ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εφαρμογή διαχείρισης και ενημέρωσης της λειτουργίας οικιακών συσκευών** **Application for controlling and monitoring the usage of household devices**
Ονοματεπώνυμο Φοιτητή	**Κανελλόπουλος Μάριος**
Πατρώνυμο	**Δημήτριος**
Αριθμός Μητρώου	**ΜΠΠΛ18025**
Επιβλέπων	**Κωνσταντίνα Χρυσαφιάδη Ε.ΔΙ.Π.**

Ημερομηνία Παράδοσης **Δεκέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Κωνσταντίνα Χρυσαφιάδη Ε.ΔΙ.Π.	Μαρία Βίρβου Καθηγήτρια	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

Περίληψη

Η παρούσα μεταπτυχιακή διατριβή, έχει ως αντικείμενο την ανάπτυξη μιας cross-platform εφαρμογής διαχείρισης έξυπνων συσκευών οικιακής χρήσης. Οι δυνατότητες που προσφέρει, απευθύνονται σε συσκευές διαφορετικών κατασκευαστών, οι οποίες υποστηρίζονται από τις τεχνολογίες της εταιρείας eWeLink.

Πρόκειται για μία Smart Home εφαρμογή, η οποία υποστηρίζει την πρόσβαση από συσκευές με λειτουργικό σύστημα:

1. Android (native app)
2. iOS (native app)
3. Wear OS (Android smartwatches - native app)
4. Τέλος, υπάρχει και η δυνατότητα πρόσβασης μέσω browser στη Web app έκδοση

Οι λειτουργίες που προσφέρονται στον χρήστη, είναι η απομακρυσμένη και άμεση διαχείριση των οικιακών του συσκευών, ο καθορισμός ρουτινών διαχείρισης αλλά και η παρακολούθηση της χρήσης τους μέσω διαγραμμάτων λειτουργίας σε ημερήσια, εβδομαδιαία και μηνιαία βάση.

Με τον όρο διαχείριση, εννοούμε την εναλλαγή της λειτουργίας των συσκευών (on/off). Έτσι, ο χρήστης έχει τη δυνατότητα να θέσει στην επιθυμητή κατάσταση τη λειτουργία των συσκευών του, από οποιαδήποτε μέρος και αν βρίσκεται, χωρίς να χρειάζεται να το κάνει χειροκίνητα. Επίσης, η δυνατότητα προγραμματισμού της κατάστασης λειτουργίας που παρέχεται, έρχεται να προσφέρει ακόμη περισσότερες ευκολίες στο χρήστη αναφορικά με την καθημερινότητά του, αφού μπορεί μέσω ρουτινών, να αυτοματοποιήσει διάφορες εργασίες με χρήσιμα προς αυτόν αποτελέσματα (π.χ., να έχει ζεστό καφέ τα πρωινά των καθημερινών, να έχει ζεστό νερό κάθε απόγευμα, να σβήνουν όλα τα φώτα κάθε βράδυ, κλπ.)

Abstract

This master's thesis has as an object the development of a cross-platform application for the management of smart household devices. The features it offers concern devices from different manufacturers, which are supported by the technologies of the eWeLink company.

It is a Smart Home application, which supports access from devices with operating system of:
1. Android (native app)
2. iOS (native app)
3. Wear OS (Android smartwatches - native app)
4. Finally, there is also the possibility of accessing the Web app version through an Internet browser

The functions offered to the user are the remote and direct control of their home devices, the scheduling of controlling routines and the monitoring of the devices' usage, through relevant charts on a daily, weekly and monthly basis.

By the term of controlling, we mean the switching of the devices' operation (on/off). Thus, the user has the ability to set the operation of his devices to the desired state, from any place, without having to do it manually. Also, the possibility of scheduling the devices' operating mode, comes to offer even more convenience to the user regarding his daily life, since they can, through routines, automate various tasks and get useful results (e.g., having hot coffee weekday mornings, to have hot water every afternoon, to turn off all the lights every night, etc.)

1. Εισαγωγή

Είναι σαφές, ότι ζούμε σε μία εποχή που χαρακτηρίζεται από ραγδαία τεχνολογική ανάπτυξη. Η είσοδος του Internet όπως και των έξυπνων συσκευών στην καθημερινότητά μας, αλλάζει συνεχώς τόσο τον τρόπο, όσο και την ποιότητα της ζωής μας.

Ένας τομέας της κοινωνίας όπου ο αντίκτυπος της τεχνολογικής εξέλιξης καθίσταται σημαντικά αισθητός, αποτελεί η έννοια του σύγχρονου νοικοκυριού. Οι ευκολίες όπου η τεχνολογία παρέχει στην καθημερινότητα ενός νοικοκυριού είναι αξιοσημείωτες, ειδικότερα τα τελευταία χρόνια με τη μαζική είσοδο και τη μαζική χρήση του Internet.

Αν εστιάσουμε λοιπόν στην αλληλεπίδραση του ανθρώπου με τις οικιακές συσκευές σε έναν χρονικό ορίζοντα 70 ετών. Κάνοντας μία αναδρομή στη δεκαετία του 1950 και στον τρόπο που οι άνθρωποι ετοίμαζαν το φαγητό τους, θα διαπιστώσουμε ότι χρησιμοποιούσαν κοινόχρηστους ξυλόφουρνους, φωτιά και αυτοσχέδια μαγειρικά σκεύη.

Σήμερα, η ιδιωτική ηλεκτρική κουζίνα είναι αναπόσπαστο κομμάτι κάθε νοικοκυριού, παρέχει πολλές δυνατότητες μαγειρέματος, και μπορεί κάποιος να τη χειριστεί ακόμη και απομακρυσμένα από ένα κινητό τηλέφωνο.

Μπορούμε εύκολα λοιπόν να αντιληφθούμε το βαθμό που η τεχνολογία και οι σύγχρονες συσκευές έχουν επηρεάσει τη ζωή μας, παρέχοντας όλο και περισσότερες δυνατότητες.

1.1. Αντικείμενο πτυχιακής εργασίας

Σκοπός της συγκεκριμένης διατριβής, είναι η δημιουργία μιας εφαρμογής όπου ο χρήστης μέσω ενός smart device (smartphone, smartwatch, ηλεκτρονικό υπολογιστή κλπ.) να έχει τη δυνατότητα να προγραμματίσει άμεσα ή ετεροχρονισμένα τη λειτουργία των ηλεκτρικών συσκευών του, και να επιβλέψει τη λειτουργία τους σε ημερήσια, εβδομαδιαία και μηνιαία βάση.

1.2. Περιγραφή του προβλήματος

Η χρήση IoT (Internet of things / Διαδίκτυο των πραγμάτων) συσκευών εντός των νοικοκυριών γίνεται όλο και περισσότερο διαδεδομένη στο κοινό, και οι εταιρείες εμπορίας προϊόντων που απευθύνονται σε αυτούς τους καταναλωτές, προσπαθούν να ενσωματώσουν τεχνολογίες διασύνδεσης στο μεγαλύτερο δυνατό μέρος αυτών των συσκευών.

Σκοπός λοιπόν των εταιρειών, είναι η διευκόλυνση χρήσης των προϊόντων τους, αφού η παροχή απομακρυσμένης διαχείρισης που προσφέρουν, παρέχει στο χρήστη σημαντικές ευκολίες.

Για παράδειγμα, ο χρήστης μιας τέτοιας τεχνολογίας, μπορεί να προγραμματίσει απομακρυσμένα τη λειτουργία του ηλεκτρικού του θερμοσίφωνα, ώστε να έχει ζεστό νερό κατά την άφιξή στην οικία του, αποφεύγοντας την ευθύνη της χειροκίνητης εναλλαγής της λειτουργίας της συσκευής του, και εξοικονομώντας χρήσιμο προς αυτόν χρόνο.

Ωστόσο, το γεγονός ότι κάθε εταιρία διάθεσης τέτοιων προϊόντων παρείχε τη δική της εφαρμογή διαχείρισης των συσκευών της, είχε σαν αποτέλεσμα είτε την υποχρέωση από τη μεριά του καταναλωτή στη χρήση πολλών διαφορετικών εφαρμογών για τη διαχείριση των συσκευών κάθε εταιρείας. Διαφορετικά, έπρεπε να περιοριστεί στη χρήση των προϊόντων κάποιου κατασκευαστή κατά αποκλειστικότητα, ώστε να μπορεί να χειριστεί όλες τις συσκευές από μία εφαρμογή.

Αυτό, οδήγησε στην ανάγκη δημιουργίας εφαρμογών που θα μπορούν να διαχειριστούν συσκευές διαφορετικών εταιρειών, ώστε να μπορούν να καλύψουν τις ανάγκες του οικοσυστήματος ενός έξυπνου σπιτιού.

Η εφαρμογή, καλύπτει τόσο τις παραπάνω, όσο και κάποιες πρόσθετες ανάγκες που ένας καταναλωτής έχει από τη χρήση μιας τέτοιας εφαρμογής, ώστε να μπορεί να χρησιμοποιηθεί ως μοναδική εφαρμογή διαχείρισης των συσκευών του χρήστη, και να του προσφέρει ένα σύνολο δυνατοτήτων που θα ικανοποιήσουν τις απαιτήσεις του.

1.3. Στόχοι της εφαρμογής

Πρωταρχικός στόχος της εφαρμογής, είναι η ευκολία πρόσβασης από τη μεριά του χρήστη.
Για να επιτευχθεί αυτός ο στόχος, η εφαρμογή αναπτύχθηκε χρησιμοποιώντας τεχνολογίες που θα μπορούσαν να την καταστήσουν cross-platform ώστε να υποστηρίζονται όσο το δυνατόν περισσότερες συσκευές.

Συγκεκριμένα, οι λειτουργίες της συγκεκριμένης εφαρμογής, είναι προσβάσιμες από συσκευές με λειτουργικό σύστημα:

1. Android
2. iOS
3. Android wear smartwatches
4. Οποιαδήποτε εφαρμογή με εγκατεστημένο Internet browser

Επίσης, ένα ακόμη κίνητρο για τη δημιουργία της συγκεκριμένης εφαρμογής αποτέλεσε η δυνατότητα προσθήκης επιπλέον παραμετροποιήσιμων επιλογών στον προγραμματισμό της λειτουργίας των συσκευών.

Συγκεκριμένα, παρέχονται οι λειτουργίες προγραμματισμού:

1. Με βάση τα λεπτά κάθε ώρας.
   Η συγκεκριμένη δυνατότητα, επιτρέπει στο χρήστη να επηρεάζει τη λειτουργία των συσκευών του στα πρώτα Χ λεπτά κάθε ώρας.

   Για παράδειγμα, για Χ=10 έχουμε αλλαγή της λειτουργίας της συσκευής σε ωριαία βάση στις 00:10, 01:10, 02:10.

2. Κάθε Χ λεπτά.
   Αυτή η δυνατότητα επιτρέπει στο χρήστη να επηρεάζει τη λειτουργία των συσκευών του κάθε Χ λεπτά της ώρας

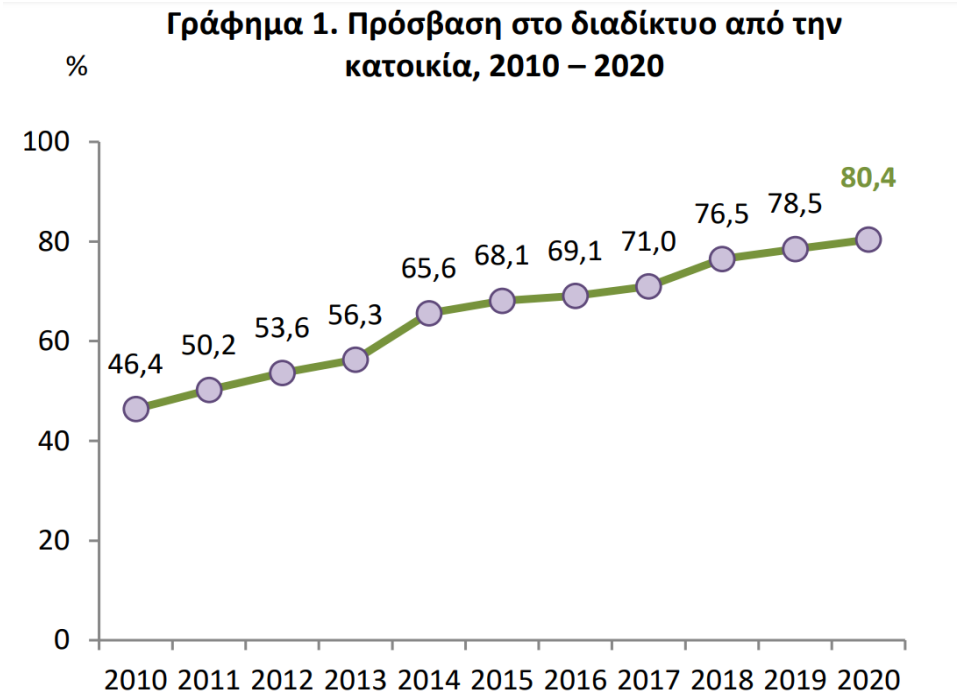
   Για παράδειγμα, για Χ=10 έχουμε αλλαγή της λειτουργίας της συσκευής σε 10-λεπτη βάση στις 00:10, 00:20, 00:30.

2. Ανασκόπηση πεδίου

Η χρήση του Internet και των έξυπνων συσκευών, εισέλθει δυναμικά στη ζωή του σύγχρονου ανθρώπου, φέρνοντας μαζικές αλλαγές τόσο στην επαγγελματική όσο και στην προσωπική του ζωή. Αξίζει να αναφέρουμε ότι σύμφωνα με έρευνα της ΕΛΣΤΑΤ το 2017, προκύπτει πως το 88,6% των ελληνικών επιχειρήσεων χρησιμοποιεί ηλεκτρονικό υπολογιστή και πρόσβαση στο διαδίκτυο για τη διεκπεραίωση των επαγγελματικών της διαδικασιών.

Επιπλέον, σύμφωνα με έρευνα της ΕΛΣΤΑΤ το 2022, το 80,4% των ελληνικών νοικοκυριών έχει τη δυνατότητα χρήσης του διαδικτύου, ενώ σε σύγκριση με το 2010, η αύξηση που έχει καταγραφεί είναι στο 73,3%. Το συγκεκριμένο ποσοστό αντικατοπτρίζει τις ριζικές αλλαγές σε όλα τα κοινωνικά επίπεδα (π.χ., κοινωνικές σχέσεις και επικοινωνία, συναλλαγές με κρατικούς και τραπεζικούς φορείς, διασκέδαση). Αξίζει να αναφέρουμε ότι απαραίτητες ενέργειες όπως η διατραπεζική μεταφορά χρημάτων, στο παρελθόν απαιτούσε τη φυσική παρουσία του καταναλωτή στο τραπεζικό κατάστημα, αλλά και σημαντικό χρόνο για την εκτέλεση της συναλλαγής. Πλέον, μπορεί να πραγματοποιηθεί άμεσα από τον ενδιαφερόμενο μέσω ενός smartphone, εξοικονομώντας σημαντικό χρόνο και κόπο.

