Ομάδα 16 : Εφαρμογή smart home

Κρίστι Τοπολλάι 1059389 $4^{ο}$ **Έτος**
Ζήσης Μανούσκος 1063974 $4^{ο}$ **Έτος**

Η παρούσα εργασία αφορά το μάθημα εβδόμου εξαμήνου 'Βάσεις Δεδομένων', θέμα της είναι η σχεδίαση μιας βάσης δεδομένων ενός μικρόκοσμου και η αξιολόγηση της λειτουργικότητας της μέσω κατασκευής της αντίστοιχης εφαρμογής. Ο μικρόκοσμος του οποίου τη βάση υλοποιούμε αφορά ένα smart home. Παρακάτω θα παρουσιαστούν βήμα βήμα οι παραδοχές που έγιναν και πως με βάση αυτές πάρθηκαν πρωτοβουλίες οι οποίες οδήγησαν στην επίτευξη των στόχων της προσδοκώμενης χρήσης.

1 ΠΕΡΙΓΡΑΦΗ, ΠΡΟΔΙΑΓΡΑΦΕΣ, ΠΑΡΑΔΟΧΕΣ ΤΟΥ ΜΙΚΡΟΚΟΣΜΟΥ

1.1 Τι είναι ένα smart home

Θα χρειαστεί αρχικά να κατανοήσουμε τι είναι ένα smart home και ποια είναι τα βασικά στοιχεία που το συνθέτουν.

Πρόκειται για ένα σύνολο οικιακών αυτοματισμών, σκοπός του οποίου είναι ο έλεγχος και η επίβλεψη διαφόρων οικιακών γνωρισμάτων. Τέτοια γνωρίσματα μπορούν να είναι ο φωτισμός, το κλίμα, συστήματα ψυχαγωγίας και γενικά κάθε είδος οικιακής συσκευής. Συχνά συμπεριλαμβάνονται και τα συστήματα ελέγχου πρόσβασης και οι συναγερμοί. Όλα τα παραπάνω μπορούν να συνδέονται στο διαδίκτυο κάτι που επιβεβαιώνει και το γεγονός πως οι οικιακές συσκευές αποτελούν σημαντικό συστατικό του διαδίκτυου των πραγμάτων (IoT).

Ένα σύστημα οικιακού αυτοματισμού συνήθως συνδέει ελεγχόμενες συσκευές με έναν κεντρικό κόμβο ή "πύλη". Η διεπαφή χρήστη για τον έλεγχο του συστήματος χρησιμοποιεί είτε επιτοίχια τερματικά, tablet ή επιτραπέζιους υπολογιστές, μια εφαρμογή κινητού τηλεφώνου ή μια διεπαφή Ιστού που μπορεί επίσης να είναι προσβάσιμη εκτός χώρου ακινήτου μέσω του Διαδικτύου.

Ιστορικά, τα συστήματα έχουν πωληθεί ως πλήρη συστήματα όπου ο καταναλωτής βασίζεται σε έναν προμηθευτή για ολόκληρο το σύστημα, συμπεριλαμβανομένου του υλικού, του πρωτοκόλλου επικοινωνιών, του κεντρικού κόμβου και της διεπαφής χρήστη. Ωστόσο, τώρα υπάρχουν συστήματα ανοιχτού υλικού και λογισμικού ανοιχτού κώδικα που μπορούν να χρησιμοποιηθούν αντί ή με ιδιόκτητο υλικό. Πολλά από αυτά τα συστήματα διασυνδέονται με ηλεκτρονικά είδη ευρείας κατανάλωσης, όπως το Arduino ή το Raspberry Pi, τα οποία είναι εύκολα προσβάσιμα στο διαδίκτυο και στα περισσότερα καταστήματα ηλεκτρονικών. Επιπλέον, οι οικιακές συσκευές αυτοματισμού διασυνδέονται όλο και περισσότερο με κινητά τηλέφωνα μέσω Bluetooth, προσφέροντας πιο προσιτές τιμές και δυνατότητες προσαρμογής στον χρήστη.

Ο οικιακός αυτοματισμός καλύπτει ένα ευρύ φάσμα οικιακών αναγκών:

- Θέρμανση, εξαερισμός και κλιματισμός : είναι δυνατόν να υπάρχει τηλεχειριστήριο όλων των οικιακών οθονών ενέργειας μέσω του Διαδικτύου με ενσωματωμένο ένα απλό και φιλικό περιβάλλον εργασίας χρήστη.
- Σύστημα ελέγχου φωτισμού: ένα «έξυπνο» δίκτυο που ενσωματώνει την επικοινωνία μεταξύ διαφόρων εισόδων και εξόδων του συστήματος φωτισμού, χρησιμοποιώντας μία ή περισσότερες κεντρικές υπολογιστικές συσκευές.
- Έλεγχος και ενσωμάτωση συσκευών με smart grid και smart meter, εκμεταλλευόμενοι, για παράδειγμα, την υψηλή απόδοση ηλιακού πλαισίου στη μέση της ημέρας για τη λειτουργία πλυντηρίων.
- Οικιακά ρομπότ και ασφάλεια: ένα οικιακό σύστημα ασφαλείας ενσωματωμένο σε σύστημα οικιακού αυτοματισμού μπορεί να παρέχει πρόσθετες υπηρεσίες, όπως απομακρυσμένη παρακολούθηση κάμερων

Fig. 1. Παραδείγματα έξυπνων συσκευών του εμπορίου

ασφαλείας μέσω Διαδικτύου, ή έλεγχο πρόσβασης και κεντρικό κλείδωμα όλων των περιμετρικών θυρών και παραθύρων.
- Ανίχνευση διαρροών, ανιχνευτές καπνού και CO.
- Οικιακός αυτοματισμός για ηλικιωμένους και άτομα με ειδικές ανάγκες.
- Φροντίδα κατοικίδιων και μωρών, για παράδειγμα, παρακολούθηση των κινήσεων των κατοικίδιων και των μωρών και έλεγχος των δικαιωμάτων πρόσβασης των κατοικίδιων.
- Έλεγχος ποιότητας αέρα. Για παράδειγμα, το Air Quality Egg χρησιμοποιείται από τους ανθρώπους στο σπίτι για να παρακολουθεί την ποιότητα του αέρα και το επίπεδο ρύπανσης στην πόλη και να δημιουργεί έναν χάρτη ρύπανσης.
- Έξυπνη κουζίνα και συνδεδεμένη κουζίνα.
- Συσκευές φωνητικού ελέγχου όπως το Amazon Alexa ή το Google Home χρησιμοποιούνται για τον έλεγχο οικιακών συσκευών ή συστημάτων.

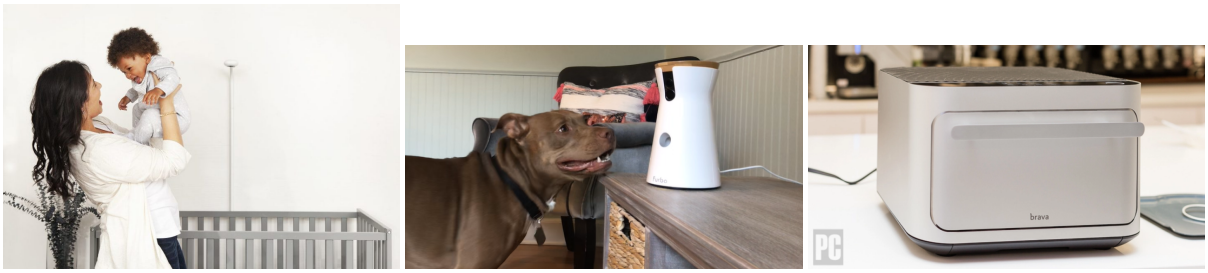

Fig. 2. Παραδείγματα έξυπνων συσκευών επίβλεψης μωρών, κατοικιδίων και συσκευών κουζίνας

Η χρήση του οικιακού αυτοματισμού θα μπορούσε να οδηγήσει σε πιο αποτελεσματικές και έξυπνες τεχνικές εξοικονόμησης ενέργειας. Με την ενσωμάτωση τεχνολογιών πληροφοριών και επικοινωνιών σε συστήματα ανανεώσιμης ενέργειας, όπως η ηλιακή ενέργεια ή η αιολική ενέργεια, τα σπίτια μπορούν να λάβουν αυτόνομα αποφάσεις σχετικά με το εάν θα αποθηκεύσουν ενέργεια ή θα την ξοδέψουν για μια δεδομένη συσκευή, οδηγώντας σε συνολικές θετικές περιβαλλοντικές επιπτώσεις και μείωση λογαριασμών ηλεκτρικού. Για να το κάνουν αυτό, οι ερευνητές προτείνουν τη χρήση δεδομένων από αισθητήρες σχετικά με τη δραστηριότητα των καταναλωτών

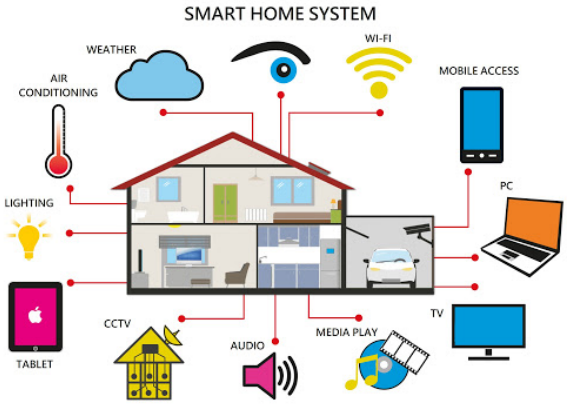

Fig. 3. Μια τυπική δομή ενός έξυπνου σπιτιού

εντός του σπιτιού για να προβλέψουν τις ανάγκες των καταναλωτών και να το εξισορροπήσουν με την κατανάλωση ενέργειας.

Επιπλέον, ο αυτοματισμός στο σπίτι έχει μεγάλες δυνατότητες όσον αφορά και την ασφάλεια της οικογένειας. Σύμφωνα με μια έρευνα του 2015 που πραγματοποιήθηκε από το iControl, οι κύριοι οδηγοί της ζήτησης για έξυπνες και συνδεδεμένες συσκευές είναι πρώτον η προσωπική και οικογενειακή ασφάλεια και δεύτερον η εξοικονόμηση ενέργειας. Ο οικιακός αυτοματισμός περιλαμβάνει μια ποικιλία έξυπνων συστημάτων ασφαλείας και ρυθμίσεων παρακολούθησης. Αυτό επιτρέπει στους καταναλωτές να παρακολουθούν τα σπίτια τους ενώ βρίσκονται μακριά και να παρέχουν στα αξιόπιστα μέλη της οικογένειας πρόσβαση σε αυτές τις πληροφορίες σε περίπτωση που συμβεί κάτι κακό.

Συνολικά, αυτό το πεδίο εξελίσσεται και η φύση κάθε συσκευής αλλάζει συνεχώς. Ενώ οι τεχνολόγοι εργάζονται για να δημιουργήσουν πιο ασφαλή, εξορθολογισμένα και τυποποιημένα πρωτόκολλα ασφαλείας, οι καταναλωτές πρέπει επίσης να μάθουν περισσότερα για το πώς λειτουργούν αυτές οι συσκευές και ποιες είναι οι επιπτώσεις της τοποθέτησής τους στα σπίτια τους. Η ανάπτυξη αυτού του πεδίου περιορίζεται προς το παρόν όχι μόνο από την τεχνολογία αλλά και από την ικανότητα ενός χρήστη να εμπιστεύεται μια συσκευή και να την ενσωματώνει με επιτυχία στην καθημερινή του ζωή.

1.2 Περιγραφή του μικρόκοσμου

Όπως αναφέρθηκε προηγουμένως ένα smart home πρακτικά είναι το σύνολο των έξυπνων συσκευών που το απαρτίζουν. Συνεπώς κεντρικη οντότητα του μικρόκοσμου μας θα είναι η συσκευή. Η συσκευή όμως σαν οντότητα είναι εξαιρετικά πολύπλοκη, οι λειτουργίες δύο διαφορετικών συσκευών συχνά είναι τελείως διαφορετικές με αποτέλεσμα η περιγραφή τους μέσω της οντότητας συσκευή και τον αντίστοιχων γνωρισμάτων της να είναι αδύνατη. Αυτός είναι και ο λόγος μάλιστα που στο εμπόριο συνήθως η κάθε συσκευή ελέγχεται απο διαφορετική εφαρμογή.

Φτάνουμε έτσι λοιπόν στην πρώτη παραδοχή. Θεωρούμε ότι όλες οι συσκευές του μικρόκοσμου ακολουθούν ένα συγκεκριμένο πρότυπο. Αυτό μπορεί να ταυτιστεί στην πραγματικότητα με την προσέγγιση διάφορων εταιριών που πράγματι διαθέτουν διάφορες έξυπνες συσκευές αλλά μόνο μια εφαρμογή ελέγχου τους π.χ. Google Nest. Μπορούμε συνεπώς να θεωρήσουμε ότι δουλεύουμε για μια τέτοια εταιρεία και παίρνουμε την πρωτοβουλία οι συσκευές μας να ακολουθούν ένα συγκεκριμένο πρότυπο, κάτι που θα καθιστά εφικτή την

κατασκευή της βάσης δεδομένων του σπιτιού και στη συνέχεια την κατασκευή μοναδικής εφαρμογής για τον έλεγχο τους.

Μένει έτσι να οριστεί αυτό το πρότυπο το οποίο θα επιβάλουμε στις συσκευές. Θεωρούμε ότι μια συσκευή μπορεί να περιγραφεί πλήρως από δυο πράγματα. Ένα σύνολο παραμέτρων και ένα σύνολο εντολών.

Το σύνολο παραμέτρων περιέχει όλες τις παράμετρους μιας συσκευής, τόσο μεταβλητές όσο και αμετάβλητες. Ας πάρουμε σαν παράδειγμα έναν έξυπνο θερμοστάτη, όπως γνωρίζουμε στην απλούστερη εκδοχή του ο θερμοστάτης χρησιμοποιείται για να ρυθμίζει τη θερμοκρασία ενός χώρου. Μια τέτοια συσκευή θα μπορούσε να περιγραφεί από τρεις παραμέτρους, το αν είναι ενεργοποιημένος, μια παράμετρος που θα μπορούσε να είναι δυαδική και να παίρνει είτε την τιμή 0 (όχι) ή 1 (ναι), τη θερμοκρασία του χώρου η οποία θα μετράται μέσω ενός αισθητήρα της συσκευής και την επιθυμητή θερμοκρασία του χώρου. Κάτι τέτοιο γίνεται προφανές πως θα μπορούσε να γενικευθεί εύκολα και σε πιο σύνθετες συσκευές. Ένα σύστημα έξυπνου συναγερμού θα μπορούσε να περιγραφεί από το εάν είναι ενεργοποιημένος ο συναγερμός (δυαδική), εάν χτυπάει ο συναγερμός (δυαδική), σε ποιες ώρες της ημέρας θέλουμε να ενεργοποιείται αυτόματα ο συναγερμός, σε ποια τηλέφωνα θα πραγματοποιείται κλήση σε περίπτωση διάρρηξης κλπ.

Το σύνολο των εντολών περιλαμβάνει όλες τις πιθανές ενέργειες που μπορεί να κάνει κάποιος χρήστης. Αυτές οι ενέργειες αφορούν τις παραμέτρους της συσκευής, έτσι σαν εντολή ορίζεται η επέμβαση ενός χρήστη στις παραμέτρους της συσκεύης. Επιστρέφοντας στο παράδειγμα του θερμοστάτη, έχουμε τρεις πιθανές ενέρεγεις, τις άναψε και σβήσε που θα θέτουν την παράμετρο ενεργοποιημένος σε αληθή και ψευδή αντίστοιχα και την εντολή θέσε θερμοκρασία που θα θέτει την παράμετρο επιθυμητή θερμοκρασία χώρου σύμφωνα με την είσοδο του χρήστη η οποία στη συνέχεια θα συγκρίνεται με τη θερμοκρασία του χώρου για να καταφέρει έτσι η συσκευή σε επίπεδο υλικού πλέον να πραγματοποιήσει την εντολή αυτή.

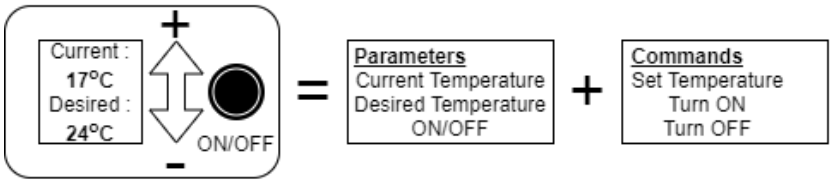

Fig. 4. Η υλοποιήση ενός θερμοστάτη σύμφωνα με το παραπάνω πρότυπο

Η δεύτερη κεντρική οντότητα δε είναι άλλη απο τον χρήστη. Ως χρήστης ορίζεται οποιοσδήποτε μπορεί να χρησιμοποιήσει τις συσκευές του έξυπνου σπιτιού. Για τον χρήστη γίνονται δυο παραδοχές.

