις καταστάσεις των μονάδων που “βλέπει”, αν το “ζητήσει” κάποιος χρήστης που εποπτεύει “ζωντανά” εκείνη την ώρα το έξυπνο σπίτι.
 - Επιπλέον, όταν κάποια μονάδα προκαλέσει event, ο Device Client στέλνει στο Web Service ανάλογο μήνυμα ώστε να καταγραφεί το γεγονός.
 - Ακόμη, ο Device Client διαβάζει από το Web Service “υπενθυμίσεις” (Reminders) που πρέπει να πραγματοποιηθούν στις μονάδες των arduino του.
-
- ◆ Ακόμη το πρόγραμμα είναι σε θέση να αναγνωρίζει και να προσθέτει συνεχώς νέα arduino αν συνδεθούν στον υπολογιστή. Όπως επίσης, καταλαβαίνει αν χαθεί η σύνδεση με κάποιο arduino και ενημερώνει κατάλληλα το Server ώστε να το πληροφορηθούν οι εκάστοτε User Clients. Μαζί με το παραπάνω είναι σε θέση, στην περίπτωση που χαθεί η σύνδεση με το Server, να κάνει ανά διαστήματα προσπάθειες να ξανά-συνδεθεί με τον εξυπηρετητή.
 - ◆ Ο Device Client δεν έχει γραφικό περιβάλλον και όλα τα στοιχεία του λογαριασμού συσκευής όπως και όλες οι υπόλοιπες ρυθμίσεις υπάρχουν σε ένα αρχείο xml (που συμπληρώνει κατάλληλα κάποιο φυσικό πρόσωπο) και διαβάζονται από το πρόγραμμα όταν πρώτο-ξεκινάει (Όπως devicename, password, ip και port του Server).

4.2.3 *User Clients (Desktop & Android)*

Οι User Clients (Android & Desktop) υλοποιήθηκαν με σκοπό οι χρήστες του συστήματος:

- Να έχουν την δυνατότητα να εποπτεύουν και να διαχειρίζονται απομακρυσμένα (ή και τοπικά) τις συσκευές (έξυπνα σπίτια) τους, σε πραγματικό χρόνο, μέσω διεπαφών με γραφικό περιβάλλον.
 - Να μπορούν να λαμβάνουν ειδοποιήσεις άμεσα όταν συμβεί κάποιο γεγονός σε κάποια συσκευή (έξυπνο σπίτι) τους, ανεξάρτητα αν κάνουν εκείνη την ώρα εποπτεία ή όχι.
-
- ◆ Το Desktop User Client αναπτύχθηκε για να λειτουργεί σε υπολογιστές (pc, laptop) ανεξαρτήτως ποιου λειτουργικού συστήματος χρησιμοποιούν (πχ Linux, macOS, Windows).

- ◆ To Android User Client αναπτύχθηκε για να λειτουργεί σε φορητές συσκευές (κινητά, tablets) που έχουν λειτουργικό σύστημα Android.

4.2.4 Server

Ο Server αναλαμβάνει αν συνδεθεί μαζί του κάποιο Java Client (Device ή User) να το αναγνωρίζει και να κάνει την ανάλογη πιστοποίηση χρήστη ή συσκευής με την βοήθεια της Database.

- ◆ Αν ο Client που συνδέθηκε μαζί του είναι μια συσκευή (Device Client) τότε δημιουργείτε ένα νέο “κανάλι” που μπορούν αργότερα να συνδεθούν σε αυτό χρήστες (User Clients) που έχουν στη συσκευή δικαιώματα. Επίσης, ενημερώνει όλους τους χρήστες (Desktop ή Android User Clients) που είναι ήδη συνδεδεμένοι και έχουν δικαιώματα στη συσκευή ότι κάποια νέα συσκευή είναι διαθέσιμη για εποπτεία.
- ◆ Αν ο Client που συνδέθηκε μαζί του είναι κάποιος χρήστης για εποπτεία (User Client), τότε ο Server με τη βοήθεια της βάσης δεδομένων στέλνει στον χρήστη τις συσκευές (Devices/Έξυπνα σπίτια) που έχει δικαιώματα και την κατάσταση τους (πχ αν είναι up ή down, δηλαδή αν είναι ήδη συνδεδεμένοι πάνω στο Server ή όχι). Σε επόμενο χρόνο, αν ο χρήστης επιλέξει να κάνει εποπτεία σε μια συσκευή, τότε ο Server στέλνει όλες τις απαραίτητες πληροφορίες που έχει ήδη από το αντίστοιχο Device Client και “βάζει” στο ίδιο “κανάλι” τον User Client με το Device Client για να μπορούν να επικοινωνούν.

4.2.5 Web App

Μέσα από το Web Application (Ιστοσελίδα) του συστήματος μπορεί κάποιος χρήστης:

- Να δημιουργεί (Register) και να διαχειρίζεται (View-Edit) έναν λογαριασμό χρήστη.
- Να δημιουργεί και να διαχειρίζεται λογαριασμούς συσκευών (έξυπνων σπιτιών).
- Να ψάχνει συσκευές άλλων χρηστών και να κάνει αίτημα ώστε να αποκτήσει δικαιώματα προσπέλασης σε άλλα έξυπνα σπίτια. Ταυτόχρονα, μπορεί να δέχεται και να δίνει τα κατάλληλα δικαιώματα σε άλλους χρήστες που του έχουν κάνει κάποιο αίτημα.
- Να βλέπει τα events που έχουν προκληθεί στις συσκευές του (έξυπνα σπίτια).
- Να προγραμματίζει πότε (συγκεκριμένη χρονική στιγμή) να γίνονται διάφορες αλλαγές στις τιμές των μονάδων των έξυπνων σπιτιών του (πχ ένα led να πάρει τιμή φωτεινότητας 200).
- Να βλέπει πληροφορίες σχετικά με την διπλωματική εργασία (θέμα, συντελεστές κλπ), να διαβάσει τη τεκμηρίωση της εργασίας (Documentation), να κατεβάσει τα επιμέρους λογισμικά του συστήματος (Των clients κυρίως αλλά και του template sketch για να προσαρμοστεί κατάλληλα), να κάνει login/logout κ.ά.

4.2.6 Web Service

Το Web Service πρακτικά είναι μια διεπαφή (API) για τις client εφαρμογές του κατανεμημένου συστήματος ώστε να έχουν ελεγχόμενη πρόσβαση στην βάση δεδομένων. Έτσι, παρέχεται μεγαλύτερη ασφάλεια, ευκολότερη μελλοντική επεκτασιμότητα και γενικότερα καλύτερη διαχείριση του συστήματος.

- Όλο το δυναμικό περιεχόμενο του Web Application γεμίζει από απαντήσεις του Web Service (μετά από αιτήσεις που κάνει το App).
- Ο Device Client, όταν συμβαίνουν events στον έξυπνο χώρο, τα στέλνει στο Web Service.
- Ο Device Client λαμβάνει μέσω του Web Service τις “υπενθυμίσεις” (reminders) που έχουν προγραμματιστεί να συμβούν στο έξυπνο σπίτι από κάποιον χρήστη.
- Οι User Clients (Android και Desktop) λαμβάνουν από το Web Service τα events που έχουν συμβεί στα smart homes τους, ώστε να ειδοποιούνται οι χρήστες.

4.3 Μοντέλο Οντοτήτων Συσχετίσεων

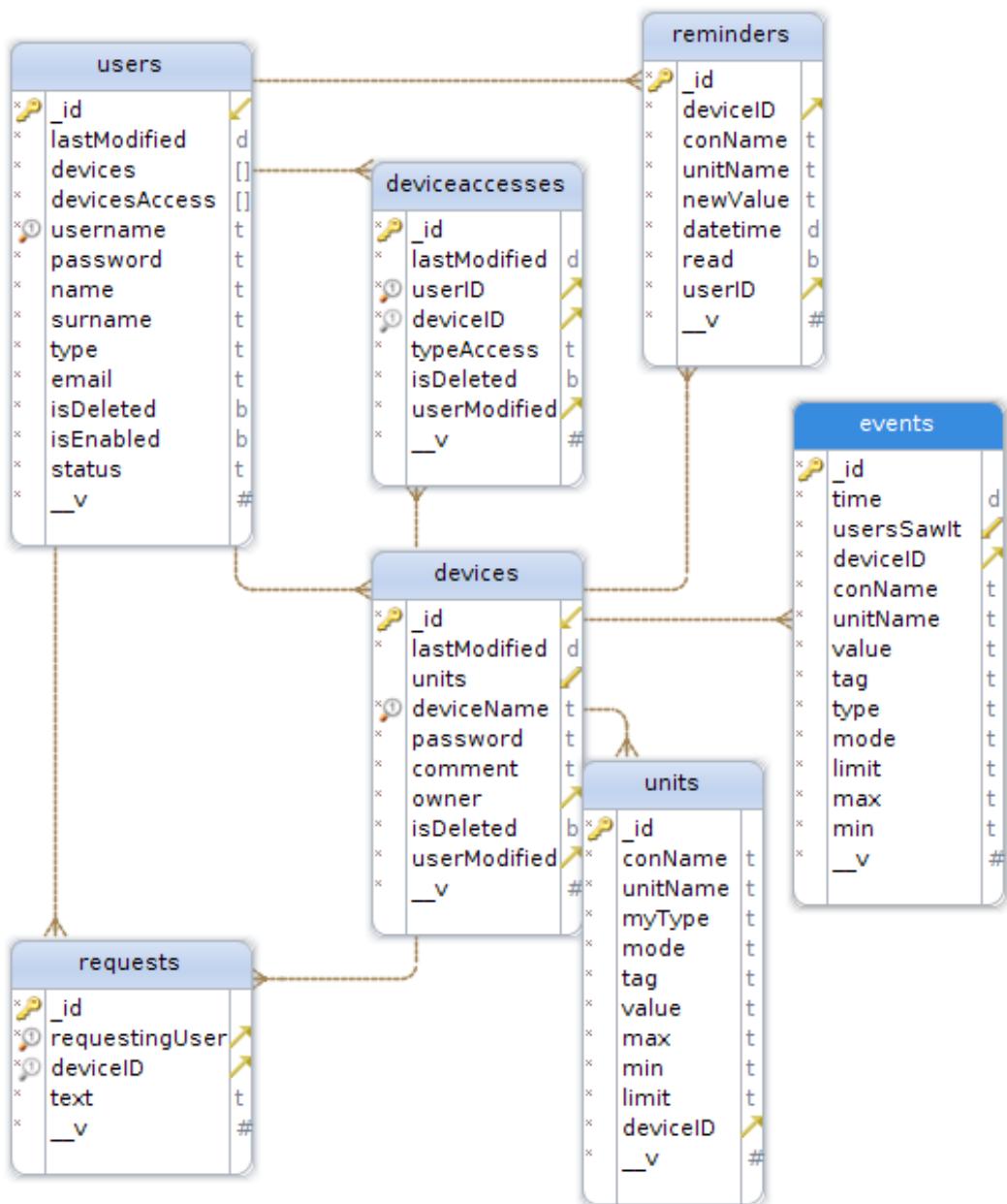
Στο σχήμα 4.3.1 που ακολουθεί διακρίνονται οι οντότητες της βάσης δεδομένων του συστήματος και πως αυτές συσχετίζονται μεταξύ τους.

Φαίνεται ότι ένας χρήστης (**Users**) μπορεί να έχει:

- πολλά δικά του σπίτια/συσκευές (**Devices**).
- δικαιώματα προσπέλασης σε πολλές συσκευές (Devices) άλλων χρηστών (**Deviceaccesses**).
- κάνει πολλές καταχωρήσεις ειδοποιήσεων (**Reminders**) για τα έξυπνα σπίτια του.
- κάνει πολλές αιτήσεις (**Requests**) για να αποκτήσει δικαιώματα σε άλλα έξυπνα σπίτια.

Και κάθε έξυπνο σπίτι (**Devices**) μπορεί να έχει:

- πολλές μονάδες* (**Units**), συνδεδεμένους στους μικρο-ελεγκτές.
- πολλά συμβάντα (**Events**) που να το αφορούν.
- πολλές ειδοποιήσεις (**Reminders**) που να το αφορούν.
- πολλές αιτήσεις (**Requests**) για δικαιώματα προσπέλασης που να το αφορούν.
- έναν χρήστη (**Users**) σαν owner (Αυτόν που δημιούργησε την συσκευή).
- πολλούς χρήστες που να έχουν δικαιώματα (**Deviceaccesses**) για εκείνη τη συσκευή.



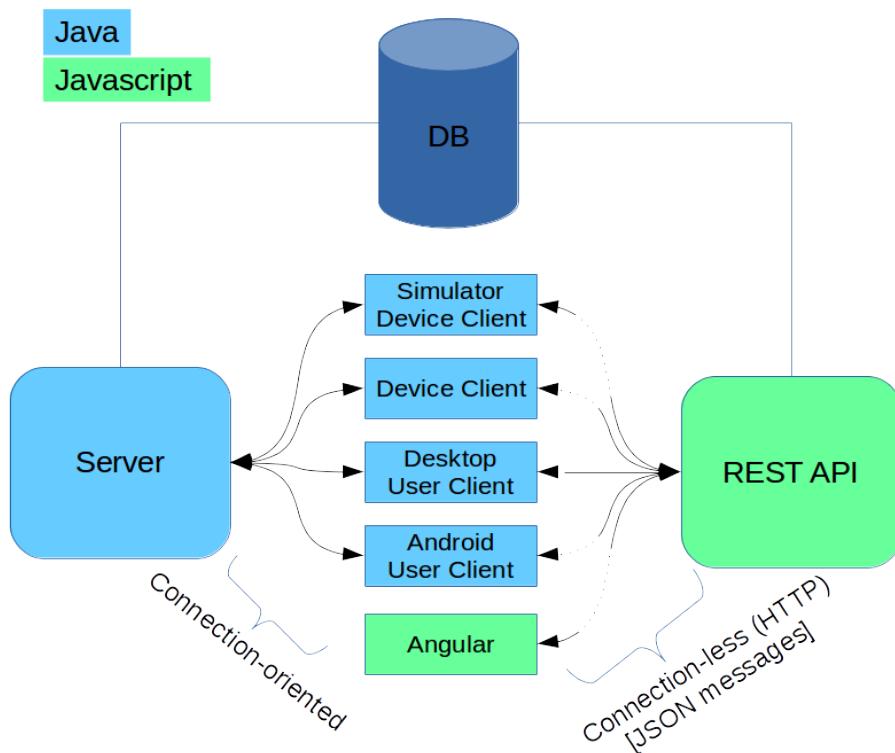
Σχήμα 4.3.1: Μοντέλο οντοτήτων συσχετίσεων της βάσης δεδομένων.

5

Σχεδίαση Συστήματος

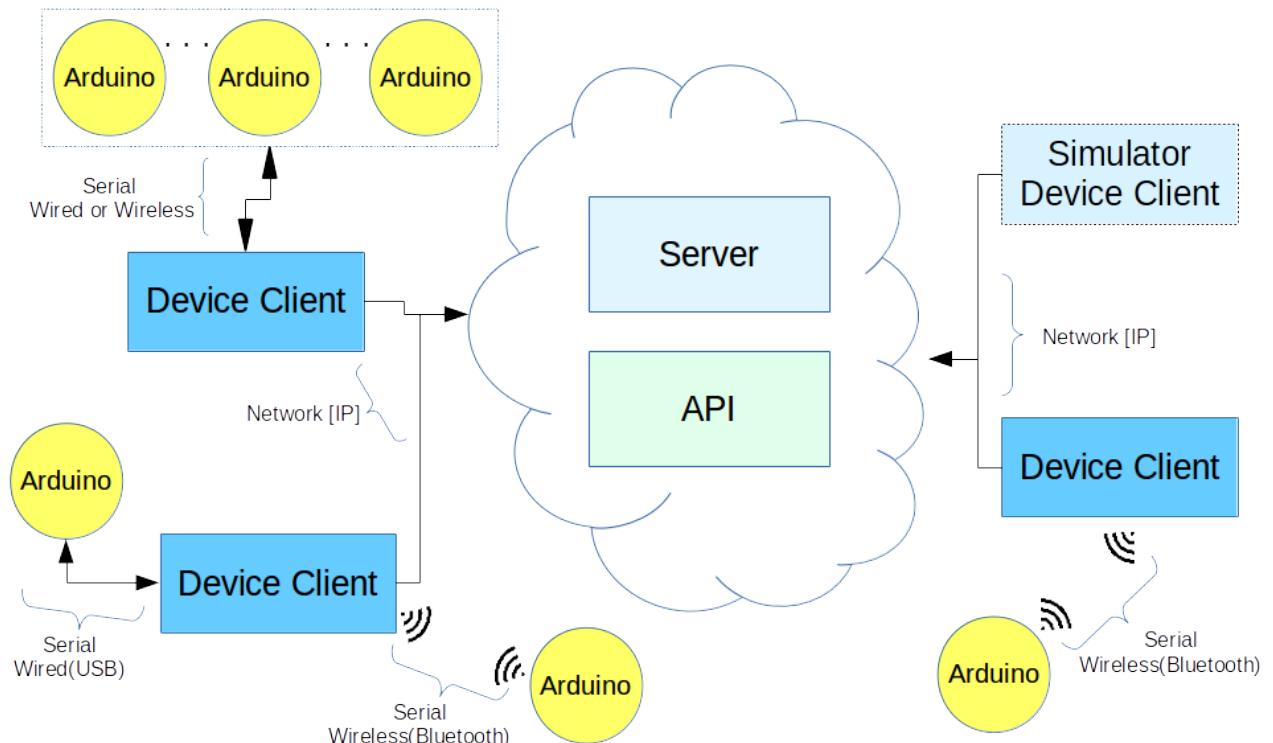
Σε αυτή την ενότητα περιγράφετε η σχεδίαση της αρχιτεκτονικής και των δυνατοτήτων του συστήματος καθώς και το πως σχεδιάστηκαν τα επιμέρους λογισμικά και η βάση δεδομένων.

5.1 Αρχιτεκτονική



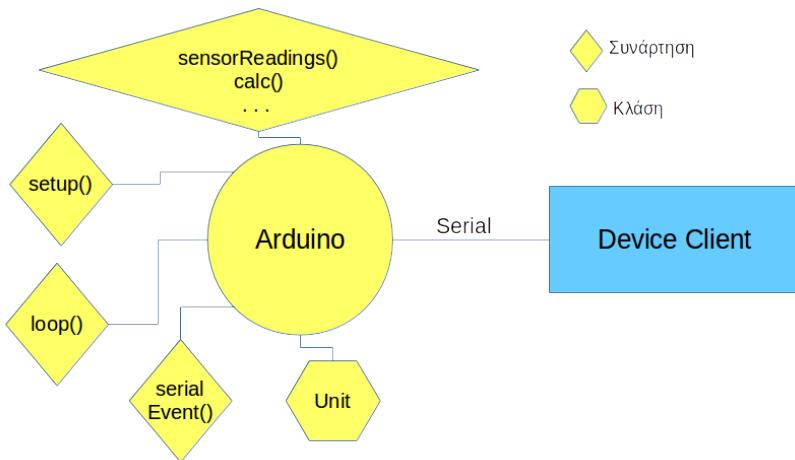
Σχήμα 5.1.1: Ένα άλλο σχέδιο της αρχιτεκτονικής του κεντρικού συστήματος.

Στο σχήμα 5.1.1 φαίνεται ένα άλλο σχέδιο της αρχιτεκτονικής του συστήματος και πιο συγκεκριμένα πως οι επιμέρους εφαρμογές αλληλεπιδράσουν μέσω δικτύου μεταξύ τους. Διακρίνεται ότι όλοι οι clients επικοινωνούν με το RESTFull API (Web service) με τη χρήση JSON μηνυμάτων και επίσης ότι οι Java Clients επικοινωνούν με τον Java Server μέσω network sockets. Ακόμη, φαίνεται ότι άμεση πρόσβαση στην βάση δεδομένων έχουν μόνο ο Java Server (που μπορεί να είναι περισσότεροι από ένας σε διαφορετικά μηχανήματα) και το Web service. Στο σχέδιο εμφανίζεται ένας Simulator Device Client, ο οποίος είναι γραμμένος σε Java και αναπτύχθηκε επειδή χρειάστηκε στην αρχή της ανάπτυξης του συστήματος, παίζοντας το ρόλο του Device client με εικονικούς μικρο-ελεγκτές και μονάδες (χωρίς πραγματικά arduino δηλαδή).