Μπορούμε εύκολα λοιπόν να αντιληφθούμε ότι η χρήση έξυπνων συσκευών και του διαδικτύου, έχει γίνει αναπόσπαστο κομμάτι της καθημερινότητάς μας τόσο στην προσωπική, όσο και την επαγγελματική ζωή.



2.1. Αναφορά σε παρόμοιες εφαρμογές

Παρατηρώντας τις τεχνολογικές εξελίξεις, καθίσταται σαφές ότι η τάση προς το IoT αυξάνεται.
Αυτό σημαίνει ότι οι άνθρωποι τείνουν να χρησιμοποιούν συσκευές οι οποίες συνδέονται μεταξύ τους μέσω του παγκόσμιου ιστού.

Αυτό το φαινόμενο, οδηγεί τις εταιρείες να ενσωματώνουν τεχνολογίες διασύνδεσης σε ολοένα και περισσότερα προϊόντα, πέρα από εκείνα όπου αυτά που θεωρούσαμε δεδομένα έως τώρα (κινητά τηλέφωνα, ηλεκτρονικοί υπολογιστές κλπ.). Βλέπουμε πλέον υποστήριξη Wi-Fi σύνδεσης σε κλιματιστικά, ψυγεία, πλυντήρια, πρίζες, τηλεοράσεις, διακόπτες, φώτα, μικροσυσκευές κλπ. Ένα σύγχρονο σπίτι λοιπόν, έχει τη δυνατότητα να δημιουργήσει ένα δίκτυο επικοινωνίας με όλες τις οικιακές του συσκευές.

Από τη μεριά του χρήστη των παραπάνω συσκευών, δημιουργείται η ανάγκη εκμετάλλευσης των δυνατοτήτων απομακρυσμένης διαχείρισης που του παρέχονται (π.χ. καθορισμός / προγραμματισμός λειτουργίας, έλεγχος λειτουργίας, έλεγχος κατανάλωσης ενέργειας κλπ.). Για να το πετύχει αυτό, είναι απαραίτητη η ύπαρξη κάποιου περιβάλλοντος διαχείρισης του συνόλου των συσκευών αυτών.

Έτσι, παρουσιάζονται ορισμένες εφαρμογές οι οποίες αναπτύχθηκαν με σκοπό να καλύψουν τις παραπάνω ανάγκες και να δώσουν στον χρήστη τη δυνατότητα πλήρους ελέγχου του έξυπνου οικοσυστήματος της οικίας του.

2.1.1. eWeLink - Smart Home

Η εφαρμογή της **eWeLink**, αποτελεί την προτεινόμενη εφαρμογή της εταιρείας Sonoff αλλά και άλλων εταιρειών για τα προϊόντα τους. Σύμφωνα με την ιστοσελίδα της εταιρείας που την ανέπτυξε, ο αριθμός των χρηστών της εκτιμάται στα 30 εκ. σε περισσότερες από 172 χώρες.

Η eWeLink έχει αναπτυχθεί ώστε να εκμεταλλευτεί τις δυνατότητες των συσκευών των κατασκευαστών που υποστηρίζει, και να προσφέρει στον χρήστη πλήρη έλεγχό τους. Χρησιμοποιώντας τη συγκεκριμένη cross platform εφαρμογή (πρόσβαση μέσω Android, iOS και Web), ο χρήστης έχει την ευκαιρία να διαχειριστεί τις συσκευές μιας πληθώρας εταιρειών και να χρησιμοποιήσει λειτουργίες όπως οι ακόλουθες:

- Απομακρυσμένος καθορισμός λειτουργίας (on/off)
- Χρονικός προγραμματισμός λειτουργίας
- Καθορισμός λειτουργίας με αντίστροφη μέτρηση
- Κοινή διαχείριση με άλλους χρήστες
- Δημιουργία και διαχείριση γκρουπ συσκευών

2.1.2. Google Home

Η εφαρμογή **Google Home**, αποτελεί την πρόταση της Google στην κάλυψη των αναγκών που αναφέρθηκαν παραπάνω. Αξίζει να αναφέρουμε ότι μόνο στο Play Store, οι μεταφορτώσεις της ξεπερνούν τα 100 εκατομμύρια. Είναι μία εύκολη στη χρήση εφαρμογή με όμορφο σχεδιασμό, όπου ο χρήστης μπορεί να συνδέσει συσκευές διαφορετικών εταιρειών, να τις ομαδοποιήσει και να τις διαχειριστεί άμεσα τη λειτουργία τους. Επιπρόσθετα των βασικών λειτουργιών μιας Smart Home εφαρμογής, στο Google Home μπορούν να καθοριστούν ρουτίνες διαχείρισης συσκευών.
Π.χ. ρουτίνα: “I am home” η οποία θα εκτελέσει την ενεργοποίηση λειτουργίας του κλιματιστικού και του φωτισμού.

Τέλος, υπάρχει η δυνατότητα διασύνδεσης με συσκευές - ψηφιακούς οδηγούς όπως το Google Assistant και η Alexa, όπου ο χρήστης μπορεί να διαχειριστεί τις συσκευές του και με φωνητικές εντολές.

2.2. Διαφοροποιήσεις της εφαρμογής μου

Η εφαρμογή καλύπτει όλες τις βασικές λειτουργίες ενός Smart Home application, όπως ο έλεγχος της κατάστασης των συσκευών του και ο άμεσος καθορισμός της λειτουργίας τους, αλλά προσφέρει και σημαντικές πρόσθετες δυνατότητες.

Την πρώτη διαφοροποίηση της εφαρμογής, αποτελεί ο χρονικός προγραμματισμός της λειτουργίας των συσκευών. Εδώ ο χρήστης έχει τις εξής επιλογές:

1. Προγραμματισμός για συγκεκριμένη χρονική στιγμή.
   Π.χ.: Ενεργοποίηση του βραστήρα στις 08:00.

2. Προγραμματισμός με βάση τα λεπτά κάθε ώρας.
   Π.χ.: Αλλαγή λειτουργίας (toggle) του κλιματιστικού κάθε 10 πρώτα λεπτά της κάθε ώρας (00:10, 01:10, 02:10 κ.ο.κ).

3. Προγραμματισμός κάθε Χ λεπτά.
   Π.χ.: Αλλαγή λειτουργίας (toggle) των φώτων της εισόδου κάθε 30 λεπτά.

Η δεύτερη σημαντική προσθήκη της εφαρμογής είναι η αυτή των γραφημάτων λειτουργίας. Εδώ ο χρήστης μπορεί να παρακολουθεί τη λειτουργία των συσκευών του σε ημερήσια, εβδομαδιαία και μηνιαία βάση. Η συγκεκριμένη λειτουργία μπορεί να βοηθήσει τον χρήστη να εντοπίσει άσκοπη λειτουργία ορισμένων συσκευών και να λάβει μέτρα με σκοπό τόσο την εξοικονόμηση ενέργειας και την οικονομία του νοικοκυριού του.

Μία ακόμη διαφοροποίηση, και συγκεκριμένα μια λειτουργία την οποία η εφαρμογή της eWeLink δεν προσφέρει ενώ αυτή της Google ενσωμάτωσε μόλις πρόσφατα (Οκτώβριος 2022), είναι η δυνατότητα εγκατάστασης σε Smartwatch. Η εφαρμογή, παρέχει τη δυνατότητα πρόσβασης από έξυπνα ρολόγια με λειτουργικό WearOS, κάτι που κάνει την διαχείριση των έξυπνων συσκευών εύκολη και άμεση.

Τέλος, σημαντική διαφοροποίηση αποτελεί και η δυνατότητα πρόσβασης στις λειτουργίες της εφαρμογής από οποιονδήποτε browser. Έτσι η πρόσβαση δεν υπόκειται σε περιορισμούς λειτουργικού συστήματος και μπορεί να επιτευχθεί από οποιαδήποτε συσκευή περιέχει Internet browser, καλύπτοντας ένα συντριπτικό ποσοστό συσκευών.
Η συγκεκριμένη δυνατότητα δεν παρέχεται από την Google home, και ενσωματώθηκε στην eWeLink τον Ιούνιο του 2021.

3. Ανάλυση και Σχεδιασμός της Εφαρμογής

3.1. Εισαγωγή

Ο στόχος αυτής της εφαρμογής είναι να σχεδιαστεί ένα σύστημα διαχείρισης του συνόλου των συσκευών ενός έξυπνου οικοσυστήματος.
Για να επιτευχθεί αυτό, ο χρήστης αλληλοεπιδρά με την εφαρμογή μέσω του front-end συστήματος και μεταφέρει τα επιθυμητά αιτήματα στον server της εφαρμογής.
Στη συνέχεια, ανάλογα με το είδος του αιτήματος, ο server επικοινωνεί είτε απευθείας με τις συσκευές μέσω API (για διαχείριση σε πραγματικό χρόνο) ή με τη βάση δεδομένων (για δημιουργία schedule, γραφημάτων, αλλαγή ονομάτων κλπ.).

3.2. Σύντομη περιγραφή της λύσης που θα αναπτυχθεί

Η συγκεκριμένη εφαρμογή, προσφέρει στο χρήστη σημαντικές δυνατότητες σχετικά με τη διαχείριση των έξυπνων συσκευών του. Οι συγκεκριμένες δυνατότητες, διαφοροποιούνται ανάλογα με τη συσκευή από την οποία θα πραγματοποιήσει σύνδεση ο χρήστης, και συγκεκριμένα διακρίνονται σε δύο κατηγορίες:

1. Είσοδο από browser / Android mobile app, iOS mobile app
2. Είσοδο από Android wear app

3.2.1. Web Browser / Android-iOS mobile

Χρησιμοποιώντας την εφαρμογή τόσο από τα Android και iOS mobile apps, όσο και μέσω web browser, ο χρήστης έχει τη δυνατότητα να:

1. Προγραμματίσει τη λειτουργία των συσκευών του σε πραγματικό χρόνο.
2. Προγραμματίσει τη λειτουργία των συσκευών του στο μέλλον, χρησιμοποιώντας προσαρμόσιμες λειτουργίες.
3. Παρακολουθήσει τη λειτουργία των συσκευών του σε πραγματικό χρόνο.
4. Παρακολουθήσει τη λειτουργία των συσκευών του σε ημερήσια, εβδομαδιαία και μηνιαία χρονική βάση.
5. Πραγματοποιήσει μετονομασία των συσκευών.

3.2.2. Android Wear OS Smartwatch

Χρησιμοποιώντας την εφαρμογή μέσω Android Wear OS Smartwatch, ο χρήστης έχει τη δυνατότητα:

1. Να προγραμματίσει τη λειτουργία των συσκευών του σε πραγματικό χρόνο.
2. Να παρακολουθήσει τη λειτουργία των συσκευών του σε πραγματικό χρόνο.

3.3. Προσφερόμενες λειτουργίες

Ακολουθεί αναλυτική περιγραφή των λειτουργιών που προσφέρονται στο χρήστη της εφαρμογής:

I. **Mobile app / Web browser:**
   1. **Login:** Είσοδος στην εφαρμογή.
   2. **Toggle:** Αλλαγή της κατάστασης λειτουργίας των συσκευών σε πραγματικό χρόνο.
   3. **Get status:** Παρακολούθηση της κατάστασης λειτουργίας των συσκευών σε πραγματικό χρόνο.
   4. **Refresh:** Ανανέωση των δεδομένων της κατάστασης λειτουργίας των συσκευών.
   5. **Tasks' monitor:** Προβολή τρεχόντων tasks
   6. **Add task:** Δημιουργία task για τον προγραμματισμό της κατάστασης λειτουργίας συσκευής σε:
      i. Συγκεκριμένη χρονική στιγμή
      ii. Κάθε Χ πρώτα λεπτά της ώρας

iii. Κάθε Χ λεπτά

7. **Delete task:** Διαγραφή task
8. **Monitor graph:** Προβολή διαγραμμάτων ημερήσιας/εβδομαδιαίας/μηνιαίας λειτουργίας
9. **Device rename:** Μετονομασία συσκευής
10. **Logout:** Αποσύνδεση από την εφαρμογή

II. <u>Smartwatch app:</u>

1. **Login:** Είσοδος στην εφαρμογή.
2. **Toggle:** Αλλαγή της κατάστασης λειτουργίας των συσκευών σε πραγματικό χρόνο.
3. **Get status:** Παρακολούθηση της κατάστασης λειτουργίας των συσκευών σε πραγματικό χρόνο.
4. **Refresh:** Ανανέωση των δεδομένων της κατάστασης λειτουργίας των συσκευών.
5. **Logout:** Αποσύνδεση από την εφαρμογή

3.4. Περιορισμοί - απαιτήσεις για την εγκατάσταση και χρήση της εφαρμογής

Για τη χρήση της εφαρμογής, είναι απαραίτητη η πρόσβαση στο Internet τόσο των τελικών συσκευών που επιθυμούμε να διαχειριστούμε, όσο και της συσκευής που θα χρησιμοποιηθεί για την παραπάνω διαχείριση.

Επιπλέον, είναι απαραίτητη η ύπαρξη λογαριασμού στην εταιρία eWelink, όπου και θα έχουν καταχωρηθεί οι συσκευές του χρήστη. Η eWelink αποτελεί την τρέχουσα προσφερόμενη λύση για τη διαχείριση των συσκευών που υποστηρίζει η εφαρμογή, και η επικοινωνία με τις συσκευές πραγματοποιείται μέσω του API της συγκεκριμένης εταιρείας.