Η πρώτη που αφορά επίσης ολόκληρη τη λειτουργία του σπιτιού είναι, ότι όλες οι εντολές και κατ'επέκταση η χρήση των συσκευών γίνεται μέσω της εφαρμογής. Αυτή η παραδοχή φαντάζει αρχικά αυστηρή αλλά είναι απαραίτητη όπως θα διαπιστώσουμε στη συνέχεια. Άλλωστε ο σκοπός μας είναι η βάση δεδομένων να χρησιμοποιείται απο την εφαρμογή συνεπώς δικαιολογημένα μπορούμε να περιοριστούμε στη χρήση των συσκευών μόνο μέσω κάποιας εφαρμογής.

Η δεύτερη παραδοχή είναι πως δεν επιθυμούμε να έχουν όλοι πρόσβαση σε όλα. Παραδείγματος χάρη σε ένα σπίτι θα θέλαμε να απαγορεύσουμε την πρόσβαση των μικρών παιδιών σε επικίνδυνες σύσκευες όπως συσκευές κουζίνας ή σε πολύπλοκες συσκευές όπως ο συναγερμός κλπ. Θα μπορούσε βέβαια κάποιος να ισχυριστεί πως το να απαγορεύεται μέσω της εφαρμογής η χρήση κάποιας συσκευής από κάποιον χρήστη δεν εξασφαλίζει ότι ο ίδιο χρήστης δε θα μπορέσει να τη χρησιμοποιήσει άμεσα. Παρόλο δηλαδή που θα φροντίσουμε κάποιο παιδί να μην μπορεί να ανάψει τον φούρνο μέσω της εφαρμογής, δε μας εξασφαλίζει κάποιος οτι δεν μπορεί να πάει στον φούρνο και να αρχίσει να πατάει κουμπιά. Αυτός έιναι και ο λόγος που κάναμε την πρώτη παραδοχή.

Μια τέτοια παραδοχή είναι πολύ λογική μιας και εξασφαλίζει την ασφάλεια των μελών της οικογένειας, αλλά θέτει και τα απαραίτητα όρια σε χρήστες εκτός της οικογένειας που θα ήθελαν να χρησιμοποιήσουν κάποια συσκευή, όπως κάποιος επισκέπτης ή ο διανομέας. Επιπλέον εξυπηρετούνται καλύτερα ακόμα πιο γενικές περιπτώσεις που αντί για έξυπνο σπίτι έχουμε κάποιο ξενοδοχείο εφοδιασμένο με έξυπνες συσκευές ή κάποιον εργασιακό χώρο. Υπενθυμίζουμε επίσης ότι η εφαρμογή δεχόμαστε πως λειτουργεί τόσο σε φορητές συσκευές όσο και σε εντοιχισμένα πάνελ και διεπαφές χρήστη.

Πρέπει να υλοποιηθεί κάπως η οντότητα του χρήστη που θα καθιστά δυνατή την επιβολή της παραπάνω προδιαγραφης. Αυτό μπορεί να γίνει εαν αντί της οντότητας χρήστη ορίσουμε την οντότητα προφίλ χρήστη. Το προφίλ χρήστη θα παρουσιάζεται με δύο εκδοχές. Η πρώτη θα είναι αυτή του πρωτεύοντος προφίλ και θα αφορά τα προφίλ εκείνα που δεν έχουν περιορισμούς και μπορούν να χρησιμοποιήσουν οποιαδήποτε συσκευή. Η δεύτερη εκδοχή είναι το δευτερεύον προφίλ το οποίο θα αφορά τα προφίλ που έχουν πρόσβαση σε συγκεκριμένες μόνο συσκευές. Με αυτόν τον τρόπο μπορούν να διαχωρίζονται τα δικαιώματα των διαφόρων χρηστών και να τίθενται τα κατάλληλα όρια.

Ποιος θα καθορίζει όμως τα δικαιώματα των δευτερευόντων προφίλ; Ερχόμαστε στην τρίτη παραδοχή που αφορά τους χρήστες, η οποία ορίζει πως κάθε δευτερεύον προφίλ πρέπει να συνοδευέται οπωσδήποτε από ένα πρωτεύον προφίλ που θα το επιβλέπει. Ένα πρωτεύον προφίλ μπορεί να επιβέπει πολλά δευτερεύοντα, αλλά ένα δευτερεύον προφίλ μπορεί να επιβλέπεται μόνο από ένα πρωτεύον. Τα δικαιώματα του κάθε δευτερεύοντος προφίλ καθορίζονται λοιπόν από το αντίστοιχο πρωτεύον που το επιβλέπει. Οπότε έαν σε κάθε μέλος της οικογένειας αντιστοιχεί κάποιο προφίλ χρήστη θα μπορούσαν τα πρωτεύοντα προφίλ να είναι των γονιών και τα δευτερεύοντα των παιδιών, ή ακόμα καλύτερα οι γονείς να μοιράζονται ένα πρωτεύον προφίλ και τα παιδιά ένα δευτερεύον.

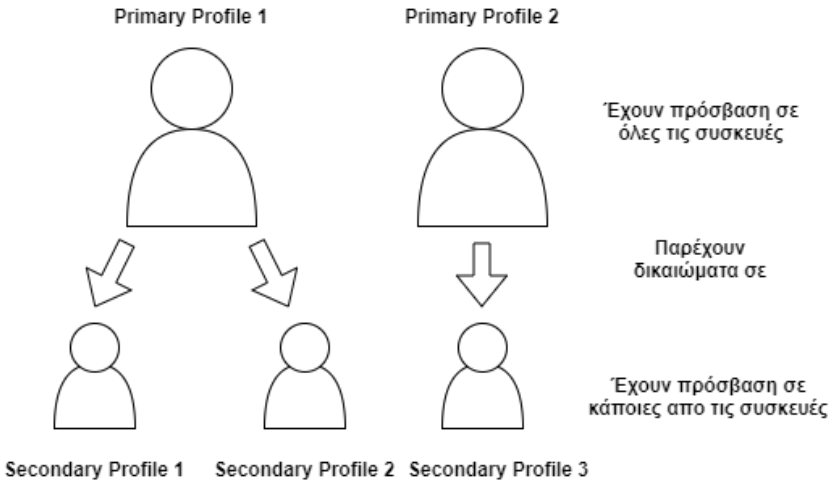

Fig. 5. Δικαιώματα των προφίλ χρηστών

Τέλος όπως έχει ήδη αναφερθεί η επικοινωνία μεταξύ χρηστών και συσκευών γίνεται μέσω των εντολών. Ως εντολές προηγουμένως ορίσαμε ένα από τα δύο βασικά συστατικά μιας συσκευής. Εκτός από αυτό οι εντολές είναι και σημαντική οντότητα του μικρόκοσμου και μέσω αυτών αποτυπώνεται κάθε ενέργεια ενός χρήση στον κόσμο αυτό. Ως προς τις εντολές δε γίνεται κάποια συγκεκριμένη παραδοχή που χρήζει έξτρα συζήτησης. Όπως είναι προφανές θα πρέπει κάθε εντολή να προέρχεται απο το σύνολο των εντολών της συσκευής που χρησιμοποιεί ο χρήστης, ενώ η κάθε εντολή αφορά μία μόνο συσκευή (δεν λαμβάνουμε υπόψη

δηλαδή εντολές του τύπου 'φτιάξε κλίμα' που θα προσάρμοζαν τον φωτισμό και θα μας έπαιζαν και κάποιο τραγούδι. Φυσική συνέπεια που μας συνδέει με όσα ειπώθηκαν παραπάνω είναι πως κάποιο δευτερεύον προφίλ μπορεί να εκτελέσει εντολές μόνο σε συσκευές στις οποιές έχει πρόσβαση.

Συνοψίζοντας ο μικρόκοσμος του smart home μας έχει ως εξής. Το εννοιολογικό μοντέλο αποτελείται από τρεις βασικές οντότητες , το προφίλ χρήστη, τη συσκευή και την εντολή. Ο χρήστης μέσω ενός προφίλ πραγματοποιεί εντολές, οι οποίες εντολές ελέγχουν την αντίστοιχη συσκευή. Τα προφίλ χρήστη είναι δύο ειδών, πρωτεύοντα και δευτερεύοντα. Τα πρωτεύοντα προφίλ έχουν πρόσβαση σε κάθε συσκευή, τα δευτερεύοντα από την άλλη μόνο σε συγκεκριμένες. Τα πρωτεύοντα προφίλ είναι αυτά που παρέχουν δικαιώματα στα δευτερεύοντα και καθορίζουν σε ποιες συσκευές έχουν πρόσβαση. Παρόλο που ο μικρόκοσμος φαντάζει συμπυκνωμένος και αρκετά αφηρημένος λόγω των παραδοχών, όπως θα φανεί παρακάτω μέσω κατάλληλων δομών θα τον καταστήσουμε πλήρως λειτουργικό και θα καλύπτει ικανοποιητικά κάθε προδιαγραφή που θα πρέπει να πληρεί μια βάση δεδομένων ενός έξυπνου σπιτιού.

2 ΑΠΟ ΤΗΝ ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΜΙΚΡΟΚΟΣΜΟΥ ΣΤΗΝ SQL

2.1 Διάγραμμα οντοτήτων συσχετίσεων

Σύμψωνα με την παραπάνω ανάλυση του μικρόκοσμου μπορούμε πλεόν να προβούμε στον σχεδιασμό του διαγράμματος οντοτήτων συσχετίσεων. Ύστερα θα ακολουθήσει περιγραφή των γνωρισμάτων και των πληθικοτήτων.

Ξεκινώντας από τις οντότητες έχουμε το προφίλ χρήστη. Οποιοδήποτε προφίλ χρήστη ανεξάρτητα απο το εάν είναι πρωτεύον ή δευτερεύον έχει τα εξής γνωρίσματα:

- username, αποτελεί το κλειδί της οντότητας άρα για κάθε προφίλ είναι διαφορετικό.
- password, κάθε προφίλ χρειάζεται κάποιον κωδικό πρόσβασης για λόγους ασφάλειας, κάτι που ενισχύεται από το γεγονός πως το κάθε προφίλ έχει διαφορετικά δικαιώματα.
- alternative password, αποτελεί εναλλακτικό κωδικό πρόσβασης, για περιπτώσεις απώλειας του κανονικού κωδικού.
- δημόσιο, μπορεί να είναι είτε αληθές είτε ψευδές, όταν είναι αληθές δε χρειάζεται κωδικός για πρόσβαση στο συγκεκριμένο προφίλ. Εξυπηρετούνται έτσι περιπτώσεις κάτα τις οποίες πχ έχουμε επισκέπτες στο σπίτι. Τότε δε θα ήταν πρακτικό να κατασκευαστεί καινούριο προφίλ, αντιθέτως θα προυπάρχει ένα δημόσιο προφίλ το οποίο θα χρησιμοποιείται από επισκέπτες και θα έχει τα αντίστοιχα δικαιώματα.
- πολλαπλών χρηστών, μπορεί να είναι είτε αληθές είτε ψευδές και εξυπηρετεί περιπτώσεις κάτα τις οποίες ένα προφίλ πρόκειται να χρησιμοποιήθει από πολλούς χρήστες.

Το προφίλ χρήστη επιπλέον όπως αποτυπώνεται και στο διάγραμμα ειδικεύεται σε πρωτεύον και δευτερεύον. Τα οποία δεν εμπεριέχουν κάποιο επιπλέον γνώρισμα αλλά συμμετέχουν σε διαφορετικές συσχετίσεις όπως θα αναλύσουμε και παρακάτω.

Η επόμενη οντότητα είναι η συσκευή και έχει τα εξής γνωρίσματα :

- device id, αποτελεί κλειδί της οντότητας και χρησιμεύει στο να ξεχωρίζονται οι συσκευές μεταξύ τους.
- είδος, λεκτική περιγραφή της συσκευής, χρησιμεύει στη διεπαφή χρήστη εφαρμογής.
- ενεργή, μπορεί να είναι αληθής ή ψευδής, ανάλογα με την τιμή της όπως θα δείξουμε παρακάτω προσαρμόζονται οι διαθέσιμες εντολές της συσκευής.
- δωμάτιο, σε ποιο δωμάτιο βρίσκεται η συσκευή.
- KWh, η κατανάλωση ισχύος της συσκευής σε κιλοβατώρες, χρησιμεύει στην επίβλεψη της συνολικής κατανάλωσης των συσκευών όπως θα δείξουμε παρακάτω.

Η τελευταία οντότητα είναι η εντολή και έχει τα εξής γνωρίσματα :

- command id, αποτελεί κλειδί της οντότητας και χρησιμεύει στο να ξεχωρίζονται οι εντολές μεταξύ τους.

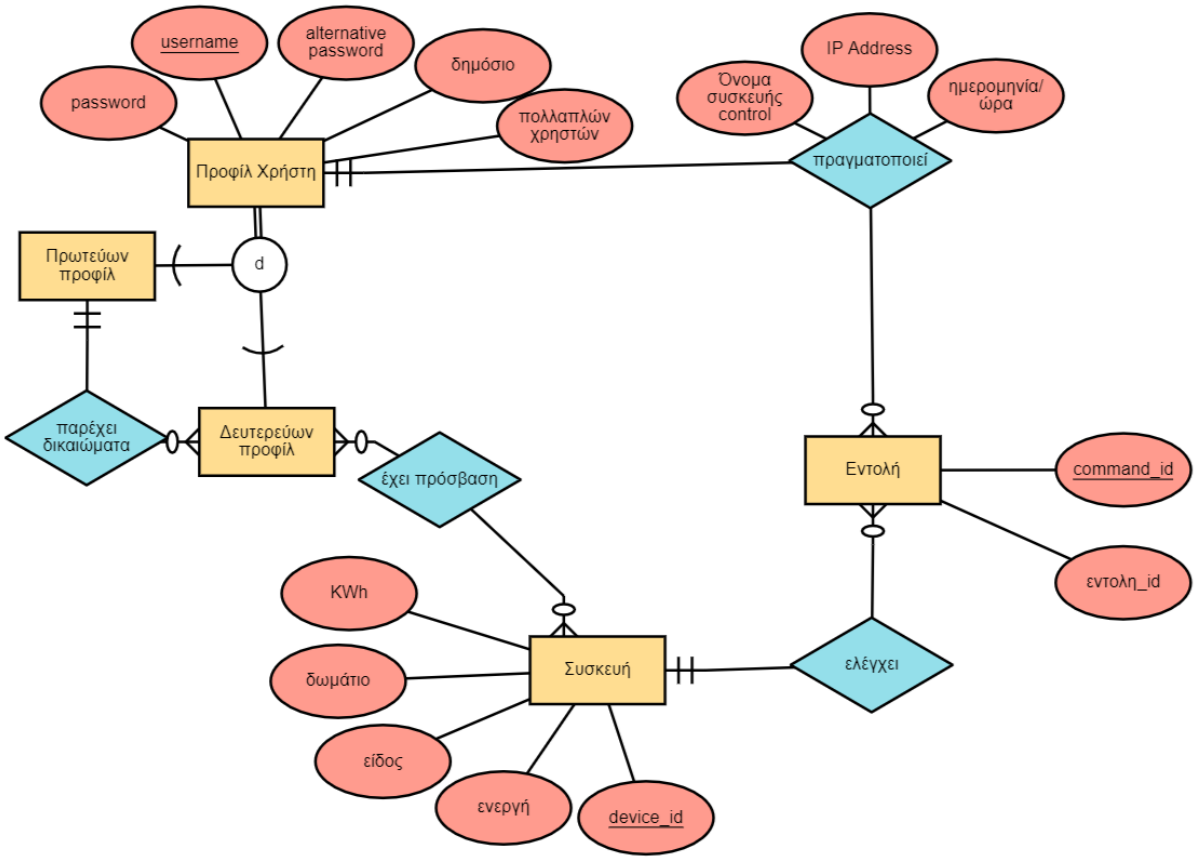

Fig. 6. Εκτεταμένο μοντέλο οντοτήτων συσχετίσεων smart home

- εντολη id, αποτελεί το κλειδι της εντολής που τη ξεχωρίζει απο το σύνολο των διαθέσιμων εντολών όλων των άλλων συσκευών, περισσότερα παρακάτω .

Ακολουθεί περιγραφή των συσχετίσεων:

- Παρέχει δικαιώματα με πληθικότητα one-one : zero-many. Όπως περιγράψαμε παραπάνω στις παραδοχές ένα πρωτεύον προφίλ μπορεί να επιβλέπει πολλά δευτευόντα, ενώ κάθε δευτερεύον προφίλ πρέπει να επιβλέπεται οπωσδήποτε από ένα και μόνο ένα πρωτεύον προφίλ.
- Έχει πρόσβαση με πληθικότητα zero-many : zero-many. Συσχέτιση που περιγράφει τα δικαιώματα ενός δευτερεύοντος προφίλ. Κάθε δευτερεύον προφίλ μπορεί να έχει πρόσβαση σε πολλές ή και καμία συσκευές (αν και αυτό καθιστά την ύπαρξη του προφίλ αυτού περιττή). Σε μία συσκευή μπορούν να έχουν πρόσβαση πολλα ή και κανένα δευτερεύον προφίλ.
- Ελέγχει με πληθικότητα zero-many : one-one. Συσχέτιση που περιγράφει ποια συσκευή αφορά μια συγκεκριμένη εντολή. Κάθε εντολή αφορά μία και μόνο μία συσκευή, παράλληλα στην ίδια συσκευή μπορεί να μη δώθηκε καμία ή να δώθηκαν πολλές εντολές.
- Πραγματοποιεί με πληθικότητα one-one : zero-many. Συσχέτιση που περιγράφει ποιο προφίλ πραγματοποιεί κάποια συγκεκριμένη εντολή. Κάθε εντολή πραγματοποιείται από ένα και μόνο ένα προφίλ χρήστη

ενώ το κάθε προφίλ χρήστη έχει δικαίωμα να πραγματοποιήσει όσες εντολές θέλει. Γνωρίσματα της συσχέτισης αυτής είναι η ημερομηνία/ώρα μέσω της οποίας θα μπορούμε να γνωρίζουμε πότε πραγματοποιήθηκε η εντολή. Το όνομα της συσκευής στην οποία είναι εγκατεστημένη η έκδοση της εφαρμογής που χρησιμοποιήσε ο χρήστης για να πραγματοποιήσει την εντολή. Η IP Address η οποία θα γνωρίζει την IP Address της συσκευής αυτής.