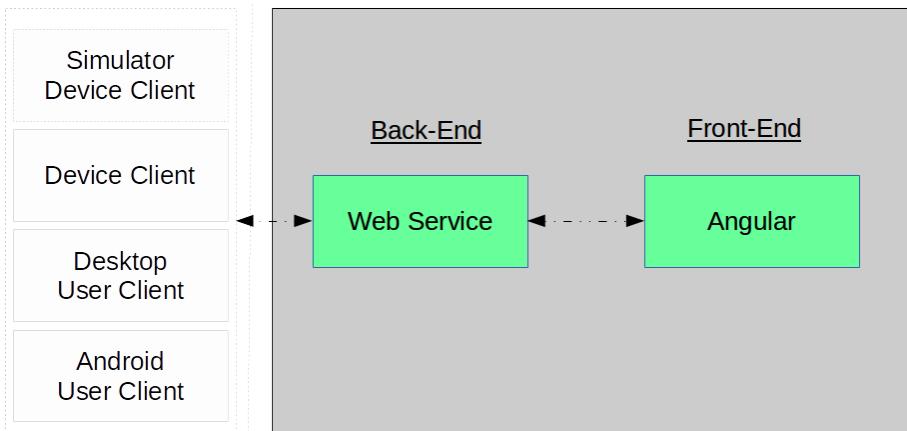
Σχήμα 5.1.2: Επικοινωνία των Device Clients (έξυπνα σπίτια) με τα arduino και με το κεντρικό σύστημα.

Στο σχέδιο του σχήματος 5.1.2 φαίνεται πως τα arduino επικοινωνούν σειριακά με τους Device Clients, είτε ενσύρματα με USB καλώδιο είτε ασύρματα (πχ με Bluetooth). Επίσης, διακρίνεται από το σχέδιο ότι οι Device Clients (ή ακόμη και κάποιος Simulator) επικοινωνεί μέσω δικτύου με το κεντρικό σύστημα, δηλαδή με τον Server (Java) και το API (Web Service) του συστήματος.



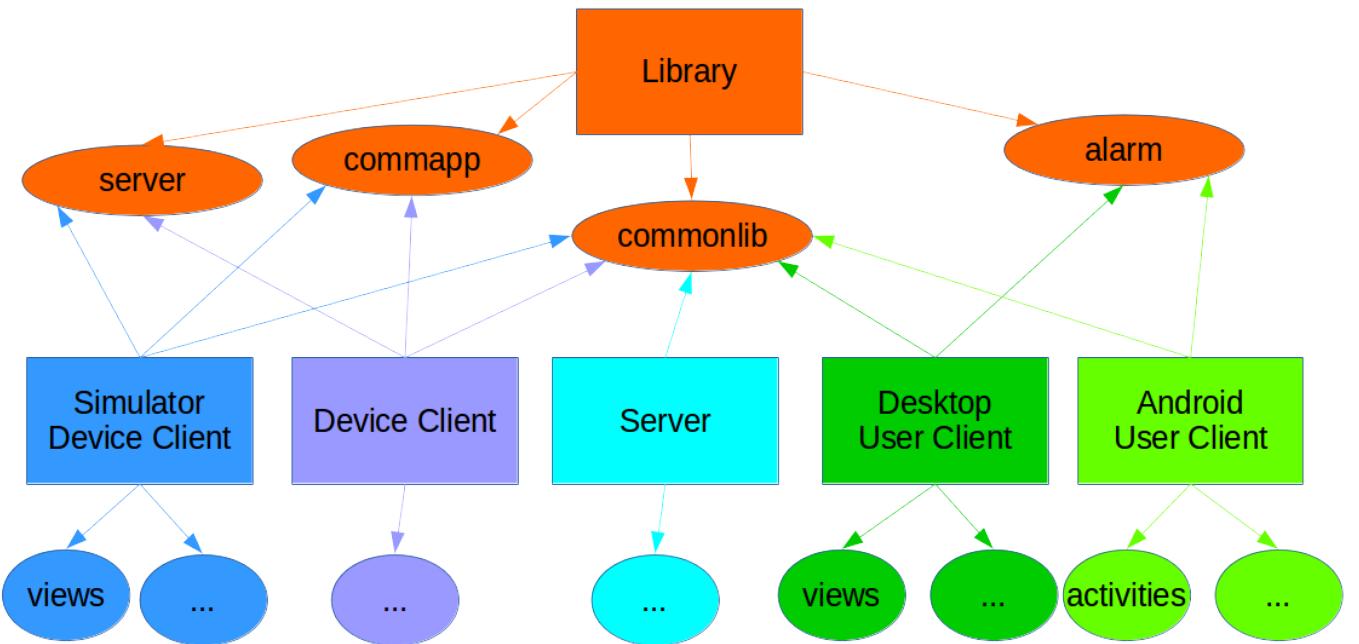
Σχήμα 5.1.3: Επικοινωνία ενός arduino με ένα Device client.

Στο σχήμα 5.1.3 φαίνεται (με έναν αφηρημένο τρόπο) πως παραμετροποιώντας ένα πρότυπο arduino sketch με βάση τις [μονάδες*](#) που θέλει κάποιος να βάλει σε ένα arduino, γίνεται να υπάρξει επικοινωνία μεταξύ ενός Device client και του μικροελεγκτή. Στην 5.3.9 παράγραφο περιγράφονται περισσότερες λεπτομέρειες.



Σχήμα 5.1.4: Επικοινωνία των clients με το Web Service.

Στο σχήμα 5.1.4 φαίνεται πως επικοινωνούν όλοι οι clients με το Web Service και δίνεται περισσότερη έμφαση στην επικοινωνία του Web Service (Back-end) με το Angular App (Front-end) που τρέχει σε κάποιον browser (όπως πχ Firefox, Chrome κ.ά.). Και οι δύο αυτές εφαρμογές έχουν γραφτεί σε Javascript και η υλοποίηση βασίστηκε στο [MEAN Stack*](#).

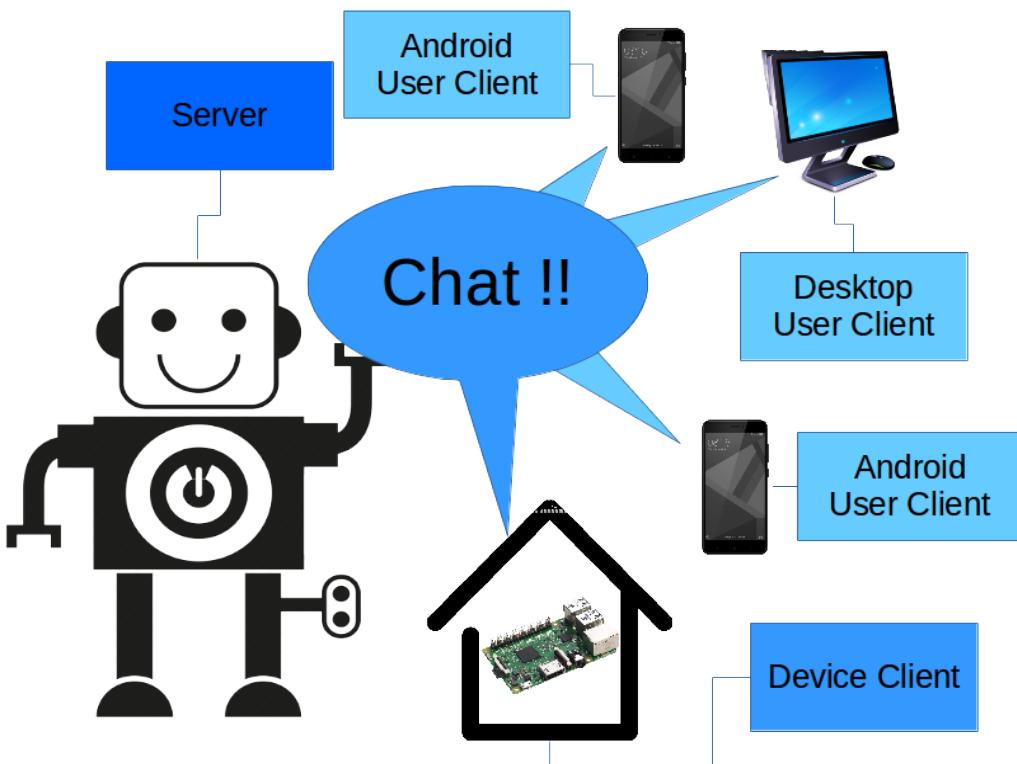


Σχήμα 5.1.5: Java πακέτα με λειτουργίες για τις java επιμέρους εφαρμογές του συστήματος.

Αναπτύχθηκε ένα Maven project (Library) ώστε να έχει το ρόλο μιας βιβλιοθήκης. Η βιβλιοθήκη αυτή περιέχει διάφορες κοινές λειτουργίες των Java εφαρμογών και γίνεται μοντελοποίηση κάποιων οντοτήτων με σκοπό να κληρονομηθούν από τις επιμέρους εφαρμογές. Συγκεκριμένα, το Library έχει 4 packages: το **server** με λειτουργίες local server του device client & Simulator, το **commapp** με λειτουργίες επικοινωνίας του Device Client & Simulator με το Web Service, το **alarm** με λειτουργίες επικοινωνίας του Desktop & Android User client με το Web Service και τέλος το **commonlib** με διάφορες κοινές κλάσεις, μεταβλητές και μεθόδους (static) που χρειάζονται σε όλα τα Java Project του συστήματος. Τα προαναφερθέντα διακρίνονται και στο σχήμα 5.1.5 όπου φαίνεται επίσης ότι κάθε ένα από τα επιμέρους projects του συστήματος έχουν και δικές τους κλάσεις (στο σχήμα: "..."). Ακόμη, στο σχήμα 5.1.5 φαίνεται ότι τα Projects που έχουν γραφικό περιβάλλον (Simulator, Desktop & Android) έχουν το κάθε ένα από ένα package (views, activities) για τις κλάσεις που αφορούν το GUI, ώστε να υπάρχει καλύτερη και ευκολότερη διαχείριση.

5.2 Σχεδιασμός των δυνατοτήτων του συστήματος

Αρχικά, να αναφερθεί ότι για να γίνεται δομημένα και αυτοματοποιημένα η επικοινωνία μεταξύ των Java εφαρμογών (με τη χρήση των sockets), σχεδιάστηκε ένα “πρωτόκολλο” επικοινωνίας. Ουσιαστικά δηλαδή μια σύμβαση ότι τα μηνύματα που θα ανταλλάσσονται μεταξύ των εφαρμογών, από την αρχή μέχρι το τέλος μιας socket σύνδεσης, θα έχουν μια συγκεκριμένη JSON δομή. Αυτό διευκολύνει στην καλύτερη διαχείριση και τη γρήγορη μετατροπή των μηνυμάτων σε objects.



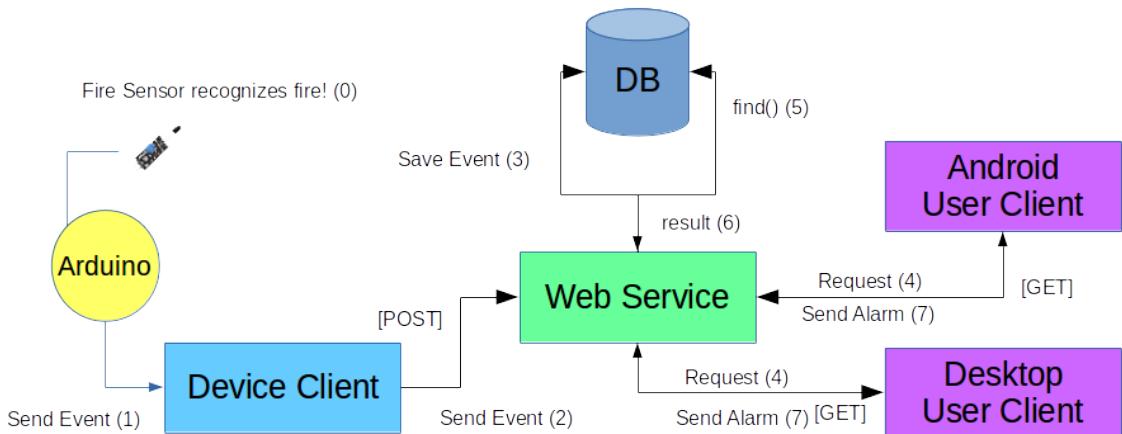
Σχήμα 5.2.1: Παρομοίωση της επικοινωνίας των Java εφαρμογών ως μια απλή chat εφαρμογή.

Η λογική της επικοινωνίας των Java εφαρμογών, ώστε να πραγματοποιούνται οι real time εποπτείες, μοιάζει πολύ με τον τρόπο που λειτουργεί ένα απλό chat συνομιλιών. Δηλαδή, θα μπορούσε κάποιος να παρομοιάσει αυτήν την λειτουργία της επικοινωνίας των εφαρμογών, ως “ιδιωτικές συνομιλίες” πολλών συνομιλητών ενός chat. Όπου, αντί συνομιλητές έχουμε Java εφαρμογές (User & Device clients) και αντί για απλά μηνύματα κειμένου ανταλλάσσονται συμβολοσειρές (με τη μορφή του “πρωτοκόλλου” που αναφέρθηκε προηγουμένως).

Πιο συγκεκριμένα, όταν προκληθεί μια αλλαγή, είτε από το ίδιο το device (πχ αλλάζει τιμή ένας αισθητήρας) είτε από κάποιον χρήστη (πχ επιλέγει να αλλάξει τιμή κάποιο led) που εποπτεύει τη συσκευή, στέλνετε (με σωστή μορφή του πρωτοκόλλου) σε όλους που είναι στο ίδιο “κανάλι” η νέα αλλαγή και έτσι ενημερώνεται κατάλληλα η κάθε “πλευρά”. Οπότε, όταν πάρει η συσκευή (Device Client) κάποιο μήνυμα κειμένου για να αλλάξει η τιμή μιας μονάδας, αλλάζει στην πράξη η τιμή ενημερώνοντας (σειριακά) το κατάλληλο arduino. Ενώ, όταν πάρει κάποιος User

Client το μήνυμα κειμένου δείχνει στον χρήστη/επόπτη με γραφικό τρόπο την αλλαγή που έχει συμβεί.

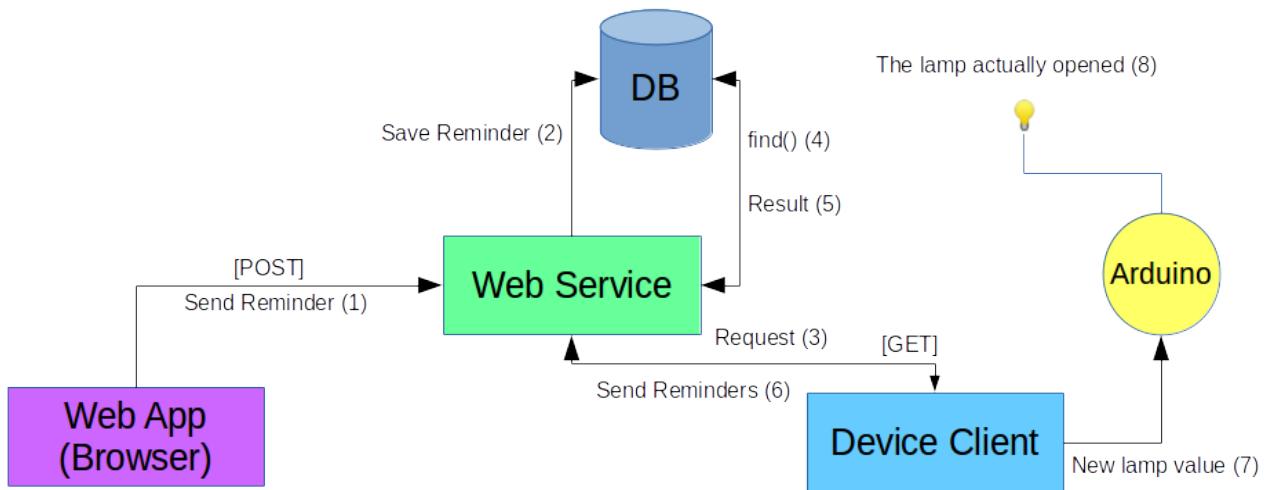
Όπως φαίνεται και στο σχήμα 5.2.1, σε κάθε “κανάλι” υπάρχει μια συσκευή/έξυπνο σπίτι (Device Client) και όλοι οι χρήστες (User Clients) που το εποπτεύουν, και η πρακτική διαχείριση όλων των “καναλιών” γίνεται από το Server.



Σχήμα 5.2.2: Μια περίπτωση αναγνώρισης γεγονότων (Events) και ειδοποίηση των χρηστών.

Το σενάριο του σχήματος 5.2.2 περιγράφεται ως εξής:

0. Προκαλείται κάποιο γεγονός. Στην περίπτωση του σχήματος κάποιος αισθητήρας φωτιάς αναγνωρίζει ότι έχουμε πυρκαγιά.
1. Το arduino στέλνει στο Device Client ότι πραγματοποιήθηκε κάποιο event.
2. Ο Device Client στέλνει στο Web Service το event που έγινε.
3. Το Web Service αποθηκεύει στην βάση δεδομένων το event.
4. Οι User Clients (Android or Desktop), ανά κάποιο χρονικό διάστημα (πχ 10 δευτερόλεπτα), “ρωτάνε” το Web service μήπως έχουν συμβεί events στα έξυπνα σπίτια που τους αφορούν και που δεν έχουν ενημερωθεί στο παρελθόν.
5. Το Web Service “ρωτάει” την βάση δεδομένων για το αν υπάρχουν events για τον χρήστη.
6. Το Web Service παίρνει τα αποτελέσματα από το query.
7. Το Web Service στέλνει τα events ώστε να ειδοποιηθούν (alarm) οι χρήστες/επόπτες.



Σχήμα 5.2.3: Μια περίπτωση καταχώρησης “υπενθύμισης” (Reminder) και η πραγματοποίηση της αλλαγής.

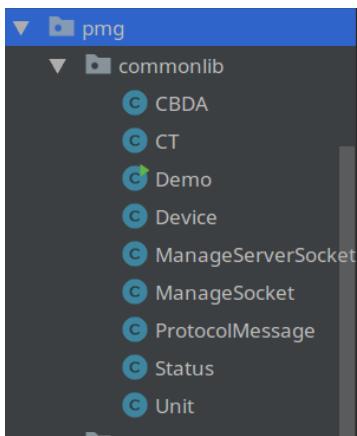
Το σενάριο του σχήματος 5.2.3 περιγράφεται ως εξής:

1. Κάποιος χρήστης μέσα από το Web Application καταχωρεί μια υπενθύμιση η οποία στέλνεται στο Web Service.
2. To Web Service αποθηκεύει την συγκεκριμένη “υπενθύμιση” στην βάση δεδομένων.
3. Ο Device Client, ανά κάποιο χρονικό διάστημα (πχ 10 δευτερόλεπτα), “ρωτάει” το Web service μήπως έχει υπενθυμίσεις που να τον αφορούν για την συγκεκριμένη χρονική στιγμή.
4. To Web Service “ρωτάει” την βάση δεδομένων για το αν υπάρχουν υπενθυμίσεις για το έξυπνο σπίτι (Device Client) για αυτήν την χρονική στιγμή.
5. To Web Service παίρνει τα αποτελέσματα από το query.
6. To Web Service στέλνει τις υπενθυμίσεις στον Device Client.
7. Εφόσον ο Device Client παραλάβει κάποια “υπενθύμιση” τότε στέλνει σειριακά στο κατάλληλο arduino την νέα τιμή της μονάδας που πρόκειται να αλλάξει.
8. To Arduino πραγματοποιεί την αλλαγή με βάση τη νέα τιμή που πήρε από τον Device Client

5.3 Περιγραφή των επιμέρους λογισμικών του συστήματος

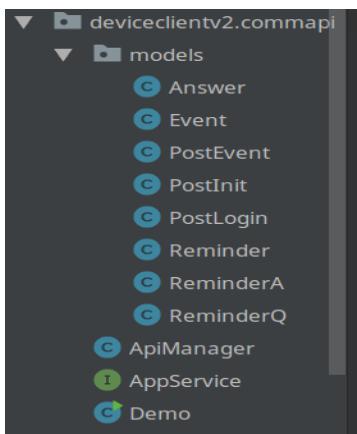
5.3.1 Library

commonlib (Χρησιμοποιείται από όλες τις επιμέρους java εφαρμογές):



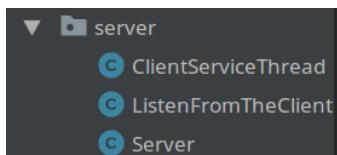
- CBDA (Communication between distributed applications). Static μέθοδοι για την επικοινωνία μεταξύ των επιμέρους λογισμικών.
- CT (Common tools). Διάφοροι static μέθοδοι σαν “εργαλεία”.
- Demo. Έχει main μέθοδο για δοκιμές (Δεν χρησιμοποιείται πρακτικά).
- Device. Μοντελοποίηση της οντότητας ενός έξυπνου χώρου/σπιτιού.
- ManageServerSocket. Για την διαχείριση των Server socket.
- ManageSocket. Για την διαχείριση των socket.
- ProtocolMessage. Μοντελοποίηση των μηνυμάτων που θα ανταλλάσσουν μεταξύ τους οι εφαρμογές (JSON μορφή).
- Status. Χρησιμοποιείται σαν τύπος επιστροφής κάποιων μεθόδων.
- Unit. Μοντελοποίηση της οντότητας μιας μονάδας.

commapi (Χρησιμοποιείται από το Device Client και τον Simulator Device Client):



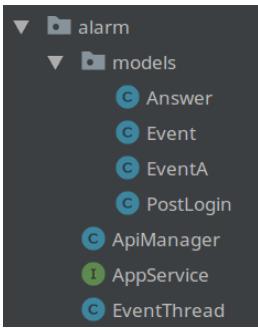
- ApiManager. Κλάση (singleton) για την διαχείριση της επικοινωνίας με το Web service (πιστοποίηση, ενημέρωση για τις μονάδες που έχει το έξυπνο σπίτι, καταχώρηση event, διάβασμα “υπενθύμισης”).
- AppService. Interface με HTTP λειτουργίες (όπως put, post, get)
- models. package με κλάσεις που μοντελοποιούν την επικοινωνία.

server (Χρησιμοποιείται από το Device Client και τον Simulator Device Client):



- ClientServiceThread. Thread για την εξυπηρέτηση (αρχικές διαδικασίες) ενός User Client (Desktop ή Android).
- ListenFromTheClient. Thread για ασύγχρονο διάβασμα μηνυμάτων από κάποιον User Client.
- Server. Thread για τη σύνδεση των User Clients στον τοπικό server.