Απαιτήσεις εγκατάστασης:

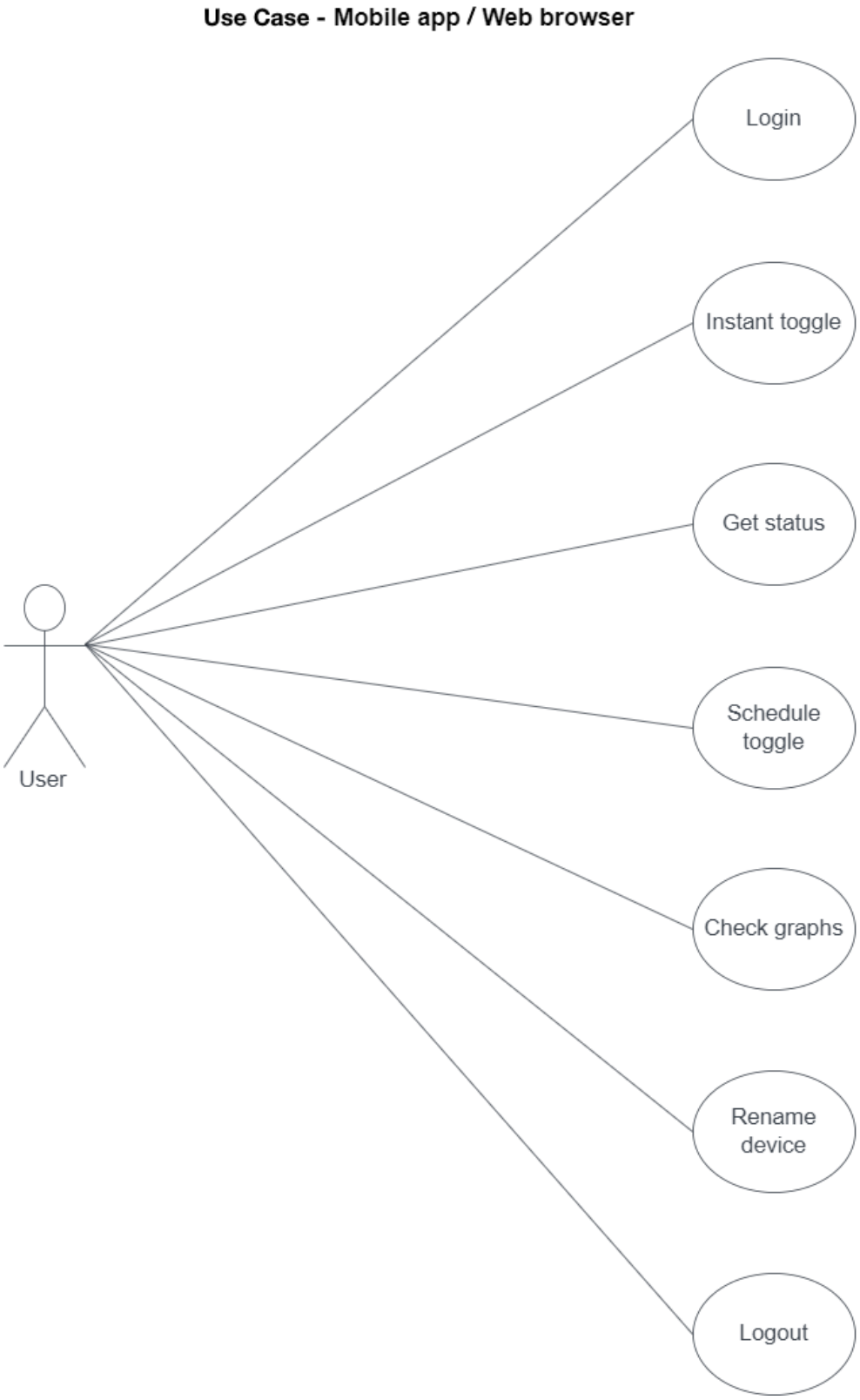
1. **Mobile Android εφαρμογής**
   Λειτουργικό σύστημα Android έκδοσης 6 ή νεότερης και ενεργή πρόσβαση στο Internet

2. **Mobile iOS εφαρμογής**
   Λειτουργικό σύστημα iOS έκδοσης 12 ή νεότερης και ενεργή πρόσβαση στο Internet

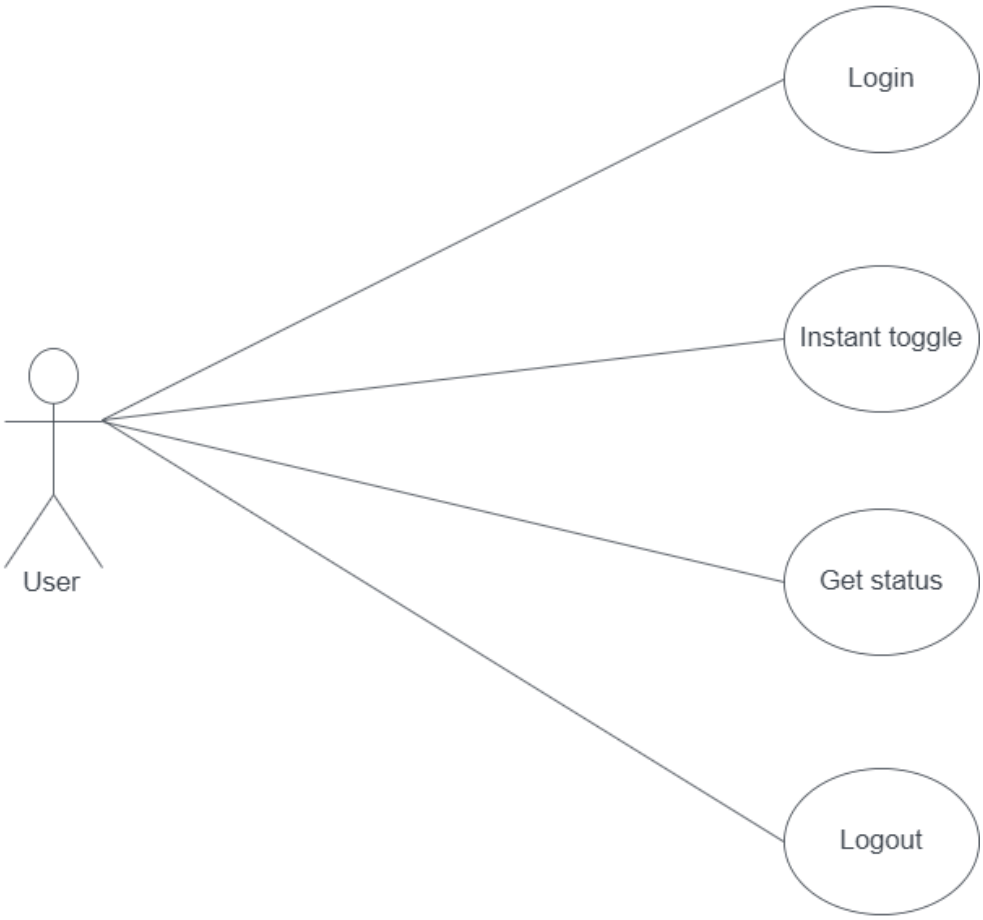
3. **Smartwatch εφαρμογής**
   Λειτουργικό σύστημα Wear OS έκδοσης 3 ή νεότερης και ενεργή πρόσβαση στο Internet

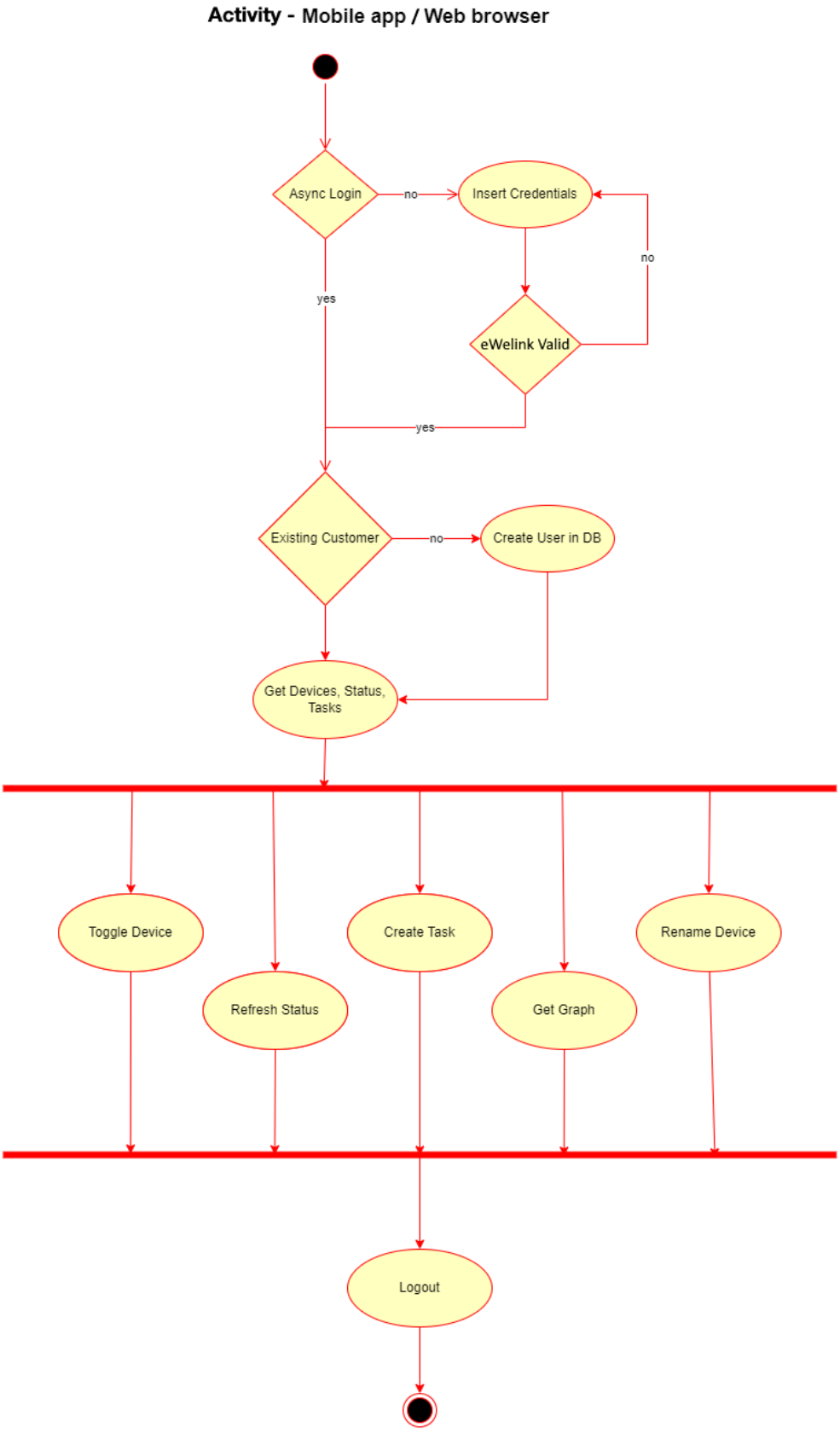
4. **Η χρήση της Web εφαρμογής**
   Web browser με ενεργή πρόσβαση στο Internet