Έχοντας πλέον το ERD διάγραμμα μπορούμε έυκολα να κατασκεύασουμε και το αντίστοιχο λογικό σχεσιακό μοντέλο με βάση τους γνωστούς μετασχηματισμούς.

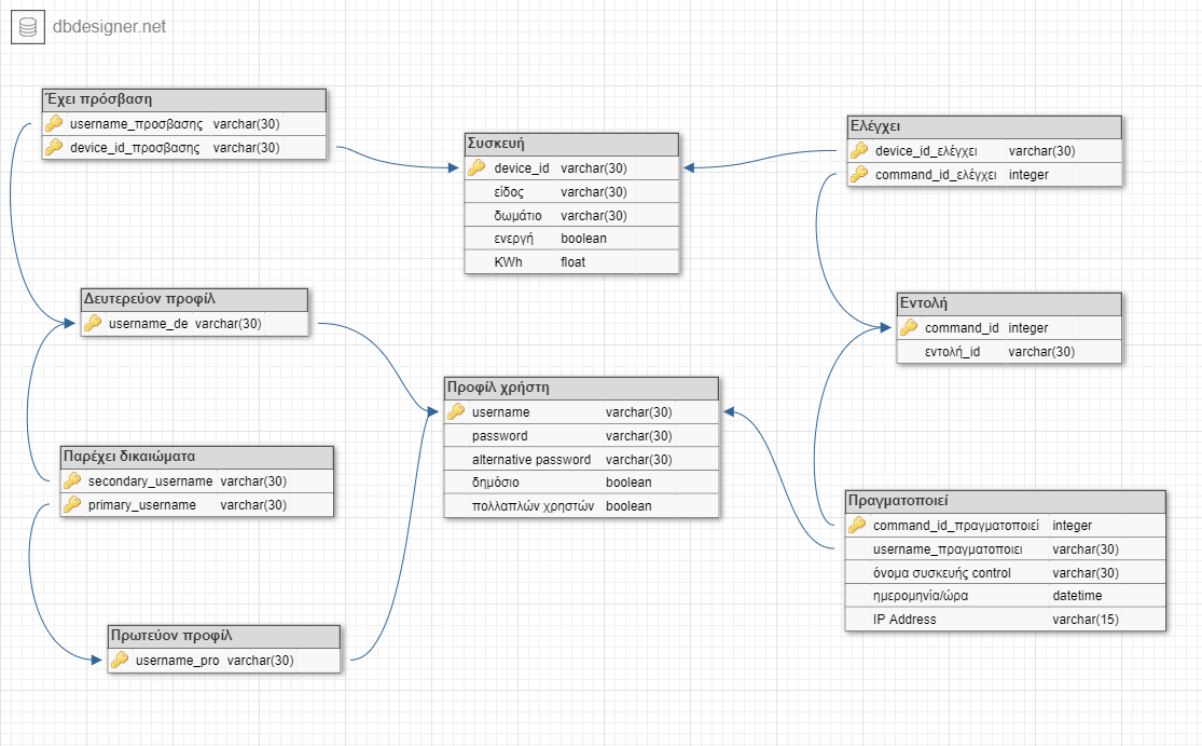

Fig. 7. Σχεσιακό μοντέλο

Μπορούμε πλέον μέσω του σχεσιακού μοντέλου να υλοποιήσουμε τη βάση δεδομένων και τους αντίστοιχους πίνακες. Αυτό αποτελεί το σχεσιακό κομμάτι της βάσης. Σημειώνουμε ότι οι οι 1:N συσχετίσεις θα μπορούσαν να αναπαράσταθουν χωρίς επιπλέον πίνακα. Παρόλαυτα η βάση είναι αρκετά μικρή ήδη και προορίζεται για αποθήκευση μικρού σχετικά μεγέθους δεδομένων, δε τίθεται δηλαδή θέμα απόδοσης. Έτσι επιλέξαμε να τους συμπεριλάβουμε για ευκολία στον χειρισμό των δεδομένων και καλύτερη κατανόηση του σχήματος.

2.2 Ανάγκη για NoSQL

Στην παραπάνω περιγραφή των οντοτήτων συσκεύη και εντολή δεν αναφέρθηκαν πουθενά το σύνολο παραμέτρων και εντολών που χαρακτηρίζει την κάθε συσκευή, ούτε οι παράμετροι της κάθε εντολής. Απλώς τους δώσαμε κάποια id αλλά δεν αναφέραμε τίποτα ουσιαστικό για τη δομή τους. Επιλέξαμε η υλοποιήση τους να μην ακολουθεί το σχεσιακό μοντέλο για τους εξής λόγους.

Όπως αναφέραμε η συσκευή είναι εξαιρετικά πολύπλοκη οντότητα, αυτός ήταν ο λόγος μάλιστα που ορίσαμε το πρότυπο των παραμέτροι + εντολές. Όπως διαπιστώνουμε όμως μια συσκευή μπορεί να είναι απλουστάτη όπως μια λάμπα ή ένας διακόπτης ή σύνθετη όπως ένα σύστημα συναγερμού. Μια λάμπα έχει δυο εντολές, άναψε και σβήσε και μοναδική της παράμετρος είναι το έαν είναι ενεργοποιημένη ή όχι. Ο συναγερμός από την άλλη έχεί πολλές παραμέτρους, πολλές εντολές και ή κάθε εντολή του μπορεί να επιδρά σε πολλές παραμέτρους του.

Παρόλες τις παραδοχές δηλαδή και την αφαίρεση εξακολουθούμε να μην έχουμε ένα πρότυπο συσκευής ικανοποιητικά αναπαραστίσιμο σε μια σχεσιακή βάση δεδομένων. Αυτός είναι και ο λόγος που καταφύγαμε σε πρότυπα μη σχεσιακής(NoSQL) βάσης δεδομένων.

Κύριο πλεονέκτημα των μη σχεσιακών βάσεων δεδομένων είναι ότι οι αντίστοιχες πλειάδες ενός αντίστοιχου πίνακα δε χρειάζεται να έχουν την ίδια δομή όπως σε μία σχεσιακή βάση δεδομένων. Κάτι που αμέσως τις καθιστά μια ελκυστική λύση στο πρόβλημα μας κύριο χαρακτηριστικό του οποίου είναι η ανομοιομορφία των διαφόρων συσκευών και εντολών.

Πιο συγκεκριμένα εξετάστηκε η χρήση της mongoDB, η οποία είναι και η noSQL βάση δεδομένων που διδάχθηκε στο μάθημα. Βασική έννοια αντί της πλειάδας είναι το document. Ένα document μπορεί να χαρακτηριστεί ως μια συλλογή 'πραγμάτων', τέτοια πράγματα μπορεί να έιναι αλφαριθμητικα, boolean κλπ αλλά και arrays ή άλλα documents. Στην περίπτωση ενός document συσκευής αύτα τα 'πράγματα' θα ήταν οι παράμετροι και οι εντολές της. Όλα τα σχετικά document σε μια τέτοια βάση συστήνουν το collection (αντίστοιχο του table στις SQL βάσεις).

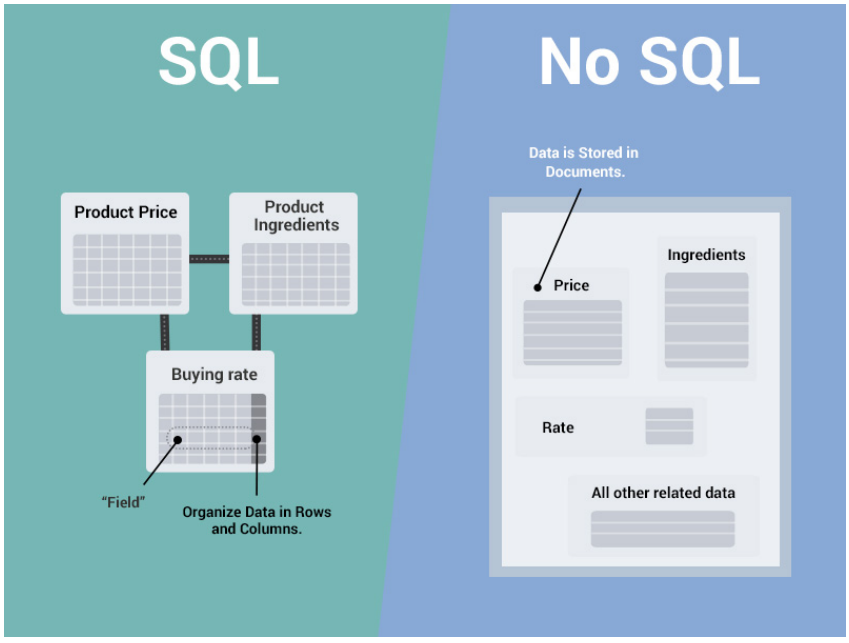

Fig. 8. SQL και NoSQL

Ορίστηκαν έτσι δύο collections, τα:

- appliances
- αρχείο εντολών

Το collection appliances περιέχει ένα document για κάθε συσκευή, τα document αυτά έχουν τη δομη:

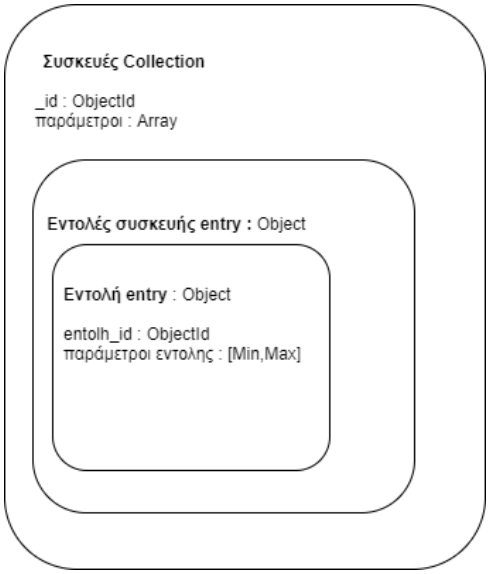

Fig. 9. Appliances document

Το document αυτό θα περιέχει το id τύπου ObjectId που αποτελεί και το αντίστοιχο πρωτεύον κλειδι. Τις παραμέτρους τύπου Array όπου θα περιέχονται οι παράμετροι τις συσκευής. Τις εντολές τύπου εμφωλευμένου document στο οποίο περιέχονται documents κλειδί των οποίων είναι το όνομα τις εντολής ενώ μέσα τους περιέχουν το id της εντολής τύπου ObjectId και ένα ακόμη εμφωλευμένο document το οποίο θα περιέχει ζεύγη παράμετρος : [Ελάχιστη τιμή παραμέτρου,Μέγιστη τιμή παραμέτρου]

Άρα η αναλυτική δομή σε αναπαράσταση key : value που είναι και η πιο συνηθισμένη για τα documents είναι :

```
{"_id":ObjectId
 "parametroi" : Array
 "entoles" : Object
    {"command_name" : Object
        {"entolh_id" : ObjectId
         "parametroi" : Object
            {"parameter name":[Parameter min value int32, Parameter max value int32]}
        }
    }
}
```

Το αρχείο εντολών είναι ένα collection στο οποίο θα αποθηκεύονται οι εντολές και οι παράμετροί τους όπως αυτές ορίστηκαν από τον χρήστη που τις πραγματοποίησε. Θα έχουν έτσι την παρακάτω δομή:

Το document αυτό θα περιέχει το id τύπου ObjectId που αποτελεί και το αντίστοιχο πρωτεύον κλειδι και τις παραμέτρους τύπου εμφωλευμένου document (object) όπου θα περιέχονται οι ζευγάρια όνομα παραμέτρου : τιμή παραμέτρου, όπως θα έχουν οριστεί από τον χρήστη.

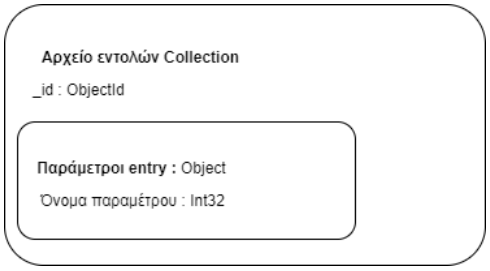

Fig. 10. Αρχείο εντολών document

Ή σε key : value μορφή:

```
{"_id":ObjectId
 "parametroi" : Object
    {"parameter name" : parameter value (Int32)}
}
```

Γίνεται έτσι φανερό ότι με κατάλληλα documents μπορούμε να αποθηκεύουμε οποιοδήποτε είδος συσκευής ανεξαρτήτως πολυπλοκότητας οι οποίες θα ακολουθούν όλες ένα συγκεκριμένο πρότυπο. Κάτι που όπως θα φανεί παρακάτω θα μας λύσει τα χέρια.

Προφανώς και η ίδια υλοποιήση θα μπορούσε να πραγματοποιηθεί και με βάση το σχεσιακό μοντέλο, κάτι το οποίο ήταν και η πρώτη σκέψη. Σύντομα όμως αντιληφθήκαμε ότι κάτι τέτοιο περιπλέκει τις αναζητήσεις μας, κάτι που στην πραγματικότητα δεν είναι απαραίτητα απαγορευτκό λόγω της φύσης της βάσης. Είναι αυτονόητο ότι σε ένα σπίτι δεν υπάρχουν και πολλές συσκευές, συνεπώς δε τίθονται ζητήματα απόδοσης. Παρόλαυτα επίλεξαμε τη συγκεκριμένη προσέγγιση μιας και είναι πολύ πιο intuitive και μας επιτρέπει ταυτόχρονα να εξασκηθούμε και να εργαστούμε και με κάποιο διαφορετικό πρότυπο βάσης δεδομένων.

Τέλος παρουσίαζουμε το τελικό διάγραμα οντοτήων συσχετίσεων που συμπεριλαμβάνει και τα δύο collection που σχολιάσαμε παραπάνω. Παρατηρούμε με τις κόκκινες γραμμές τις αναφορές στα διάφορα κλείδια των documents. Κάτι τέτοιο συνηθίζεται σε σχεσιακά διαγράμματα, αλλά απο τη στιγμή που όλο διάγραμμα προήλθε από δική μας πρωτοβουλία και δεν διέπεται από κάποιος αυστηρούς κανόνες και πρότυπα επιλέξαμε να τις συμπεριλάβουμε.

Βλέπουμε ότι για να μπορούμε να γνώριζουμε ποιο document αντιστοιχεί σε ποια συσκευή φροντίζουμε τόσο αυτό όσο και η οντότητα συσκεύη να έχουν το ίδιο πρωτεύον κλειδί. Ομοίως και για το αρχείο εντολών όπου κάθε document του πρέπει να έχει το ίδιο πρωτεύον κλειδί με την αντίστοιχη οντότητα. Τέλος θυμόμαστε ότι κάθε εντολή πρέπει να ανήκει σε κάποιο σύνολο διαθέσιμων εντολών μιας συσκευής, έτσι δικαιολογούμε και τη σύνδεση της εντολη id με την entolh id.

Ο τρόπος με τον οποίο επιτυγχάνεται αυτή η συνέπεια μεταξύ των δύο βάσεων θα αναλυθέι παρακάτω και βασίζεται στην ίδια την εφαρμογή και υλοποιείται μέσω του προγράμματος της ύστερα απο μεσολάβηση της γλώσσας προγραμματισμού Python.