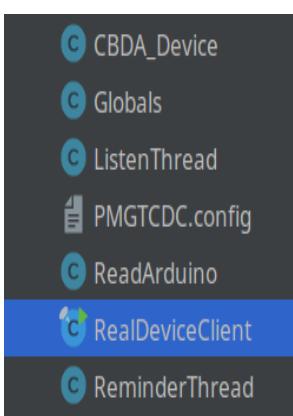
alarm (Χρησιμοποιείται από το Desktop & Android User Client):



- ApiManager. Singleton κλάση για την διαχείριση της επικοινωνίας με το Web service (πιστοποίηση, διάβασμα event για ειδοποίηση του χρήστη).
- AppService.Interface με HTTP λειτουργίες (όπως put, post, get).
- models. package με κλάσεις που μοντελοποιούν την επικοινωνία.
- EventThread: Ξεχωριστό thread που ασχολείται αποκλειστικά με το διάβασμα από το Web Service των ειδοποιήσεων για τον χρήστη.

5.3.2 Device Client

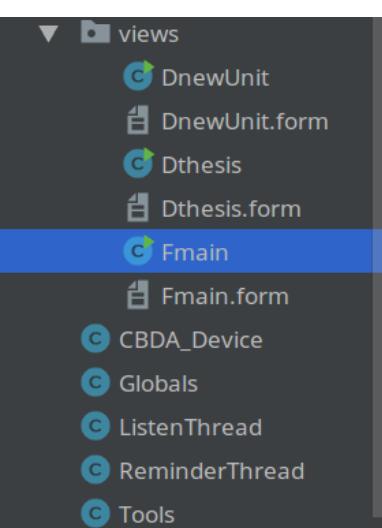
deviceclientv2:



- CBDA_Device. Κλάση που κληρονομεί την CBDA και έχει κάποιες επιπλέον διαδικασίες για την επικοινωνία του Device Client με τον Server.
- Globals. Static μεταβλητές και συναρτήσεις με σκοπό να χρησιμοποιούνται από παντού μέσα στο project.
- ListenThread. Thread για ασύγχρονο διάβασμα μηνυμάτων από τον server.
- PMGTCDC.config. XML με ρυθμίσεις(IP, port, deviceName, password κ.ά)
- ReadArduino. Κλάση που αναλαμβάνει την επικοινωνία του device client με τα arduino.
- RealDeviceClient. Κλάση που ξεκινάει τον device client.
- ReminderThread. Thread για να διαβάζονται ασύγχρονα μηνύματα (“υπενθυμίσεις”) από το web Service.

5.3.3 Simulator Device Client

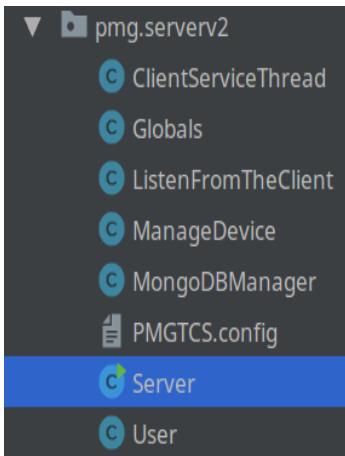
devicecliensimulatortv2:



- [views].package με 3 κλάσεις (Jframe) για το GUI της εφαρμογής
- CBDA_Device. Κλάση που κληρονομεί την CBDA και έχει κάποιες επιπλέον static διαδικασίες για την επικοινωνία του Simulator Device Client με τον Server.
- Globals. Static μεταβλητές και συναρτήσεις με σκοπό να χρησιμοποιούνται από παντού μέσα στο project.
- ListenThread. Thread για να διαβάζονται ασύγχρονα μηνύματα από τον Server.
- ReminderThread. Thread για να διαβάζονται ασύγχρονα μηνύματα (“υπενθυμίσεις”) από το web Service.
- Tools. Κλάση με static συναρτήσεις/εργαλεία.

5.3.4 Server

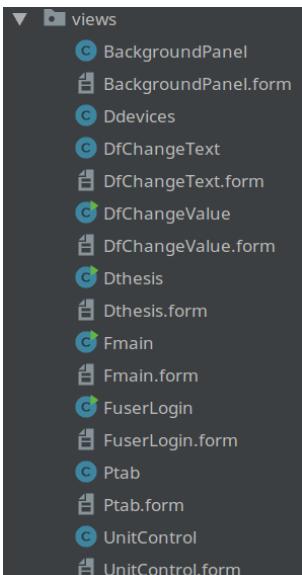
serverv2:



- ClientServiceThread. Thread για την εξυπηρέτηση (αρχικές διαδικασίες) των Device Clients (& Simulator) και User Clients (Desktop ή Android).
- Globals. Static μεταβλητές και συναρτήσεις με σκοπό να χρησιμοποιούνται από παντού μέσα στο project.
- ListenFromTheClient. Thread για ασύγχρονο διάβασμα μηνυμάτων από τους Java Clients.
- ManageDevice. Διαχείριση device καναλιού (deviceClient & userClients)
- MongoDBManager. Διαχειριστής της βάσης δεδομένων.
- PMGTCS.config. XML με ρυθμίσεις (Database κ.ά)
- Server. Κλάση όπου ξεκινάει ο Server.
- User. Μοντελοποίηση ενός χρήστη με τα στοιχεία που έχει η database.

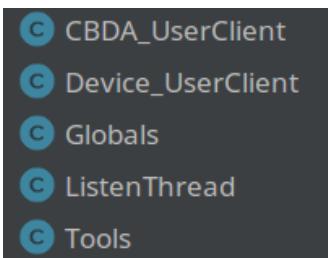
5.3.5 Desktop User Client

views:



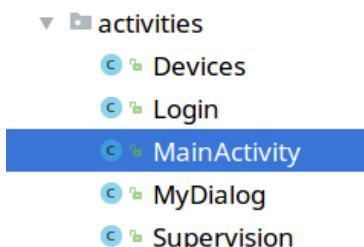
- [views].package με κλάσεις για το GUI της εφαρμογής.
- CBDA_UserClient. Κλάση που κληρονομεί την CBDA και έχει κάποιες επιπλέον static διαδικασίες για την επικοινωνία του Desktop User Client με τον Server.
- Device_UserClient. Κλάση που κληρονομεί την Device από το commonlib package.
- Globals. Static μεταβλητές και συναρτήσεις με σκοπό να χρησιμοποιούνται από παντού μέσα στο project.
- ListenThread. Thread για να διαβάζονται ασύγχρονα μηνύματα από τον server.
- Tools. Κλάση με static συναρτήσεις/εργαλεία.

userclientv2:



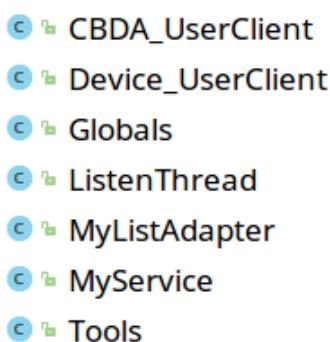
5.3.6 Android User Client

activities:



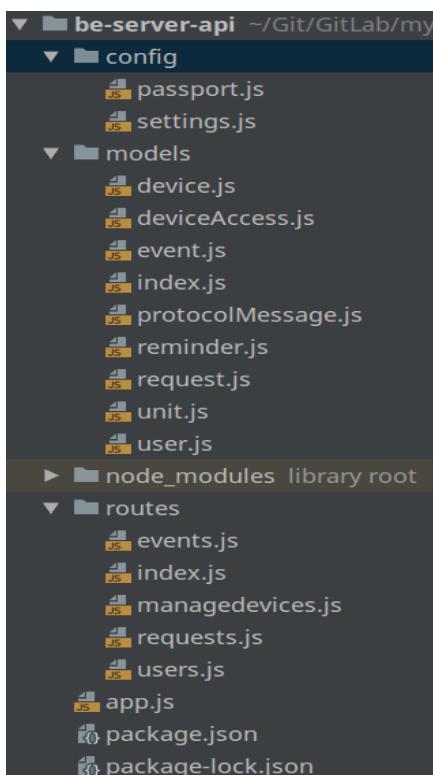
- [activities]. Package με το GUI της εφαρμογής, δηλαδή activities (διάφορες “οθόνες”) που έχει η android εφαρμογή.
- CBDA_UserClient. Κλάση που κληρονομεί την CBDA και έχει κάποιες επιπλέον static διαδικασίες για την επικοινωνία του Desktop User Client με τον Server.
- Device_UserClient. Κλάση που κληρονομεί την Device από το commonlib package.

userandroidclientv2:



- Globals. Static μεταβλητές και συναρτήσεις με σκοπό να χρησιμοποιούνται από παντού μέσα στο project.
- ListenThread. Thread για να διαβάζονται ασύγχρονα μηνύματα από τον Server.
- MyListAdapter. Απαραίτητος ArrayAdapter για να φαίνονται σε λίστα οι διάφορες μονάδες της εποπτείας.
- MyService. Android Service, για να λειτουργεί στο background η διαδικασία των ειδοποιήσεων.
- Tools. Κλάση με static συναρτήσεις/εργαλεία.

5.3.7 Web Service (Back-end)



● config

Φάκελος με διάφορες ρυθμίσεις (πχ database, port κ.ά) και με διαδικασίες πιστοποίησης (με βάση το token που έχει πάρει κάποιος client, μετά από την διαδικασία του login).

● models [χρήση mongoose/mongoDB από το MEAN stack]

Ουσιαστικά είναι μοντελοποιημένες όλες οι οντότητες του συστήματος, όπως ακριβώς βρίσκονται στην βάση δεδομένων. Επίσης, έχει μοντελοποιηθεί το ProtocolMessage. Το αρχείο index λειτουργεί απλά ως κατάλογος όλων των μοντέλων. Να αναφερθεί ότι εφόσον φτιάχτηκαν τα μοντέλα, δημιουργήθηκαν αυτόματα μέσω της εφαρμογής, όλα τα collections που βρίσκονται στην mongoDB.

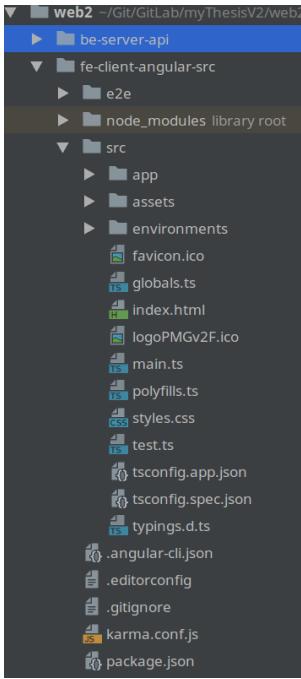
● routes [Controllers. Χρήση express από το MEAN stack]

Σε αυτόν τον φάκελο ουσιαστικά βρίσκονται όλες οι λειτουργίες CRUD (HTTP post, get, put, delete) που έχει το RESTful API.

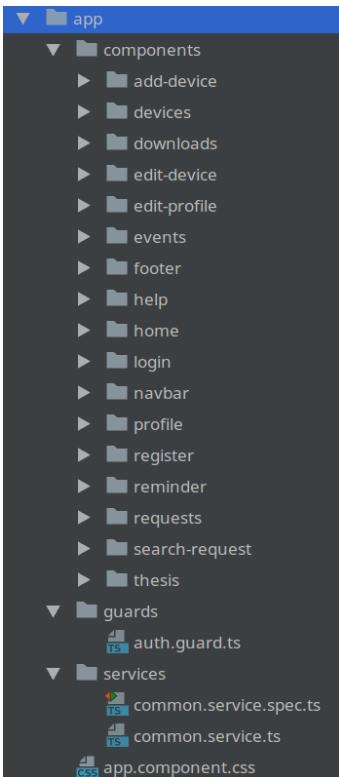
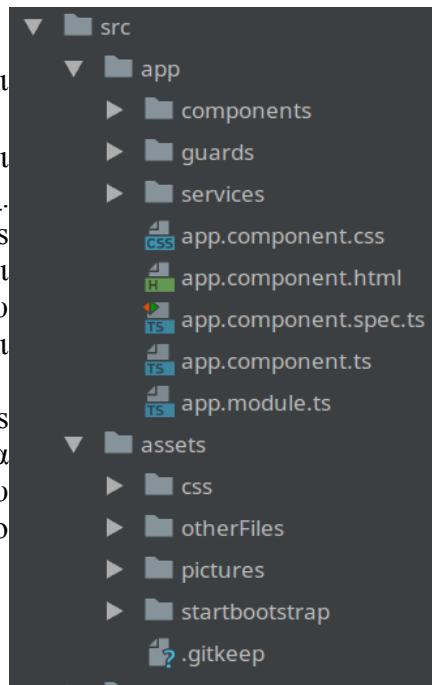
● app.js

Ουσιαστικά, αρχικοποιείται και ξεκινάει το web service.

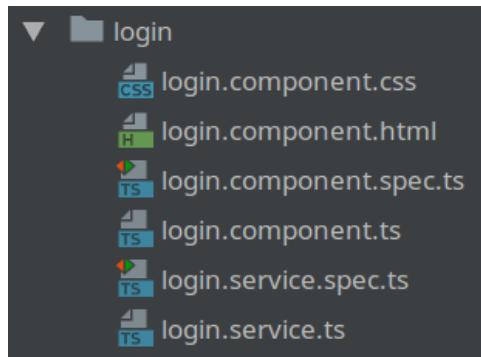
5.3.8 Angular (Front-end)



- ◆ Στην αριστερή εικόνα φαίνονται όλοι οι φάκελοι του Angular project και κάποια αρχεία.
- ◆ Στην δεξιά εικόνα φαίνονται ανοιχτοί οι φάκελοι `app` και `assets`. Ο φάκελος `app` έχει όλα τα `components` (διαφορετικές σελίδες της ιστοσελίδας), και διάφορα άλλα `css`, `html` και `typescript` αρχεία, που επιτρέπουν δυναμικό περιεχόμενο στην σελίδα (και χωρίς την ανάγκη ανανέωσης). Ο φάκελος `assets` έχει διάφορους φακέλους με `css` αρχεία, εικόνες, αρχεία (επιμέρους λογισμικά) για να τα κατεβάσει ο χρήστης και τον φάκελο του `bootstrap` (έτοιμα `css` αρχεία και άλλα για το interface της ιστοσελίδας).



- ◆ Αριστερά φαίνονται όλα τα `components`, τα οποία ουσιαστικά είναι όλες οι διαφορετικές σελίδες που έχει το Webs Site. Ο φάκελος `guards` και `services` έχουν `typescript` αρχεία που χρησιμοποιούνται από διάφορα `components`.
- ◆ Ενδεικτικά, από κάτω φαίνεται τι περιέχει το component `login`:



5.3.9 Arduino template sketch

Το template arduino sketch έχει φτιαχτεί με σκοπό να παραμετροποιείται εύκολα, ανάλογα με τις μονάδες που θέλουμε να έχει ο εκάστοτε μικρο-ελεγκτής.

Έτσι, το πρότυπο έχει μια κλάση που ονομάζεται Unit (ώστε να μοντελοποιηθεί η οντότητα της μονάδας) και κάποιες συναρτήσεις που είτε δεν χρειάζονται καμία αλλαγή είτε πρέπει να παραμετροποιηθούν ανάλογα τις ανάγκες της κάθε υλοποίησης.

Οι συναρτήσεις που έχει το πρότυπο είναι οι εξής (όπως φαίνονται και στο σχήμα 5.1.3):

- setup() [Χρειάζεται απλά να αρχικοποιηθούν οι μονάδες]
- loop() [Δεν χρειάζεται παραμετροποίηση]
- serialEvent [Δεν χρειάζεται παραμετροποίηση. Είναι υπεύθυνη για να διαβάζει τα μηνύματα από τον Device Client]
- calc() [Δεν χρειάζεται παραμετροποίηση. Είναι υπεύθυνη για την επεξεργασία των μηνυμάτων που διαβάζονται από τον Device Client]
- SensorReadings() [Χρειάζεται παραμετροποίηση ανάλογα τις μονάδες που έχει το arduino].

Να αναφερθεί ότι μπορεί να προσθέσει και άλλες συναρτήσεις κάποιος ανάλογα τις ανάγκες των μονάδων που θέλει να έχει το arduino. Επίσης, ο αναγνώστης αν θέλει μπορεί να ανατρέξει σε ένα παράρτημα (βρίσκεται στο [Github](#)) με διάφορα παραδείγματα μονάδων και με αναλυτικές οδηγίες για το πως λειτουργεί το πρότυπο sketch.

5.4 Βάση δεδομένων

Collection users:

users	
	<u>_id</u>
	<u>lastModified</u>
	<u>devices</u>
	<u>devicesAccess</u>
	<u>username</u>
	<u>password</u>
	<u>name</u>
	<u>surname</u>
	<u>type</u>
	<u>email</u>
	<u>isDeleted</u>
	<u>isEnabled</u>
	<u>status</u>
	<u>V</u>

- _id[object](κλειδί): Μοναδικό id για κάθε χρήστη.
- username[text]: Μοναδικό όνομα χρήστη.
- password[text]: Κωδικός χρήστη.
- name[text]: Πραγματικό όνομα χρήστη.
- surname[text]: Επίθετο χρήστη.
- type[text]: Τύπος χρήστη (“U”: απλός, “S”:συστήματος, “V”:VIP).
- email[text]: Email χρήστη.
- isDeleted[boolean]: true αν είναι διαγραμμένος από το σύστημα (Υπάρχει ανάγκη για λογικές διαγραφές).
- isEnabled[boolean]: false αν για κάποιο λόγο δεν είναι ενεργοποιημένος ο χρήστης (πχ δεν έχει επικυρώσει το email του [Δεν έχει φτιαχτεί τέτοια λειτουργία])
- status[text]: Μήνυμα κατάστασης του χρήστη. Πχ Δεν έχει επικυρώσει το email του.
- devices[device/object]: Λίστα με τις συσκευές που είναι owner ο συγκεκριμένος χρήστης.
- devicesAccess[deviceaccess/object]: Λίστα με τις συσκευές που έχει δικαιώματα προσπέλασης ο συγκεκριμένος χρήστης.
- lastModified[date]: Ημερομηνία τελευταίας αλλαγής των στοιχείων του χρήστη.

```
> db.users.find().pretty();
{
    "_id" : ObjectId("5a8a2d3a365cf85d0dbe90e3"),
    "lastModified" : ISODate("2018-02-19T01:49:46.063Z"),
    "devices" : [
        ObjectId("5a8a2e03365cf85d0dbe90e4"),
        ObjectId("5a8f32f989b1a777289796af")
    ],
    "devicesAccess" : [
        ObjectId("5a8fd53c24b252113e857879")
    ],
    "username" : "maria",
    "password" : "$2a$10$dmT1jv8VAhWlWaks7.9aquvK9C5QJABGilmzJ3/hqXPNuTH/GYK.u",
    "name" : "maria",
    "surname" : "maria",
    "type" : "S",
    "email" : "maria@yahoo.gr",
    "isDeleted" : false,
    "isEnabled" : true,
    "status" : "",
    "__v" : 0
}
```

Σχήμα 5.4.1: Ενδεικτικά μια εγγραφή του χρήστη “maria” στην βάση δεδομένων.