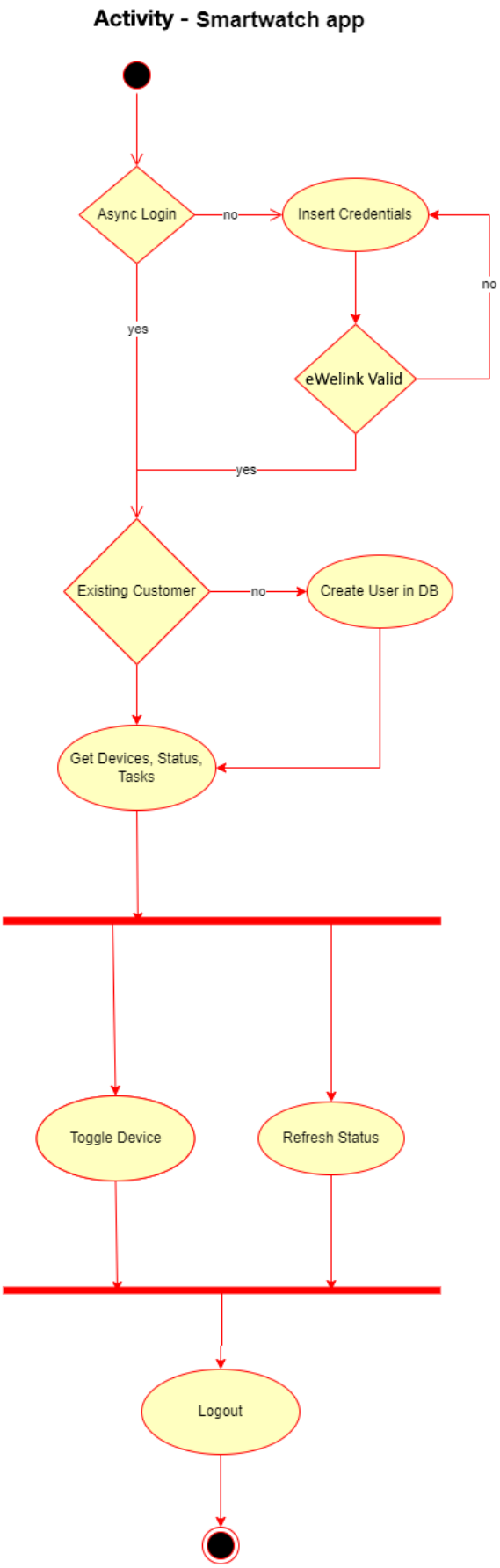
3.5. Διαγράμματα UML

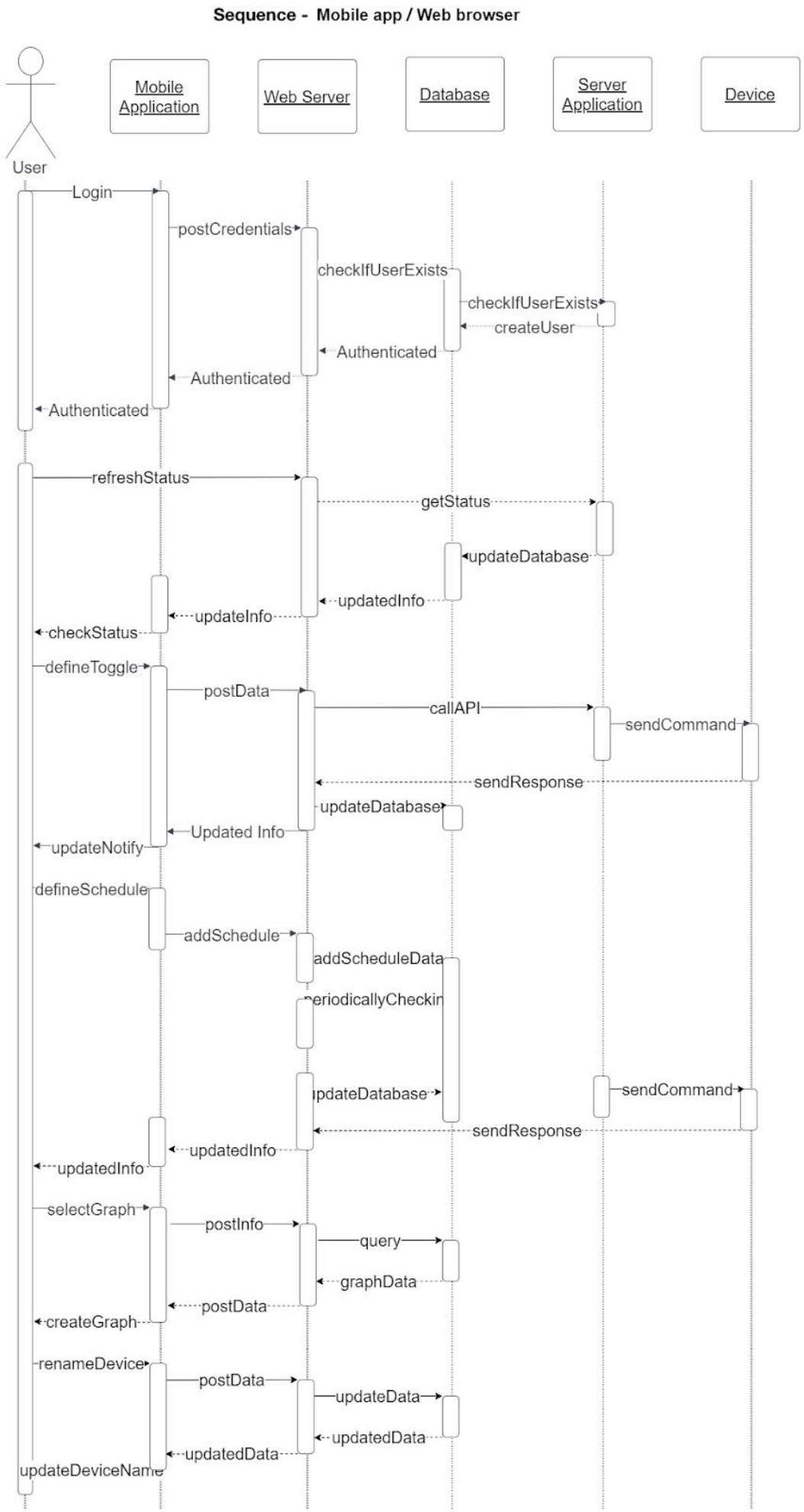


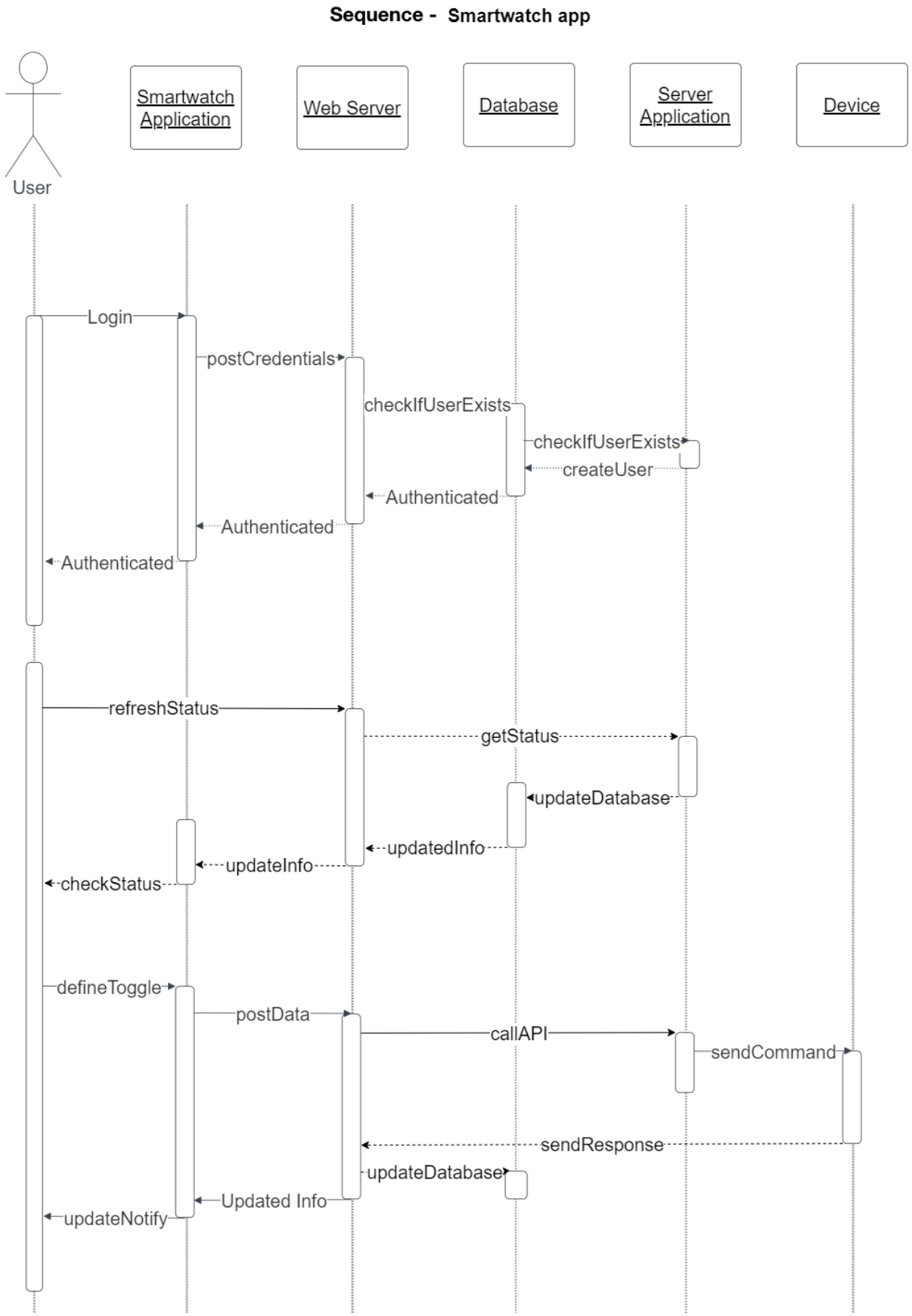
Use Case - Smartwatch app




Activity - Mobile app / Web browser
Async Login
no
Insert Credentials
no
yes
eWelink Valid
yes
Existing Customer
no
Create User in DB
Get Devices, Status, Tasks
Toggle Device
Refresh Status
Create Task
Get Graph
Rename Device
Logout


Activity - Smartwatch app
Async Login
no
Insert Credentials
no
yes
eWelink Valid
yes
Existing Customer
no
Create User in DB
Get Devices, Status, Tasks
Toggle Device
Refresh Status
Logout


Sequence - Mobile app / Web browser
User
Mobile Application
Web Server
Database
Server Application
Device
Login
postCredentials
checkIfUserExists
checkIfUserExists
createUser
Authenticated
Authenticated
Authenticated
refreshStatus
getStatus
updateDatabase
updatedInfo
updateInfo
checkStatus
defineToggle
postData
callAPI
sendCommand
sendResponse
updateDatabase
Updated Info
updateNotify
defineSchedule
addSchedule
addScheduleData
eriodicallyCheckin
pdateDatabase
sendCommand
sendResponse
updatedInfo
updatedInfo
selectGraph
postInfo
query
graphData
postData
createGraph
renameDevice
postData
updateData
updatedData
updatedData
updateDeviceName


Sequence - Smartwatch app
User
Smartwatch Application
Web Server
Database
Server Application
Device
Login
postCredentials
checkIfUserExists
checkIfUserExists
createUser
Authenticated
Authenticated
Authenticated
refreshStatus
getStatus
updateDatabase
updatedInfo
updateInfo
checkStatus
defineToggle
postData
callAPI
sendCommand
sendResponse
updateDatabase
Updated Info
updateNotify

4. Αρχιτεκτονική συστήματος

Στο συγκεκριμένο κεφάλαιο, θα παρουσιαστούν οι τεχνολογίες που χρησιμοποιήθηκαν κατά την διαδικασία ανάπτυξης της εφαρμογής, συμπεριλαμβανομένων των γλωσσών προγραμματισμού και των σχετικών εργαλείων. Επιπλέον, θα αναλυθεί ο κώδικας που αναπτύχθηκε και θα περιγράφει η ροή των πληροφοριών μεταξύ front-end - back-end και βάσης δεδομένων.

4.1. Τεχνολογίες που χρησιμοποιήθηκαν

4.1.1. Τεχνολογία Front-end

React Native

Η React-native είναι ένα ευρέως γνωστό framework της γλώσσας προγραμματισμού JavaScript και χρησιμοποιείται για την ανάπτυξη mobile native εφαρμογών για τα λειτουργικά συστήματα android και iOS. Η εταιρεία που δημοσίευσε τη συγκεκριμένη τεχνολογία είναι η Facebook και το έτος κυκλοφορίας της είναι το 2015. Η React native κατάφερε σε μικρό χρονικό διάστημα να γίνει ένα πολύ σημαντικό εργαλείο για τους προγραμματιστές σχετικά με την ανάπτυξη mobile εφαρμογών. Αξίζει να αναφέρουμε ότι μερικές από τις πιο γνωστές εταιρείες του χώρου της τεχνολογίας όπως Facebook, Instagram και Skype, χρησιμοποιούν το συγκεκριμένο framework στην ανάπτυξη των εφαρμογών τους.

Πλεονεκτήματα χρήσης της React Native:

Το πιο βασικό πλεονέκτημα στη χρήση της React Native είναι η cross platform ιδιότητα που προσφέρει στην ανάπτυξη των εφαρμογών. Με τον όρο cross platform, εννοούμε την λογική ανάπτυξης εφαρμογών οι οποίες θα είναι συμβατές σε συσκευές διαφόρων λειτουργικών συστημάτων. Έτσι οι επιχειρήσεις έχουν τη δυνατότητα να αναπτύξουν κώδικα ο οποίος θα απευθύνεται σε έναν πολύ μεγάλο όγκο συσκευών του κοινού τους, χωρίς να χρειάζεται να συντηρούν διαφορετικά project για κάθε λειτουργικό σύστημα. Με αυτόν τον τρόπο, η εταιρεία εξοικονομεί πολύ σημαντικά κόστη σε οικονομικό και παραγωγικό πλαίσιο.

Ένα ακόμη σημαντικό πλεονέκτημα της React Native είναι ότι έχει αναπτυχθεί χρησιμοποιώντας ως σημείο αναφοράς τη React.js, η οποία αποτελεί μία πολύ δημοφιλή βιβλιοθήκη της JavaScript. Το γεγονός αυτό, έδωσε πολύ σημαντικά εφόδια στη χρήση της React Native, αφού οι προγραμματιστές μπορούσαν να χρησιμοποιήσουν πολλά έτοιμα εργαλεία χωρίς να χρειάζεται να αναπτυχθούν νέα. Έτσι, οι front-end προγραμματιστές είχαν στα χέρια τους πολλές χρήσιμες επιλογές και μπορούσαν πλέον χρησιμοποιώντας project περιεκτικότερου κώδικα, να αναπτύξουν εφαρμογές που θα απευθύνονταν σε πολύ μεγαλύτερο κοινό.

Συμπερασματικά, χρησιμοποιώντας το συγκεκριμένο framework, έχουν επιτευχθεί κάποια πολύ σημαντικά ζητήματα αναφορικά με τις mobile εφαρμογές, με βασικότερα αυτά της συνοχής του κώδικα μεταξύ διαφορετικών λειτουργικών συστημάτων και την εξοικονόμηση χρόνου και κόστους κατά στην ανάπτυξή τους.

Έτσι, λαμβάνοντας υπόψη τα σημαντικά πλεονεκτήματα που προσφέρει η React Native, αποφασίστηκε η ανάπτυξη του front-end μέρους της εφαρμογής να πραγματοποιηθεί χρησιμοποιώντας το συγκεκριμένο framework της JavaScript, κάτι που καθιστά τη δυνατότητα χρήσης της εφαρμογής τόσο σε λειτουργικά συστήματα android και iOS, όσο και μέσω web browsers.

Java

Η Java είναι μία γλώσσα προγραμματισμού που βασίζεται στην αντικειμενοστρέφεια (Object-Oriented Programming) και βρίσκει εφαρμογή σε διάφορα λειτουργικά συστήματα μέσω μεταγλώττισης. Οι γνώσεις που οι προγραμματιστές έχουν σε βασικές γλώσσες προγραμματισμού όπως η C και η C++ είναι χρήσιμες, αφού υπάρχουν αρκετές ομοιότητες στον τρόπο ανάπτυξης του κώδικα.

Η συγκεκριμένη γλώσσα αριθμεί περισσότερα από 30 χρόνια ύπαρξης και εντάσσεται στις 3 πιο δημοφιλείς γλώσσες προγραμματισμού παγκοσμίως.

Ένα μεγάλο μέρος της δημοφιλίας της συγκεκριμένης γλώσσας οφείλεται στο γεγονός ότι μπορεί να λειτουργήσει σε διαφορετικά λειτουργικά συστήματα χρησιμοποιώντας ένα απαραίτητο εργαλείο, το Java Runtime Environment. Έτσι έπειτα από την εγκατάσταση του συγκεκριμένου εργαλείου ο ίδιος κώδικας μπορεί να εκτελεστεί σε διαφορετικές πλατφόρμες, κάτι που βοηθά πολύ στην εξοικονόμηση απαραίτητων πόρων για τις εταιρείες και τους προγραμματιστές. Επίσης, ένα πλεονέκτημα της συγκεκριμένης γλώσσας αποτελεί η δυνατότητα ανάπτυξης διαδικτυακών εφαρμογών. Εδώ, αξίζει να σημειώσουμε ότι στη Java οφείλεται η μετάβαση από στατικές ιστοσελίδες σε web εφαρμογές με διαδραστικό περιεχόμενο.

Λύσεις οι οποίες έχουν αναπτυχθεί χρησιμοποιώντας την java χρησιμοποιούνται καθημερινά από ένα πολύ ευρύ κοινό (π.χ., ηλεκτρονικά συστήματα συναλλαγών, κρατικών φορέων, νοσοκομείων και επιχειρήσεων).

Τέλος, η Java είναι πολύ χρήσιμη στην δημιουργία mobile εφαρμογών μιας και αποτελεί τη native γλώσσα προγραμματισμού ανάπτυξης mobile εφαρμογών για το λειτουργικό σύστημα του Android.

Android

Το Android αποτελεί το πιο διαδεδομένο λειτουργικό σύστημα κινητών συσκευών. Βασίζεται στο λειτουργικό ανοιχτού κώδικα Linux και οι εφαρμογές που αναπτύσσονται και αφορούν το συγκεκριμένο λειτουργικό σύστημα βασίζονται στη γλώσσα προγραμματισμού Java.

Έκανε για πρώτη φορά την εμφάνισή του το 2007 από την τεχνολογική εταιρεία Google και από τότε έχει καταφέρει να βρίσκεται στην πρώτη θέση της δημοφιλίας των λειτουργικών συστημάτων που αφορούν κινητές συσκευές σε παγκόσμιο επίπεδο.

Τα βασικά πλεονεκτήματα του Android είναι τα εξής:

- Open source λειτουργικό σύστημα με πολλές επιλογές προσαρμογής
- Μεγάλη κοινότητα χρηστών και προγραμματιστών
- Δυνατότητα ενσωμάτωσης εφαρμογών
- Μειωμένο κόστος ανάπτυξης εφαρμογών
- Πλούσιο περιβάλλον ανάπτυξης
- Υποστήριξη ευρείας γκάμας συσκευών

Το front-end κομμάτι της εφαρμογής που αφορά συσκευές smartwatch με λειτουργικό σύστημα android wear OS είναι υλοποιημένο σε γλώσσα Java για Android.

4.1.2. Τεχνολογία Back-end

Node.js

Το Node.js, είναι μία τεχνολογία της γλώσσας προγραμματισμού JavaScript πού απευθύνεται κυρίως στην ανάπτυξη του back-end ιστοσελίδων και εφαρμογών. Με τον όρο back-end εννοούμε το μέρος της εφαρμογής που εκτελείται από τον server (ένα φυσικό μηχάνημα) και αφορά τόσο την εκτέλεση μεγάλου όγκου εντολών όσο και τη διασύνδεση του front-end με τα υπόλοιπα μέρη της εφαρμογής όπως τη βάση δεδομένων.