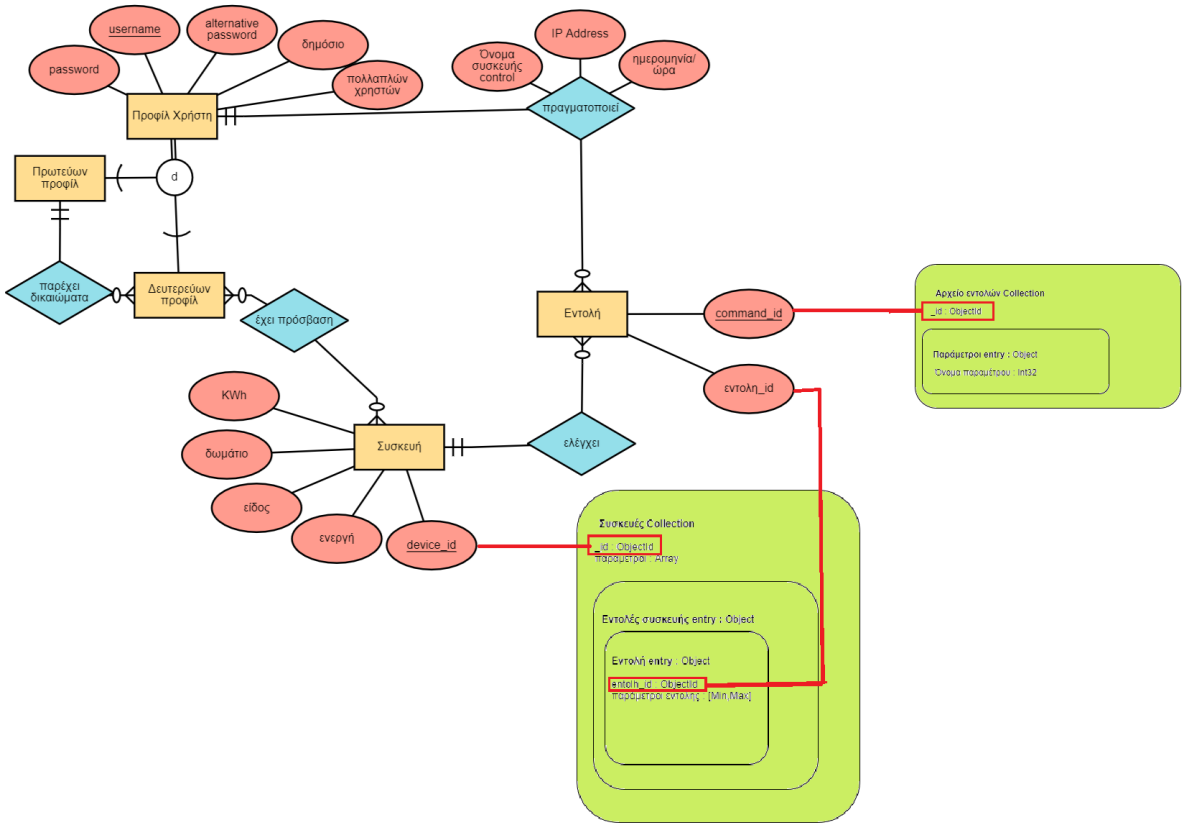

Fig. 11. Τελικό EER

2.3 SQLite και γλωσσα ορισμού δεδομένων

Έχουμε το σχήμα της βάσης και το σχεσιακό διάγραμμα οπότε μπορούμε πλέον εύκολα να μεταβούμε στην κατασκευή των πινάκων. Αρχικά η RDBMS που σκεφτήκαμε να χρησιμοποιήσουμε ήταν η MySQL, η οποία ήταν και η πρώτη που διδάχθηκε στο μάθημα και η πρώτη με την οποία προγραμματίσαμε. Σε μεταγενέστερα όμως εργαστήρια έιχαμε την ευκαιρία να δουλέψουμε και σε SQLite μέσω του περιβάλλος DB Browser.

Τα δύο αυτά RDBMS έχουν κάποιες διαφορές οι κυριότεροι λόγοι όμως για τους οποίους προτιμήθηκε η SQLite για τους εξής λόγους.

- Η SQLite δε χρειάζεται σέρβερ βάσης δεδομένων για να τρέξει, σε αντίθεση με τη MySQL.
- Μας άρεσε άρκετα και μας φάνηκε πολύ πιο εύχρηστο για τις ανάγκες της βάσης μας το περιβάλλον DB Browser.
- Πρόκειται για βάση μικρού μεγέθους, κάτω από φυσιολογική χρήση δε θα ξεπερνάει μέγεθος μερικών MB. Για βάσεις τέτοιου μεγέθους δεδομένων προτιμάται η SQLite.
- Η Python την οποία και χρησιμοποιήσαμε για την κατασκευή της εφαρμογής και για τα μεικτά queries έχει by default τη βιβλιοθήκη sqlite3, όποτε θεωρήσαμε ότι θα ύπαρχει πιο πλούσιο documentation σε σχέση με αντίστοιχες βιβλιοθήκες της MySQL και επιπλέον ή εφαρμογή θα έχει ένα requirement λιγότερο.

S.NO.	MySQL	SQLite
1.	Developed by Oracle on May 1995.	Developed By D. Richard Hipp on August 2000.
2.	MySQL is developed in C and C++ languages.	SQLite is developed only in C language.
3.	MySQL requires a database server for its functioning. Hence, it follows client/server architecture.	SQLite does not require a server to run. Hence, it is serverless.
4.	It can handle multiple connections simultaneously.	It can handle only one connection at a time.
5.	It is highly scalable and can handle a large volume of data very efficiently.	It can handle only small set of data if the volume of data increased its performance degrades.
6.	It requires large space in the memory for its functioning (approx 600 Mb).	It requires only some KBs of space as it is very lightweight approx (250Kb-300Kb).
7.	MySQL supports multiple user environment.	SQLite does not support multiple user environment.
8.	It also supports XML format.	It does not supports XML format.

Fig. 12. Διαφορές MySQL και SQLite

Θα δείξουμε τώρα τόν κώδικα SQL του κάθε table και θα γίνουν οι απαραίτητοι σχολιασμοί.

Προφίλ χρήστη :

```
CREATE TABLE ”Προφίλ χρήστη” (
”username” VARCHAR(30) NOT NULL,
”password” varchar(30) NOT NULL,
”alternative_password” varchar(30),
”δημόσιο” boolean,
”πολλαπλών χρηστών” boolean,
PRIMARY KEY(”username”)
);
```

Ακριβής αντιγραφή του λογικού σχεσιακού διαγράμματος, βλέπουμε ότι ο εναλλακτικός κωδικός μπορεί να είναι NULL για χρήστες που αρέσκονται μόνο σε έναν κωδικό. Είπαμε επίσης προηγουμένως πως τα δημόσια προφίλ δε θα απαιτούν κωδικό, μπορούμε πλέον να πούμε πως όπως έχει υλοποιηθεί και στην εφαρμογή παρακάτω τα δημόσια προφίλ έχουν κωδικό ένα κενό string.

Πρωτεύον προφίλ :
CREATE TABLE ”Πρωτεύον προφίλ” (
”username_pro” varchar(30) NOT NULL,
PRIMARY KEY(”username_pro”),
FOREIGN KEY(”username_pro”) REFERENCES ”Προφίλ χρήστη”(”username”) ON DELETE CASCADE ON UPDATE CASCADE
);

Το username ενός πρωτεύοντος προφίλ όπως φαίνεται και στο διάγραμμα αναφέρει το username του πίνακα προφίλ χρήστη. Για ευνόητους λόγους όταν διαγράφεται ένα προφίλ χρήστη θέλουμε να διαγράφεται η αντίστοιχη αναφορά του απο τον πίνακα πρωτεύοντος προφίλ. Το ίδιο θέλουμε να γίνεται και όταν πραγματοποιείται κάποια άλλαγη στο πεδίο username. Ορίσαμε έτσι αντίστοιχα τους περιορισμούς αναφορικής ακεραιότητας.

Δευτερεύον προφίλ :
CREATE TABLE ”Δευτερεύον προφίλ” (
”username_de” varchar(30) NOT NULL,
PRIMARY KEY(”username_de”),
FOREIGN KEY(”username_de”) REFERENCES ”Προφίλ χρήστη”(”username”) ON DELETE CASCADE ON UPDATE CASCADE
);

Ακριβώς ότι ισχύει και για τα πρωτεύοντα προφίλ ισχύει και για τα δευτερεύοντα.

Συσκευη :
TABLE ”Συσκευή” (
”device_id” varchar(30) NOT NULL,
”είδος” varchar(30) NOT NULL,
”δωμάτιο” varchar(30) NOT NULL,
”ενεργή” boolean NOT NULL,
”KWh” real NOT NULL,
PRIMARY KEY(”device_id”)
);

Και πάλι απλή αντιγραφή του σχεσιακού σχήματος, μόνη παρατήρηση έιναι πώς δεν επιτρέπουμε σε κανένα γνώρισμα να είναι NULL μιας και είναι όλες απαραίτητες και εύβρετες πληροφορίες.

Εντολή :
CREATE TABLE ”Εντολή” (
”command_id” integer NOT NULL,
”εντολή_id” varchar(30) NOT NULL,
PRIMARY KEY(”command_id” AUTOINCREMENT)
);

Βλέπουμε πώς το πρωτεύον κλειδί του πίνακα αυτού περιέχει την παράμετρο AUTOINCREMENT, αυτό σημαίνει ότι δε χρείαζεται να προσδιορίζουμε εμείς το κλειδί όταν εισάγουμε κάτι στον πίνακα αυτον, αλλά το φροντίζει η sql από μόνη της. Τα κλειδία αυτά είναι ακέραιοι αριθμοί μεγαλύτεροι του μηδέν και με κάθε εισαγωγή ενός νέου τέτοιου κλειδιού αποθηκεύεται η τιμή του σε έναν νέο πίνακα που δήμιουργεί αυτόματα η SQLite και ονομάζεται sqlite_sequence.

Ελέγχει :

CREATE TABLE ”Ελέγχει” (
”command_id_ελέγχει” integer NOT NULL,
”device_id_ελέγχει” varchar(30) NOT NULL,
PRIMARY KEY(”command_id_ελέγχει”),
FOREIGN KEY(”command_id_ελέγχει”) REFERENCES ”Εντολή”(”command_id”) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(”device_id_ελέγχει”) REFERENCES ”Συσκευή”(”device_id”) ON DELETE SET NULL ON UPDATE CASCADE
);

Άμα διαγράφεται μια εντολή από τη βάση δεδομένων θέλουμε να διαγράφεται από παντού, ενώ αν αλλάζει το κλείδι της το οποίο ορίζει αυτόματα η βάση (για οποιονδήποτε λόγο) θέλουμε να αλλάζει παντού. Από την άλλη εάν διαγράφεται μια συσκευή απο τη βάση δεδομένων δε θέλουμε να διαγράφεται κάθε εντολή που αφορά τη συσκευή αυτή, συνεπώς τίθεται η τιμή της σε NULL. Γένικα επιθυμούμε να έχουμε όλο το ιστορικό εντολών ακόμα και εάν το προφίλ χρήστη που την εκτέλεσε ή η συκευή διαγράφονται.

Έχει πρόσβαση :
CREATE TABLE ”Έχει πρόσβαση” (
”username_πρόσβασης” varchar(30) NOT NULL,
”device_id_πρόσβασης” varchar(30) NOT NULL,
PRIMARY KEY(”username_πρόσβασης”,”device_id_πρόσβασης”),
FOREIGN KEY(”username_πρόσβασης”) REFERENCES ”Δευτερεύον προφίλ”(”username_de”) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(”device_id_πρόσβασης”) REFERENCES ”Συσκευή”(”device_id”) ON DELETE CASCADE ON UPDATE CASCADE
);

Ο πίνακας έχει πρόσβαση περιέχει περιέχει τα δικαιώματα ενός δευτερεύοντος προφίλ, τις συσκευές δηλαδή στις οποίες έχει προσβασή, οπότε όταν είτε το προφίλ έιτε η συσκευή διαγράφεται ή ανανεώνεται, θέλουμε το ίδιο να συμβαίνει και στην αντίστοιχη πλειάδα..

Πραγματοποιεί :
CREATE TABLE IF NOT EXISTS ”Πραγματοποιεί” (
”username_πραγματοποιεί” varchar(30),
”command_id_πραγματοποιεί” int NOT NULL,
”όνομα συσκευής control” varchar(30),
”ημερομηνία/ώρα” datetime NOT NULL,
”IP_Address” varchar(15),
PRIMARY KEY(”command_id_πραγματοποιεί”),
FOREIGN KEY(”command_id_πραγματοποιεί”) REFERENCES ”Εντολή”(”command_id”) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(”username_πραγματοποιεί”) REFERENCES ”Προφίλ χρήστη”(”username”) ON DELETE SET NULL ON UPDATE CASCADE
);

Για τον πίνακα αυτό βλέπουμε ότι επιτρέπεται τα γνωρίσματα όνομα συσκευής control και IP_Address να είναι NULL, αυτό οφείλεται στο ότι ίσως σε μία μελλοντική βελτίωση της εφαρμογής να υπάρχει η δυνατότητα να αποθηκεύονται και όσες εντολές εκτελούνται άμεσα στη συσκευή και όχι μόνο όσες εκτελούνται μέσω της εφαρμογής. Ως προς τους περιορισμούς αναφορικής ακεραιότητας θέλουμε όταν διαγράφεται η εντολή να

διαγράφεται και ολόκληρη πλειάδα ενώ όταν διαγράφεται ένας χρήστης τίθεται το αντίστοιχο πεδίο σε NULL για τους ίδιος λόγους με πάνω.

Παρέχει δικαιώματα :
CREATE TABLE ”Παρέχει δικαιώματα” (
”primary_username” varchar(30) ,
”secondary_username” varchar(30) NOT NULL,
PRIMARY KEY(”primary_username”,”secondary_username”),
FOREIGN KEY(”primary_username”) REFERENCES ”Πρωτεύον προφίλ”(”username_pro”)
ON DELETE SET NULL ON UPDATE CASCADE,
FOREIGN KEY(”secondary_username”) REFERENCES ”Δευτερεύον προφίλ”(”username_de”)
ON DELETE CASCADE ON UPDATE CASCADE
);

Ο πίνακας αυτός περιέχει τα ζευγάρια επιβλέποντος πρωτεύοντος προφίλ και επιβλεπόμενου δευτερεύοντος προφίλ. Όταν διαγράφεται ένα δευτερεύον προφίλ θέλουμε προφανώς να διαγράφεται η αντίστοιχη πλειάδα από τον πίνακα αυτό, χωρίς δευτερεύον προφίλ δεν υπάρχουν δικαιώματα. Η αντίθετη περίπτωση όμως έχει περισσότερο ενδιαφέρον. Εάν διαγραφεί ένα πρωτεύον προφίλ τα αντίστοιχα επίβλεπόμενα δευτερεύοντα προφίλ θα μείνουν χωρίς επιβλέποντα. Η λύση που προτείναμε εμείς είναι να δίνονται τα δικαιώματα επίβλεψης σε ένα άλλο πρωτεύον προφίλ (το πρώτο πρωτεύον προφίλ του πίνακα Πρωτεύον προφίλ), σε περίπτωση που δεν υπάρχει άλλο πρωτεύον προφίλ τοτέ θα διαγράφονται μαζί του και όλα τα προφίλ που επιβλέπει. Αυτό δεν μπορεί να οριστεί κάτα την κατασκευή των πινάκων (ή τουλάχιστον δεν καταφέραμε να βρούμε τρόπο), για την ώρα στους περιορισμούς αναφορικής ακεραιότητας θέτουμε το πεδίο primary_username σε NULL σε περίπτωση διαγραφής. Στην εφαρμογή όμως όπως θα δείξουμε παρακάτω υπάρχει η επιλογή να διαγράψουμε κάποιο προφίλ, εκεί θα ξαναναφερθούμε λοιπόν στο πως αντιμετωπίζεται η περίπτωση αυτή.

Όσο για το mongoDB κομμάτι της βάσης, δε γίνεται εκ των πρωτέρων ο ορισμός της μορφής των documents, παρόλο που εμείς ορίσαμε ένα συγκεκριμένο πρότυπο που επιβάλουμε στα document αυτά. Ο λόγος υπενθυμίζουμε ότι είναι το ότι η mongoDB αφορά unstructured data ,συνεπώς ο εκ των προτέρων ορισμός της μορφής των document θα αναιρούσε τη βασικότερη λειτουργία της.

Για τα id των διαφόρων documents είδαμε επίσης στο σχ.11 πως επικρατεί μια σχέση αντίστοιχη των ξένων κλειδιών. Η σχέση αυτή στη λογική της SQL θα μπορούσε να περιγραφεί ως πχ **appliances._id REFERENCES ”Συσκευή”(”device_id”).** Προφανώς κάτι τέτοιο δε γίνεται, συνεπώς φροντίζουμε οι ίδιοι σε προγραμματιστικό επίπεδο να διατηρείται η συνέπεια αυτή μεταξύ των δύο βάσεων.

3 ΕΦΑΡΜΟΓΗ ΚΑΙ ΑΞΙΟΠΟΙΗΣΗ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

3.1 Προδιαγραφές και σχεδιασμός μιας εφαρμογής Smart Home

Όπως έχουμε αναφέρει επανειλημμένα σκοπός της βάσης είναι να αποθηκευόνται δεδομένα τα οποία θα αξιοποιούνται από μια εφαρμογή. Στην περίπτωση μας θα θέλαμε ο χρήστης να είναι σε θέση τόσο να χρησιμοποιεί τις διάφορες συσκευές του σπιτιού όσο και να έχει εποπτεία τών δευτερευόντων προφίλ, των συσκευών και τον προηγούμενων εντολών.