Στο σχήμα 5.4.1 διακρίνεται μεταξύ άλλων, ότι ο χρήστης maria έχει 2 δικές του συσκευές (έξυπνα σπίτια) και σε μια ακόμη συσκευή έχει δικαιώματα προσπέλασης. Επίσης, φαίνεται ότι το password αποθηκεύεται στην βάση δεδομένων κρυπτογραφημένο για μεγαλύτερη ασφάλεια.

Collection **devices** (έξυπνα σπίτια):

- **_id[object](κλειδί):** Μοναδικό id για κάθε συσκευή.
- **deviceName[text]:** Μοναδικό όνομα συσκευής.
- **password[text]:** Κωδικός συσκευής.
- **comment[text]:** Κάποιο σχόλιο για την συσκευή.
- **isDeleted[boolean]:** true αν είναι διαγραμμένη από το σύστημα. (Υπάρχει ανάγκη για λογικές διαγραφές).
- **owner[user/object]:** Ο χρήστης που δημιούργησε την συσκευή.
- **units[unit/object]:** Λίστα με τις μονάδες* που έχει κάθε συσκευή.
- **lastModified[date]:** Ημερομηνία τελευταίας αλλαγής των στοιχείων της συσκευής.
- **userModified[user/object]:** ο χρήστης που έκανε την τελευταία αλλαγή στα στοιχεία της συσκευής.

```
> db.devices.find().pretty();
{
    "_id" : ObjectId("5a8a2e03365cf85d0dbe90e4"),
    "lastModified" : ISODate("2018-02-19T01:53:07.863Z"),
    "units" : [
        ObjectId("5a9e7ff68a4f5228604b3db7"),
        ObjectId("5a9e7ff68a4f5228604b3db8"),
        ObjectId("5a9e7ff68a4f5228604b3db9"),
        ObjectId("5a9e7ff68a4f5228604b3dba"),
        ObjectId("5a9e7ff68a4f5228604b3dbb")
    ],
    "deviceName" : "perama",
    "password" : "$2a$10$UxCJhUyD/TbUn3n0vmrbQerSco2CfZUvdHg.1kaHB97xButf1amYS",
    "comment" : "perama",
    "owner" : ObjectId("5a8a2d3a365cf85d0dbe90e3"),
    "isDeleted" : false,
    "userModified" : ObjectId("5a8a2d3a365cf85d0dbe90e3"),
    "__v" : 0
}
```

Σχήμα 5.4.2: Ενδεικτικά μια εγγραφή της συσκευής “perama” στην βάση δεδομένων.

Στο σχήμα 5.4.2 διακρίνεται μεταξύ άλλων ότι η συσκευή “perama” έχει owner τον χρήστη “maria” (ίδιο ObjectId χρήστη με το σχήμα 5.4.1) και μια λίστα με 5 μονάδες.

Collection **requests** (αιτήσεις χρηστών για να πάρουν δικαιώματα από άλλους χρήστες):



- **_id[object](κλειδί):** Μοναδικό id για κάθε αίτηση.
- **requestingUser[user/object]:** Ο χρήστης που έκανε την αίτηση.
- **deviceID[device/object]:** Για ποια συσκευή απευθύνεται η αίτηση.
- **text[text]:** Κάποιο συνοδευτικό σχόλιο μαζί με την αίτηση (όπως πχ “Θα μπορούσα να έχω δικαιώματα administrator για την συσκευή;”).

Collection **deviceaccesses** (δικαιώματα προσπέλασης χρηστών σε συσκευές):

- **_id[object](κλειδί)**: Μοναδικό id για κάθε εγγραφή δικαιωμάτων.
- **userID[user/object]**: Ο χρήστης που έχει δικαιώματα στην συσκευή.
- **deviceID[device/object]**: Η συσκευή που έχει δικαιώματα ο χρήστης.
- **typeAccess[text]**: Τύπος δικαιωμάτων
 - “R”: Readonly (Μόνο εποπτεύει).
 - “F” : Full Access (Εποπτεύει & αλλάζει κατάσταση σε μονάδες)
 - “A” : Administrator (Εποπτεύει, αλλάζει κατάσταση σε μονάδες και διαχειρίζεται δικαιώματα της συσκευής άλλων χρηστών όπως ακριβώς και ο Owner της συσκευής. Δηλαδή, εγκρίνει αιτήσεις δικαιωμάτων και διαγράφει δικαιώματα από χρήστες).
- **isDeleted[boolean]**: true αν είναι διαγραμμένη από το σύστημα. (Λογική διαγραφή).
- **lastModified[date]**: Ημερομηνία τελευταίας τροποποίησης των δικαιωμάτων προσπέλασης ενός χρήστη σε μια συσκευή.
- **userModified[user/object]**: ο χρήστης που έκανε την τελευταία αλλαγή στα στοιχεία της εγγραφής.

Collection **units** (Οι μονάδες των συσκευών):

units		
_id	conName	t
unitName	unitName	t
myType	myType	t
mode	mode	t
tag	tag	t
value	value	t
max	max	t
min	min	t
limit	limit	t
deviceID	deviceID	#

- **_id[object](κλειδί)**: Μοναδικό id για κάθε μονάδα.
- **conName[text]**: Όνομα μικρο-ελεγκτή.
- **unitName[text]**: Όνομα μονάδας.
- **myType[text]**: Τύπος μονάδας (πχ διακόπτης).
- **mode[text]**: Mode μονάδας (πχ Auto).
- **tag[text]**: Tag μονάδας.
- **value[text]**: Value μονάδας (πχ “155”).
- **max[text]**: Μέγιστη τιμή που μπορεί να πάρει η μονάδα.
- **min[text]**: Ελάχιστη τιμή που μπορεί να πάρει η μονάδα.
- **limit[text]**: Όριο τιμής της μονάδας.
- **deviceID[device/object]**: Σε ποια συσκευή απευθύνεται η μονάδα.

Collection **reminders** (Οι αλλαγές καταστάσεων των μονάδων που θα γίνουν στα έξυπνα σπίτια):

reminders	
<code>_id</code>	[object](κλειδί)
<code>deviceID</code>	
<code>conName</code>	t
<code>unitName</code>	t
<code>newValue</code>	t
<code>datetime</code>	d
<code>read</code>	b
<code>userID</code>	#
	v

- **_id[object](κλειδί):** Μοναδικό id για κάθε ειδοποίηση.
- **deviceID[device/object]:** Σε ποια συσκευή απευθύνεται η ειδοποίηση.
- **conName[text]:** Σε ποιο arduino της συσκευής απευθύνεται η ειδοποίηση.
- **unitName[text]:** Σε ποια μονάδα του arduino απευθύνεται η ειδοποίηση.
- **newValue[text]:** Η νέα τιμή της μονάδας που θέλουμε να αλλάξει.
- **datetime[date]:** Ποια χρονική στιγμή θέλουμε να αλλάξει value η μονάδα.
- **read[boolean]:** “false” αν δεν έχει διαβαστεί από την συσκευή και “true” αν έχει διαβαστεί κανονικά (και άρα έχει γίνει η αλλαγή).
- **userID[user/object]:** Ποιος χρήστης καταχώρησε την ειδοποίηση.

Collection **events** (Τα συμβάντα που έχουν γίνει σε συσκευές):

events	
<code>_id</code>	
<code>time</code>	d
<code>usersSawIt</code>	
<code>deviceID</code>	
<code>conName</code>	t
<code>unitName</code>	t
<code>value</code>	t
<code>tag</code>	t
<code>type</code>	t
<code>mode</code>	t
<code>limit</code>	t
<code>max</code>	t
<code>min</code>	t
	#

- **_id[object](κλειδί):** Μοναδικό id για κάθε event.
- **time[date]:** Χρονική στιγμή που συνέβη το συμβάν.
- **deviceID[device/object]:** Σε ποια συσκευή έγινε το event.
- **conName[text]:** Από ποιον μικρο-ελεγκτή έγινε το event.
- **unitName[text]:** Από ποια μονάδα έγινε το event.
- **value[text]:** Τιμή που είχε η μονάδα όταν προκάλεσε το event.
- **tag[text]:** Tag που έχει η μονάδα που έκανε το event.
- **type[text]:** Ο τύπος της μονάδας που προκάλεσε το event.
- **mode[text]:** To mode της μονάδας που προκάλεσε το event.
- **limit[text]:** To limit της μονάδας που προκάλεσε το event.
- **max[text]:** To max της μονάδας που προκάλεσε το event.
- **min[text]:** To min της μονάδας που προκάλεσε το event.
- **usersSawIt[user/object]:** Λίστα με τους χρήστες που έχουν ειδοποιηθεί για το συγκεκριμένο event.

```

> db.events.find().pretty();
{
    "_id" : ObjectId("5a9dc93301e53c4a8323178d"),
    "time" : ISODate("2018-03-05T22:48:19.336Z"),
    "usersSawIt" : [
        ObjectId("5a8a2d3a365cf85d0dbe90e3")
    ],
    "deviceID" : ObjectId("5a8a2e03365cf85d0dbe90e4"),
    "conName" : "Con0",
    "unitName" : "Unit0",
    "value" : "162.0",
    "tag" : "Event",
    "type" : "1",
    "mode" : "2",
    "limit" : "128SW",
    "max" : "255",
    "min" : "0",
    "v" : 1
}

```

Σχήμα 5.4.3: Ενδεικτικά μια εγγραφή ενός event της συσκευής “perama” στην βάση δεδομένων.

Στην εγγραφή του σχήματος 5.4.3 διακρίνεται μεταξύ άλλων ότι το συγκεκριμένο event αφορά το έξυπνο σπίτι “perama” και ότι έχει ειδοποιηθεί ο χρήστης “maria” (με βάση και τα σχήματα 5.4.1 και 5.4.2), όποτε δεν θα ξαναεμφανιστεί κάποια ειδοποίηση στον ίδιο χρήστη. Αν υπάρχει κάποιος άλλος χρήστης με δικαιώματα στο σπίτι “perama”, μόλις θα ειδοποιηθεί για το event θα προστεθεί το ObjectId του χρήστη στην λίστα usersSawIt.

6

Υλοποίηση

Σε αυτή την ενότητα ακολουθούν κάποια ενδεικτικά κομμάτια της υλοποίησης του συστήματος και αναφέρονται οι πλατφόρμες και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν.

6.1 Ενδεικτικά κάποιες λεπτομέρειες υλοποίησης (software)

6.1.1 Μήνυμα πρωτοκόλλου

Ένα παράδειγμα ενός μηνύματος είναι το ακόλουθο, όπου ουσιαστικά πρόκειται για ένα μήνυμα νέας τιμής μιας μονάδας (Από την εκτέλεση ενός Simulator):

```
{  
    "sID": "5a8a2e03365cf85d0dbe90e4",  
    "sT": "DSC",  
    "c": "NewValues",  
    "noP": 1,  
    "p": [  
        ["Con0", "Unit1", "184.0"]  
    ]  
}
```

- sID: ID αποστολέα.
- sT: Τύπος αποστολέα.
- c: Εντολή (command) αποστολέα.
- noP: Αριθμός παραμέτρων.
- p: Παράμετροι (Λίστες μέσα σε λίστα).



Άλλα commands των μηνυμάτων είναι: Error, Quit, Failed, Login, LoginReply, Certification, GetDevicesInfo, PutDevicesInfo, BringMeDevice, InitControllers, NewController, AlterController, DeleteController, NewValues, InitializationFinished, ChangeValues, ChangeModes, Answer, GetTypeUser, PutTypeUser, TSNC, Event.

6.1.2 Αλλαγή τιμής μιας μονάδας από απομακρυσμένο χρήστη

```

public void run() {
    ProtocolMessage proMess = new ProtocolMessage() ;
    try {
        while (running) {
            proMess.reload(manSocket.getIn().readLine()); //Διαβάζεται από το socket ένα μήνυμα
            switch (proMess.getCommand()) {
                case Error:
                    CT.Debug.print("Stopping Listen thread - ERROR (by read)! ");
                    Globals.mainDeviceClient.correctClose();
                    running = false;
                    break;
                case Quit:
                    CT.Debug.print("Quit From Server!");
                    Globals.mainDeviceClient.correctClose();
                    running = false;
                    break;
                case ChangeValues: //Εάν το "command" των μηνύματος είναι "ChangeValues"
                    CT.Debug.print("Read from server : " + proMess.getProtocolMessage(), MSGFROMSERVER);
                    //Ενημερώνονται για την αλλαγή οι μονάδες που πρόκειται να αλλάξουν τιμή:
                    ListenThread.changeValues(proMess);
                    //Ενημερώνονται για την αλλαγή όλοι οι τοπικοί επόπτες:
                    for (ManageServerSocket item : ClientServiceThread.listWithConnectedUserClientInDevice) {
                        item.getOut().println(proMess.getProtocolMessage());
                        item.getOut().flush();
                    }
                    break;
                case ChangeModes:

```



Φωτογραφία 6.1.2.1: Κομμάτι κώδικα από την συνάρτηση run() της ListenThread.

Στην φωτογραφία 6.1.2.1 φαίνεται τι θα γίνει αν ο Device Client διαβάσει από τον Server ένα μήνυμα για αλλαγή τιμής μιας μονάδας (Δηλαδή, ένα μήνυμα σαν αυτό του παραδείγματος 6.1.1 με τη διαφορά ότι σαν command θα έχει “ChangeValues”). Το μήνυμα αυτό ουσιαστικά το έχει δημιουργήσει κάποιος απομακρυσμένος επόπτης (User Client).

Στην Φωτογραφία 6.1.2.1 φαίνεται ότι θα εκτελεστεί η static μέθοδος changeValues() της ListenThread της οποίας ο κώδικας φαίνεται στην φωτογραφία 6.1.2.2.

```

public static synchronized void changeValues(ProtocolMessage proMess) throws IOException {
    //Για όσες είναι οι μονάδες που θα αλλάξουν, φτιάχνονται και στέλνονται
    //τόσα μηνύματα στα arduino για να αλλάξουν τις τιμές τους στην πραγματικότητα.
    for (ArrayList<String> alParam : proMess.getParameters()) {
        String conName = alParam.get(0);
        String unitName = alParam.get(1);
        String value = alParam.get(2);

        //Δημιουργείται το μήνυμα αλλαγής τιμής
        String post = "#c" + unitName + ":" + value + "#";

        //Στέλνεται σειριακά το μήνυμα στο κατάλληλο arduino, ώστε να αλλάξει η τιμή.
        Globals.hmAllReadArduino.get(conName).output.write(post.getBytes());
        Globals.hmAllReadArduino.get(conName).output.flush();
    }
    //Ενημερώνεται κατάλληλα και το object device (Device)
    Globals.device.setValuesOfDevice(proMess);
}

```

Φωτογραφία 6.1.2.2: Η συνάρτηση changeValues() της ListenThread.

Στην Φωτογραφία 6.1.2.2 φαίνεται πως θα ενημερωθούν τα arduino για να πραγματοποιηθούν στην πραγματικότητα οι αλλαγές στις μονάδες. Επίσης, φαίνεται

πως ενημερώνονται οι μονάδες του αντικειμένου device (Device) μέσα από το μήνυμα πρωτοκόλλου που αρχικά είχε διαβαστεί από τον απομακρυσμένο Server.

Στην φωτογραφία 6.1.2.3 φαίνεται ο κώδικας που εκτελείται στην setValuesOfDevice() της κλάσης Device, και πως ενημερώνεται το εκάστοτε unit.

```

    public boolean setValuesOfDevice(ProtocolMessage pm)
    {
        boolean allWentWell = true ;
        String conName,unitName,unitValue ;
        try {
            for (int i = 0 ; i<pm.getParameters().size();i++)
            {
                conName = pm.getParameters().get(i).get(0) ;
                unitName = pm.getParameters().get(i).get(1) ;
                unitValue = pm.getParameters().get(i).get(2) ;

                allUnits.get(conName).get(unitName).setValue(unitValue);
            }
        } catch(Exception ex){
            allWentWell = false ;
        }
        return allWentWell ;
    }
    /**

```

Φωτογραφία 6.1.2.3: Η συνάρτηση setValuesOfDevice() της Device.

```

public class Device {
    protected final String deviceName ;
    protected final String devicePassword ;
    protected String deviceID ;
    protected SenderType systemType ;
    protected HashMap<String,HashMap<String,Unit>> allUnits ;
    protected ProtocolMessage pm ;
    int sumOfControllers ;
}

```

Φωτογραφία 6.1.2.4: Η δήλωση της allUnits (HashMap δομής) μέσα στην κλάση Device.

6.1.3 Διαδικασία ειδοποίησης των User Clients όταν συμβεί κάποιο event

```

public void run() { //Σε ξεχωριστό thread
    ApiManager.getInstance().login(username, password); } Login
    timer = new Timer();
    TimerTask task = () -> {
        try {
            ApiManager.getInstance().getEvents(); } Get Events
            if (ApiManager.getInstance().getAllEvents() != null && !ApiManager.getInstance().getAllEvents().isEmpty()) {
                String message = "";
                for (Event e : ApiManager.getInstance().getAllEvents())
                    message += e.toString() + '\n'; } Χτίσε το μήνυμα
                playClip(); Εξεκίνησε την μουσική
                JOptionPane.showMessageDialog(frame, message, title: "Ειδοποίηση", JOptionPane.WARNING_MESSAGE);
                stopClip(); Σταμάτησε την μουσική
                for (Event e : ApiManager.getInstance().getAllEvents())
                    ApiManager.getInstance().putSawEvent(e); } Ενημέρωσε ότι ο χρήστης
                είδε τις ειδοποιήσεις
                ApiManager.getInstance().getAllEvents().clear();
                Thread.sleep( millis: 2000 );
            }
        } catch (InterruptedException ex) {
            System.out.println("Exception!");
        }
    };
    timer.schedule(task, delay: 4000, period: 10000); } Κάνε την διαδικασία κάθε 10 δευτερόλεπτα
}

```

Φωτογραφία 6.1.3.1: Κώδικας της διαδικασίας ειδοποίησης του Desktop User Client όταν συμβεί event.

Στην φωτογραφία 6.1.3.1 φαίνεται πως αρχικά γίνεται login για να παρθεί να απαραίτητο token πρόσβασης από το Web Service. Στην συνέχεια δημιουργείται ένα task που με την βοήθεια ενός timer εκτελείται κάθε 10 δευτερόλεπτα. Στο task αυτό διαβάζονται από το Web Service τα events που πιθανόν έχουν συμβεί στα έξυπνα σπίτια του χρήστη που έκανε login. Έπειτα, για όσα είναι τα events (εφόσον υπάρχουν) δημιουργείται ένα μήνυμα το οποίο εμφανίζεται στο γραφικό περιβάλλον του χρήστη, ενώ έχει ξεκινήσει να ακούγεται και ένας ήχος ειδοποίησης. Μόλις ο χρήστης δει το μήνυμα με τα events και πατήσει το πλήκτρο “OK”, θα σταματήσει ο ήχος ειδοποίησης και θα ενημερωθεί το Web Service ότι ο χρήστης ενημερώθηκε για τα events.

```
public interface AppService {
    @POST("/users/authenticate")
    Call<Answer> getUserToken(@Body PostLogin pl);

    @GET("/events/notSeen")
    Call<EventA> getEvents(@Header("authorization") String token);

    @PUT("/events/event/{id}")
    Call<Answer> getPutSawEvent(@Header("authorization") String token, @Path("id") String eventID);
}
```

Φωτογραφία 6.1.3.2: Το περιεχόμενο του AppService του package alarm
(Χρήση από User Clients).