Η JavaScript είναι μία ευρέως διαδεδομένη γλώσσα που αφορά την ανάπτυξη του front-end μέρους διαδικτυακών εφαρμογών και στο παρελθόν ήταν απαραίτητη η χρήση διαφορετικής γλώσσας για την ανάπτυξη του back-end μέρους αυτών. Η χρήση λοιπόν δύο διαφορετικών γλωσσών στην ίδια εφαρμογή, καθιστά την ανάπτυξη της περισσότερο απαιτητική και κοστοβόρα.

Αυτό το φαινόμενο, οδήγησε την κοινότητα της JavaScript στη δημιουργία ενός "εξατομικευμένου λειτουργικού συστήματος" που δίνει τη δυνατότητα εκτέλεσης κώδικα JavaScript ώστε να χρησιμοποιηθεί από σέρβερ μηχανήματα.
Το περιβάλλον αυτό ονομάστηκε Node.js και είναι μία ιδιαίτερα δημοφιλής επιλογή για back-end προγραμματισμό.

Τα βασικά πλεονεκτήματα του Node.js είναι τα εξής:

- Εύκολη εκμάθηση
- Ενιαία γλώσσα προγραμματισμού back-end και front-end
- Cross-platform ανάπτυξη
- Υψηλή απόδοση
- Επεκτασιμότητα
- Μεγάλη κοινότητα χρήσης και υποστήριξης

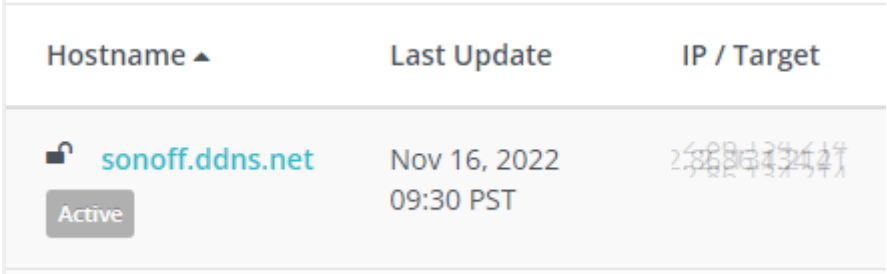
Για την ανάπτυξη του back-end μέρους της εφαρμογής, χρησιμοποιήθηκε το Node.js.
Ο server της εφαρμογής τρέχει σε ένα μεμονωμένο μηχάνημα με το οποίο επικοινωνούν οι συσκευές των χρηστών της εφαρμογής για την εκτέλεση των λειτουργιών της.

No-IP

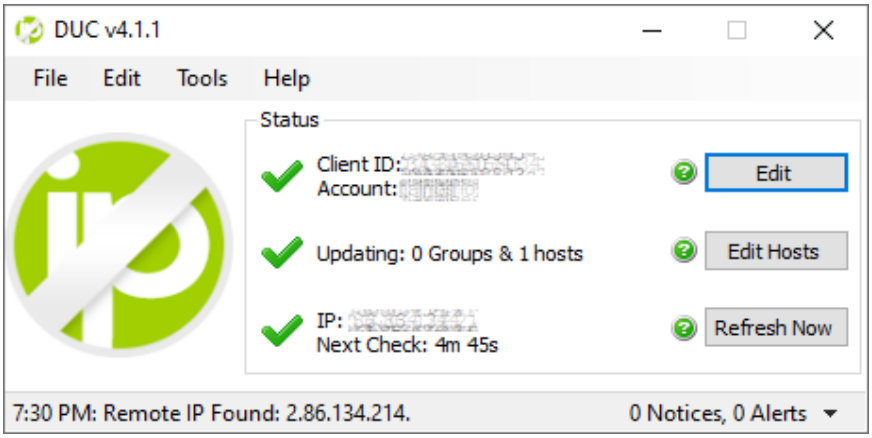
Για την επικοινωνία μεταξύ back-end και front-end, ήταν απαραίτητη η κατοχύρωση κάποιας σταθερής IP διεύθυνσης για το μηχάνημα server που εκτελεί τον κώδικα του backend.

Για το λόγο αυτό, χρησιμοποιήθηκε η υπηρεσία No-IP, μέσω της οποίας η εφαρμογή έχει κατοχυρώσει το hostname **sonoff.ddns.net** το οποίο δείχνει πάντα την εξωτερική IP του server. Έτσι, όλη η επικοινωνία μεταξύ front-end και back-end που γίνεται με requests, χρησιμοποιεί το συγκεκριμένο hostname το οποίο οδηγεί στην IP όπου λειτουργεί ο server.

Παράλληλα, στον server λειτουργεί ένας client της εφαρμογής No-IP, ο Dynamic Update Client (DUC), ο οποίος έχει αναλάβει να δηλώνει στην υπηρεσία της No-IP την τρέχουσα εξωτερική IP του server. Έτσι, παρέχεται η διασφάλιση ότι ακόμα και στην περίπτωση που η IP αλλάξει, θα πραγματοποιηθεί η απαιτούμενη ενημέρωση ώστε το hostname sonoff.ddns.net να συνεχίσει να δείχνει στην νέα IP του server.

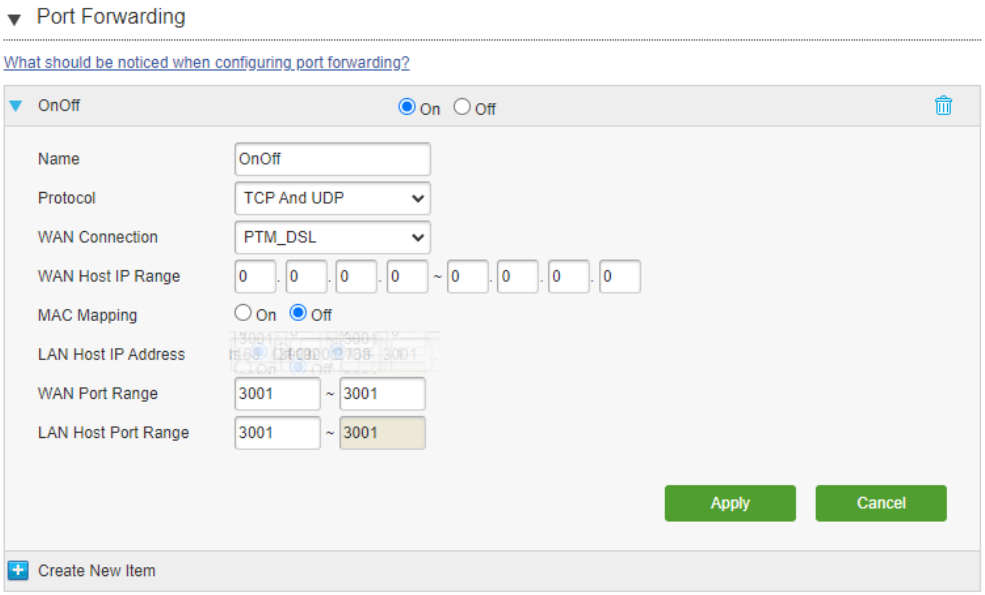


Το hostname στην υπηρεσία No-IP



Ο Dynamic Update Client

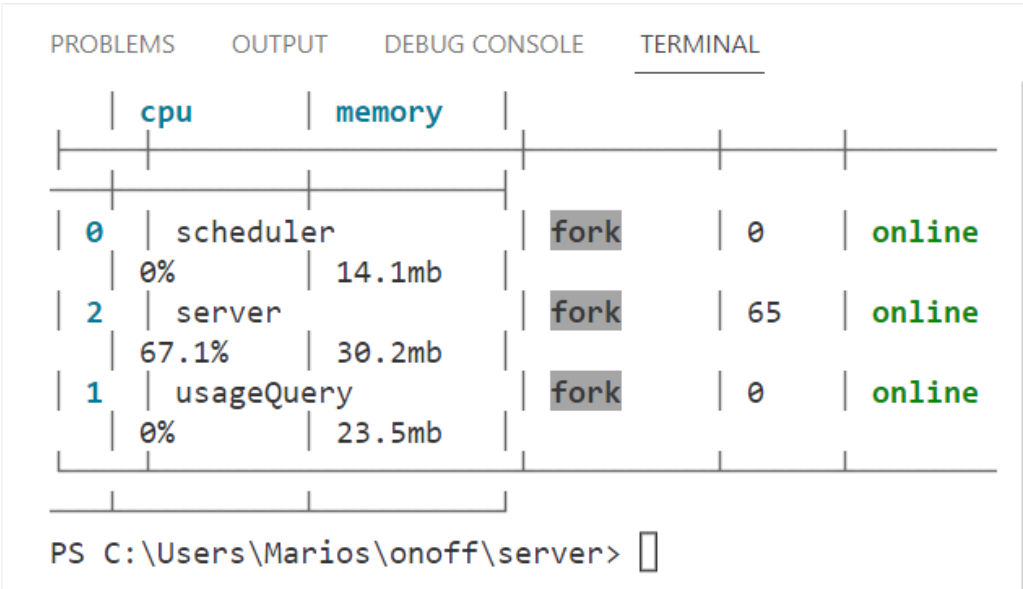
Επιπλέον, ήταν απαραίτητη η ρύθμιση Port Forwarding στο router που είναι συνδεδεμένος ο server, ώστε να προωθεί όλα τα request που πραγματοποιούνται στο **sonoff.ddns.net:3001** (3001 είναι η πόρτα στην οποία εκτελείται ο κώδικας του server) στην local IP του server στο τοπικό δίκτυο, ώστε να επιτευχθεί η απαιτούμενη επικοινωνία.



Η ρύθμιση για το Port Forwarding στην local IP του server στις ρυθμίσεις του router.

PM2

Για να μην προκύψει πρόβλημα στην εκτέλεση του κώδικα του backend, και για να διασφαλιστεί η αδιάκοπη λειτουργία του, ο κώδικας του server εκτελείται μέσω της διεργασίας PM2, η οποία βοηθά να διατηρηθεί η εφαρμογή σε λειτουργία 24/7 στην περίπτωση εμφάνισης οποιουδήποτε σφάλματος.



Η διεργασία PM2 σε λειτουργία για τις εργασίες του χρήστη

4.1.3. Βάση Δεδομένων

Στο μεγαλύτερο μέρος των εφαρμογών που χρησιμοποιούνται από ηλεκτρονικές συσκευές, υπάρχει ανάγκη αποθήκευσης και ανάκτησης συγκεκριμένων δεδομένων.
Έτσι δημιουργήθηκε η έννοια της βάσης δεδομένων όπου με το συγκεκριμένο όρο εννοούμε μία μεμονωμένη εφαρμογή η οποία εκτελεί ενέργειες δημιουργίας, αποθήκευσης, επεξεργασίας, διαγραφής ανάκτησης δεδομένων. Μαζί με την ύπαρξη του front-end και του back-end, η βάση δεδομένων συμπληρώνει τα τρία βασικά στοιχεία για τη δημιουργία μιας σύγχρονης εφαρμογής.

Στις σύγχρονες βάσεις δεδομένων χρησιμοποιούνται κυρίως πίνακες τους οποίους οι προγραμματιστές δημιουργούν ανάλογα με τις ανάγκες της εκάστοτε εφαρμογής ώστε να επιτύχουν την αποθήκευση και διαχείριση των απαραίτητων δεδομένων.

Ένα σημαντικό στοιχείο που αφορά τους πίνακες της βάσης δεδομένων είναι η δυνατότητα δημιουργίας σχέσης μεταξύ των δεδομένων τους χρησιμοποιώντας "κλειδιά". Αυτή η ιδιότητα, είναι ιδιαίτερα χρήσιμη τόσο στην αποθήκευση και διαχείριση μεγάλου όγκου δεδομένων, όσο και στις δυνατότητες που παρέχονται στους προγραμματιστές για την ανάκτηση και τη γενικότερη διαχείριση αυτών.

MySQL

Η MySQL είναι ένα ευρέως γνωστό σύστημα βάσης δεδομένων και κατακτά τη δεύτερη θέση στη δημοφιλία των προγραμματιστών. Ωστόσο, μιλώντας για βάσεις δεδομένων ανοιχτού κώδικα, εκεί η MySQL έχει τη θέση του πιο δημοφιλούς συστήματος βάσης δεδομένων.
Τα πιο σημαντικά στοιχεία που έχουν οδηγήσει στην κατάκτηση της συγκεκριμένης επιτυχίας είναι η αξιοπιστία και η αποδοτικότητά της.

Τα βασικά πλεονεκτήματα της MySQL είναι τα εξής:

1. Δωρεάν διάθεση
2. Open source κώδικας, όπου δίνει τη δυνατότητα παραμετροποίησης στους προγραμματιστές
3. Υψηλή ασφάλεια δεδομένων
4. Διαχείριση μεγάλου όγκου δεδομένων
5. Μεγάλη κοινότητα χρηστών και υποστήριξης
6. Μικρές απαιτήσεις σε επίπεδο hardware

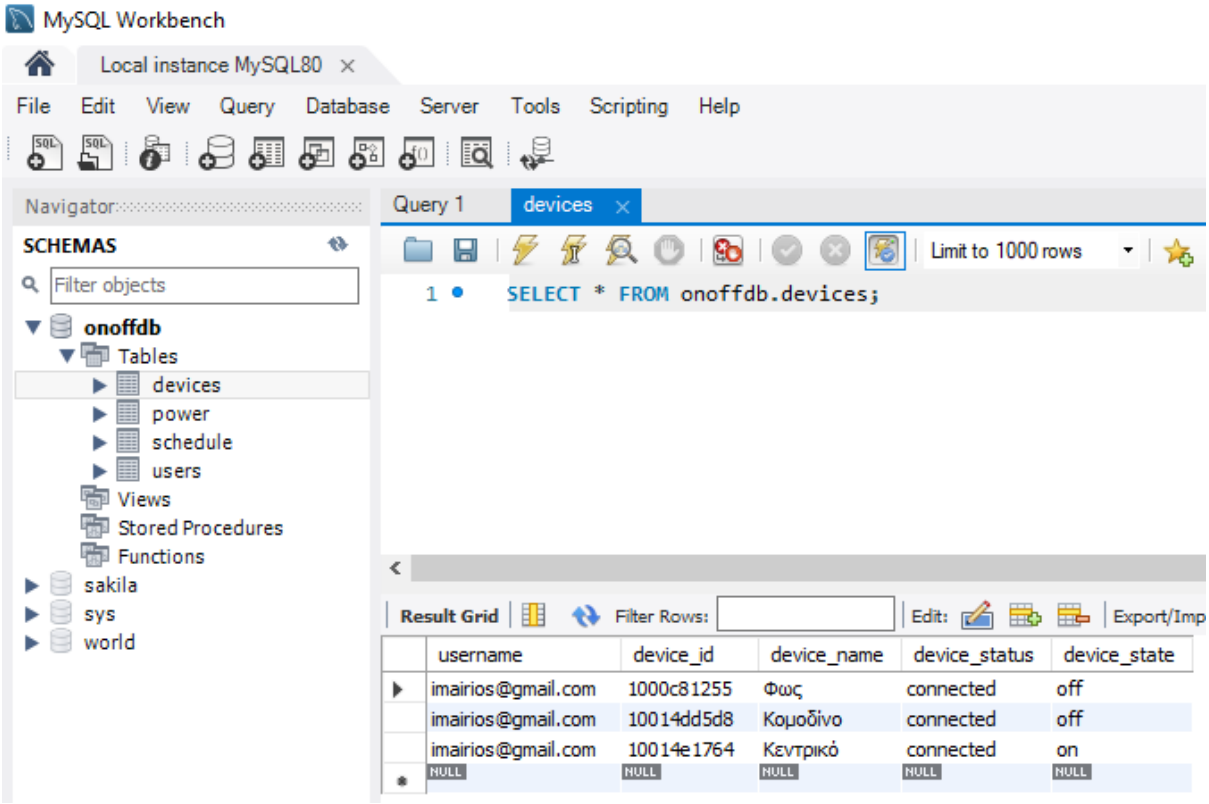
Η ανάπτυξη της βάσης δεδομένων της εφαρμογής, πραγματοποιήθηκε χρησιμοποιώντας την τεχνολογία της MySQL, η οποία έχει εγκατασταθεί στο ίδιο φυσικό μηχάνημα όπου λειτουργεί και ο server της εφαρμογής, ώστε να κάνει την μεταξύ τους επικοινωνία περισσότερο άμεση και αποδοτική.