Επιπλέον επιθυμούμε η συσκευή μας να επιτρεπεί τη δημιουργία νέων προφίλ, τη διαγραφή παλιών και την προσαρμογή των επιλογών ανάλογα με τα δικαιώματα πρόσβασης του χρήστη. Όλα τα παραπάνω αποτελούν και τις προδιαγραφές πράξεων (δοσοληψίες) με βάση τις οποίες θα βασιστεί ο σχεδιασμός της εφαρμογής. Οι βασικές λειτουργίες που θέλουμε η εφαρμογή μας να είναι σε θέση να πραγματοποιεί είναι οι εξής :

- Δημιουργία νέου προφίλ

- Δημιουργία είτε πρωτεύοντος προφίλ, είτε δευτερεύοντος,ενώ για τη δημιουργία δευτερεύοντος προφίλ πρέπει να είναι δυνατή η επιλογή του επιβλέποντος προφίλ.
- Διαγραφή κάποιου προφίλ και σωστή μεταφορά των δικαιωμάτων επίβλεψης σε περίπτωση διαγραφής πρωτεύοντος προφίλ.
- Επιλογή προφίλ και έλεγχος κωδικού πρόσβασης.
- Παράβλεψη της εισαγωγής κωδικού πρόσβασης για δημόσια προφίλ.
- Επιλογή συσκευής προς χρήση.
- Περιορισμός των συσκευών προς χρήση μόνο σε αυτές με δικαίωμα πρόσβασης για τα δευτερεύοντα προφίλ.
- Επιλογή εντολής από το σύνολο εντολών της συσκευής.
- Εαν η συσκευή είναι η κλειστή η μόνη εντολή που θα εμφανίζεται θα είναι αυτη της ενεργοποίησης, αλλιώς θα είναι όλες εκτός αυτής.
- Εισαγωγή παραμέτρων εντολής.
- Δυνατότητα τα πρωτεύοντα προφίλ να μπορούν να βλέπουν σε ποιές συσκευές έχουν πρόσβαση τα δευτερεύοντα προφίλ που επιβλέπουν και να μπορούν να τροποποιούν τα δικαιώματα πρόσβασης τους.
- Δυνατότητα προβολής του ιστορικού των εντολών.
- Δυνατότητα προβολής της συνολικής κατανάλωσης ισχύος της κάθε συσκευής.

Επιπλέον θέλουμε όλες αυτές ο λειτουργίες να επικοινώνουν σωστά, να μη δημιουργούνται ασυνέπειες και πάνω από όλα το ίδιο το περιβάλλον της εφαρμογής να είναι χρηστικό και να θυμίζει, όσο είναι δυνατόν και όσο το επιτρέπουν οι παραδοχές που έχουν γίνει, αντίστοιχες εφαρμογές του εμπορίου.

Έτσι θα μπορέσουμε να αξιολογήσουμε σωστά τη βάση δεδομένων, μια επαρκής βάση δεδομένων θα είναι σε θέση να μπορεί να πραγματοποιεί αυτές τουλάχιστον τις λειτουργίες. Θα μπορέσουμε επίσης με εύκολο τρόπο μέσω της εφαρμογής αυτής να γεμίσουμε με δεδομένα τη βάση μας με τρόπο που θυμίζει παιχνίδι, αυστηρά και αποφεύγοντας σφάλματα που θα προέκυπταν από άλλους τρόπους συλλογης δεδομένων.

Παρακάτω βλέπουμε ένα τύπου flowchart/application-FSM, το οποίο περισσότερο διαισθητική σημασία έχει και δεν είναι αυστηρά δομημένο. Μας δίνει όμως μια καλή ιδέα της συμπεριφοράς που επιθυμούμε να έχει η εφαρμογή μας. Με κόκκινο δηλώνονται τα διάφορα panel της εφαρμογής μας, ένα είδος τρέχουσας κατάστασης δηλαδή. Με πράσινο οι είσοδοι του χρήστη που μπορούν να έχουν την μορφή κειμένου, κουμπίων και οποιουδήποτε πρακτικά widget της εφαρμογης.

3.2 Αξιόλογα σημεία του προγράμματος της εφαρμογής

Αρχικά θα γίνει μια σύντομη περιγραφή για το γραφικό κομμάτι της εφαρμογής. Χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python ενώ για το γραφικό κομμάτι η βιβλιοθήκη PyQt5.

Η γραφική διεπαφή βασίστηκε στο stacked layout widget, αυτό που μας προσφέρει αυτό το widget ουσιαστικά είναι η δυνατότητα να κατασκευάζουμε το κάθε layout ξεχωριστά, τα δίαφορα κουμπία, text edits κλπ που απαρτίζουν ένα παράθυρο δηλαδή, και να τα στοιβάζουμε το ένα πάνω στο άλλο. Με αποτέλεσμα να είμαστε σε θέση να αλλάζουμε από το ένα layout στο άλλο χωρίς να χρειάζεται να καταστρέφουμε το τρέχον layout και να το κατασκευάζουμε από την αρχή. Κάθε κόκκινος κύκλος στο παραπάνω σχήμα αντιπροσωπεύει ένα τέτοιο layout και κάθε τέτοιο layout αντιπροσωπεύει μια λειτουργία της εφαρμογής.

Θα συζητήσουμε τώρα ένα ένα τα σημαντικά κομμάτια της προγραμματιστικής υλοποίησης.

Homepage

Το homepage θα περιλαμβάνει ένα κουμπί create new profile μέσω του οποίου μεταβαίνουμε στη σελίδα δημιουργίας νέου προφίλ και ένα κουμπί για κάθε διαθέσιμο προφίλ.

Για τη δημιουργία των κουμπιών των προφίλ γίνεται το παρακάτω απλούστατο query:

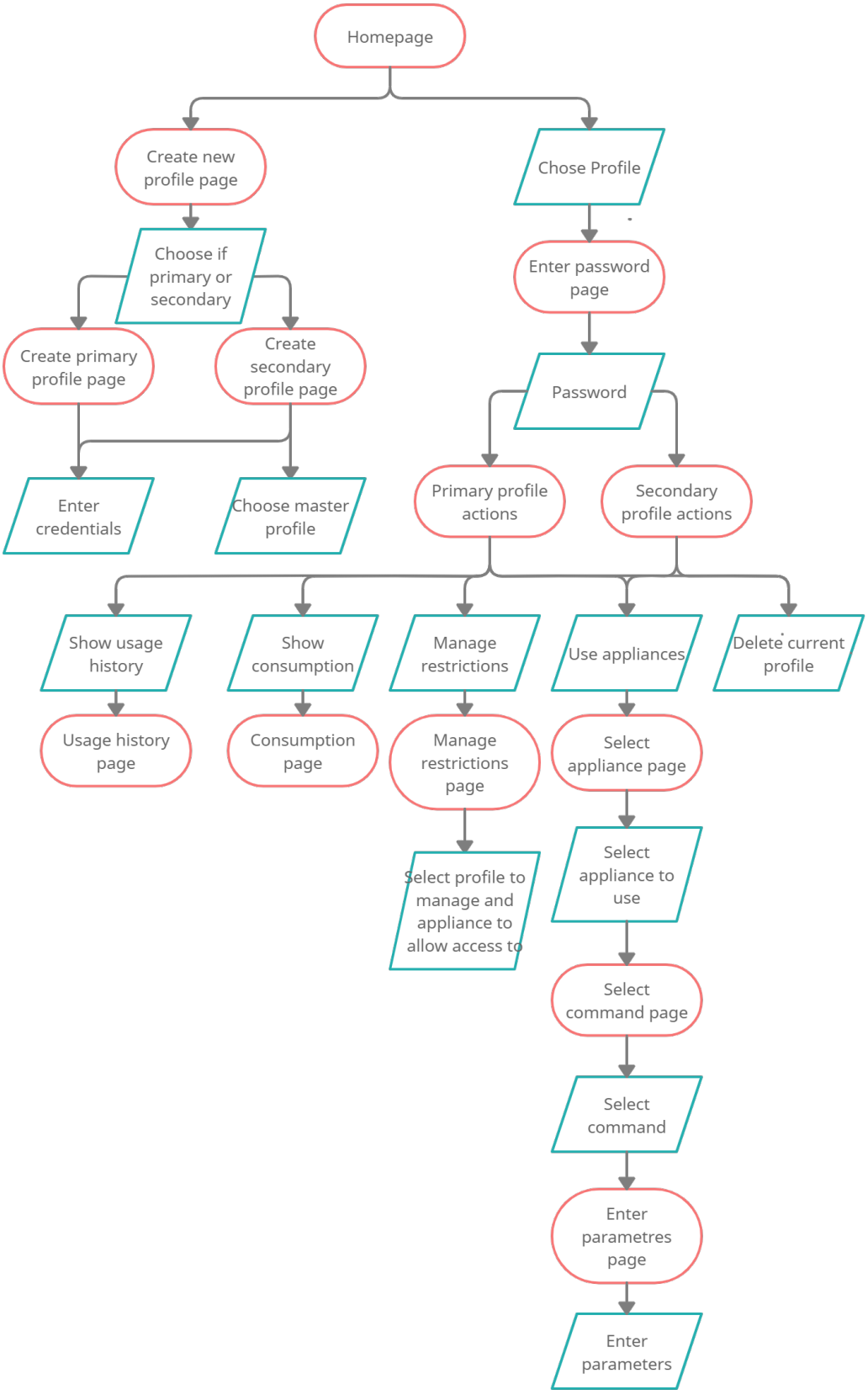

Fig. 13. Flowchart εφαρμογής

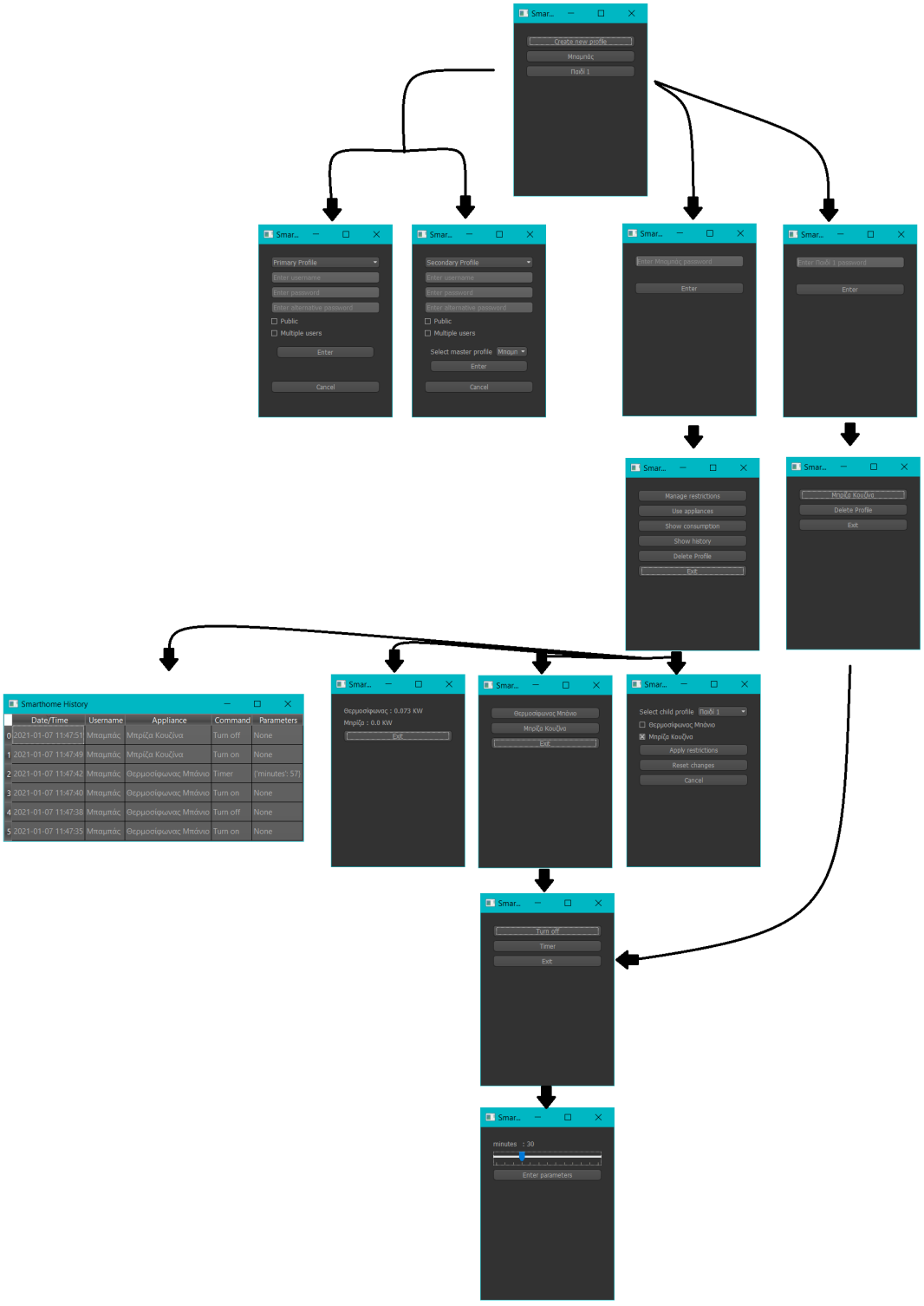

Fig. 14. Ροή τυπικής χρήσης της εφαρμογής

```
SELECT * FROM 'Προφίλ_χρήστη';
```

Μας επιστρέφονται έτσι τα δεδομένα του κάθε χρήστη και μπορούμε να φτίαξουμε ένα κουμπί με το username κάθε χρήστη.

Create new profile page

Πατώντας το κουμπί create new profile του homepage μεταφερόμαστε στη σελίδα κατασκευής νέου προφίλ. Η σελίδα αυτή υλοποιείται ως ένα δεύτερο stacked layout, ο λόγος είναι πως τόσο για την κατασκευή πρωτεύοντος όσο και για την κατασκευή δευτερεύοντος απαιτούνται κάποιο συγκεκριμένα πράγματα (username, password κλπ). Συνεπώς και τα widget που αφορούν τις εισόδους αυτές μπορούν να είναι κοινά. Για τη κατασκεύη ενός δευτερεύοντος προφίλ όμως απαιτείται και η επιλογή πρωτεύοντος επιβέποντος προφίλ. Τόσο η επιλογή του τίνος προφίλ θέλουμε να δημιουργήσουμε όσο και η επιλογή του επιβλέποντα γίνοντα μέσω ενός Combo Box.

Το κουμπί Enter εξετάζει αν ο χρήστης συμπλήρωσε κατάλληλα τα στοιχεία, εμφανίζει τα κατάλληλα μηνύματα σε περίπτωση λάθους και εάν όλα είναι εντάξει δημιουργεί το νέο προφίλ. Τα σχετιζόμενα query είναι τα:

```
"'INSERT INTO Προφίλ_χρήστη(username,password,alternative_password,δημόσιο,
πολλαπλών_χρηστών)

VALUES(?,?,?,?,?)"',(self.new_username.text(),self.new_password.text(),
None if self.new_alternative_password.text()==" else
self.new_alternative_password.text(),self.isPublic.isChecked(),
self.isMulti.isChecked())

INSERT INTO Πρωτεύον_προφίλ(username_pro)
VALUES('{self.new_username.text()}')
```

Η χρήσης της SQLite μέσω της βιβλιοθήκης sqlite3 γίνεται ορίζοντας το αντίστοιχο query ως string. Στην περίπτωση των insert μπορούμε να ορίσουμε τις εισαγώμενες τιμές απευθείας στο string με χρήση των f-string της Python (πρώτη περίπτωση). Ή μπορούμε να τίς κρατήσουμε με ερωτηματικά '?' και να ορίσουμε τις εισαγώμενες τιμές ως tuple (δεύτερη περίπτωση).

Οι εισαγώμενες τιμές προέρχονται από χρήση των widget και είναι διαφόρων ειδών, πχ η self.new_username.text() επιστρέφει το κείμενο ενός text edit ενώ self.isMulti.isChecked() επιστρέφει True εάν το αντίστοιχο Check Box είναι checked, αλλιώς επιστρέφει False.

Για τα δευτερεύοντα προφίλ πρέπει να γίνεται αποθήκευση και των αντίστοιχων δικαιωμάτων τους, τα query είναι :

```
"'INSERT INTO Προφίλ_χρήστη(username,password,alternative_password,δημόσιο,
πολλαπλών_χρηστών)
VALUES(?,?,?,?,?)"',(self.new_username.text(),self.new_password.text(),
None if self.new_alternative_password.text()==" else
self.new_alternative_password.text(),self.isPublic.isChecked(),
self.isMulti.isChecked()))
```

```
INSERT INTO Δευτερεύον_προφίλ(username_de)
VALUES('{self.new_username.text()}')

INSERT INTO Παρέχει_δικαιώματα(primary_username,secondary_username)
VALUES('{self.master_profiles.currentText()}','{self.new_username.text()}')
```

Πρώτα εισάγουμε στον πίνακα προφίλ χρήστη μιας και και οι επόμενοι πίνακες περιέχουν κλειδία τα οποία αναφέρουν το πρωτεύον κλειδί του. Αυτό βέβαια δεν είναι τόσο αυστηρό μιας και διαπιστώσαμε ότι οι περιορίσμοι αυτοί στην sqlite3 είναι πιο ρευστοί, όπως θα δείξουμε και παρακάτω στη διαγραφή χρήστη. Παρόλαυτα για λόγους καλής πρακτικής οι εισαγωγές γίνονται με τη σωστή σειρά.