Στην φωτογραφία 6.1.3.2 φαίνονται οι τρεις μέθοδοι που χρησιμοποιούνται για να επικοινωνήσουν οι User Clients (Android & Desktop) με το Web Service. Συγκεκριμένα, με την μέθοδο getToken() γίνεται **http post** ώστε να παρθεί ένα token πιστοποίησης για τον συγκεκριμένο χρήστη που χρησιμοποιεί τον User client. Με την μέθοδο getEvents() γίνεται **http get** ώστε να παρθούν τα events που έχουν συμβεί ώστε να ειδοποιηθεί ο χρήστης. Και τέλος, με την μέθοδο getPutSawEvent() γίνεται **http put** ώστε να ενημερωθεί το Web Service για το ποια events έχει ειδοποιηθεί ο χρήστης. Αντίστοιχες λειτουργίες γίνονται και από τον Device Client & Simulator, όταν προσθέτουν events που συμβαίνουν ή όταν λαμβάνουν Reminders από το Web Service.

```
public boolean login(String username, String password) {
    boolean returnValue = true;
    PostLogin pl = new PostLogin(username, password);Αρχικοποίησε το μοντέλο PostLogin
    try {
        appService.getUserToken(pl).enqueue(new Callback<Answer>() {
            @Override
            public void onResponse(Call<Answer> call, Response<Answer> response) {
                if (response.code() == 200) {
                    Answer a = response.body();
                    a.printValues();
                    if (a.isSuccess()) { //Αν όλα πήγαν καλά
                        getInstance().setLoggedIn(true);
                        getInstance().setToken(a.getToken());
                        getInstance().getStatus().ok(); } Καταχώρησε το token που έλαβες από το Web Service στην μεταβλητή token
                    } else {
                        getInstance().setLoggedIn(false);
                        getInstance().getStatus().error();
                    }
                }
            }
            @Override
            public void onFailure(Call<Answer> call, Throwable t) {
                System.out.println("Error while posting to the api" + t.getMessage());
            }
        });
    } catch (Exception ex) {
        returnValue = false;
    }
    return returnValue;
}
```

Φωτογραφία 6.1.3.3: Η μέθοδος login() της singleton κλάσης ApiManager.

Στην φωτογραφία 6.1.3.3 φαίνεται πως αρχικοποιείται ένα αντικείμενο της κλάσης PostLogin και στην συνέχεια πως αυτό στέλνεται ως body της συνάρτησης getToken() στο Web Service. Η εκτέλεση γίνεται ασύγχρονα και όταν απαντήσει το Web Service, ανάλογα με το HTTP code που έχει η απάντηση, εκτελείται η ανάλογη μέθοδος. Στην περίπτωση που όλα πάνε καλά και η απάντηση έχει code 200, θα παραλάβουμε ένα JSON μήνυμα που θα έχει μέσα και ένα token. Το token είναι απαραίτητο για την συνέχεια της επικοινωνίας των clients του συστήματος με το Web Service, διότι όλα τα requests προς το API (εκτός από το authenticate) χρειάζονται πιστοποίηση.

```
public class PostLogin {
    @SerializedName("username")
    @Expose
    private String username;
    @SerializedName("password")
    @Expose
    private String password;

    public PostLogin(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }
}
```

Φωτογραφία 6.1.3.4: Η Κλάση PostLogin που στέλνεται ως body μέσω της getToken() στο API.

```
public void getEvents() {
    if (isLoggedIn()) {
        try {
            appService.getEvents(token).enqueue(new Callback<EventA>() {
                @Override
                public void onResponse(Call<EventA> call, Response<EventA> response) {
                    System.out.println("Response code:" + response.code());
                    if (response.code() == 200) {
                        EventA ea = response.body();
                        //ea.printValues();
                        if (ea.isSuccess()) {
                            getInstance().getStatus().ok();
                            getInstance().setAllEvents(ea.getEvents());
                        } else {
                            getInstance().getStatus().error();
                            getInstance().setLoggedIn(false);
                        }
                    }
                }
                @Override
                public void onFailure(Call<EventA> call, Throwable t) {
                    System.out.println("Error while posting to the api" + t.getMessage());
                    getInstance().getStatus().error();
                }
            });
        } catch (Exception ex) {
    }
}
```

} Τα events που
έστειλε το
Web Service

Φωτογραφία 6.1.3.5: Η μέθοδος getEvents() της singleton κλάσης ApiManager.

Στην φωτογραφία 6.1.3.5 φαίνεται πως χρησιμοποιείται το token (που έχει παρθεί από προηγούμενη επικοινωνία με την μέθοδο login()) ώστε πλέον ως πιστοποιημένος χρήστης, να του απαντήσει το Web Service με τα events που τον αφορούν.

6.1.4 Κάποιες λειτουργίες των Web Service (Back-end)

```
router.get('/notSeen', passport.authenticate('jwt', {session: false}), function (req, res) {
  Device.getDevicesByIds(req.user.devices, function (err, devices) {Φέρει όλα τα devices του χρήστη
    if (err) return res.json({success: false, msg: 'Get Events Error'}); που είναι owner.
    DeviceAccess.getDevicesByUserIdAllRights(req.user._id, function (err, devicesWithRights) {
      if (err) return res.json({success: false, msg: 'Get Events Error'}); ↑
      console.log(devices);
      devicesWithRights.forEach(function (deviceAccess) { } ← Φέρει όλα τα devices που
        devices.push(deviceAccess['deviceID']); } ← έχει δικαιώματα ο χρήστης
      });
      console.log(devices);
      Event.getEventsByDevicesNoSeenUser(devices, req.user._id, function (err, events) { ← Συναθροίζονται όλα τα devices μαζί
        if (err) return res.json({success: false, msg: 'Get Events Error'}); ↑
        return res.json({success: true, events: events});
      });
    });
  });
});
```

Φωτογραφία 6.1.4.1: Κώδικας που απαντάει ένα σύνολο από events που αφορούν κάποιον χρήστη.

Στην φωτογραφία 6.1.4.1 φαίνεται ο τρόπος που απαντάει το Web Service, όταν κάποιο πρόγραμμα (πχ User Client) εκτελέσει την μέθοδο http get στο path: **(domain/ip):port+“/events/notSeen”**. Διακρίνεται μεταξύ άλλων στην φωτογραφία, ότι για να εκτελεστεί ο κώδικας πρέπει να έχει γίνει πιστοποίηση του χρήστη με βάση το token (πρώτη γραμμή: “authenticate”) που πρέπει να έχει στο header το http get request. Με πράσινο φαίνονται οι εκτελέσεις μεθόδων των οντοτήτων της βάσης.

```
module.exports.getEventsByDevicesNoSeenUser = function (deviceIDs, userID, callback) {
  var query = {
    "$and": [
      {'deviceID': {"$in": deviceIDs}},
      {'usersSawIt': {"$ne": userID}}
    ]
  };
  Event.find(query)
    .populate('deviceID')
    .select('_id conName unitName value time type deviceName')
    .exec(function (err, post) {
      callback(err, post);
    });
}
```

Φωτογραφία 6.1.4.2: Η μέθοδος `getEventsByDevicesNoSeenUser()` του μοντέλου `Event` της βάσης.

Ενδεικτικά φαίνεται στην φωτογραφία 6.1.4.2 πως το Web Service κάνει ένα query στην βάση δεδομένων. Συγκεκριμένα, το “ερώτημα” της φωτογραφίας ζητάει από το collection events (Κλάση/Mοντέλο Event) να του φέρει τα events που το deviceID τους να ανήκει στην λίστα deviceIDs (πρώτη παράμετρος) και που η λίστα usersSawIt των events να μην περιέχει το userID (δεύτερη παράμετρος) του χρήστη (Δηλαδή, να μην έχει ενημερωθεί στο παρελθόν ο χρήστης...). Στην φωτογραφία 6.1.4.1 φαίνεται η εκτέλεση της μεθόδου getEventsByDevicesNoSeenUser() με τους κατάλληλους παραμέτρους.

```
// Delete Reminder
router.delete('/reminder/:id', passport.authenticate('jwt', {session: false}), function (req, res) {
  var reminderID = req.params['id'];

  if (!reminderID) {
    return res.json({success: false, msg: 'Delete Reminder error (Not found Reminder ID)'});
  } else {
    Reminder.deleteReminderById(reminderID, function (err, done) {
      if (err) return res.json({success: false, msg: 'Error during Delete Reminder!'});

      if (done) {
        return res.json({success: true, msg: 'Reminder deleted!'});
      } else {
        return res.json({success: false, msg: 'Reminder not deleted (Failed)'});
      }
    })
  }
});
```

Φωτογραφία 6.1.4.3: Κώδικας που διαγράφει έναν Reminder.

Στην φωτογραφία 6.1.4.3 φαίνεται ο τρόπος που απαντάει το Web Service, όταν κάποιο πρόγραμμα (πχ Angular) εκτελέσει την μέθοδο http delete στο path: **(domain/ip):port**+“/devices/reminder”+id.

```
router.post('/changeAccess', passport.authenticate('jwt', {session: false}), function (req, res) {
  var access = req.body.access;
  var deviceAccessID = req.body.deviceAccessID;

  if (!deviceAccessID || (access != 'A' && access != 'F' && access != 'R')) {
    return res.json({success: false, msg: 'ChangeAccess error'});
  } else {
    DeviceAccess.updateDeviceAccess(deviceAccessID, access, function (err, done) {
      if (err) throw err;
      if (done){
        return res.json({success: true, msg: 'Changes saved'});
      } else {
        return res.json({success: false, msg: 'Change Access failed!'});
      }
    })
  }
});
```

Φωτογραφία 6.1.4.4: Κώδικας που αλλάζει τα δικαιώματα προσπέλασης.

Στην φωτογραφία 6.1.4.4 φαίνεται ο τρόπος που απαντάει το Web Service, όταν κάποιο πρόγραμμα (πχ Angular) εκτελέσει την μέθοδο http post στο path: **(domain/ip):port**+“/devices/changeAccess”.

```
// Request Schema
var RequestSchema = mongoose.Schema({
  requestingUser: {
    type: mongoose.Schema.ObjectId, ref: "User"
  },
  deviceID: {
    type: mongoose.Schema.ObjectId, ref: "Device"
  },
  text: {
    type: String
  }
});

RequestSchema.index({requestingUser: 1, deviceID: 1}, {unique: true});
```

Φωτογραφία 6.1.4.5: Η κλάση/μοντέλο Request με στοιχεία όπως ακριβώς αποθηκεύονται στην mongoDB.

```

// Device Schema
var DeviceSchema = mongoose.Schema({
  deviceName: {
    type: String,
    index: true,
    unique: true
  },
  password: {
    type: String
  },
  comment: {
    type: String
  },
  owner: {type: Schema.ObjectId, ref: "User", childPath: "devices"}, 
  isDeleted: {
    type: Boolean
  },
  lastModified: {
    type: Date,
    default: Date.now
  },
  userModified: {type: mongoose.Schema.ObjectId, ref: "User"}, 
  units: [{type: mongoose.Schema.ObjectId, ref: "Unit"}]
});

```

Φωτογραφία 6.1.4.6: Η κλάση/μοντέλο Device με στοιχεία όπως ακριβώς αποθηκεύονται στην mongoDB.

Στις φωτογραφίες 6.1.4.5 και 6.1.4.6 φαίνεται ο τρόπος που μοντελοποιούνται οι οντότητες (collections) της βάσης δεδομένων (mongoDB) με την nodeJS (χρήση mongoose). Συγκεκριμένα, εφόσον φτιαχτούν τα σχήματα (κλάσεις/μοντέλα) με τα συγκεκριμένα στοιχεία τους (μεταβλητές μιας κλάσης), η nodeJS αναλαμβάνει να φτιάξει αυτόματα για κάθε σχήμα ένα collection αλλά και τις συσχετίσεις μεταξύ τους στην mongoDB.

6.1.5 Κάποιες λειτουργίες του Angular project (Front-end)

```

this.reminderService.getReminders(deviceId) Θέρε όλους τους Reminders, για ένα συγκεκριμένο device, από το Web Service
  .subscribe( next: data => {
    if (data['success']) { Εάν η απάντηση από το Web Service είναι success (=true)
      this.reminders = data['reminders'];   Κάνε "bind" τους Reminders που ήρθαν από το Web Service με την
                                              μεταβλητή reminders (Θα περιέχει μια συλλογή από Reminders).
    } else {
      this.flashMessage.show(data['msg'], {cssClass: 'alert-danger', timeout: 3000});
      this.router.navigate(commands: ['/devices']);
    }
  },
  error: err => {
    this.flashMessage.show('Error during get Reminders by Server', {cssClass: 'alert-danger', timeout: 3000});
    this.router.navigate(commands: ['/devices']);
    return false;
  });

```

Φωτ. 6.1.5.1: Javascript/TypeScript κώδικας, που φέρνει όλους τους Reminders & τους κάνει bind στην μετ. Reminders.

conName	unitName	Datetime	New Value	Read	Delete
Reminder	Reminder	2023-09-15T10:00:00Z	100	False	<a (click)="deleteReminder(reminder)" class="btn btn-danger fa fa-trash" href="#">Delete
Reminder	Reminder	2023-09-15T10:00:00Z	100	False	<a (click)="deleteReminder(reminder)" class="btn btn-danger fa fa-trash" href="#">Delete
Reminder	Reminder	2023-09-15T10:00:00Z	100	False	<a (click)="deleteReminder(reminder)" class="btn btn-danger fa fa-trash" href="#">Delete
Reminder	Reminder	2023-09-15T10:00:00Z	100	False	<a (click)="deleteReminder(reminder)" class="btn btn-danger fa fa-trash" href="#">Delete

Φωτογραφία 6.1.5.2: HTML κώδικας, που εμφανίζει όλους τους Reminders ενός Device με μορφή πίνακα.

```
getReminders(deviceID) {
  var authToken = this.commonService.loadToken(); Φόρτωσε το token του χρήστη
  const headers = new HttpHeaders().set('Authorization', authToken); Βάλε στους Headers το token
  console.log(globals.ROOT_URL + globals.PATH_REMINDER + "/" + deviceID);
  return this.httpClient.get(url: globals.ROOT_URL + globals.PATH_REMINDERS + "/" + deviceID, options: {headers})
}

Εκτέλεσε ένα http get request στο Web Service για να παρθούν οι Reminders ενός device
```

Φωτογραφία 6.1.5.3: Javascript/Typescript κώδικας που φέρνει όλους τους Reminders ενός Device από το Web Service.

```
deleteReminder(reminder) { Διέγραψε τον Reminder από το σύστημα
  this.reminderService.deleteReminder(reminder); ←
  var removedReminder = this.reminders.indexOf(reminder);
  this.reminders.splice(removedReminder, deleteCount: 1)
} Διέγραψε τον Reminder από την html σελίδα που βλέπει ο χρήστης ←
```

Φωτογραφία 6.1.5.4: Javascript/Typescript κώδικας που διαγράφει έναν Reminder

```
deleteReminder(reminder) {
  var authToken = this.commonService.loadToken(); Φόρτωσε το token του χρήστη
  const headers = new HttpHeaders().set('Authorization', authToken); Βάλε στους Headers το token
  this.httpClient.delete(url: globals.ROOT_URL + globals.PATH_DELREMINDER + '/' + reminder['_id'], options: {headers})
    .subscribe();
}

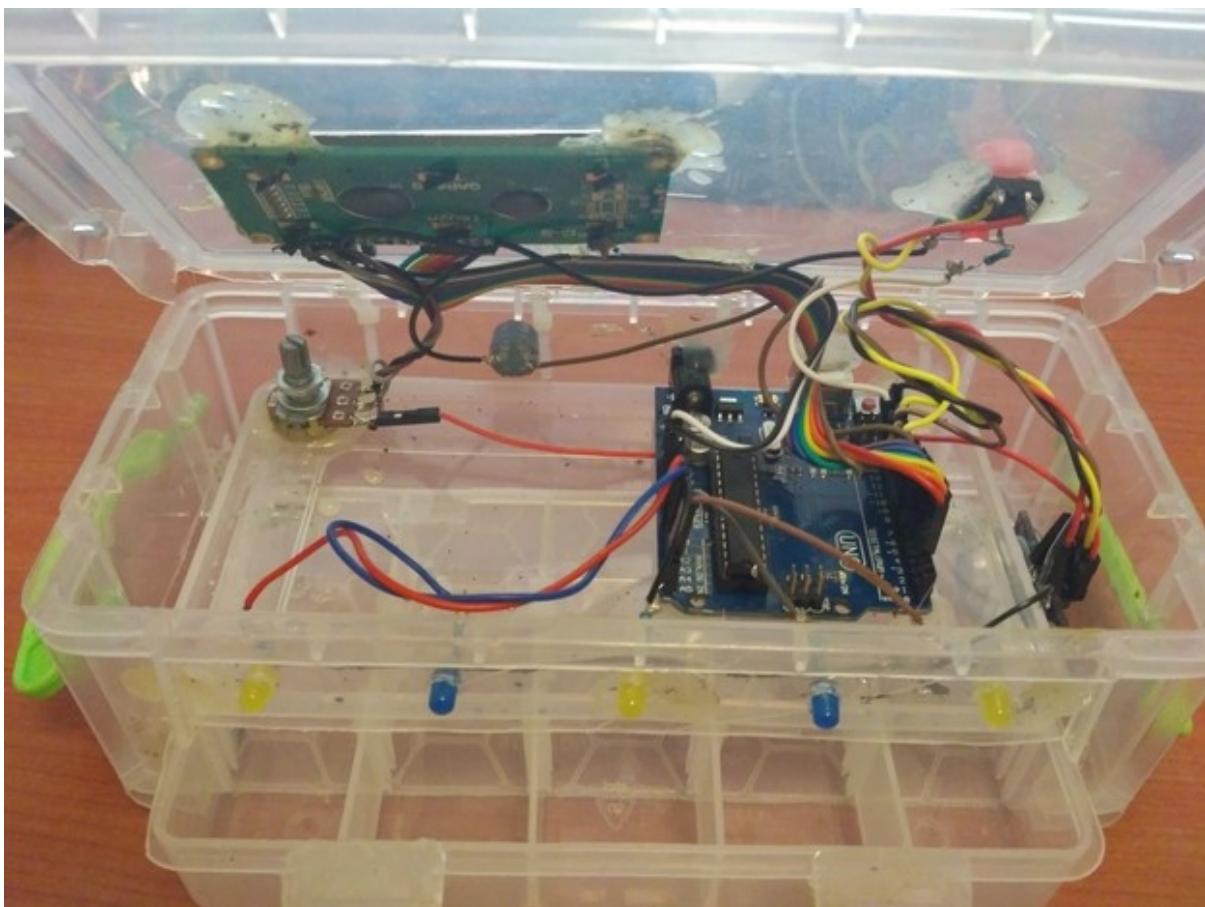
Εκτέλεσε ένα http delete request στο Web Service για να διαγραφεί ο Reminder από το σύστημα
```

Φωτ. 6.1.5.5: Javascript/Typescript κώδικας που διαγράφει έναν Reminder από το σύστημα (μέσω του Web Service).

Στις φωτογραφίες 6.1.5.1, 6.1.5.2, 6.1.5.3, 6.1.5.4 & 6.1.5.5 φαίνεται με ενδεικτικά screenshots ο τρόπος που επικοινωνεί το Angular Project (Front-end) με το Web Service (Back-end).