4.2. Πίνακες βάσης δεδομένων

Ακολουθούν χρήσιμες πληροφορίες σχετικά με τους πίνακες της βάσης δεδομένων της εφαρμογής.

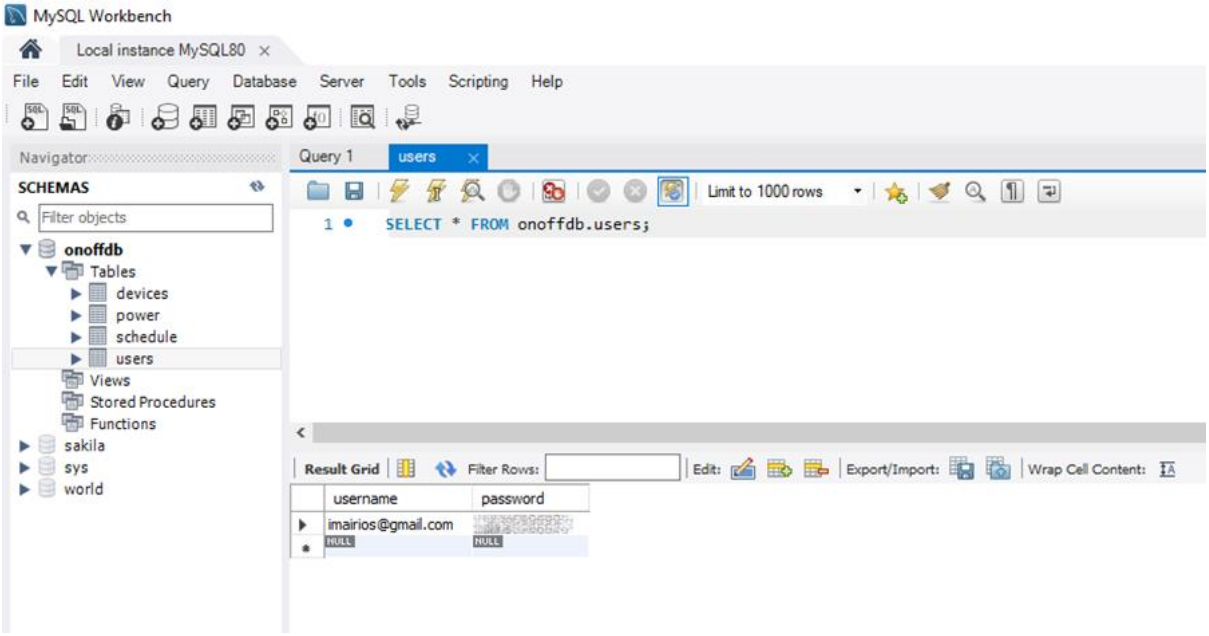
4.2.1. Πίνακας devices

Ο ακόλουθος πίνακας, περιέχει τις απαραίτητες πληροφορίες για τις συσκευές των χρηστών της εφαρμογής. Κατά την είσοδο στην εφαρμογή, εκτελείται μία συνάρτηση ελέγχου ύπαρξης των συσκευών του χρήστη στην εφαρμογή, και στην περίπτωση νέων συσκευών, εκτελείται η σχετική εγγραφή των δεδομένων του. Επίσης πραγματοποιείται ενημέρωση των απαραίτητων στοιχείων του συγκεκριμένου πίνακα κάθε φορά που εκτελείται μία ενέργεια αλλαγής κατάστασης λειτουργίας κάποιας συσκευής.



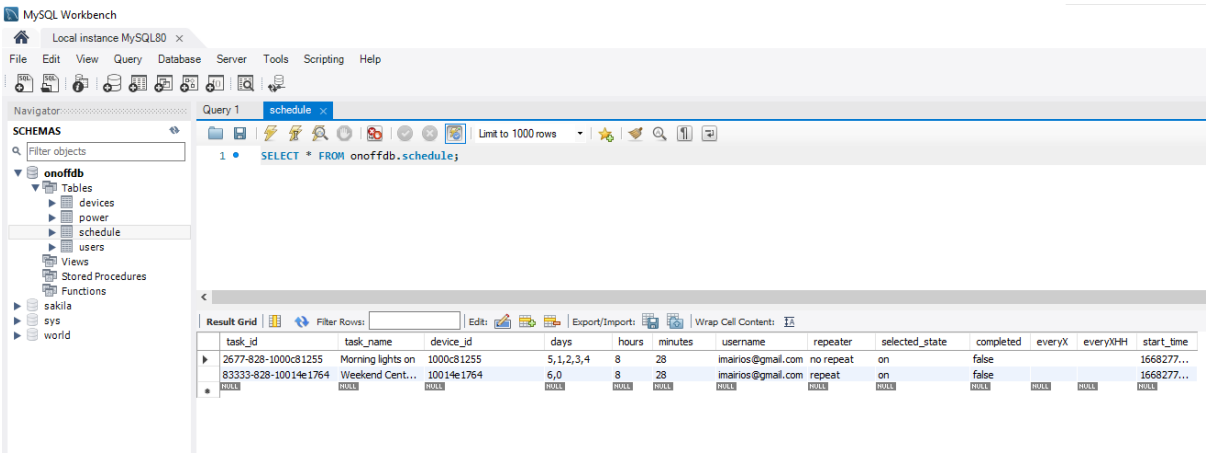
4.2.2. Πίνακας users

Στο συγκεκριμένο πίνακα εμπεριέχεται η πληροφορία με τα διαπιστευτήρια εισόδου του κάθε χρήστη της εφαρμογής. Κατά την είσοδο στην εφαρμογή, εκτελείται μία συνάρτηση ελέγχου ύπαρξης του συγκεκριμένου χρήστη στην εφαρμογή, και στην περίπτωση που δεν υπάρχει ήδη, εκτελείται η σχετική εγγραφή των δεδομένων του.



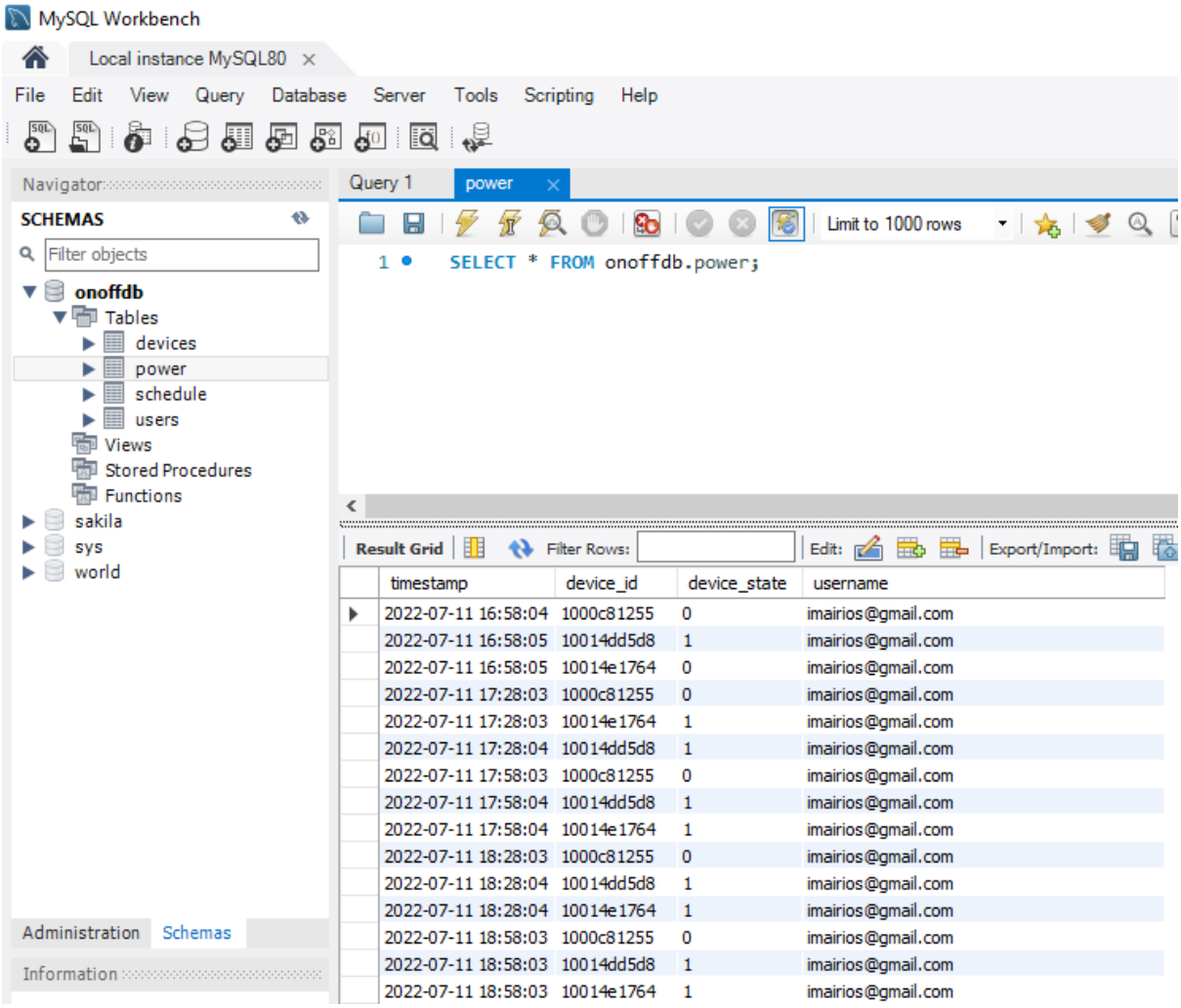
4.2.3. Πίνακας schedule

Στον ακόλουθο πίνακα, αποθηκεύονται οι πληροφορίες που αφορούν τα tasks που δημιουργεί ο χρήστης στη σελίδα Schedule της εφαρμογής. Μέσω της συνάρτησης scheduler η οποία τρέχει σε επανάληψη με interval ενός λεπτού, πραγματοποιείται έλεγχος για το αν κάποιο task πρέπει να εκτελεστεί. Κατά την εκτέλεση ενός task, ενημερώνεται η βάση δεδομένων στον πίνακα devices με το νέο state της συσκευής (on/off), και αν δεν υπάρχει επιλεγμένη επόμενη ημερομηνία εκτέλεσης του συγκεκριμένου task, ενημερώνεται η στήλη completed ώστε να μην εκτελεστεί ξανά από την σχετική συνάρτηση.



4.2.4. Πίνακας power

Στον ακόλουθο πίνακα, αποθηκεύονται οι πληροφορίες που αφορούν την κατάσταση λειτουργίας κάθε συσκευής μέσω της συνάρτησης usageQuery. Η συγκεκριμένη συνάρτηση εκτελείται κάθε 30 λεπτά και ενημερώνει τις τιμές του πίνακα. Η πληροφορία του πίνακα power, χρησιμοποιείται για τη λειτουργία γραφημάτων της σελίδας Schedule.



4.3. Κώδικας λειτουργιών της εφαρμογής mobile/web

4.3.1. Login - Front-end

Η συνάρτηση onSubmit εκτελείται ασύγχρονα κατά το πάτημα του σχετικού button στη Login σελίδα της εφαρμογής. Χρησιμοποιεί ένα post request, όπου στέλνει στο back-end τα username και password που εισήγαγε ο χρήστης. ‘Επειτα, ο server ελέγχει τα παραπάνω δεδομένα και επιστρέφει στο front-end την πληροφορία για την εγκυρότητα των στοιχείων που εισήγαγε, ώστε να οδηγήσει τον χρήστη στη Home σελίδα της εφαρμογής, ή να τον κρατήσει στη σελίδα Login. Επίσης, δημιουργεί ένα token το οποίο χρησιμοποιείται από το AsyncStorage της εφαρμογής ώστε να μπορεί ο χρήστης να παραμένει συνδεδεμένος στο λογαριασμό του ακόμα και αν η εφαρμογή κλείσει ή η συσκευή απενεργοποιηθεί.

```
const onSubmit = async () => {
    setIsLoading(true);
    console.log("first");
    if (username == "") {
      navigation.navigate("Home");
      setModalVisible(false);
      setIsLoading(false);
    }
    axios
      .post(
        "http://sonoff.ddns.net:3001/testLogin",
        {
          credentials: [
            {
              username: username,
              password: password,
            },
          ],
        },
        {}
      )
      .then((response) => {
        console.log(response.data.message);
        if (response.data.message == "Login successful") {
          console.log("Button logged");
          AsyncStorage.setItem("token", username);
          navigation.navigate("Home");
          setIsLoading(false);
          setModalVisible(false);
        } else {
          console.log("button login failed");
          setDetails(response.data.message);
          console.log(response.data.message);
          setIsLoading(false);
          setModalVisible(true);
        }
      })
      .catch((err) => {
        console.log(err);
      });
  };
```

4.3.2. Async login - Front-end

Η παρακάτω συνάρτηση ελέγχει για την ύπαρξη token ώστε να εκτελέσει την αυτόματη σύνδεση-login στο λογαριασμό του χρήστη. Το συγκεκριμένο token, το οποίο δημιουργείται κατά τη διαδικασία του login, παραμένει αποθηκευμένο στο storage της εφαρμογής αφού επιτευχθεί ο έλεγχος εγκυρότητας των διαπιστευτηρίων εισόδου του χρήστη.

```
const tokenlogin = async () => {
  const value = await AsyncStorage.getItem("token");
  if (value !== null) {
    navigation.navigate("Home");
    console.log("async logged", value);
    setModalVisible(false);
    setIsLoading(false);
  } else {
    console.log("async failed");
  }
};
```