Enter password page

Ανάλογα με το ποιο προφίλ επιλέξουμε πατώντας το αντίστοιχο κουμπί μεταφερόμαστε στη σελίδα εισαγωγής κωδικού πρόσβασης. Ο χρήστης εισάγει τον κωδίκο, εάν είναι σωστός τότε έαν το προφίλ είναι πρωτεύον μεταφέρεται στη σελίδα ενεργειών πρωτεύοντος προφίλ, εάν είναι δευτερεύον μεταφέρεται στη σελίδα ενεργειών δευτερεύοντος προφίλ. Η πρώτη όπως φάνηκε και στα παραπάνω σχήματα περιλαμβάνει πολλές ενέργειες ενώ η δεύτερη αφορά μόνο τη χρήση συσκευών στις οποίες έχει πρόσβαση το προφίλ. Τα query είναι :

```
SELECT password,alternative_password,δημόσιο FROM 'Προφίλ_χρήστη'
WHERE username = '{self.profile}' ;

SELECT * FROM 'Πρωτεύον_προφίλ' WHERE username_pro = '{self.profile}' ;

SELECT Έχει_πρόσβαση.device_id_πρόσβασης,είδος,δωμάτιο,ενεργή from Έχει_πρόσβαση
join Συσκευή on Συσκευή.device_id=Έχει_πρόσβαση.device_id_πρόσβασης
and username_πρόσβασης='{self.profile}'
```

Το πρώτο βρίσκει τους έγκυρους κωδικούς με τους οποίους μετά θα συγκρίνει με την είσοδο του χρήστη.

Το δεύτερο εξετάζει αν το προφίλ είναι πρωτεύον ή δευτερεύον.

Το τρίτο αφορά δευτερεύοντα προφίλ και μέσω συνένωσης των πινάκων Συσκευή και Έχει πρόσβαση βρίσκει τις συσκευές στις οποίες έχει πρόσβαση.

User options page

Πρόκειται για δύο διαφορετικές σελίδες, η μία αφορά τις επιλογές των πρωτεύοντων προφίλ και περιέχει ένα κουμπί για κάθε ενέργεια. Οι επιλογές που μπορεί να έχει ένα πρωτεύον προφίλ είναι:

- Επίβλεψη δευτερεύοντων προφίλ.
- Χρήση κάποιας συσκευής.
- Προβολή ιστορικού εντολών.
- Προβολή στοιχείων κατανάλωσης.
- Διαγραφή τρέχοντος προφίλ.

Οι επιλογές που μπορεί να έχει ένα δευτερεύον προφίλ είναι:

- Χρήση κάποιας συσκευής.
- Διαγραφή τρέχοντος προφίλ.

Έτσι η σελίδα ενεργείων πρωτεύοντος προφίλ θα περιέχει κουμπιά για την κάθε ενέργεια. Ενώ για τα δευτερεύοντα προφίλ μπορούμε αντί να έχουμε κουμπί χρήση κάποιας συσκευής που θα μας μετέφερε σε

καινούρια σελίδα, να έχουμε για απλότητα όλες τις διαθέσιμες συσκευές στη σελίδα αυτή. Οι συσκευές στις οποίες θα έχει πρόσβαση έχουν οριστεί όπως είδαμε παραπάνω κάτα την εισαγωγή του κωδικού.

Manage restrictions page

Αφού πατήσουμε το αντίστοιχο κουμπί μεταφερόμαστε στη σελίδα επίβλεψης δευτερευόντων προφίλ. Δίνεται η δυνατότητα να δώσουμε πρόσβαση σε μια συσκευή ή να την απαγορεύσουμε. Αρχικά μέσω ενός Combo Box επιλέγουμε το επιβλεπόμενο προφίλ απο το σύνολο των προφίλ για τα οποία ο χρήστης έχει δικαιώμα επίβλεψης. Υπάρχει ένα Check Box για κάθε συσκευή, εάν το προφίλ έχει πρόσβαση σε μια συσκεύη το αντίστοιχο Check Box θα είναι συμπληρωμένο, αλλιώς θα έιναι ασυμπλήρωτο. Μπορεί λοιπόν το επιβλέπων προφίλ να κάνει check ή uncheck τα Check Box και να προσδιορίσει έτσι τα δικαιώματα πρόσβασης. Το κουμπί Enter αποθηκεύει τα νέα δικαιώματα εξετάζοντας ένα ένα τα Check Box και συγκρίνοντας την τρέχουσα κατάσταση τους με το εάν προηγουμένως το προφίλ είχε πρόσβαση στην αντίστοιχη συσκευή, ενώ το κουμπι set default τα επαναφέρει. Τα σχετιζόμενα query είναι :

```
SELECT secondary_username FROM 'Παρέχει_δικαιώματα'
WHERE primary_username='{self.profile}';

SELECT * FROM Έχει_πρόσβαση
WHERE username_πρόσβασης='{self.child_profiles.currentText()}' and
device_id_πρόσβασης='{self.appliances[i-2][0]}';

DELETE FROM Έχει_πρόσβαση
WHERE username_πρόσβασης='{self.child_profiles.currentText()}' and
device_id_πρόσβασης='{self.appliances[i-2][0]}';

INSERT INTO Έχει_πρόσβαση(username_πρόσβασης,device_id)
VALUES('{self.child_profiles.currentText()}','{self.appliances[i-2][0]}')
```

Με το πρώτο query βρίσκουμε για το δεδομένο πρωτεύον προφίλ τα προφίλ που επιβλέπει, με τα οποία θα γεμίσουμε μετά το Combo Box.

Με το δεύτερο βρίσκουμε αν ένα δεδομένο δευτερεύον προφίλ έχει πρόσβαση σε μια δεδομένη συσκευή.

Με το τρίτο διαγράφουμε δικαίωμα πρόσβασης, διαγράφοντας από τον πίνακα Έχει πρόσβαση το ζευγάρι δευτερεύοντος username, συσκευής. Ενώ με το τέταρτο εισάγουμε δικαίωμα πρόσβασης.

Select appliance page

Μας μεταφέρει στη σελίδα επιλογής συσκευής, πρόκειται για πρωτεύον προφίλ και περιλαμβάνει ένα κουμπί για κάθε συσκευή. Το μόνο query που πραγματοποιείται είναι αυτό που βρίσκει όλες τις συσκευές.

```
"SELECT * FROM Συσκευή"
```

Select command page

Έχοντας επιλέξει τη συσκευή που θέλουμε να χρησιμοποιήσουμε μεταφερόμαστε στη σελίδα που περιλαμβάνει τις διαθέσιμες εντολές της συσκευής αυτής. Κάθε κουμπί αφορά και μια εντολή. Θέλουμε όταν η συσκευή είναι κλειστή ή μόνη διαθέσιμη εντολή να είναι ή Turn on ενώ όταν είναι ανοιχτή να είναι όλες οι άλλες διαθέσιμες. Αυτό εισάγει και μια καινούρια παραδοχή, το ότι κάθε συσκεύη πρέπει να περιλαμβάμει τις εντολές Turn on και Turn off στο σύνολο εντολών της. Θυμόμαστε όμως ότι το σύνολο εντολών μιας συσκεύης είναι αποθηκευμένο στο NoSQL κομμάτι της βάσης μας, συνεπώς δε μας αρκούν απλά queries για την ανάκτηση τους. Θα πρέπει

κάπως να είμαστε σε θέση να πραγματοποιούμε μεικτά query. Αυτό επιτυγχάνεται εύκολα μέσω της Python και των βιβλιοθηκων sqlite3 που χρησιμοποιούμε μέχρι τώρα και της pymongo.

Θυμόμαστε ότι η δομή των document θυμίζει αυτή των JSON και τη δομή του λεξικού της Python, κάτι που καθιστά την εκτέλεση mongo queries μέσω Python πανεύκολη και πολλη φυσική διαδικασία. Μπορούμε πολύ απλά να ανακτούμε το document που θέλουμε και στη συνέχεια να το χρησιμοποήσουμε όπως θα χρησιμοποιούσαμε ένα λεξικό, με όλες τις μεθόδους και συναρτήσεις που το συνοδεύουν.

```
self.entoles = db['appliances'].find({'_id':ObjectId(self.appliance_id)})[0]['entoles']
```

Όπου db = client['smarthome'] έιναι ή mongoDB βάση δεδομένων μας που περιέχει τα collections appliances και αρχείο εντολών, ενώ client είναι ο σέρβερ στον οποίο είμαστε συνδεδεμένοι, στην περίπτωση μας local. Βλέπουμε πως η συνάρτηση find λειτουργεί με ακριβώς τον ίδιο τρόπο και μας επιστρέφει τα documents με το ζητούμενο _id. Το _id αυτό είχαμε πει ότι για λόγους συνέπειας είναι ίδιο με το id της συσκεύης. Στον πίνακα Συσκευή όμως η δομή του κλειδιού είναι varcar(30) (string) ενώ στην mongo είναι τύπου ObjectId, έτσι για να γίνει σωστά η αναζήτηση προηγείται η μετατροπή του κλείδιου σε ObjectId με τη συνώνυμη συνάρτηση που περιέχεται στη βιβλιοθήκη bson που περιλαμβάνεται στη default έκδοση της Python.

Το query αυτό επιστρέφει ένα document, το οποίο η pymongo μεταφράζει σε λεξικό. Συνεπώς βρίσκουμε πολύ απλά τις εντολές της συσκευής παίρνοντας τη τιμή του κλειδιού 'entoles' που έχουμε προκαθορίσει ότι θα περίεχει το σύνολο εντολών της κάθε συσκευής. Το αν η συσκευή είναι ενεργοποιημένη η όχι υπάρχει ήδη σαν πληροφορία μιας και ξέρουμε τι κουμπί πατήθηκε για να φτάσουμε στη συγκεκριμένη σελίδα.

Insert parameters page

Αφού επιλέξουμε την εντολή μεταφερόμαστε στη σελίδα εισαγωγής των τιμών των παραμέτρων της. Χρησιμοποιουνται Sliders ως widget είσοδου, ελάχιστη τιμή των οποίων είναι ο πρώτος όρος του array που αντίστοιχει στην κάθε παράμετρο μιας εντολής και μέγιστη ο δεύτερος. Αφού ο χρήστης θέσει τις επιθυμητές τιμές στα sliders θα πατάει το κουμπί Enter και θα αποθηκεύονται οι εντολές στις δύο βάσεις. Τα query και οι εντολές Python για αναζητήσεις στη mongo βάση είναι:

```
entoli_name = list(self.entoles.keys())[cntr]
entoli = self.entoles[entoli_name]
minValue = entoli['parametroi'][i][0]
maxValue = entoli['parametroi'][i][1]
```

Από το σύνολο εντολών της συσκευής που είχαμε βρει προηγουμένως βρίσκουμε την εντολή αυτή που επέλεξε ο χρήστης. Άρα με βάση αυτή τοποθετούνται τα widget στη σελίδα εισόδου παραμέτρων. Αν η εντολή ήταν turn on ή turn off πρέπει να αλλάξη η τιμή ενεργή του πίνακα Συσκευή.

```
UPDATE Συσκευή SET ενεργή=true
WHERE device_id = '{self.appliance_id}'

UPDATE Συσκευή SET ενεργή=false
WHERE device_id = '{self.appliance_id}'
```

Για να αποθηκεύσουμε τις εντολές θέλουμε 2 πράγματα, την εντολή id που είναι το id της εντολής στο document της mongo, και το command id που είναι το auto incremental key του πίνακα Εντολή. Αρχικά θα γίνει η αποθήκευση της εντολής στην SQLite βάση. Τα query είναι :

```
_id = str(entoli['entolh_id'])

INSERT INTO Εντολή(εντολή_id)
VALUES('{_id}')

INSERT INTO Ελέγχει(command_id_ελέγχει,device_id_ελέγχει)
VALUES(last_insert_rowid(),'{self.appliance_id}')

INSERT INTO Πραγματοποιεί(username_πραγματοποιεί,command_id_πραγματοποιεί,
όνομα_συσκευής_control,ημερομηνία_ώρα,IP_Address)
VALUES('{self.profile}',last_insert_rowid(),'{socket.gethostname()}',
CURRENT_TIMESTAMP,'{socket.gethostbyname(socket.gethostname())}')
```

Αρχικά παίρνουμε το id της εντολής απο το document της συσκευής, και εισάγουμε στη βάση την εντολή, το κλειδί είναι auto incremental και γιαυτό δεν το γράφουμε.

Με την sql εντολή last_insert_rowid() παίρνουμε το κλειδί της τελευταίας εντολής και εισάγουμε στον πίνακα ελέγχει.

Τέλος με τη βιβλιοθήκη socket που υπάρχει στο default πακέτο της Python βρίσκουμε το όνομα της συσκευής του χρήστη, τη διεύθυνση IP του και κάνουμε την εισαγωγή στον πίνακα Πραγματοποιεί.

Για την εισαγωγή στο αρχείο εντολών της mongo βάσης.

```
parametroi.update({self.page11Layout.itemAt(i).widget().text():
self.page11Layout.itemAt(i+2).widget().value()})

db.arxeio_entolwn.insert_one({"_id":ObjectId('0'*(24-len(self.entoli_id))
+self.entoli_id),"parametroi":parametroi})
```

Από τα αντίστοιχα widget κάθε παραμέτρου παίρνουμε το όνομα της παραμέτρου, την τιμή της και τις εισάγουμε πολυ απλά σε ένα λεξικό που θα περιέχει τα ζευγάρια αυτά για κάθε παράμετρο. Στη συνέχεια εισάγουμε στο αρχείο εντολών και το λεξικό παραμέτρων αφού γίνει προσαύξηση της εντολής id για να γίνει μήκους 24 δεκαεξαδικών που είναι και προυπόθεση για να τη μετατρέψουμε από string σε ObjectId.

Show history page

Αρχικά θα ανοίξουμε μια παρένθεση. Αυτό ήταν και το κομμάτι που προγραμματιστικά μας δυσκόλεψε περισσότερο. Το ιστορικό θέλαμε να το υλοποιήσουμε με τη χρήση του widget QTableView. Μέχρι τώρα μέσω του stacked layer καταφέρναμε να έχουμε σταθερό μέγεθος παραθύρου που ήταν και το επιθυμητό μιας και θεωρητικά όπως και οι περισσότερες εμπορικές εφαρμογές θα θέλαμε να προσομοιώσουμε εφαρμογή κινητού, στις οποίες υπάρχουν περιορισμοί για το μέγεθος παραθύρου. Παρόλαυτα για τις ανάγκες του πρότζεκτ την αφήσαμε πίσω αυτή τη προδιαγραφή και επιλέξαμε το παράθυρο του ιστορικού να έχει μεγαλύτερο μέγεθος.

Αυτό που μας δυσκόλεψε είναι, πως η δομή του stacked layer θεωρεί ως συνολικό μέγεθος παραθύρου το μέγεθος της μεγαλύτερης επι μέρους σελίδας. Συνεπώς το μέγεθος της σελίδας ιστορικού τράβαγε πάνω το μέγεθος όλης της εφαρμογής. Δε βρέθηκε ικανοποιητικός τρόπος να αποφευχθεί αυτό, δοκιμάστηκαν εντολές που αφορούσαν restrictions στο size adjustment αλλά χωρίς επιτυχία.

Επιπλέον έγινε χρήση της βιβλιοθήκης Pandas που εφοδιάζει το widget αυτό με όλες τις ιδιότητες ενός DataFrame το οποίο είναι η βασική δομή δεδομένων της βιβλιοθήκης αυτής. Αυτό μας έδωσε τη δυνατότητα να έχουμε indexes στον πίνακα, να έχουμε column headers και βασικότερο όλων να έχουμε τη δυνατότητα

ταξινόμήσης. Πατώντας την αντίστοιχη στήλη υπάρχει η δυνατότητα αύξουσας ή φθίνουσας ταξινόμησης χωρίς ουσιαστικά να χρειαστεί να γράψουμε κάτι έξτρα σε κώδικα. Εδώ κλείνει και η παρένθεση.

Σχετικά με τη διασύνδεση προγράμματος βάσης που είναι και αυτή που μας ενδιαφέρει, θέλουμε στο ιστορικό να εμφανίζονται οι εξής πληροφορίες, ημερομηνία/ώρα, προφίλ που εκτέλεσε την εντολή, συσκευή, όνομα εντολής και παράμετροι εντολής. Οι πληροφορίες δηλαδή προέρχονται και από τις δύο βάσεις.

Τα query που τις συγκεντρώνουν:

```
SELECT ημερομηνία_ώρα,username_πραγματοποιεί,είδος,δωμάτιο,εντολή_id,command_id,
device_id FROM
((Πραγματοποιεί JOIN Ελέγχει ON Ελέγχει.command_id_ελέγχει =
Πραγματοποιεί.command_id_πραγματοποιεί) a1
JOIN Συσκευή ON a1.device_id_ελέγχει = Συσκευή.device_id) a2
JOIN Εντολή ON Εντολή.command_id = a2.command_id_ελέγχει
ORDER BY ημερομηνία_ώρα desc;
```

Υλοποιούμε μια συνένωση αρχικά των πινάκων Πραγματοποιεί και Ελέγχει ως πρός τα command id τους για να έχουμε σε έναν πίνακα στοιχεία που μας απασχολούν (device_id, command_id, username, ημερομηνία/ώρα). Ύστερα τον πίνακα αυτόν τον συνενώνουμε με τον Συσκευή για να πάρουμε το είδος της συσκευής και το δωμάτιο της. Τέλος γίνεται και συνένωση με την εντολή για να κρατήσουμε στον πίνακα και την εντολη_id μέσω της οποίας θα πάμε να βρούμε μετά το όνομα της εντολής απο τη mongo βάση.