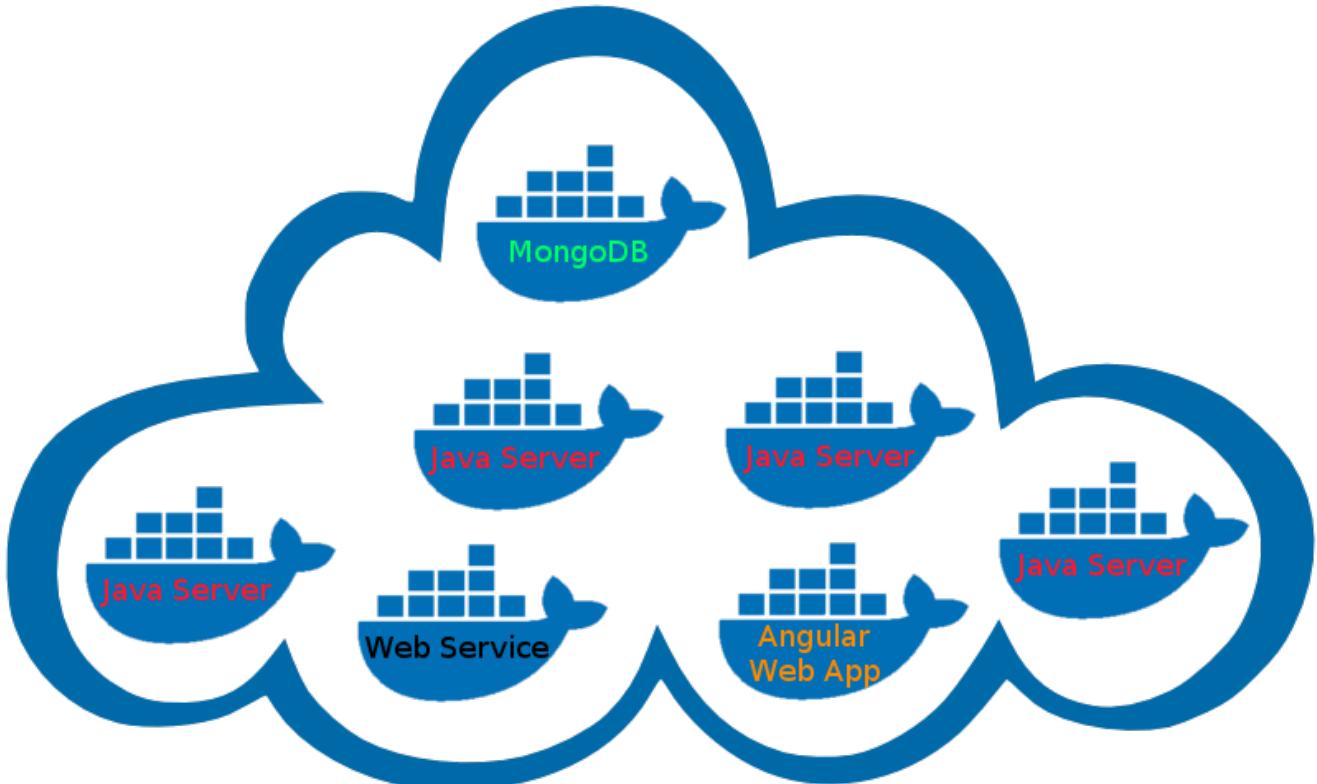
6.2 Υλοποίηση κατασκευών (hardware)

Στα πλαίσια της εργασίας φτιάχτηκαν ενδεικτικά 2 κατασκευές και προσαρμόστηκαν 3 arduino sketch (με βάση το πρότυπο arduino sketch) ώστε να λειτουργήσουν 3 διαφορετικά arduino uno. Πιο συγκεκριμένα, κατασκευάστηκε μια συσκευή υπενθύμισης χαπιών όπου χρησιμοποιήθηκαν εκτός από ένα arduino και ένα sonar, ένα buzzer, 6 led λαμπάκια, ένα κουμπί, ένα ποτενσιόμετρο και μια LCD οθόνη. Για την δεύτερη κατασκευή, που ήταν μια μικρογραφία ενός έξυπνου σπιτιού, χρησιμοποιήθηκαν εκτός από ένα arduino και 2 κουμπιά, 5 led λαμπάκια, ένα RGB led, ένας φωτοαισθητήρας, ένα θερμόμετρο & υγρασιόμετρο και ένας αισθητήρας αιγίχνευσης φωτιάς. Επίσης, για τις παραπάνω κατασκευές χρειάστηκε ένα κολλητήρι, καλάι, ένα κοπίδι και κόλλα θερμής έγχυσης. Το 3^o arduino της εργασίας είχε συνδεδεμένα πάνω του (με την βοήθεια ενός breadboard) έναν αισθητήρα κίνησης, έναν αισθητήρα ήχου, 3 led, ένα θερμόμετρο, ένα κουμπί και ένα bluetooth adapter., ώστε να επικοινωνεί με κάποιον Device client ασύρματα. Για τροφοδοσία, το 3^o arduino παίρνει ρεύμα από 6 μπαταρίες των 1.5V (Συνολικά 9V). Σε όλες τις παραπάνω συνδέσεις, χρησιμοποιήθηκαν μικρά καλώδια και αντιστάσεις. Να αναφερθεί, ότι επιλέχθηκε τυχαία η 2^η κατασκευή (μικρογραφία ενός έξυπνου σπιτιού) να είναι συνδεδεμένη και με ένα ρελέ, ώστε να χειρίζεται μια λάμπα υψηλής τάσης. Ο χειρισμός ενός ρελέ είναι ίδιος με ενός led. Δηλαδή, όπως με το HIGH ανάβει το led λαμπάκι και με το LOW κλείνει, αντίστοιχα ανοίγει και κλείνει μια συσκευή υψηλότερης τάσης όπως ένα πορτατίφ (κλείνει και ανοίγει το κύκλωμα).



Φωτογραφία. 6.2.1: Μια ενδεικτική φωτογραφία από το εσωτερικό της κατασκευής υπενθύμισης χαπιών.

6.3 Deploy του συστήματος στο Cloud



Σχήμα. 6.3.1: Μια ενδεικτική απεικόνιση των επιμέρους εφαρμογών στο Cloud μέσα σε docker containers.

Στο σχήμα 6.3.1 φαίνεται με έναν αφηρημένο τρόπο, πως μπορούν τα διαφορετικά κομμάτια του πυρήνα του συστήματος (Angular Web App, Web Service, MongoDB και Java Servers) να εκτελούνται σε τελείως διαφορετικά περιβάλλοντα (ακόμη και σε διαφορετικά μηχανήματα) μέσα στο Cloud. Οι Java Servers είναι ενδεικτικά 4 στο σχήμα. Το πλήθος τους θα μπορούσε να εξαρτάτε από την εκάστοτε ζήτηση, δηλαδή τον αριθμό των online έξυπνων χώρων (Devices) και εποπτών.

6.4 Πλατφόρμες και προγραμματιστικά εργαλεία

- ✓ [IntelliJ IDEA](#) (Java IDE) για όλες τις Java εφαρμογές (εκτός από την Android)
- ✓ [WebStorm](#) (Javascript IDE) για Web Service και Angular.
- ✓ [Android Studio](#) (Android IDE) για την Android εφαρμογή.
- ✓ [Arduino IDE](#) για τους κώδικες των arduino.
- ✓ [Git](#) (Repository στο Gitlab.com) για versioning όλων των εφαρμογών.
- ✓ LibreOffice για τα σχεδιαγράμματα και την συγγραφή.
- ✓ DBSchema (Database Diagram Designer & Admin tool) για σχεδιάγραμμα της database.

7

Ελεγχος

Σε αυτή την ενότητα ακολουθεί ένα πραγματικό σενάριο χρήσης, μέσα από το οποίο θα φανούν έμπρακτα οι υλοποιημένες λειτουργίες του συστήματος.

The screenshot shows a web application interface. At the top, there is a dark header bar with various icons (calculator, gear, user, etc.) and two buttons: 'Login' and 'Register'. Below the header is a sidebar on the left with icons for home, info, and a graduation cap. The main content area has a title 'Register' and a sub-section 'Register an Account'. The form fields are as follows:

Username	newUser
Name	Michael
Surname	Galliakis
Email address	michaelgalliakis@yahoo.gr
Password	*****
Confirm password	*****

At the bottom of the form is a large blue 'Register' button. A small footer at the bottom of the page reads: ΑΙΓΑΚ * IT * (Student Michael Galliakis) * Thesis 2010 * (Supervisor Christos Skarlatos).

Screenshot 7.1 Δημιουργία λογαριασμού χρήστη (newUser)

Login

Username
newUser

Password
.....

Login

Screenshot 7.2 Login του χρήστη newUser (που μόλις δημιουργήθηκε στο 7.1)

Devices

Devices

+Add a new Device

Screenshot 7.3 Εμφάνιση συσκευών (κενό) και κουμπί για την δημιουργία νέου λογαριασμού συσκευής

Devices / New Device account

Add a new Device account

Device name
pliroforiki

Password
.....

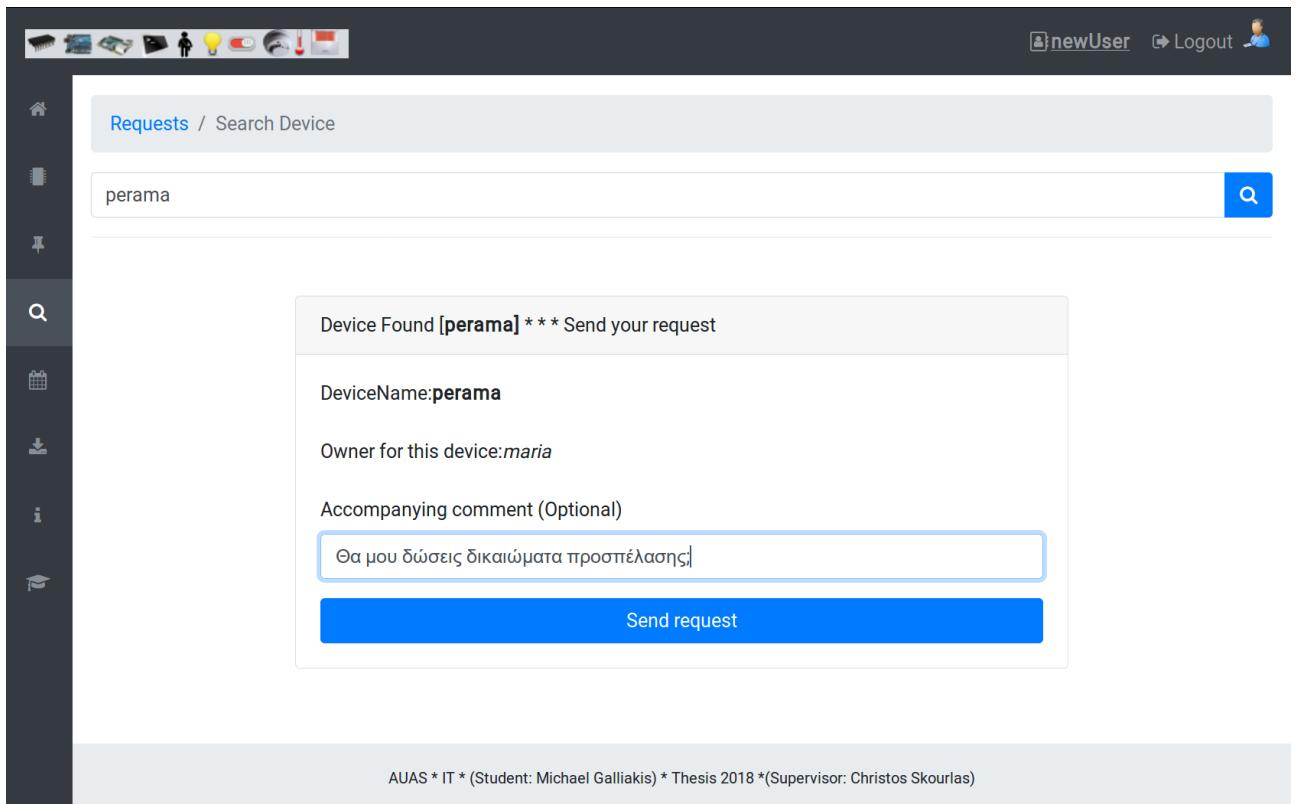
Confirm password
.....

Comment
Έξυπνος χώρος στο Πανεπιστήμιο Δυτικής Αττικής

Save new device

AUAS * IT * (Student: Michael Galliakis) * Thesis 2018 *(Supervisor: Christos Skourlas)

Screenshot 7.4 Δημιουργία λογαριασμού συσκευής (pliroforiki)



Screenshot 7.5 Αναζήτηση της συσκευής perama και αίτηση για απόκτηση δικαιωμάτων προσπέλασης.

This screenshot shows a list of requests made by the user 'newUser'. The title of the section is 'Requests I have made'. It displays a table with columns for 'DeviceName' and 'Delete'. One row is visible, showing 'perama' in the 'DeviceName' column and a red 'Delete' button with a white 'X' icon in the 'Delete' column. The 'Delete' button is circled in green. The table also contains a row of text: 'Θα μου δώσεις δικαιώματα προσπέλασης;'. On the far left, there is a vertical sidebar with icons for search, calendar, download, and other functions.

Screenshot 7.6 Η αίτηση όπως την βλέπει ο “newUser,” με δυνατότητα να την διαγράψει

The screenshot shows a user interface for managing requests. At the top, there is a toolbar with various icons. On the left, a sidebar contains icons for Home, Requests, Requests I have made, Requests which concern me, Device Management, and Help.

The main area is titled "Requests" and contains two sections:

- Requests I have made**: An empty list.
- Requests which concern me**: A table showing a single request for device "perama" from user "newUser". The table has columns: DeviceName, User, Access and Accept, and Delete.

DeviceName	User	Access and Accept	Delete
perama	newUser	<input type="radio"/> Admin (🔒) <input type="radio"/> Full (🔓) <input checked="" type="radio"/> Read (➡)	✓ ✗

Below the table, a message reads: "Θα μου δώσεις δικαιώματα προσπέλασης;"

At the bottom right of the main area, there is a footer bar with the text "AUAS * IT * (Student: Michael Galliakis) * Thesis 2018 *(Supervisor: Christos Skourlas)".

Screenshot 7.7 Η αίτηση όπως την βλέπουν οι διαχειριστές και ο owner (maria) της συσκευής “perama”, με δυνατότητα να δώσουν δικαιώματα προσπέλασης (Admin, Full & Read) ή να την απορρίψουν.

This screenshot shows the "Access users for this device" section. It has a similar layout to the previous screenshot, with a sidebar on the left and a main content area.

The main content area displays a table for device "perama" with columns: DeviceName, User, Access, Change access and Save, and Release.

DeviceName	User	Access	Change access and Save	Release
perama	newUser	 <input checked="" type="radio"/> Admin (🔒) <input type="radio"/> Full (🔓) <input type="radio"/> Read (➡)	☒ ☒	☒

Screenshot 7.8 Εφόσον έχουν δοθεί δικαιώματα, οι admins και ο owner της συσκευής perama, έχουν την δυνατότητα να αλλάξουν ή να αναιρούν τα δικαιώματα προσπέλασης για τον χρήστη “newUser”.

Devices

Devices

pliroforiki

Έξυπνος χώρος στο Πανεπιστήμιο Δυτικής Αττικής

perama

perama

Διαγραφή συσκευής για τον χρήστη newUser

Διαχείριση υπενθυμίσεων (Reminders) για την συσκευή

AUAS * IT * (Student: Michael Galliakis) * Thesis 2018 *(Supervisor: Christos Skourlas)

Screenshot 7.9 Οι συσκευές του χρήστη “newUser”.

Devices / Change Device account

Update Device account

DeviceName: pliroforiki

Owner for this device: newUser

Password Confirm password

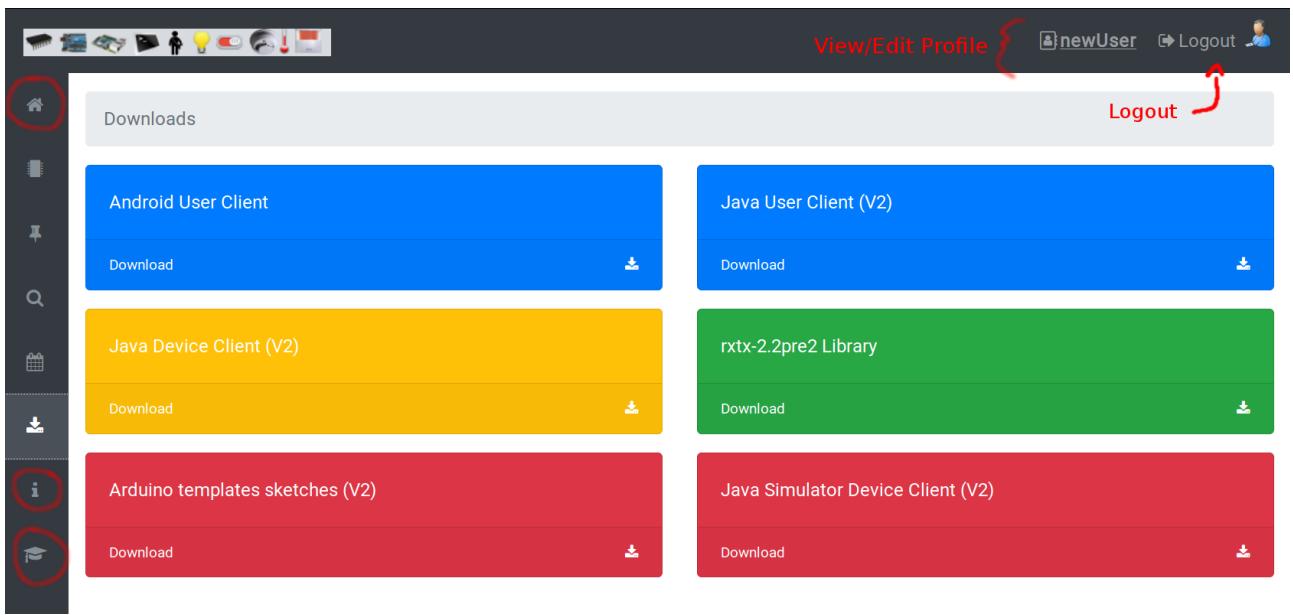
Comment

Έξυπνος χώρος στο Πανεπιστήμιο Δυτικής Αττικής

Save device

Access users for this device

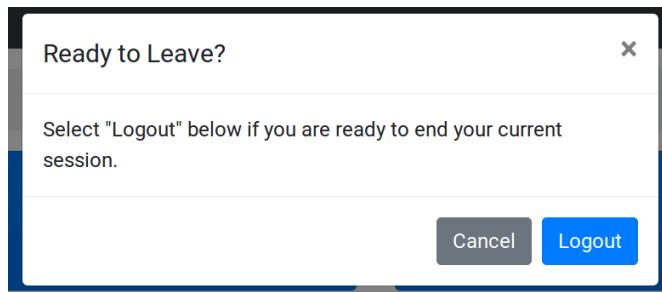
Screenshot 7.10 Edit της συσκευής “pliroforiki”.



Screenshot 7.11 Σελίδα όπου μπορεί κάποιος να κατεβάσει τις επιμέρους εφαρμογές του συστήματος.

Στο screenshot 7.11 διακρίνονται τα κουμπιά του menu, όπου μπορεί κάποιος να δει την αρχική σελίδα, το documentation του συστήματος και πληροφορίες για την εργασία. Ακόμη, φαίνεται ο τρόπος που μπορεί κάποιος χρήστης να δει και να επεξεργαστεί το profile του ή να κάνει logout.

Screenshot 7.12 Ενδεικτικές εικόνες για το View και Edit ενός λογαριασμού χρήστη.



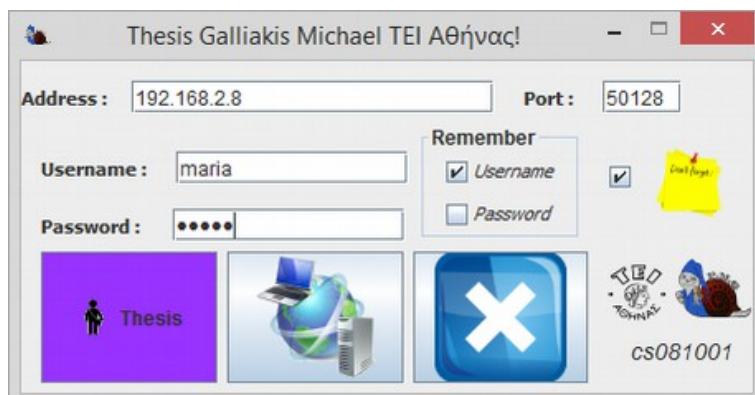
Screenshot 7.13 Popup που εμφανίζεται όταν πατηθεί το “logout”, για να δοθεί η δυνατότητα αποσύνδεσης.

Στα προηγούμενα screenshots (7.1-7.13) φαίνεται ο τρόπος που χρησιμοποιείται το Web App. Υποτίθεται ότι το Web Application (μαζί με το Web Service) και ένας ή περισσότεροι Java Servers “τρέχουν” στο Cloud. Επομένως, σε αυτό το πραγματικό σενάριο παρουσιάζονται μόνο οι ενέργειες που χρειάζεται να κάνει κάποιος χρήστης για να χρησιμοποιεί το σύστημα.