4.3.3. Login - Backend

Από τη μεριά του backend, ο server λαμβάνει το post request του χρήστη το οποίο περιέχει τις τιμές που εισήγαγε στα πεδία της login φόρμας. Χρησιμοποιώντας αυτά τα δεδομένα και το API της eWelink πραγματοποιεί έλεγχο των διαπιστευτηρίων εισόδου.

```
app.post("/testLogin", (req, res) => {
  var jsonData = req.body;
  var jsonParsed = JSON.parse(JSON.stringify(jsonData))
  var username = jsonParsed.credentials[0].username;
  var password = jsonParsed.credentials[0].password;
  console.log(username);
  console.log(password);
  const ewelink = require("ewelink-api");
  var dev = [];
  (async () => {
    const connection = new ewelink({
      email: username,
      password: password,
      region: "eu",
    });

```

Στη συνέχεια, χρησιμοποιώντας το δοθέν username, πραγματοποιείται έλεγχος για την ύπαρξη του συγκεκριμένου λογαριασμού στον πίνακα users της βάσης δεδομένων. Αν δεν υπάρχει λογαριασμός με αυτό το username, δημιουργείται μέσω του σχετικού query στη βάση δεδομένων.

```
con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    for (let i = 0; i < data2.length; i++) {
      var sql0 = `SELECT username FROM users WHERE username LIKE '${username}'`;
      con.query(sql0, function (err, results) {
        if (err) throw err;
        if (results.length > 0) {
          console.log("user already exist");
        } else {
          console.log(data[i]);
          var sql = `INSERT INTO users (username, password) VALUES ('${username}', '${password}')`;
          con.query(sql, function (err, result) {
            if (err) throw err;
            console.log("record/s inserted");
          });
        }
      });
    }
  });
```

Η συνάρτηση συνεχίζει ελέγχοντας τις τιμές που γυρίζει το API, και επιστρέφει το αντίστοιχο μήνυμα για την επιτυχία/αποτυχία εισόδου. Στη συνέχεια, πραγματοποιεί έλεγχο στις συνδεδεμένες συσκευές του χρήστη και αποθηκεύει τις απαιτούμενες πληροφορίες για κάθε μία από αυτές: Username, Device id, Device name, Device status, Device state

```
17      const devices = await connection.getDevices();
18      const mess = JSON.stringify(devices);
19      if (mess.includes("failed")) {
20        console.log("Login Failed");
21        res.json({ message: "Login Failed" });
22      } else {
23        console.log("Login successful");
24        res.json({ message: "Login successful" });
25      }
26      var data = [];
27      var data2 = [];
28      for (device in devices) {
29        var sonoff = {};
30        var deviceId = devices[device]["deviceid"];
31        var deviceName = devices[device]["name"];
32        const status = await connection.getDevicePowerState(
33          devices[device]["deviceid"]
34        );
35        if (status.status == "ok") {
36          var deviceStatus = "connected";
37          var deviceState = status.state;
38        } else {
39          device.status = "disconnected";
40          var deviceState = "not available";
41        }
42        console.log(deviceStatus, deviceState);
43        const auth = await connection.getCredentials();
44        sonoff.username = username;
45        sonoff.id = deviceId;
46        sonoff.name = deviceName;
47        sonoff.status = deviceStatus;
48        sonoff.state = deviceState;
49        data.push(sonoff);
50        data2.push(sonoff);
51      }
52
```

Τέλος, πραγματοποιείται έλεγχος στη βάση δεδομένων για την ύπαρξη νέας συσκευής στο account του χρήστη. Αν πρόκειται για νέα συσκευή, θα πραγματοποιηθεί η αντίστοιχη καταχώρηση στον πίνακα devices της βάσης δεδομένων.

```
    var mysql = require("mysql");
    var con = mysql.createConnection({
      host: "localhost",
      user: "root",
      password: "[redacted]",
      database: "onoffdb",
    });
    var data = data.map(Object.values);
    // console.log(data2[i].id)
    con.connect(function (err) {
      if (err) throw err;
      console.log("Connected!");
      for (let i = 0; i < data2.length; i++) {
        var sql0 = `SELECT device_id FROM devices WHERE device_id LIKE '${data2[i].id}'`;
        con.query(sql0, function (err, results) {
          if (err) throw err;
          // console.log(results);
          if (results.length > 0) {
            console.log("device already exist");
          } else {
            console.log(data[i]);
            var sql = `INSERT INTO devices (username, device_id, device_name, device_status, device_state)
            VALUES ('${data2[i].username}', '${data2[i].id}', '${data2[i].name}', '${data2[i].status}', '${data2[i].state}')`;
            con.query(sql, function (err, result) {
              if (err) throw err;
              console.log("record/s inserted");
            });
          }
        });
      }
    });
  })();
});
```

4.3.4. Get devices - Front-end

Η συγκεκριμένη συνάρτηση καλείται από το front-end για να φέρει από το back-end τη λίστα των συσκευών του χρήστη, την οποία αποθηκεύει σε ένα state, που το χρησιμοποιεί ώστε να δημιουργήσει τα απαραίτητα button στο UI.

```
useEffect(() => {
  axios.get("http://sonoff.ddns.net:3001/devices").then((response) => {
    const data = response.data.dev;
    setDevices([...data]);
  });
}, []);
```

4.3.5. Get devices - Backend

Ο server δέχεται το request και αναζητεί στον πίνακα devices της βάσης δεδομένων τις ζητούμενες πληροφορίες, τις οποίες και επιστρέφει στο front-end.

```
app.get("/devices", (req, res, next) => {
  console.log("Getting devices...");
  var dev = [];
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "[redacted]",
    database: "onoffdb",
  });
  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = `SELECT device_name AS label, device_name AS label0, device_id AS value FROM devices`;
    con.query(sql, function (err, results) {
      if (err) throw err;
      const dev = Object.values(JSON.parse(JSON.stringify(results)));
      res.json({ dev });
      console.log(dev);
    });
  });
});
```

4.3.6. Toggle - Front-end

Η ακόλουθη συνάρτηση onPressHandler, καλείται όταν ο χρήστης επιλέξει κάποιο device button ώστε να κάνει toggle (on/off) κάποιο διακόπτη. Έτσι, πραγματοποιείται ένα post request στο backend, το οποίο περιέχει το id της συγκεκριμένης συσκευής.

```
onPressHandler(id) {
  this.setState({ selectedItem: id });
  axios.post("http://sonoff.ddns.net:3001/toggle/" + id);
}
```

4.3.7. Toggle - Backend

Ο server λαμβάνει το post request από το front-end και καλεί τη συνάρτηση onoff ώστε να εκτελεστούν οι απαιτούμενες εντολές για τη διαχείριση της λειτουργίας της συσκευής με το συγκεκριμένο id.

```
1  app.post("/toggle/:id", (req, res) => {
2    onoff(req.params.id);
3    console.log("Toggling...");
4    res.send(req.params.id);
5  });
```

4.3.8. Onoff -Backend

Σε επίπεδο back-end η συνάρτηση onoff πραγματοποιεί αναζήτηση στον πίνακα users της βάσης δεδομένων, για την εύρεση των username και password του account όπου η συγκεκριμένη συσκευή είναι καταχωρημένη. Στη συνέχεια, χρησιμοποιώντας το API της eWeLink, πραγματοποιεί την απαραίτητη αλλαγή στη λειτουργία της συσκευής. Στη συνέχεια, θα ενημερώσει τον σχετικό πίνακα devices στη βάση δεδομένων, με τη νέα κατάσταση της συσκευής.

```
module.exports = function (id) {

  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "mrskrls699697",
    database: "onoffdb",
  });

  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var sql0 = `SELECT username FROM devices WHERE device_id = '${id}'`;
    con.query(sql0, function (err, results) {
      if (err) throw err;
      console.log(results);
      const user = Object.values(JSON.parse(JSON.stringify(results)));
      console.log(user);
      usr = user[0].username;
      var sql1 = `SELECT password FROM users WHERE username = '${usr}'`;
      con.query(sql1, function (err, results) {
        if (err) throw err;
        console.log(results);
        const pass = Object.values(JSON.parse(JSON.stringify(results)));
        console.log(pass);
        pswrd = pass[0].password;
        const ewelink = require("ewelink-api");
        (async () => {
          const connection = new ewelink({
            email: usr,
            password: pswrd,
            region: "eu",
          });
          await connection.toggleDevice(id);
          const status = await connection.getDevicePowerState(id)
          var devstat = status.state;
          console.log(devstat)
          var sql3 = `UPDATE devices SET device_state = '${devstat}' WHERE device_id = '${id}';`;
          con.query(sql3, function (err, result) {
            if (err) throw err;
            console.log("record/s updated");
          });
        })();
      });
    });
  });
};
```

4.3.9. Refresh button - Front-end

Το συγκεκριμένο button καλεί την statusRefresh συνάρτηση για να πραγματοποιηθεί ανανέωση στις πληροφορίες των συσκευών (status: online/offline, state: on/off etc.).

```
<TouchableOpacity onPress={this.statusRefresh}>
            <Image
              style={styles.refresh}
              source={require("../assets/refresh.png")}
            />
          </TouchableOpacity>
```

4.3.10. Refresh - Front-end

Η statusRefresh, η οποία καλείται κατά την εκκίνηση της εφαρμογής και κατά το πάτημα του refreshButton, στέλνει στο back-end ένα post request το οποίο περιέχει το username του χρήστη (το οποίο είναι αποθηκευμένο στο AsyncStorage), ζητώντας του την τρέχουσα κατάσταση των συσκευών. Όταν λάβει τη ζητούμενη πληροφορία, θα την αποθηκεύσει σε ένα state το οποίο θα χρησιμοποιήσει από κάθε στοιχείο του UI ώστε να ενημερώσει αντίστοιχα τα στοιχεία του.

```
statusRefresh = async () => {
    this.setState({ isLoading: true });
    const username = await AsyncStorage.getItem("token");
    axios
      .get("http://sonoff.ddns.net:3001/status", {
        params: {
          username: username,
        },
      })
      .then((response) => {
        this.setState({ ...this.state, devices: response.data.stat });
        this.setState({ isLoading: false });
      })
      .catch((error) => {
        console.log(error);
        return error;
      });
  };
```

Refresh - Backend

Ο server λαμβάνει το request και αντλεί τις απαιτούμενες πληροφορίες για τον λογαριασμό του χρήστη από τον πίνακα users της βάσης δεδομένων.
Έπειτα, χρησιμοποιώντας αυτές τις πληροφορίες, χρησιμοποιεί το eWeLink API για να αποθηκεύσει την τρέχουσα κατάσταση των συσκευών του.

```
app.get("/status", (req, res, next) => {
  console.log("Getting status...");
  let param = req.query.username;
  console.log(param, "usrnm");
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: [redacted],
    database: "onoffdb",
  });

  const ewelink = require("ewelink-api");
  var stat = [];
  var pswrd = "";

  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var sql0 = `SELECT password FROM users WHERE username = '${param}'`;
    con.query(sql0, function (err, results) {
      if (err) throw err;
      console.log(results);
      const pass = Object.values(JSON.parse(JSON.stringify(results)));
      console.log(pass);
      pswrd = pass[0].password;
    });
    var sql = `SELECT device_name, device_id, device_status FROM devices WHERE username = '${param}'`;
    con.query(sql, function (err, results) {
      if (err) throw err;
      const resulter = Object.values(JSON.parse(JSON.stringify(results)));
      (async () => {
        const connection = new ewelink({
          email: param,
          password: pswrd,
          region: "eu",
        });
```

Τέλος, αποθηκεύει όλη την πληροφορία της κατάστασης των συσκευών του χρήστη σε έναν πίνακα, τον οποίο και επιστρέφει στο front-end. Η σελίδα Home, θα χρησιμοποιήσει αυτά τα δεδομένα για να ενημερώσει τα στοιχεία του UI.

```
38          /* get all devices */
39          const devices = await connection.getDevices();
40          for (device in devices) {
41            const status = await connection.getDevicePowerState(
42              devices[device]["deviceid"]
43            );
44            var devstat = status.state;
45            var dName;
46            for (let i = 0; i < resulter.length; i++) {
47              if (resulter[i].device_id == devices[device]["deviceid"]) {
48                dName = resulter[i].device_name;
49                dConnection = resulter[i].device_status;
50                dSsid = devices[device]["params"]["ssid"];
51                if (dSsid == undefined) {
52                  dSsid = "-";
53                }
54              }
55            }
56            stat.push({
57              name: dName,
58              id: devices[device]["deviceid"],
59              connection: dConnection,
60              status: devstat,
61              ssid: dSsid,
62              brand: devices[device]["brandName"],
63            });
64          }
65          res.json({ stat });
66        })();
67      });
68    });
69  });
```

4.3.11. Get tasks - Front-end

Η συνάρτηση getTasks στέλνει ένα post request στον server το οποίο περιέχει το username του χρήστη αναζητώντας τα tasks τα οποία έχει καταχωρήσει ο χρήστης και είναι ακόμα εν ενεργεία. Αυτήν την πληροφορία, θα την αποθηκεύσει σε ένα state και θα τη χρησιμοποιήσει για να δημιουργήσει τη σχετική λίστα με τα tasks.

```
const getTasks = async () => {
    const username = await AsyncStorage.getItem("token");
    console.log(username);
    axios
      .post("http://sonoff.ddns.net:3001/tasks", {
        user: [
          {
            username: username,
          },
        ],
      })
      .then((response) => {
        const data = response.data;
        console.log(data);
        setTaskItems(data);
      })
      .catch((err) => {
        console.log(err);
      });
  };
```

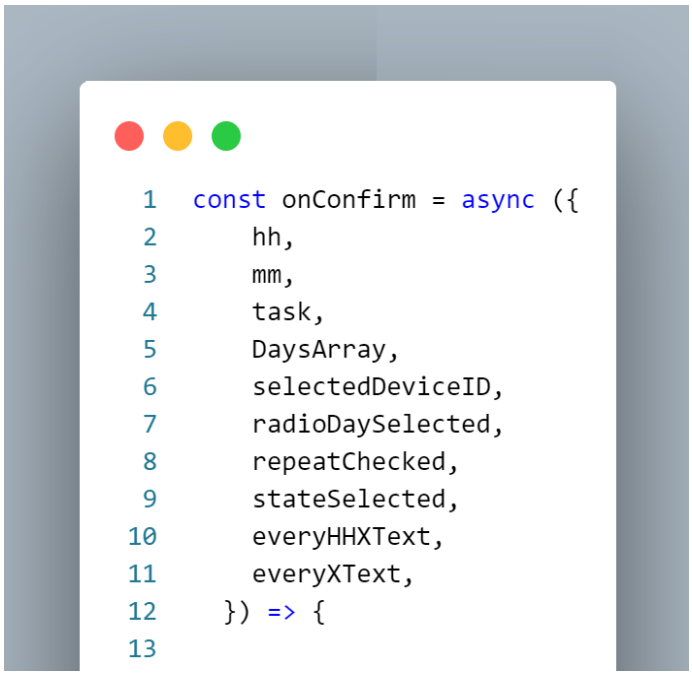
4.3.12. Get tasks - Backend

Ο server λαμβάνει το request και αναζητεί στον πινακα tasks τις καταχωρήσεις του χρήστη με το δοθέν username. Επιστρέφει την πληροφορία στο front-end μέσω της μεταβλητής tasks.

```
app.post("/tasks", (req, res) => {
  var jsonData = req.body;
  // var jsonParsed = JSON.parse(JSON.stringify(jsonData));
  console.log(jsonData, "sk");
  var username = jsonData.user[0].username;
  console.log("got", username);
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "[redacted]",
    database: "onoffdb",
  });

  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = `SELECT task_name FROM schedule WHERE username LIKE '${username}'" `;
    con.query(sql, function (err, results) {
      if (err) throw err;
      // console.log(results);
      var tasks = results.map(function (result) {
        return result["task_name"];
      });
      console.log(tasks), res.json(tasks);
    });
  });
});
```

4.3.13. Add task - Front-end

Κατά καταχώρηση νέου task, ο χρήστης καλεί τη συνάρτηση onConfirm, στην οποία εκχωρεί όλη την πληροφορία που έχει επιλέξει μέσω του UI της σχετικής φόρμας.



```
1  const onConfirm = async ({
2      hh,
3      mm,
4      task,
5      DaysArray,
6      selectedDeviceID,
7      radioDaySelected,
8      repeatChecked,
9      stateSelected,
10     everyHHXText,
11     everyXText,
12   }) => {
13
```

Οι συγκεκριμένες πληροφορίες αποθηκεύονται στα αντίστοιχα state μέσω των στοιχείων του UI που χρησιμοποιεί ο χρήστης.