Μένει να ανακτήσουμε και τις απαραίτητες πληροφορίες που είναι αποθηκευμένες στη mongo βάση. Οι εντολές Python:

```
appliance_commands = db['appliances'].find({'_id':ObjectId(i[6])})[0]['entoles']
for j in appliance_commands.keys():
if appliance_commands[j]['entolh_id'] == ObjectId(i[4]):
...
parameters=['arxeio_entolwn'].find({"_id":ObjectId('0'*(24-len(str(i[5])))+str(i[5]))})[0]['parametroi']
```

Αρχικά με βάση το id της συσκευής βρίσκουμε το σύνολό εντολών της, στη συνέχεια με βάση το εντολή_id, προσπελαυνοντας όλες τις εντολές βρίσκουμε αυτή με το ίδιο id και παίρνουμε έτσι και τό όνομα της εντολής. Τέλος με βάση το command_id με παρόμοια διαδικασία αυτής στο insert βρίσκουμε το document από το αρχείο εντολών και κρατάμε τις παραμέτρους της εντολής. Έτσι έχουμε πλήρη εποπτεία των εντολών που έχουν πραγματοποιηθεί στο παρελθόν.

Show consumption page

Σελίδα που περιέχει τη συνολική κατανάλωση της κάθε συσκευής σε KW. Query για τον υπολογισμό τους:

```
SELECT device_id_ελέγχει,

CASE
WHEN ενεργή=0 THEN sum(opened_for)*KWh
ELSE sum(opened_for)*KWh + (julianday(CURRENT_TIMESTAMP) -
julianday(max(opened_on)))* 24*KWh END total_consumption FROM
(SELECT *,(julianday(closed_on) - julianday(opened_on))* 24 as
opened_for FROM
(SELECT command_id,εντολή_id,ημερομηνία_ώρα as
opened_on,device_id_ελέγχει,
LEAD (ημερομηνία_ώρα) OVER(order by ημερομηνία_ώρα) closed_on
FROM
(SELECT command_id,εντολή_id,ημερομηνία_ώρα,device_id_ελέγχει FROM
(SELECT * FROM Πραγματοποιεί JOIN Ελέγχει ON
Πραγματοποιεί.command_id_πραγματοποιεί =
Ελέγχει.command_id_ελέγχει) e1
JOIN Εντολή ON Εντολή.command_id = e1.command_id_ελέγχει
WHERE device_id_ελέγχει = '{i[0]}' AND (εντολή_id = '{anapse_id}'
OR εντολή_id = '{sbhse_id}')))
WHERE εντολή_id = '{anapse_id}') JOIN Συσκευή ON
device_id_ελέγχει = Συσκευή.device_id
GROUP BY εντολή_id
```

Για να υπολογίσουμε τη συνολική κατανάλωση δεχτήκαμε ότι η κατανάλωση μιας συσκευής είναι σταθερή και ίση με την τιμή του πεδίου KWh, έτσι αρκεί να βρούμε τις συνολικές ώρες που είναι ενεργοποιημένη και να τις πολλαπλασιάσουμε με τις κιλοβατώρες. Ο χρόνος που μια συσκευή είναι ενεργοποιημένη είναι το άθροισμα όλων των χρονικών διαστημάτων μεταξύ ενός turn on και ενός turn off συν τον χρόνο από το τελευταίο turn on έως τώρα εάν η τελευταία εντολή ήταν turn on.

Το query λοιπόν υλοποιεί την παραπάνω λεκτική περιγραφή. Αρχικά συνενώνονται οι πίνακες Πραγματοποιεί, Συσκευή και Εντολή ως προς το command id για να μαζέψουμε όλη την απαραίτητη πληροφορία. Στη συνέχεια κρατάμε απο τον πίνακα αυτό μόνο τις εντολές turn on, turn off που αφορούν μια συγκεκριμένη συσκευή. Τα id των εντολών αυτών τα βρίσκουμε από τη mongo βάση με τις εξής εντολές.

```
anapse_id=str(db['appliances'].find({'_id':ObjectId(i[0])})[0]['entoles']['Turn on']['entolh_id'])
sbhse_id = str(db['appliances'].find({'_id':ObjectId(i[0])})[0]['entoles']['Turnoff']['entolh_id'])
```

Πλεόν στον πίνακα μας υπάρχουν εναλλάξ εντολές turn on και turn off. Μέσω της εντολής lead δημιουργούμε μια νέα στήλη στον πίνακα με όνομα closed_on τιμής της οποίας είναι η ημερομηνία/ώρα κάθε εντολής turn off που διαδέχεται μια turn on. Στη συνέχεια αφαιρούμε τη στήλη closed on από την ημερομηνία/ώρα που ονομάζουμε opened on και περιέχει το datetime του turn on και δημιουργούμε τη στήλη opened for την οποία και μετατρέπουμε σε ώρες. Τέλος για να συμπεριλάβουμε και την περίπτωση που η τελευταία ήταν turn on δημιουργούμε ένα case. Εάν η συσκευή είναι ενεργή, τότε στις συνολικές ώρες θα προστίθεται η δίαφορά της τρέχουσας datetime από το μέγιστο datetime τής στήλης opened on του πίνακα, το οποίο ισοδυναμεί με την πιο πρόσφατη εντολή. Αλλιώς εάν η συσκευή είναι κλειστή δεν κάνουμε τίποτα. Μετατρέπουμε τις ώρες σε KW πολλαπλασιάζοντας τις με την KWh και έχουμε έτσι τη συνολική κατανάλωση μιας δεδομένης συσκευής.

Delete profile

Η διαγραφή προφίλ δεν υπάρχει σε ξεχωριστή σελίδα όπως τις υπόλοιπες λειτουργίες, βρίσκεται στη σελίδα ενεργειών τόσο του πρωτεύοντος όσο και του δευτερεύοντος προφίλ και πραγματοποιείται όταν ο χρήστης πατάει το αντίστοιχο κουμπί.

Υπενθυμίζουμε τι ισχύει για τις διαγραφές. Όταν διαγράφεται ένα δευτερεύον προφίλ δεν προκύπτει κάποιο ζήτημα και όλα αντιμετωπίζονται από τους περιορισμούς αναφορικής ακεραιότητας. Όταν όμως διαγράφεται ένα πρωτεύον προφίλ θέλουμε τα δικαιώματα επίβλεψης να μεταφέρονται σε άλλο πρωτεύον προφίλ. Εάν πάλι δεν υπάρχει άλλο πρωτεύον προφίλ τότε θέλουμε να διαγράφεται τόσο αυτό όσο και τα υπόλοιπα δευτερεύοντα προφίλ.

Θυμίζουμε επίσης ότι κάτα τη διαγραφή λόγω τον περιορισμών αναφορικής ακεραιότητας, το πρωτεύον username στον πίνακα Παρέχει δικαιώματα τίθεται σε NULL συνεπώς πρέπει με κάποιον τρόπο να συμπληρώνεται αυτή η τιμή και να υπακούει στις προδιαγραφές που έχουμε θέσει. Τα query που χειρίζονται τη διαγραφή προφίλ είναι τα εξής:

```
PRAGMA foreign_keys = 1
DELETE FROM Προφίλ_χρήστη WHERE username = '{self.profile}'
PRAGMA foreign_keys = 0

UPDATE Παρέχει_δικαιώματα
SET primary_username=(CASE
WHEN (SELECT count(username_pro)
FROM Πρωτεύον_προφίλ
ORDER BY username_pro )>0
THEN (SELECT username_pro
FROM Πρωτεύον_προφίλ
ORDER BY username_pro limit 1)
ELSE NULL
END)
WHERE primary_username IS NULL
```

Αρχικά για να μπορέσει να γίνει κανονικά η διαγραφή και να επιβληθούν οι περιορισμοί αναφορικής ακεραιότητας θα πρέπει να εκτελεστεί η εντολή της πρώτης γραμμής. Γίνεται κανονικά η διαγραφή με το δεύτερο query και ξαναπενεργοποιούμε τους περιορισμούς.

Στη συνέχεια βρίσκουμε από τον πίνακα Παρέχει δικαιώματα τις πλειάδες εκείνες που έχουν NULL επιβλέπων πρωτεύον username. Για τα δευτερεύοντα προφίλ εκείνα μέσω ενός case θέτουμε σαν επιβλέπων προφίλ το πρώτο προφίλ του πίνακα Πρωτεύον προφίλ, αλλοώς αν δεν υπάρχει άλλο πρωτεύον προφίλ το αφήνουμε ως έχει.

```
PRAGMA foreign_keys = 1

DELETE FROM Προφίλ_χρήστη
WHERE username IN
(
SELECT secondary_username
FROM Παρέχει_δικαιώματα As B
Where B.primary_username IS NULL
)
PRAGMA foreign_keys = 0
```

Ξαναενεργοποιούμε τους περιορισμούς και τώρα διαγράφουμε τα προφίλ εκείνα τα οποία στο πίνακα έχει πρόσβαση έχουν επιβλέπων προφίλ NULL, όσα προφίλ δηλαδή απέμειναν από την προηγούμενη φάση. Απενεργοποιούμε και πάλι περιορισμούς και ολοκληρώνεται έτσι η διαδικασία της διαγραφής προφίλ που καλύπτει κάθε περίπτωση.

Ολοκληρώνεται έτσι η περιγραφή της εφαρμογής ως προς τις λειτουργίες της και την επικοινωνία της με τις βάσεις. Θα μπορούσαμε να εμβαθύνουμε περισσότερο ως προς τις προγραμματιστικές επιλογές και λεπτομέριες αλλά δε θέλαμε να επεκταθούμε άλλο και να γίνουμε κουραστικοί. Άλλωστε στο λινκ που θα δωθεί σε παράρτημα στο τέλος θα μπορεί κάποιος να επισκεφθεί το repository και να δει όλον τον κώδικα που θα συνοδεύεται από σχολιασμό της κάθε εντολής.

4 ΤΕΛΙΚΗ ΑΞΙΟΛΟΓΗΣΗ ΤΗΣ ΒΑΣΗΣ, ΠΕΡΙΓΡΑΦΗ ΔΙΑΔΙΚΑΣΤΙΚΩΝ ΚΑΙ ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ

4.1 Αξιολόγηση της βάσης, προβλήματα και μελλοντικές βελτιώσεις

Όπως διαπιστώνει κανείς ύστερα απο χρήση της εφαρμογής, επιτυγχάνεται μια πολύ καλή προσομοίωση μια εφαρμογής smart home. Ο χρήστης έχει αρκετές ελευθερίες και επιλογές και μπορεί να πραγματοποιήσει όλες τις απαραίτητες ενέργειες που θα θέλαμε να μπορεί να πραγματοποιεί κάποιος που χρησιμοποιεί έξυπνες συσκευές στην καθημερινότητα του.

Η βάσεις συνεργάζονται σωστά, δε τίθονται θέματα απόδοσης, ούτε χωρικής ούτε χρονικής, λόγω της φύσης της που δεν απαιτεί αποθήκευση μεγάλου ποσού δεδομένων, και πράγματι το πρότυπο συσκευής που ορίστηκε σε συνδυασμό με την υλοποιήση της σε NoSQL DBMS μας επιτρέπει να μοντελοποιούμε αρκετά σύνθετες συσκευές.

Παρόλαυτα δεν μπορούμε να τη θεωρήσουμε ολοκληρωμένη. Όπως ορίσαμε παραπάνω η συσκευή περιγράφεται από ένα σύνολο εντολών και ένα σύνολο παραμέτρων. Οι εντολές περιγράφουν το τι μπορεί να κάνει ο χρήστης με την εντολή αυτή ενώ οι παράμετροι περιγράφουν τη τρέχουσα κατάσταση της συσκευής. Θα θέλαμε λοιπόν να φαίνεται κάπως η επίδραση μιας εντολής στις παραμέτρους της συσκευής. Παραδείγματος χάρην στο παράδειγμα του θερμοστάτη όταν ο χρήστης δίνει εντολή να γίνει η θερμοκρασία του χώρου 23 βαθμούς, θα θέλαμε να φαίνεται αυτή η τιμή στις παραμέτρους της συσκευής και σε συνδυάσμο με τη τιμή της παραμέτρου τρέχουσας θερμοκρασίας χώρου να μπορεί να μοντελοποιηθεί κάπως η μεταβολή της θερμοκρασίαςνα ανανεώνονται σε πραγματικό χρόνο οι τιμές των παραμέτρων.

Κάτι τέτοιο βέβαια στην πραγματικότητα θα γινόταν κατευθείαν απο το μικρουπολογιστικό σύστημα της συσκευής, συνέπως θα μπορούσαμε απλώς να διαβάζουμε τις εξόδους της και να ενημερώνονται με αυτόν τον τρόπο οι παράμετροι. Στην περίπτωση μας θα έπρεπε να μοντελοποιηθεί η κάθε συσκευή ξεχωριστά, κάτι που κρίναμε περιττό και θεωρήσαμε ξεφεύγει από τους στόχους της εργασίας.

Όμως παρόλο που δεν τις αξιοποιούμε επιλέξαμε να συμπεριλάβουμε τις παραμέτρους με το όνομα τους σε ένα Array χωρίς τη δυνατότητα να τους δίνουμε κάποια τιμή για να φαίνεται ακριβώς αυτή η δυνατότητα. Μια πιθανή μελλοντική βελτίωση λοιπόν θα ήταν να ορίσουμε τις παραμέτρους ως ένα εμφωλευμένο document που θα περιέχει ζευγάρια παράμετρος : τιμή.

Ένα άλλο κομμάτι που δεν αξιοποιήθηκε όσο θα θέλαμε είναι αυτό της έξυπνης συσκευής κοντρόλ (κινητό, ντεσκτοπ κλπ). Αρχική επιθυμίας μας ήταν να υπάρχει άμεση επικοινωνία μεταξύ εφαρμογής και διαδικτύου και συνεχής ανταλλαγή δεδομένων. Με αυτόν τον τρόπο θα προσομοιώναμε πραγματικές εφαρμογές του εμπορίου στις οποίες αξιοποιούνται τα δεδομένα χρήστη για να γίνεται personalization διαφημίσεων. Δεν ξεχνάμε ότι στον μικρόκοσμο μας εκπροσωπούμε μια εταιρεία παροχής υπηρεσιών smart home και θα χρειαζόταν να υπακούσουμε τους ανωτέρους μας και να φροντίσουμε η εφαρμογή μας να συμφωνεί σε αυτές τις συχνά ανήθικες αλλά πάντα προσοδοφόρες πρακτικές.

Μια πιθανή μελλοντική βελτίωση λοιπόν θα ήταν εκτός από το όνομα της συσκευής και τη διεύθυνση IP της να αποθηκεύονται στη βάση και άλλα στοιχεία της συσκευής και της διαδικτυακής δραστηριότητας του χρήστη.

Κάτι άλλο που δεν υλοποιήσαμε το οποίο θα ήταν ενδιαφέρον, θα ήταν να εμφανίζεται η κατανάλωση κάθε συσκευής εντός κάποιου χρονικού διαστήματος που θα μπορεί να επιλέξει ο χρήστης. Λογώ χρονικών περιορισμών δεν έγινε αυτή η επέκταση, πολύ πιθανό όμως να είναι έτοιμη για την τελική παρουσίαση.

Επίσης σχετικά με το κομμάτι των κωδικών καλό θα ήταν να υιοθετηθούν πρακτικές ασφάλειας της βάσης. Θα μπορούσαμε να αξιοποιήσουμε τεχνικές hashing και salt για αποθήκευση των κωδικών. Θα μπορούσαν επίσης να χρησιμοποιήθουν μέθοδοι προστασίας απο SQL Injections που θα μπορούσαν να προφυλάξουν τη βάση και τα δεδομένα μας από κακόβουλες προθέσεις.

Η τελευταία πιθανή βελτίωση αφορά της κατηγορικές παραμέτρους. Σαν παράδειγμα θέτουμε μια έξυπνη λάμπα τριών χρωμάτων. Ο καλύτερος σχεδιασμός θα ήταν να υπάρχει εντολή διάλεξε χρώμα η οποία να δίνει στον χρήστη τις τρεις αυτές επιλογές. Οι τιμές των παραμέτρων όμως θέσαμε να είναι μόνο ακέραιες. Θα έπρεπε έτσι να θεωρήσουμε μέγιστη τιμή παραμέτρου το 0 μέγιστη το 2 και να μπορεί έτσι ο χρήσης να επιλέξει μεταξύ των 0, 1, 2. Κάτι τέτοιο όμως δε θα ήταν πρακτικό μιας και δε θα μπορούσε να ξέρει ο χρήστης ποια τιμή αντιστοιχεί σε πιο χρώμα. Ένας τρόπος να επιλυθεί αυτό θα ήταν όπως περιγράφηκε παραπάνω, να ενημερώνονται οι παράμετροι της συσκευής μετά από κάθε εντολή. Ένας άλλος θα ήταν στο document των παραμέτρων των εντολών η κάθε παράμετρος να περιέχει και ένα mapping από ακέραια τιμή σε categorical. Άρα παρόλο που η αναπαράσταση κάθε παραμέτρου με ακεραίους μπορεί με σωστό χειρισμό να περιγράψει κάθε παράμετρο, ακόμα και δεκαδικές ορίζοντας ως μια δεύτερη παράμετρο το δεκαδικό μέρος, δεν είναι πάντα πρακτική.