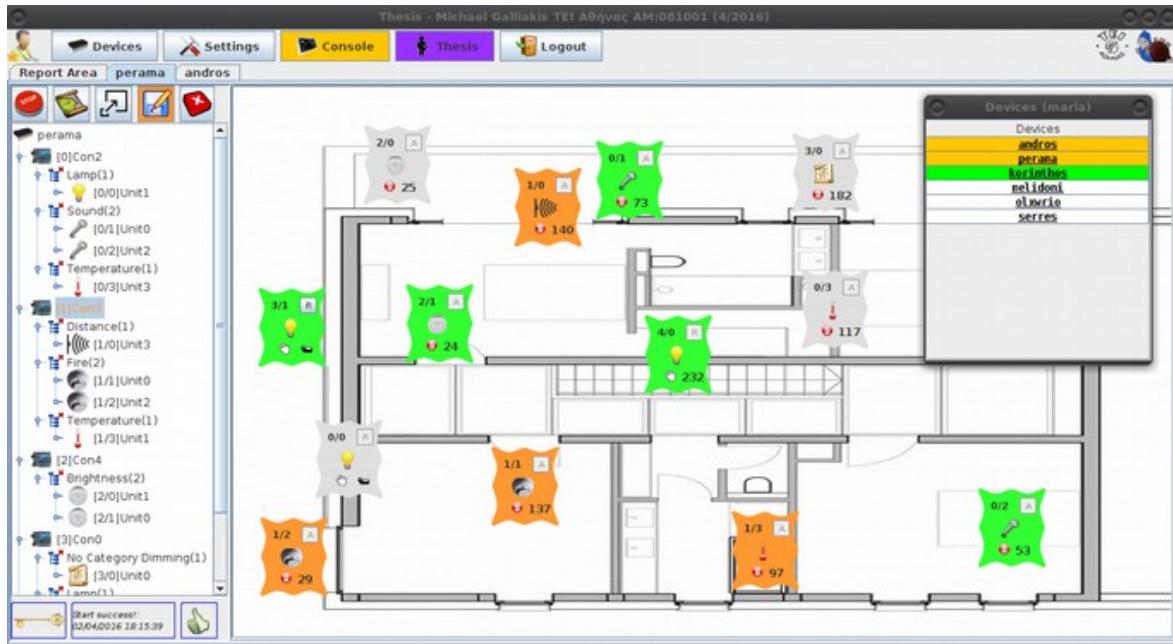
Σύμφωνα με τις μονάδες που έχει ο έξυπνος χώρος “pliroforiki” του newUser, προσαρμόζονται κατάλληλα κάποια arduino sketch με βάση το template sketch της διπλωματικής εργασίας (μπορεί να κατεβάσει κάποιος παραδείγματα του προτύπου από το Web Application, ss 7.11). Εφόσον γραφτούν τα arduino sketch και “καούν” στα arduino του έξυπνου χώρου “pliroforiki”, εκτελείται σε κάποιον υπολογιστή (πχ σε ένα raspberry) ο Device Client με σκοπό να συνδεθεί με τα arduino και με έναν απομακρυσμένο Java Server που βρίσκεται στο Cloud. Πριν την εκτέλεση του Device client, πρέπει να έχουν καθοριστεί το devicename, password, server ip/port κ.ά.

```
Input - Run (DeviceClientV2) X
RXTX Warning: Removing stale lock file. /var/lock/LCK..ttyACM0
RXTX Warning: Removing stale lock file. /var/lock/LCK..rfcomm0
/dev/rfcomm0
s{"ID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "46.39"}]}
s{"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "31.74"}]}
s{"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "46.39"}]}
s{"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "32.23"}]}
s{"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "46.39"}]}
s{"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "31.74"}]}
sStopped!
[ | A]: Version : PMG:V2
[ | A]: deviceNameConfirm : 0
[ | A]: info : {"sID": "0","sT": "DC","c": "NewController","noP": 6,"p": [{"Bluetooth"], ["thermometro", "9", "0", "", "31.74", "100.00", "-40.00", "NL"], ["Mikrofono", "6", "0", "", "31.00", "300.00"]}, {"sID": "0","sT": "DC","c": "NewController","noP": 6,"p": [{"Bluetooth"], ["thermometro", "9", "0", "", "31.74", "100.00", "-40.00", "NL"], ["Mikrofono", "6", "0", "", "31.00", "300.00", "0.00", "NL"]}], "/dev/ttyACM0
s{"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"ArdPill", "EmergencyButton", "2.00"}]}
sStopped!
[ | A]: Version : PMG:V2
[ | A]: deviceNameConfirm : 0
[ | A]: info : {"sID": "0","sT": "DC","c": "NewController","noP": 8,"p": [{"ArdPill"], ["firstPill", "15", "2", "", "0.00", "255.00", "0.00", "128SW"], ["secondPill", "12", "2", "", "0.00", "255.00", "0.00", "128SW"], {"sID": "0","sT": "DC","c": "NewController","noP": 8,"p": [{"ArdPill"], ["firstPill", "15", "2", "", "0.00", "255.00", "0.00", "128SW"], ["secondPill", "12", "2", "", "0.00", "255.00", "0.00", "128SW"]}], [ | A]: {"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "46.39"}]}]
[ | ] certification OK!
[ | ] DeviceName and Password Correct - Login OK!
[ | ] Prepare Local Device success!
[ | ] "Saae6625e32d5a54fa0fc141", "sT": "DC", "c": "InitControllers", "noP": 2, "p": [{"Bluetooth", "5"}, {"ArdPill", "7"}]}
[ | ] InitControllers OK!
[ | ] NewController OK!
[ | ] NewController OK!
[ | ] InitializationFinished OK!
[ | A]: {"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "32.23"}]} ]
*****
[ "sID": "Saae6625e32d5a54fa0fc141", "sT": "DC", "c": "NewController", "no": 6, "p": [{"Bluetooth"], ["Diakoptis", "11", "2", "", "0.00", "255.00", "0.00", "128SW"], ["M
[ "sID": "Saae6625e32d5a54fa0fc141", "sT": "DC", "c": "NewController", "noP": 8, "p": [{"ArdPill"], ["EmergencyButton", "1", "2", "Event", "2.00", "2.00", "0.00", "1SW"]}], [
*****#
[ | ] Thesis Michael Galliakis 2018.
[ | ] TEI Athens - IT department.
[ | ] Email : cmt16003@teiath.gr & michaelgalliakis@yahoo.gr .
[ | ] All files can be found :
[ | ] "https://github.com/michaelgalliakis/myThesisV2.git"
[ | ] #####
[ | ] Starting Server ...
[ | ] Opened successfully the server at the door 50127.
[ | ] Started Server.
[ | ] Listen ...
[ | A]: {"sID": "0","sT": "DC","c": "NewValues","noP": 1,"p": [{"Bluetooth", "thermometro", "46.39"}]} ]
```

Screenshot 7.14 Ενδεικτικά, η έξοδος από ένα Device Client ενώ εκτελείται.



Screenshot 7.15 To interface που μπορεί κάποιος να κάνει login στον Desktop User Client.



Screenshot 7.16 Μια περίπτωση εποπτείας από έναν Desktop User Client.



Screenshot 7.17 Αλλαγή τιμής μιας μονάδας με χρήση trackbar

Device	Value	Control
ArdPill/secondPill	0.00	Switch (B)
ArdPill/fourthPill	255.00	Switch (B)
ArdPill/firstPill	0.00	Switch (B)
ArdPill/thirdPill	0.00	Switch (B)
ArdPill/EmergencyButton	2.00	Switch (B)
ArdPill/Sonar	4.00	Switch (A)
Bluetooth/Kinisis	0.00	Switch (A)
Bluetooth/Diakoptis	255.00	Switch (B)
Bluetooth/Mikrofono	31.00	Switch (A)
Bluetooth/thermometro	56.15	Switch (A)
Bluetooth/ledYellow	157.00	Slider (R)

Screenshot 7.18 Μια περίπτωση login και εποπτείας από έναν Android User Client.

perama

Units of this device								
conName	unitName	Type	mode	tag	max	limit	min	Chose
ArdPill	EmergencyButton		Both	Event	2.00	1SW	0.00	<input type="radio"/>
ArdPill	Sonar		Auto		3000.00	NL	0.00	
ArdPill	ledYellow		Both		255.00	128SW	0.00	<input type="radio"/>
ArdPill	ledBlue		Both		255.00	128SW	0.00	<input type="radio"/>
ArdPill	ledRed		Both		255.00	128SW	0.00	
Bluetooth	Diakoptis		Both		255.00	128SW	0.00	<input type="radio"/>
Bluetooth	Mikrofono		Auto		300.00	NL	0.00	
Bluetooth	ledYellow		Remote		255.00	1TB	0.00	<input type="radio"/>
Bluetooth	thermometro		Auto		100.00	NL	-40.00	
Bluetooth	Kinisis		Auto		255.00	0.55SW	0.00	

ArdPill - ledRed

<	March 2018	>				
Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

^ ^
22 : 03
v v

Enter Value:

Save

Reminders

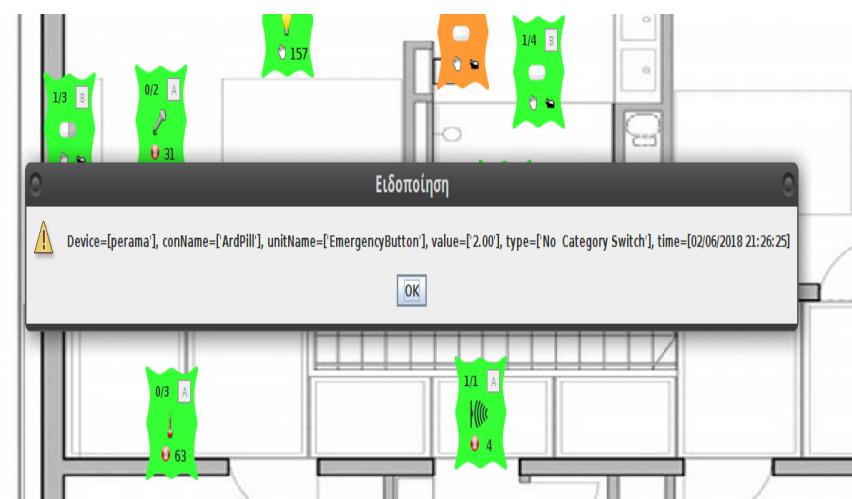
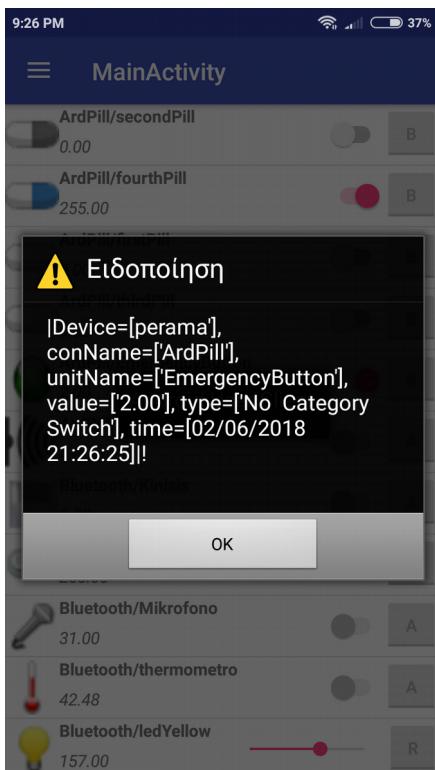
conName	unitName	Datetime	New Value	Read	Delete
ArdPill	ledYellow	06/03/2018 * 13:51:19	Aspirine*13:51 06/03/2018	true	
ArdPill	ledYellow	07/03/2018 * 18:33:42	Aspirine*18:33 07/03/2018	true	

Screenshot 7.19 Ο τρόπος που μπορεί κάποιος να δημιουργήσει Reminders.

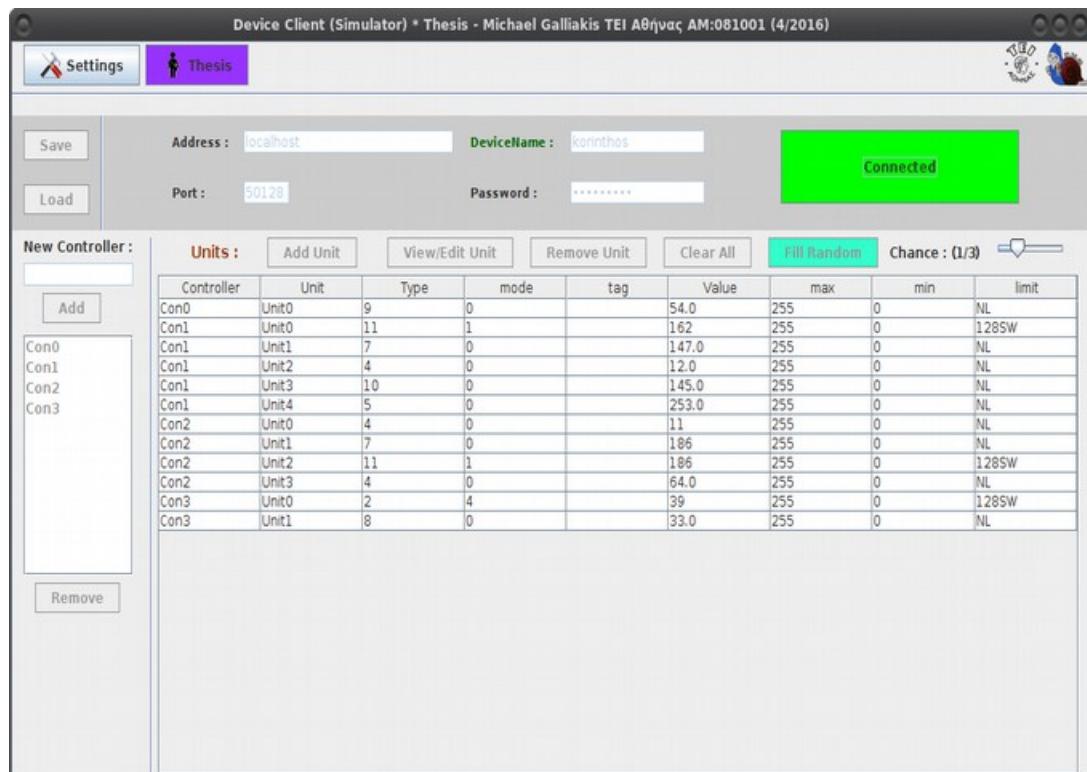
My Events

DeviceName	Time	conName	unitName	Type	Value	mode	tag	limit	max	min	Del
perama	02/06/2018 * 20:56:48	ArdPill	EmergencyButton		2.00	2	Event	1SW	2.00	0.00	
perama	02/06/2018 * 20:56:52	ArdPill	EmergencyButton		2.00	2	Event	1SW	2.00	0.00	
perama	02/06/2018 * 21:04:21	ArdPill	EmergencyButton		2.00	2	Event	1SW	2.00	0.00	
perama	02/06/2018 * 21:26:25	ArdPill	EmergencyButton		2.00	2	Event	1SW	2.00	0.00	
pliroforiki	02/06/2018 * 21:31:40	ArdPill	EmergencyButton		2.00	2	Event	1SW	2.00	0.00	
perama	02/06/2018 * 21:40:09	ArdPill	EmergencyButton		2.00	2	Event	1SW	2.00	0.00	

Screenshot 7.20 Εμφάνιση όλων των καταγεγραμμένων events που αφορούν κάποιον χρήστη.



Screenshot 7.21 Ειδοποίηση events στους User Clients (Android & Desktop).



Screenshot 7.22 Εκτέλεση του Simulator, που έχει το ρόλο ενός Device Client.



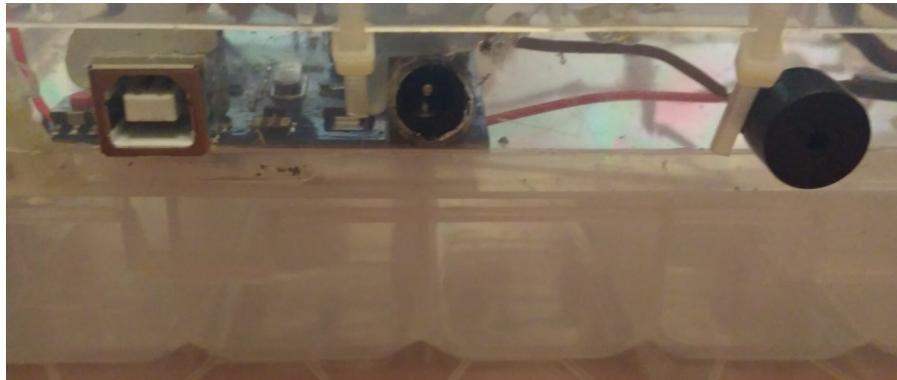
Φωτογραφία 7.23 Συσκευή για να υπενθυμίζει σε κάποιον πότε να παίρνει τα χάπια του (1).



Φωτογραφία 7.24 Συσκευή για να υπενθυμίζει σε κάποιον πότε να παίρνει τα χάπια του (2).

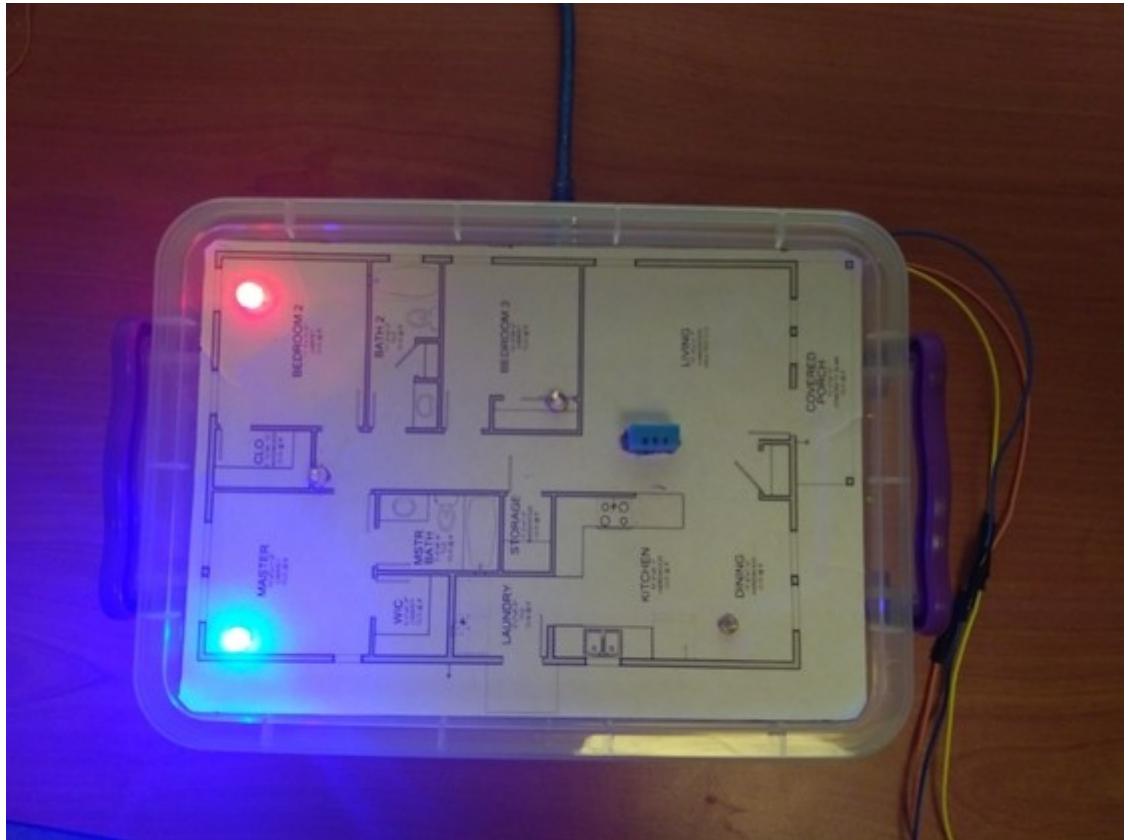


Φωτογραφία 7.25 Συσκευή για να υπενθυμίζει σε κάποιον πότε να παίρνει τα χάπια του (3).

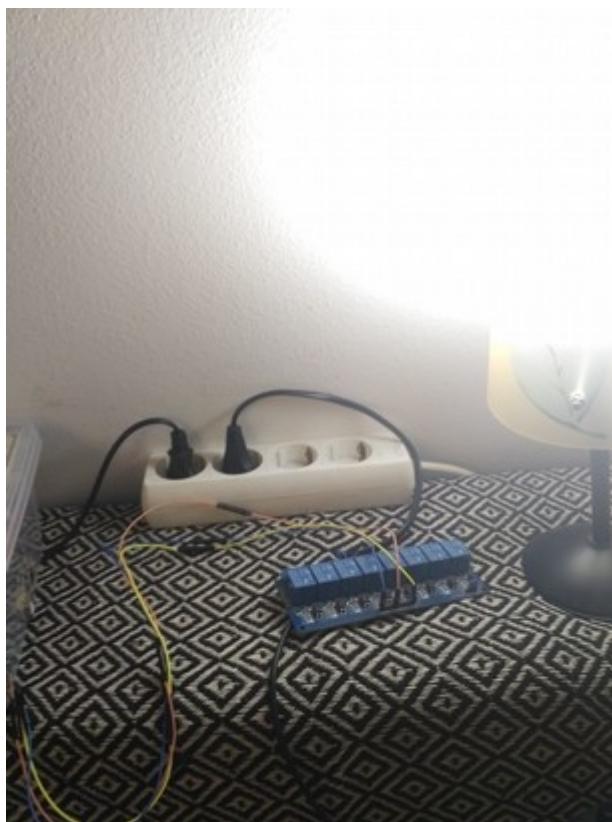


Φωτογραφία 7.26 Πίσω μεριά της συσκευής που υπενθυμίζει σε κάποιον πότε να παίρνει τα χάπια του.