Θα θέλαμε να σημειώσουμε επίσης ότι στη τρέχουσα εκδοσή της η εφαρμογή επιτρέπει στον οποιονδήποτε να δημιουργεί προφίλ, θα μπορούσε δηλαδή ένα άτακτο παιδί να δημιουργήσει ένα πρωτεύοντα λογαριασμό από τον οποίο θα μπορεί να έχει πρόσβαση παντού. Αυτό επιλύεται εύκολα, είτε προσθέτοντας την επιλογή δημιουργίας προφίλ στη σελίδα ενεργειών των πρωτεύοντων προφίλ αντί για το homepage. Είτε ζητώντας κάποιον κωδικό κάτα τη δημιουργία προφίλ, ίσως κάποιο license το οποίο θα ερχόταν πακέτο θεωρητικά με την αγορά τον υπηρεσιών smart home της εταιρείας που εκπροσωπούμε. Οι δύο λύσεις είναι εξίσου εύκολα υλοποιήσιμες, παρόλαυτα νιώσαμε ότι η τρέχουσα διαμόρφωση προβάλει καλύτερα τις δυνατότητες της βάσης.

Τέλος σχετικά με τα bug, βρέθηκαν αρκετά κάτα τη διάρκεια χρήσης της και αφορούσαν κυρίως corner cases λόγω μη προβλεπόμενης χρήσης της εφαρμογής απο τον χρήστη. Προσπαθήσαμε να τα εντοπίσουμε όλα, έγινε το απαραίτητο exception handling, αλλά δε γίνεται να είμαστε 100% σίγουροι ότι τα εντοπίσαμε όλα. Ολοκληρώνουμε λέγοντας πως είμαστε ικανοποιημένοι από το τελικό αποτέλεσμα και κρίνουμε πως τόσο η βάση όσο και η εφαρμογή ανταποκρίνονται πλήρως στις προδιαγραφές του μικρόκοσμου μας και στη προβλεπόμενη χρήση της.

4.2 Πως εργαστήκαμε και τι πλατφόρμες αξιοποιήθηκαν

Αρχική υποχρέωση ήταν η κατανόηση των αναγκών ενός smart home και στη συνέχεια ο ορισμός των παραδοχών, η εύρεση των οντοτήτων που απαρτίζουν τον μικρόκοσμο μας και η κατάστρωση ενός διαγράμματος οντοτήτων συσχτίσεων.

Την αρχική αυτή προεργασία παρουσιάσαμε στους υπεύθυνους του μαθήματος οι οποίοι με υποδείξεις και παρατηρήσεις μας βοήθησαν να τη φέρουμε στη τελική της μορφή. Η επικοινωνία μεταξύ των μελών της ομάδας, λόγω της κατάστασης, έγινε μέσω τηλεδιασκέψεων στη πλατφόρμα discord, η σχεδίαση του διαγράμματος ERD έγινε μέσω της διαδικτυακής εφαρμογής ERDMaker που αποτελεί δουλεία στα πλαίσια της διπλωματικής ενός φοιτητή στη σχολή μας και τέλος το λογικό σχεσιακό διάγραμμα σχεδιάστηκε μέσω της διαδικτυακής εφαρμογής DBDesigner.

Στη συνέχεια εντοπίστηκε η ανάγκη για χρήση NoSQL βάσης δεδομένων για την περιγραφή της συσκευής και για τη διατήρηση αρχείου εντολών. Αφού ολοκληρώθηκε η κατασκευή του προτύπου του document της συσκευής ξεκίνησε η κατασκευή της βάσης.

Χρησιμοποιήθηκε η SQLite και το DB Browser για το SQL κομμάτι της βάσης, δημιουργήσαμε έτσι το κάθε μέλος local βάση στο μηχάνημα μας. Για το NoSQL κομμάτι της βάσης επιλέχθηκε η mongoDB και χρησιμοποιήθηκε το λογισμικό Compass της mongoDB για ευκολότερη επίβλεψη της βάσης. Αρχική επιθυμία ήταν η βάση να βρίσκεται στο cloud μέσω του mongo Atlas άλλα για ακόμα άγνωστο λόγο δε μας επιτράπηκε να επιλέξουμε το free cluster που θα στέγαζε τη βάση. Συνεπώς και τα δύο κομμάτια της βάσης κατασκευάστηκαν locally, κάτι που εν τέλει αποδείχθηκε οφέλιμο μιας και μπορούσαν και τα δυο μέλη να πειραματίζονται χωρίς να εμπλέκονται στη δουλειά του άλλου.

Σε αυτό το σημείο δημιουργήθηκε η ανάγκη να υπάρχουν τα αρχεία του κώδικα κάπου συγκεντρώμενα όπου θα μπορούσαν και τα δύο μέλη να έχουν πρόσβαση. Δημιουργήθηκε έτσι το Smarthome Repository στο github το οποίο αξιοποιήσαμε μέχρι το τέλος για να μοιράζονται τα αρχεία κώδικα. Παράλληλα κρατήσαμε και ένα google doc στο οποίο γράφαμε τις εκκρεμότητες και τις ιδέες μας για να μην τις ξεχνάμε.

Αργότερα αφού ολοκληρώθηκε η κατασκευή των βάσεων ξεκινήσαμε να δουλεύουμε την εφαρμογή. Αρχικά πειραματιστήκαμε με τις βιβλιοθήκες sqlite3 και pymongo στο περιβάλλον Jupyter, αφού εξοικιωθήκαμε με τη χρήση τους υλοποιήσαμε όλες τις λειτουργίες που θα θέλαμε να πραγματοποιεί η εφαρμογή μας σε μορφή διαλόγου, χρησιμοποιώντας δηλαδή εντολές εισόδου για τις εισόδους και εντολές print για τις εξόδους.

Στη συνέχεια εργαστήκαμε στο γραφικό κομμάτι της εφαρμογής, επιλέχθηκε η χρήση της βιβλιοθήκης PyQt5 και από δω και πέρα χρησιμοποιήθηκαν το Visual Studio και το IDLE ως IDE για τη συγγραφή του κώδικα. Αφού ολοκληρώθηκε και το γραφικό κομμάτι επόμενα βήματα ήταν η χρήση της για εντοπισμό bug και γέμισμα της βάσης με δεδομένα. Η διαδικασία αυτή συνεχίστηκε μέχρι και την παράδοση της άσκησης.

Τέλος ολοκληρώθηκε η εργασία με τη συγγραφή της παρούσας αναφοράς. Γράφτηκε σε LaTeXστο διαδικτυακό γραφικό περιβάλλον Overleaf με βάση το πρότυπο ACM Master Article, ενώ αρκετά από τα διαγράμματα σχεδιάστηκαν με χρήση του Microsoft Paint ή με τις διαδικτυακές εφαρμογές Creatly(free version) και app.diagrams. Με βάση την αναφορά έγινε και η συγγραφή της παρουσίασης σε Power Point.

Χρονοδιάγραμμα:

- **10/11/2020 - 19/11/2020** Κατάστρωση του μικρόκοσμου και του ERD διαγράμματος και από τα δύο μέλη.
- **19/11/2020 - 30/11/2020** Διόρθωση του διαγράμματος με βάση τις παρατηρήσεις στην ενδιάμεση παρουσίαση και από τα δύο μέλη.
- **1/12/2020 - 8/12/2020** Σχεδίαση του λογικού σχεσιακού διαγράμματος και των πινάκων από τον Ζήση Μανούσκο.
- **8/12/2020 - 10/12/2020** Σχεδίαση του προτύπου συσκεύων και mongoDB κομμάτι της βάσης από τον Κρίστι Τοπολλάι.

- **17/12/2020 - 24/12/2020** Υλοποιήση βασικών query για τις διάφορες λειτουργίες της εφαρμογής από τον Ζήση Μανούσκο.
- **24/12/2020 - 2/1/2021** Κατασκεύη γραφικών και εφαρμογής από τον Κρίστι Τοπολλάι.
- **3/1/2021 - 8/1/2021** Συγγραφή αναφοράς από τον Κρίστι Τοπολλάι.
- **9/1/2021** Συγγραφή της παρουσίασης από τον Ζήση Μανούσκο.
- **2/1/2021 - 10/1/2021** Χρήση της εφαρμογής και debugging από τα δύο μέλη.

Εννοείται πως καθόλη τη διάρκεια του πρότζεκτ η επικοινωνία των δύο μελών ήταν συνεχής και όλοι βοηθούσαν παντού. Σε κανένα κομμάτι δε συμμετείχε αποκλειστικά ένα μέλος.

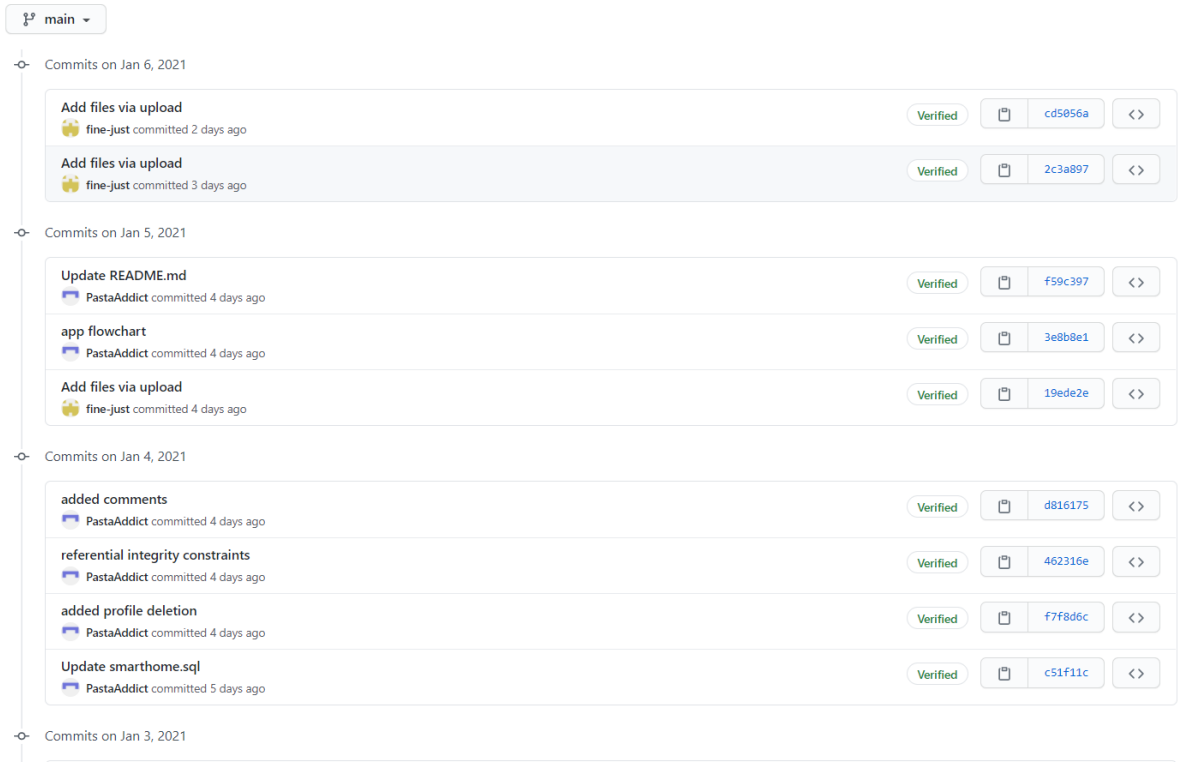

Fig. 15. Κάποια από τα commit στο repository

4.3 Οδηγίες εγκατάστασης και απαραίτητοι σύνδεσμοι

Αρχικά να αναφέρουμε ότι δε συμπεριλάβαμε επιλογή εισαγωγής νέας συσκευής στην εφαρμογή για το λόγο ότι δε θεωρήσαμε ρεαλιστικό το να δημιουργούμε μια συσκευή την οποία μετά ο χρήστης θα μπορεί να χρησιμοποιήσει. Αντιθέτως τις συμπεριλαμβάνουμε εξαρχής στις δύο βάσεις, είτε εκτελώντας τις αντίστοιχες εντολές στο DB Browser και mongo shell. Είτε τρέχοντας το αρχείο initialize_database.py που θα βρείτε στο repository.

Requirements:

Αρχικά πρέπει να υπάρχει εγκατεστημένη η Python, η έκδοση που χρησιμοποιήθηκε απο εμάς είναι η Python 3.8.3. Χρειάζονται επίσης οι βιβλιοθήκες:

- PyQt 5.9.7
- pandas 1.0.5
- pymongo 3.11.2
- bson 0.5.10

Οι εκδόσεις που δίνονται είναι αυτές που χρησιμοποήθηκαν από εμάς.

Επιπλέον πρέπει να υπάρχει εγκατεστημένη η SQLite, η έκδοση που χρησιμοποιήθηκε απο εμάς είναι η 3.33.0.

Πρέπει επίσης να υπάρχει εγκατεστημένη η mongoDB, η έκδοση που χρησιμοποιήθηκε απο εμάς είναι η 4.4.2.

Οδηγίες εγκατάστασης:

Δίνουμε δύο τρόπους εγκατάστασης της εφαρμογής.

1os **Τρόπος**

Από το Smarthome Repository κατεβάζουμε το φάκελο project. Σε αυτόν τον φάκελο περιέχονται όλα τα απαραίτητα αρχεία:

- Η SQLite βάση μας smarthome.db που περιέχει έτοιμα τα tables.
- Το style.txt που περιέχει το stylesheet που διακοσμεί το gui της εφαρμογής.
- Τον φάκελο images που περιέχει κάποια διακοσμητικά που αξιοποιεί το stylesheet.
- Το insert_appliances.py το οποίο γεμίζει τις δύο βάσεις με συσκευές.
- Το gui.py που είναι το πρόγραμμα της εφαρμογής.

Αφού κατεβάσουμε το αρχείο εκτελούμε το insert_appliances.py. Θα μας εμφανιστεί μια λίστα διαθέσιμων συσκευών. Πληκτρολογούμε το νούμερο κάθε συσκεύης που επιθυμούμε διαχωρίζοντας τις με κενό και προσέχοντας μη συμπεριλάβουμε την ίδια συσκευή δυο φορές. Συμπεριλάβαμε 11 συσκευές, στο μέλλον θα προστεθούν και άλλες.

Αφού γεμίσουμε με αυτόν τον τρόπο τις δύο βάσεις εκτελούμε το gui.py.

2os **Τρόπος**

Αντί να χρησιμοποιήσουμε την έτοιμη βάση smarthome.db, μπορούμε να δημιουργήσουμε από μόνοι ένα αρχείο db στο αντίγραφο του φακέλου project στον υπολογιστή μας.Αφού το δημιουργήσουμε αντιγράφουμε τις εντολές από το αρχείο smarthome.sql και τις εκτελούμε. Προτείνεται η αντιγραφή αντί για τη λήψη του αρχείου μιας και παρατηρήθηκε ότι κατά τη λήψη του συγκεκριμένου αρχείου δε διατηρούνται οι ελληνικοί χαρακτήρες. Κλείνουμε το DB Browser ή το αντίστοιχο λογισμικό που χρησιμοποιούμε και εκτελούμε το insert_appliances.py . Η διαδικασία ολοκληρώνεται ξανά εκτελόντας το gui.py.

Εάν θέλουμε να συμπεριλάβουμε δικές μας συσκευές θα πρέπει να φροντίσουμε να ακολουθούν ακριβώς το ίδιο πρότυπο που περιγράφηκε παραπάνω και να κάνουμε τα αντίστοιχα insert, τόσο στην SQLite βάση όσο και στο collection appliances της smarthome mongoDB βάσης μας.

Τέλος ιδιαίτερη προσοχή όταν αδειάζουμε τη βάση για οποιοδήποτε λόγο. Θα πρέπει να διαγράφουμε και τα δύο collection από τη mongoDB βάση.

5 ΠΗΓΕΣ ΚΑΙ ΒΟΗΘΗΜΑΤΑ

- https://eclass.upatras.gr/modules/document/index.php?course=EE766openDir=/5f7ce163t0qg
- https://en.wikipedia.org/wiki/Home_automation
- https://www.geeksforgeeks.org/difference-between-mysql-and-sqlite/
- https://sqlite.org/docs.html
- https://docs.python.org/3/library/sqlite3.html
- https://www.sqlitetutorial.net/
- https://www.sqlitetutorial.net/sqlite-python/
- https://www.mongodb.com/nosql-explained/nosql-vs-sql
- https://docs.mongodb.com/
- https://pymongo.readthedocs.io/en/stable/tutorial.html
- https://doc.qt.io/qtforpython/
- https://www.learnpyqt.com/tutorials/qtableview-modelviews-numpy-pandas/
- https://www.learnpyqt.com/tutorials/layouts/
- https://discourse.techart.online/t/release-qt-dark-orange-stylesheet/2287post14381