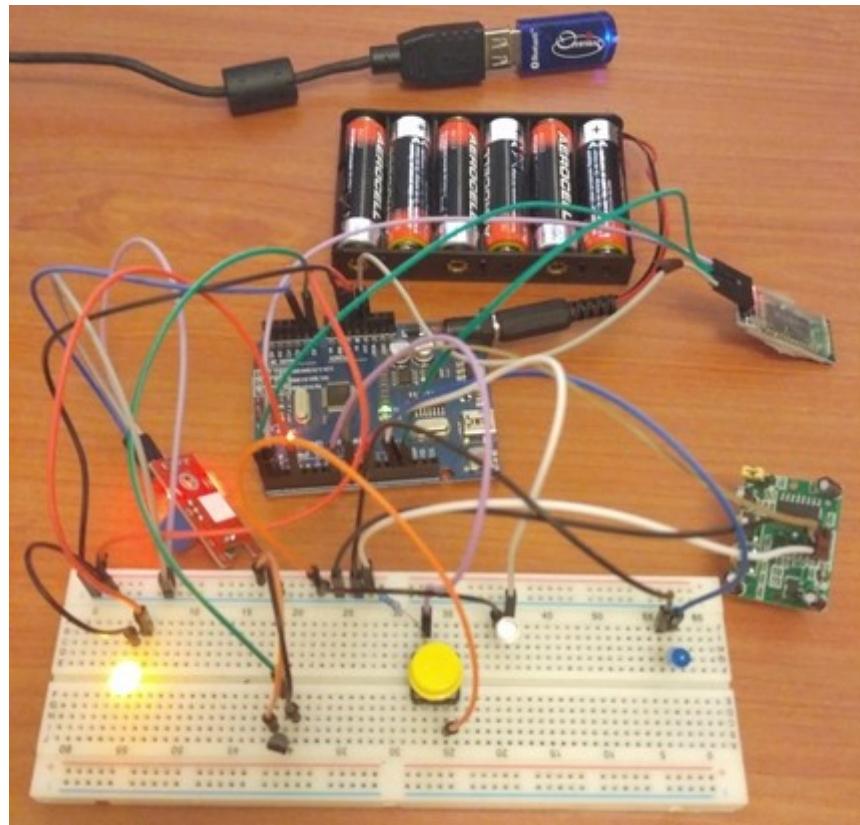
Στις φωτογραφίες 7.23 & 7.24 φαίνεται πως γίνεται η ειδοποίηση σε κάποιον για να πάρει κάποιο συγκεκριμένο φάρμακο. Διακρίνεται, ότι ανάβει το ανάλογο led λαμπάκι πάνω από την συγκεκριμένη θέση του χαπιού, όπως επίσης φαίνεται στην LCD οθόνη η περιγραφή του χαπιού και η ακριβής ώρα που πρέπει να παρθεί. Στην πίσω μεριά της συσκευής (φωτογραφία 7.26) υπάρχει ένα buzzer όπου κάνει κάποιον χαρακτηριστικό ήχο όσο κάποιο λαμπάκι από τα χάπια είναι αναμμένο. Στην φωτογραφία 7.25 φαίνεται πως αφού έχει ανοίξει η θήκη των χαπιών, έχει σβήσει το led λαμπάκι και δεν γράφει πλέον κάτι στην LCD οθόνη. Επίσης, στην φωτογραφία 7.25 φαίνεται αναμμένο το κόκκινο led λαμπάκι, επειδή έχει πατηθεί το κόκκινο emergency button.



Φωτογραφία 7.27 Μικρογραφία ενός έξυπνου σπιτιού.



Φωτογραφία 7.28 Χρήση ρελέ για χειρισμό συσκευών υψηλής τάσης, δύο καταστάσεων (on/off).



Φωτογραφία 7.29 Arduino που επικοινωνεί ασύρματα (με Bluetooth) με κάποιον Device Client.

8

Επίλογος

8.1 Σύνοψη και συμπεράσματα

Μέσα από αυτήν εδώ την διπλωματική εργασία σχεδιάστηκε και αναπτύχθηκε ένα σύστημα με απότερο σκοπό να καλυφθούν κάποιες περιπτώσεις υποβοήθησης υπερηλίκων ή ασθενών κατ' οίκουν. Συγκεκριμένα, με την χρήση του συστήματος γίνεται η υπενθύμιση/ειδοποίηση για την λήψη χαπιών/φαρμάκων. Επίσης, δημιουργούνται ειδοποιήσεις σε οικεία πρόσωπα ή στους φροντιστές/ιατρούς των ατόμων αυτών, όταν συμβαίνουν σημαντικά events (πχ πυρκαγιά ή όταν πατηθεί κάποιο κουμπί έκτακτης ανάγκης). Επιπλέον, υπάρχει η δυνατότητα στους χρήστες του συστήματος, να ανοίγουν-κλείνουν οποιαδήποτε ηλεκτρική συσκευή (δύο καταστάσεων, on/off) που έχουν στο σπίτι τους, είτε τοπικά από μέσα από το σπίτι, είτε απομακρυσμένα από οποудήποτε μέσα στο κόσμο. Παράλληλα, οι χρήστες του συστήματος μπορούν να παρακολουθούν την “αίσθηση” του χώρου ζωντανά με την χρήση διαφόρων αισθητήρων και να χρονοπρογραμματίζουν πότε να γίνονται κάποιες ενέργειες (πχ στις 23:00 να κλείνουν τα φώτα του κήπου).

Σκοπός ήταν να αναπτυχθεί ένα εύχρηστο σύστημα για τους υπερηλίκους/ασθενείς και να χρησιμοποιηθούν τεχνολογίες, γλώσσες προγραμματισμού και άλλα προγραμματιστικά εργαλεία αιχμής. Επιπλέον, το σύστημα χρησιμοποιεί μικρο-ελεγκτές (Arduino) open-hardware και τα υλικά που χρειάζονται για την δημιουργία των έξυπνων μηχανισμών είναι σχετικά οικονομικά.

Από την εργασία φαίνεται ακόμη πως μπορούν συνδυαστικά διαφορετικές μεταξύ τους εφαρμογές και τεχνολογίες να χρησιμοποιηθούν ώστε να υπάρξει ένα ενιαίο αποτέλεσμα και να υφίσταται ένα κατανεμημένο σύστημα. Παράλληλα, φαίνεται στην πράξη μια εφαρμογή του internet of things (διαδίκτυο πραγμάτων) και το πως μπορεί να αξιοποιηθεί το hardware με το software και οι υπολογιστές με τα δίκτυα, ώστε να κάνουν ευκολότερη και καλύτερη την ζωή των ανθρώπων. Διότι, με το σύστημα της εργασίας μπορεί να γίνει η διαχείριση διάφορων πραγμάτων μέσω

του διαδικτύου. Για παράδειγμα, γίνεται εν δυνάμει να ανοίγουν και να κλείνουν όλες οι ηλεκτρικές συσκευές (λάμπες, ανεμιστήρες, ηλεκτρικά παράθυρα, θερμοσίφωνες κ.ά.) που υπάρχουν σε κάποιον χώρο/κτήριο από οποιαδήποτε άλλο σημείο του κόσμου. Όπως επίσης, γίνεται να μπορεί κάποιος χρήστης να γνωρίζει την “αίσθηση” κάποιου χώρου με τη χρήση διάφορων ειδών αισθητήρων. Και μάλιστα όλα αυτά γίνονται σε πραγματικό χρόνο (live) ακόμη και όταν πρόκειται κάποιος χρήστης να βρίσκεται σε κάποιο πολύ μακρινό μέρος (γεωγραφικά) από τον “έξυπνο” χώρο που εποπτεύει. Και αναφέρετε αυτό διότι έγιναν επιτυχημένες δοκιμές πχ στο να ανοίξει και να κλείσει μια λάμπα που βρισκόταν στην Αθήνα από άτομα που ήταν στην Κρήτη και στην Γερμανία.

Η πιλοτική αυτή εφαρμογή πιθανόν θα μπορούσε με ενδεχομένως κάποιες βελτιώσεις και προσθήκες (πχ χρήση καμερών), να χρησιμοποιηθεί σε πραγματικές συνθήκες χρήσης, ώστε να βοηθήσει στην βελτίωση της ποιότητας ζωής των υπερηλίκων/ασθενών.

8.2 Μελλοντικές επεκτάσεις

- ✓ Να ενσωματωθεί η δυνατότητα πραγματικού χρόνου εποπτείας video.
- ✓ Να αναπτυχθεί η User Client εφαρμογή για κινητά τηλέφωνα και άλλων λειτουργικών συστημάτων, όπως IOS (Apple/iPhone) και Windows phone. *Επειδή το σύστημα είναι κατανεμημένο, όλες οι επιμέρους εφαρμογές είναι ανεξάρτητες μεταξύ τους, που σημαίνει πρακτικά ότι μπορεί να επικοινωνήσει μια οποιαδήποτε εφαρμογή (από όποια γλώσσα προγραμματισμού και αν έχει φτιαχτεί) με το σύστημα εφόσον τηρεί το πρωτόκολλο επικοινωνίας/μηνυμάτων που έχει οριστεί.*
- ✓ Να φτιαχτούν arduino sketches (προσαρμογή του προτύπου) ώστε να καλύπτονται περισσότερες μονάδες (πχ και άλλοι τύποι αισθητήρων). Έτσι, θα μπορούσε να μεγαλώσει η λίστα των τύπων μονάδων που υπάρχουν μέχρι τώρα, και να μπορούν οι User Clients να υποστηρίζουν περισσότερα εικονίδια μονάδων (πχ να υπάρχει εικονίδιο ανεμιστήρα, εικονίδιο βρύσης για αυτόματο πότισμα και άλλα πολλά).
- ✓ Να εξελιχθούν περισσότερο τα arduino sketches ώστε να είναι πιο κοντά στην πραγματικότητα και με πιο αξιοποιήσιμες τελικές τιμές οι μετρήσεις από κάποιους αισθητήρες που χρησιμοποιήθηκαν στην εργασία.
- ✓ Να γίνει χρήση βιοαισθητήρων, ώστε να μπορεί να γίνει εποπτεία βιομετρήσεων όπως θερμοκρασία σώματος, καρδιακοί παλμοί, αρτηριακή πίεση και άλλα.
- ✓ Να προγραμματιστούν μικροελεγκτές άλλου τύπου (πχ PIC) και να επικοινωνούν με το σύστημα χρησιμοποιώντας το ίδιο “πρωτόκολλο” μηνυμάτων/επικοινωνίας που έχει οριστεί.
- ✓ Να μπορούν να επικοινωνούν με το σύστημα και wearables συσκευές όπως έξυπνα ρολόγια.
- ✓ Να προσαρμοστεί ο Java Device Client σε android εφαρμογή, ώστε να μην χρειάζεται κάποιο raspberry στα έξυπνα σπίτια, αλλά να αρκεί κάποιο παλιό android κινητό (Επικοινωνία arduino και smartphone μέσω ενσωματωμένου bluetooth του κινητού).
- ✓ Να γίνει χρήση zigBee για ασύρματη επικοινωνία των arduino με τον Device Client.

- ✓ Να φτιαχτούν arduino sketches (προσαρμογή του προτύπου) ώστε να γίνονται περισσότεροι αυτοματισμοί στον έξυπνο χώρο. Πχ όταν ανεβαίνει η θερμοκρασία σε ένα χώρο να ξεκινάει αυτόματα η λειτουργία ενός ανεμιστήρα (Κάτι που είναι ανεξάρτητο από τον άμεσο έλεγχο του χρήστη μέσω εποπτείας ή χρόνοπρογραμματισμού).
- ✓ Να σχεδιαστεί η πλακέτα της έξυπνης συσκευής υπενθύμισης χαπιών, ώστε να τυποποιηθεί η διαδικασία και να μπορούν να παραχθούν τέτοιες συσκευές σε μηχανές παραγωγής.
- ✓ Να εφαρμοστεί το σύστημα πειραματικά σε γηροκομεία/νοσοκομεία ή ακόμη και σε άλλους δημόσιους οργανισμούς όπως σχολεία.
- ✓ Να αναπτυχθεί κατάλληλη εφαρμογή εποπτείας (Οπως ο User Client), ώστε με χρήση BCI (Brain computer interface) συσκευής, να μπορεί κάποιος να αλληλεπιδράσει με το έξυπνο σπίτι του μόνο με τη χρήση του μυαλού του. Για παράδειγμα, να μπορεί να ανοίγει και να κλείνει μια λάμπα μόνο με την σκέψη.
- ✓ Να γίνονται διάφορες καταγραφές (όπως τιμές μονάδων, συνδέσεις χρηστών κ.ά) στην βάση δεδομένων. Παράλληλα, θα πρέπει να εξελιχθεί και να προστεθούν δυνατότητες στην angular εφαρμογή, ώστε να γίνεται ανάκτηση και να αξιοποιούνται αυτές οι νέες πληροφορίες.
- ✓ Το Web Service να έχει το ρόλο και του διακομιστή, ώστε να μοιράζει το φόρτο δυναμικά στους Java Servers. Τώρα, κάθε User & Device Client ρυθμίζεται σε ποιο Java Server (ip & port) θα συνδεθεί. Μελλοντικά, αυτές τις πληροφορίες (διεύθυνση και πόρτα) θα τις παίρνει αυτόματα από το Web Service. Ετσι, στην περίπτωση που θα υπάρχουν πάρα πολλοί ταυτόχρονοι χρήστες/επόπτες και συσκευές/σπίτια συνδεδεμένοι στο σύστημα, θα γίνεται αυτοματοποιημένα η διαδικασία κατανομής του φόρτου στους κατανεμημένους Java Servers. Δηλαδή, ανάλογα τη ζήτηση (online έξυπνα σπίτια και επόπτες), θα δεσμεύονται δυναμικά οι πόροι που θα χρειάζονται στο cloud. Πχ θα ξεκινάνε ή θα κλείσουν Java Servers που θα τρέχουν σε διαφορετικά μηχανήματα, με δικό του επεξεργαστή, μνήμη κλπ.
- ✓ Μέσα από την angular εφαρμογή να γίνεται εποπτεία/διαχείριση της λειτουργίας του συστήματος. Για παράδειγμα, να μπορούν οι χρήστες (τύπου συστήματος) να βλέπουν ποιοι χρήστες (απλοί) είναι συνδεδεμένοι στους Java Servers, τι ip έχουν, πόσα sockets έχει ανοίξει ο κάθε Java Server, τι κίνηση έχει κάποιο συγκεκριμένο “κανάλι” εποπτείας, να παίρνουν στατιστικά στοιχεία, να παρατηρούν τυχόν κακόβουλες ενέργειες κ.ά.
- ✓ Να γίνουν πειράματα μεγαλύτερης κλίμακας.

9

Βιβλιογραφία

Alam, M., Reaz, M. and Ali, M. (2012). A Review of Smart Homes—Past, Present, and Future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6), pp.1190-1203.

De Silva, L., Morikawa, C. and Petra, I. (2012). State of the art of smart homes. *Engineering Applications of Artificial Intelligence*, 25(7), pp.1313-1321.

Wilson, C., Hargreaves, T. and Hauxwell-Baldwin, R. (2014). Smart homes and their users: a systematic analysis and key challenges. *Personal and Ubiquitous Computing*, 19(2), pp.463-476.

Batty, M., Axhausen, K., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G. and Portugal, Y. (2012). Smart cities of the future. *The European Physical Journal Special Topics*, 214(1), pp.481-518.

Amiribesheli, M., Benmansour, A. and Bouchachia, A. (2015). A review of smart homes in healthcare. *Journal of Ambient Intelligence and Humanized Computing*, 6(4), pp.495-517.

Lim, J., Zhan, A., Ko, J., Terzis, A., Szanton, S. and Gitlin, L. (2012). A closed-loop approach for improving the wellness of low-income elders at home using game consoles. *IEEE Communications Magazine*, 50(1), pp.44-51.

Megalingam, R., Unnikrishnan, D., Radhakrishnan, V. and Jacob, D. (2012). HOPE: An electronic gadget for home-bound patients and elders. *2012 Annual IEEE India Conference (INDICON)*.

Mageroski, A., Alsadoon, A., Prasad, P., Pham, L. and Elchouemi, A. (2016). Impact of wireless communications technologies on elder people healthcare: Smart home in Australia. *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*.

Megalingam, R., Pocklassery, G., Jayakrishnan, V., Mourya, G. and Thulasi, A. (2014). Smartphone based continuous monitoring system for home-bound elders and patients. *2014 International Conference on Communication and Signal Processing*.

Basanta, H., Huang, Y. and Lee, T. (2017). Using voice and gesture to control living space for the elderly people. *2017 International Conference on System Science and Engineering (ICSSE)*.

Tsai, H., Tseng, C., Wang, L. and Juang, F. (2017). Bidirectional smart pill box monitored through internet and receiving reminding message from remote relatives. *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*.

Ou, Y., Shih, P., Chin, Y., Kuan, T., Wang, J. and Shih, S. (2013). Framework of ubiquitous healthcare system based on cloud computing for elderly living. *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*.

Wu, L., Lu, J., Zhang, T. and Gong, J. (2016). Robot-assisted intelligent emergency system for individual elderly independent living. *2016 IEEE Global Humanitarian Technology Conference (GHTC)*.

Caranica, A., Cucu, H., Burileanu, C., Portet, F. and Vacher, M. (2017). Speech recognition results for voice-controlled assistive applications. *2017 International Conference on Speech Technology and Human-Computer Dialogue (Sped)*.

Anido, L., Valladares, S., Fernandez-Iglesias, M., Rivas, C. and Gomez, M. (2013). Adapted interfaces and interactive electronic devices for the smart home. *2013 8th International Conference on Computer Science & Education*.

Fuxreiter, T., Mayer, C., Hanke, S., Gira, M., Sili, M. and Kropf, J. (2010). A modular platform for event recognition in smart homes. *The 12th IEEE International Conference on e-Health Networking, Applications and Services*.

Panicker, N. and Sukesh Kumar A (2015). Design of a telemonitoring system for detecting falls of the elderly. *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*.

Kenny Chieng Tze Hing, Chai Chin Tin, Tan Kock Li, YeaDat Chuah, Alvey Hau Lee Cheun and Tan Rong Wei (2012). Development of home monitoring system with integration of safety and security modules. *2012 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT)*.

Ransing, R. and Rajput, M. (2015). Smart home for elderly care, based on Wireless Sensor Network. *2015 International Conference on Nascent Technologies in the Engineering Field (ICNTE)*.

Saldana-Jimenez, D., Rodriguez, M., Espinoza, A. and Garcia-Vazquez, J. (2009). A context-aware component for identifying risks associated to elders' activities of daily living. *Proceedings of the 3d International ICST Conference on Pervasive Computing Technologies for Healthcare*.

Teng-Fa, T. and Cheng-Chien, K. (2013). A smart monitoring and control system for the household electric power usage. *2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*.

Gaddam, A., Mukhopadhyay, S. and Gupta, G. (2008). Development of a Bed Sensor for an Integrated Digital Home Monitoring System. *2008 IEEE International Workshop on Medical Measurements and Applications*.

Gaddam, A., Mukhopadhyay, S. and Gupta, G. (2011). Elder Care Based on Cognitive Sensor Network. *IEEE Sensors Journal*, 11(3), pp.574-581.

Chen, Y., Tsai, M., Fu, L., Chen, C., Wu, C. and Zeng, Y. (2015). Monitoring Elder's Living Activity Using Ambient and Body Sensor Network in Smart Home. *2015 IEEE International Conference on Systems, Man, and Cybernetics*.

Demiris, G. (2009). Privacy and social implications of distinct sensing approaches to implementing smart homes for older adults. *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

Suryadevara, N., Mukhopadhyay, S., Rayudu, R. and Huang, Y. (2012). Sensor data fusion to determine wellness of an elderly in intelligent home monitoring environment. *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*.

Gaddam, A., Mukhopadhyay, S. and Gupta, G. (2009). Smart home using optimized number of wireless sensors for elderly care. *2009 Applied Electromagnetics Conference (AEMC)*.

Zhang, X., Wang, H. and Yu, Z. (2010). Toward a Smart Home Environment for Elder People Based on Situation Analysis. *2010 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing*.

Gaddam, A., Mukhopadhyay, S. and Gupta, G. (2010). Towards the Development of a Cognitive Sensors Network Based Home for Elder Care. *2010 6th International Conference on Wireless and Mobile Communications*.

Gaddam, A., Mukhopadhyay, S. and Gupta, G. (2011). Trial & experimentation of a smart home monitoring system for elderly. *2011 IEEE International Instrumentation and Measurement Technology Conference*.

Suryadevara, N. and Mukhopadhyay, S. (2012). Wireless Sensor Network Based Home Monitoring System for Wellness Determination of Elderly. *IEEE Sensors Journal*, 12(6), pp.1965-1972.