Τέτοια στοιχεία είναι:

- Text input για την εισαγωγή ονόματος για το task
- Radio buttons για την επιλογή του χρόνου εκτέλεσης του task
- Time picker για την επιλογή συγκεκριμένης ώρας εκτέλεσης
- Dropdown menu για την επιλογή συσκευής
- Multiple buttons για την επιλογή ημέρας/ών
- Checkbox για τη επιλογή της εβδομαδιαίας επανάληψης

Στη συνέχεια, πραγματοποιείται έλεγχος εγκυρότητας των τιμών που έχει καταχωρήσει ο χρήστης.

```
14      if (
15        task === undefined ||
16        task === null ||
17        task == "" ||
18        !task.replace(/\s/g, "").length
19      ) {
20        console.log("no task");
21        chek.push("Task Name, ");
22      }
23
24      if (radioTimeSelected == "Specific Time") {
25        if (hh === undefined || hh === null || hh == "") {
26          console.log("no hh");
27          chek.push("Picked Hours, ");
28        }
29        if (mm === undefined || mm === null || mm == "") {
30          console.log("no mm");
31          chek.push("Picked minutes, ");
32        }
33      } else if (radioTimeSelected == "Every HH:X") {
34        if (
35          everyHHXText === undefined ||
36          everyHHXText === null ||
37          everyHHXText == ""
38        ) {
39          console.log(everyHHXText, "no HH:X");
40          chek.push("HH:X minutes, ");
41        }
42      } else if (radioTimeSelected == "Every X Mins") {
43        if (everyXText === undefined || everyXText === null || everyXText == "") {
44          console.log("no X Mins");
45          chek.push("X Mins minutes, ");
46        }
47      } else {
48        chek.push("Time selection, ");
49      }
50
51      if (radioDaySelected == "Other") {
52        if (DaysArray.length == 0) {
53          console.log("4", DaysArray.length);
54          chek.push("Selected Days, ");
55        }
56      }
57      if (selectedDeviceID.length == 0) {
58        console.log("no devices");
59        chek.push("Selected Devices, ");
60      }
```

Στην περίπτωση που οι καταχωρήσεις είναι έγκυρες, καλείται η συνάρτηση handleAddTask για να προσθέσει το task στη σχετική λίστα του UI και συνεχίζει με τις απαραίτητες ενέργειες επικοινωνίας με το backend. Δημιουργείται ένα μοναδικό task_id για κάθε task, το οποίο θα χρησιμοποιηθεί ως primary key στον πίνακα schedule της βάσης δεδομένων όπου πρόκειται να εκχωρηθεί το task.

```
61      setChecker(chek);
62      if (chek.length == 0) {
63        setModalVisible(!modalVisible);
64        handleAddTask();
65        setVisible(false);
66        var now = new Date();
67        var dayNow = now.getDay();
68        console.log(dayNow);
69        if (radioDaySelected == "Today") {
70          DaysArray = [dayNow];
71        }
72        var devicePostID = selectedDeviceID.toString();
73        console.log(selectedDeviceID);
74        devicePostID = devicePostID.replace(/,/g, "-");
75        const username = await AsyncStorage.getItem("token");
76        var task_id =
77          Math.floor(Math.random() * 100000) +
78          1 +
79          "-" +
80          hh +
81          mm +
82          "-" +
83          devicePostID;
```

Εδώ, η συνάρτηση ολοκληρώνει τις εντολές της, στέλνοντας όλη την πληροφορία του χρήστη στο back-end μέσω post request.

Οι παρακάτω μεταβλητές αφορούν:

1. **selectedDeviceID**: το id της συσκευής
2. **days:** τις ημέρες εκτέλεσης
3. **hours:** την ώρα εκτέλεσης
4. **minutes**: τα λεπτά εκτέλεσης
5. **task_name:** το όνομα του task
6. **username**: το username του χρήστη
7. **repeat:** την επιλογή εβδομαδιαίας επανάληψης
8. **selectedState:** την επιλεγμένη λειτουργία για τη συσκευή
9. **everyX:** την επιλογή για εκτέλεση ανά Χ λεπτά
10. **everyXHH:** την επιλογή για ωριαία εκτέλεση στα επιθυμητά λεπτά

```
84        axios
85          .post(
86            "http://sonoff.ddns.net:3001/schedule",
87            {
88              newSchedule: [
89                {
90                  devices_ids: selectedDeviceID,
91                  days: DaysArray,
92                  hours: hh,
93                  minutes: mm,
94                  task_name: task,
95                  task_id: task_id,
96                  username: username,
97                  repeat: repeatChecked,
98                  selectedState: stateSelected,
99                  everyX: everyXText,
100                 everyXHH: everyHHXText,
101               },
102             ],
103           },
104           {}
105         )
106         .then((response) => {})
107         .catch((err) => {
108           console.log(err);
109         });
110     } else {
111       setModalVisible(false);
112       setModal3Visible(true);
113     }
114   };
```

4.3.14. Add task - Backend

Ο server, λαμβάνει από το request τις τιμές που έχει επιλέξει ο χρήστης, πραγματοποιεί τους απαιτούμενους ελέγχους και καταχωρεί τις τιμές στη βάση δεδομένων, στον πίνακα schedule.

```
app.post("/schedule", (req, res) => {
  var jsonData = req.body;
  var jsonParsed = JSON.parse(JSON.stringify(jsonData));
  console.log(jsonParsed);
  var device_id = jsonParsed.newSchedule[0].devices_ids;
  var days = jsonParsed.newSchedule[0].days;
  var hours = jsonParsed.newSchedule[0].hours;
  hours = parseInt(hours);
  var minutes = jsonParsed.newSchedule[0].minutes;
  minutes = parseInt(minutes);
  var task_name = jsonParsed.newSchedule[0].task_name;
  var task_id = jsonParsed.newSchedule[0].task_id;
  var username = jsonParsed.newSchedule[0].username;
  var repeater = jsonParsed.newSchedule[0].repeat;
  var selected_state = jsonParsed.newSchedule[0].selectedState;
  var completed = "false";
  var everyX = jsonParsed.newSchedule[0].everyX;
  var everyXHH = jsonParsed.newSchedule[0].everyXHH;
  var now = new Date();

  if (isNaN(hours)) {
    hours = now.getHours();
  }
  if (isNaN(minutes)) {
    minutes = now.getMinutes();
  }
  if (everyX != "") {
    days = ["1", "2", "3", "4", "5", "6", "0"];
    everyX = everyX * 60000;
  }
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "[redacted]",
    database: "onoffdb",
  });

  var start = Date.now();
  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = `INSERT INTO schedule (task_id, task_name, device_id, days, hours,
    minutes, username, repeater, selected_state, completed, everyX, everyXHH, start_time)
    VALUES ('${task_id}', '${task_name}', '${device_id}', '${days}', '${hours}', '${minutes}',
    '${username}', '${repeater}', '${selected_state}', '${completed}', '${everyX}', '${everyXHH}', '${start}')`;
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("1 record inserted");
    });
  });
  res.json(
    device_id,
    days,
    hours,
    minutes,
    task_name,
    task_id,
    username,
    repeater,
    selected_state,
    completed
  );
});
```

4.3.15. Delete task - Front-end

Επιλέγοντας τη διαγραφή κάποιου task από το UI, καλείται η συνάρτηση cancelTask, η οποία στέλνει στο back-end ενα post request το οποίο περιέχει το όνομα του task και το username του χρήστη.

```
const cancelTask = async (item) => {
  const username = await AsyncStorage.getItem("token");
  axios
    .post("http://sonoff.ddns.net:3001/scheduleCancel", {
      cancel: [
        {
          task_name: item,
          username: username,
        },
      ],
    })
    .then((response) => {
      const data = response.data;
    })
    .catch((err) => {
      console.log(err);
    });
};
```

4.3.16. Delete task - Backend

Ο server λαμβάνει το request από το front-end σχετικά με τη διαγραφή του task. Στη συνέχεια, αναζητεί στο σχετικό πίνακα της βάσης δεδομένων schedule την αντίστοιχη εγγραφή με βάση το username του χρήστη και το task name, ώστε να εντοπίσει το μοναδικό task_id της επιθυμητής εγγραφής. Τέλος, αναζητεί στον ίδιο πίνακα, την εγγραφή με βάση το task_id, και τη διαγράφει.

```
app.post("/scheduleCancel", (req, res) => {
  var jsonData = req.body;
  var jsonParsed = JSON.parse(JSON.stringify(jsonData));
  var task_name = jsonParsed.cancel[0].task_name;
  var username = jsonParsed.cancel[0].username;
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "[redacted]",
    database: "onoffdb",
  });

  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = `SELECT task_id FROM schedule WHERE username LIKE '${username}' AND task_name LIKE '${task_name}' `;
    con.query(sql, function (err, results) {
      if (err) throw err;
      var tasksToCancel = results.map(function (result) {
        return result["task_id"];
      });
      console.log(tasksToCancel, "will be canceled");
    });
    var sql2 = `DELETE FROM schedule WHERE username LIKE '${username}' AND task_name LIKE '${task_name}' `;
    con.query(sql2, function (err, results) {
      if (err) throw err;
    });
    res.json("task deleted and canceled");
  });
});
```

4.3.17. Day graph - Front-end

Κατά την επιλογή συσκευής στο σελίδα Graphs, καλείται η συνάρτηση onConfirmDay. Μέσω των εντολών της, στέλνει ένα request στον server, περιμένοντας τα ημερήσια δεδομένα λειτουργίας της συγκεκριμένης συσκευής, ώστε να τα εμφανίσει στο σχετικό γράφημα.

```
const onConfirmDay = async (selectedDeviceID) => {
  setIsLoading(true);
  const username = await AsyncStorage.getItem("token");
  if (selectedDeviceID.selectedDeviceID.length == 0) return "";
  const devicePostID = selectedDeviceID.selectedDeviceID.toString();
  axios
    .post("http://sonoff.ddns.net:3001/graphDay", {
      daily: [
        {
          username: username,
          device_id: devicePostID,
        },
      ],
    })
    .then((response) => {
      console.log(response.data);
      setDayData(response.data);
    })
    .catch((err) => {
      console.log(err);
    });
  onConfirmWeek(devicePostID);
  console.log(devicePostID, "a");
};
```

4.3.18. Day graph - Backend

Ο server λαμβάνει το σχετικό request και αναζητά τις απαιτούμενες πληροφορίες στον πίνακα power της βάσης δεδομένων. Τα εκχωρεί σε έναν πίνακα και τα επιστρέφει στο front-end.

```
app.post("/graphDay", (req, res) => {
  var jsonData = req.body;
  var jsonParsed = JSON.parse(JSON.stringify(jsonData));

  var device_id = jsonParsed.daily[0].device_id;
  var username = jsonParsed.daily[0].username;
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "[redacted]",
    database: "onoffdb",
  });

  con.connect(function (err) {
    if (err) throw err;
    con.query(
      `SELECT device_state FROM power WHERE date(timestamp) = CURRENT_DATE AND device_id = '${device_id}'
      AND username = '${username}' AND HOUR(timestamp) BETWEEN 0 AND 23 AND MINUTE(timestamp) BETWEEN 0 AND 29`,
      function (err, result, fields) {
        if (err) throw err;
        var todayHourly = result.map(function (result) {
          return result["device_state"];
        });
        var len = todayHourly.length;
        var len2 = 24 - len;
        for (let i = 0; i < len2; i++) {
          todayHourly.push(0);
        }
        res.json(todayHourly);
      }
    );
    con.end();
  });
});
```

Στη ίδια λογική βασίζεται και ο κώδικας του server που αφορά τα weekly και monthly γραφήματα της σελίδας graphs.

4.3.19. Rename - Front-end

Κατά την καταχώρηση αιτήματος μετονομασίας κάποιας συσκευής, μέσω της σελίδας Settings, καλείται η ακόλουθη συνάρτηση η οποία αποθηκεύει τις εισαχθείσες τιμές σε έναν πίνακα, τον οποίο στέλνει στο back-end μέσω post request.

```
useEffect(() => {
    console.log("Changed devices: ", devices);
    setItems(
      devices.map((device) => {
        return { label: device_name, label0: device_name0, value: device_id };
      })
    );
  }, [devices]);

  const [devices, setDevices] = useState([]);
  const [items, setItems] = useState([]);

  useEffect(() => {
    for (let i = 0; i < devices.length; i++) {
      if (devices[i].value == idd) {
        devices[i].label = onoma.newOnoma;
      }
      console.log(devices);
    }
  }, [onoma]);

  const handleChange = (tags) => (event) => {
    console.log(onoma);
  };

  const onConfirm = () => {
    axios
      .post(
        "http://sonoff.ddns.net:3001/rename",
        {
          devices,
        },
        {}
      )
      .then((response) => {
        console.log(response.data);
      })
      .catch((err) => {
        console.log(err);
      });
    setShowDevices(false);
  };
```

Rename -Backend

Ο server, λαμβάνει το request και εκτελεί το σχετικό update στον πίνακα devices της βάσης δεδομένων με βάση τα δοθέντα device_id.

```
app.post("/rename", (req, res) => {
  var jsonData = req.body;
  var jsonParsed = JSON.parse(JSON.stringify(jsonData));
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "[redacted]",
    database: "onoffdb",
  });
  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var deviceId;
    var deviceName;
    for (i = 0; i < jsonParsed.devices.length; i++) {
      deviceId = jsonParsed.devices[i].value;
      deviceName = jsonParsed.devices[i].label;
      console.log(deviceId, deviceName);
      var sql = `UPDATE devices SET device_name = '${deviceName}' WHERE device_id = '${deviceId}';`;
      con.query(sql, function (err, result) {
        if (err) throw err;
        console.log("record/s updated");
      });
    }
  });
});
```

4.3.20. Logout

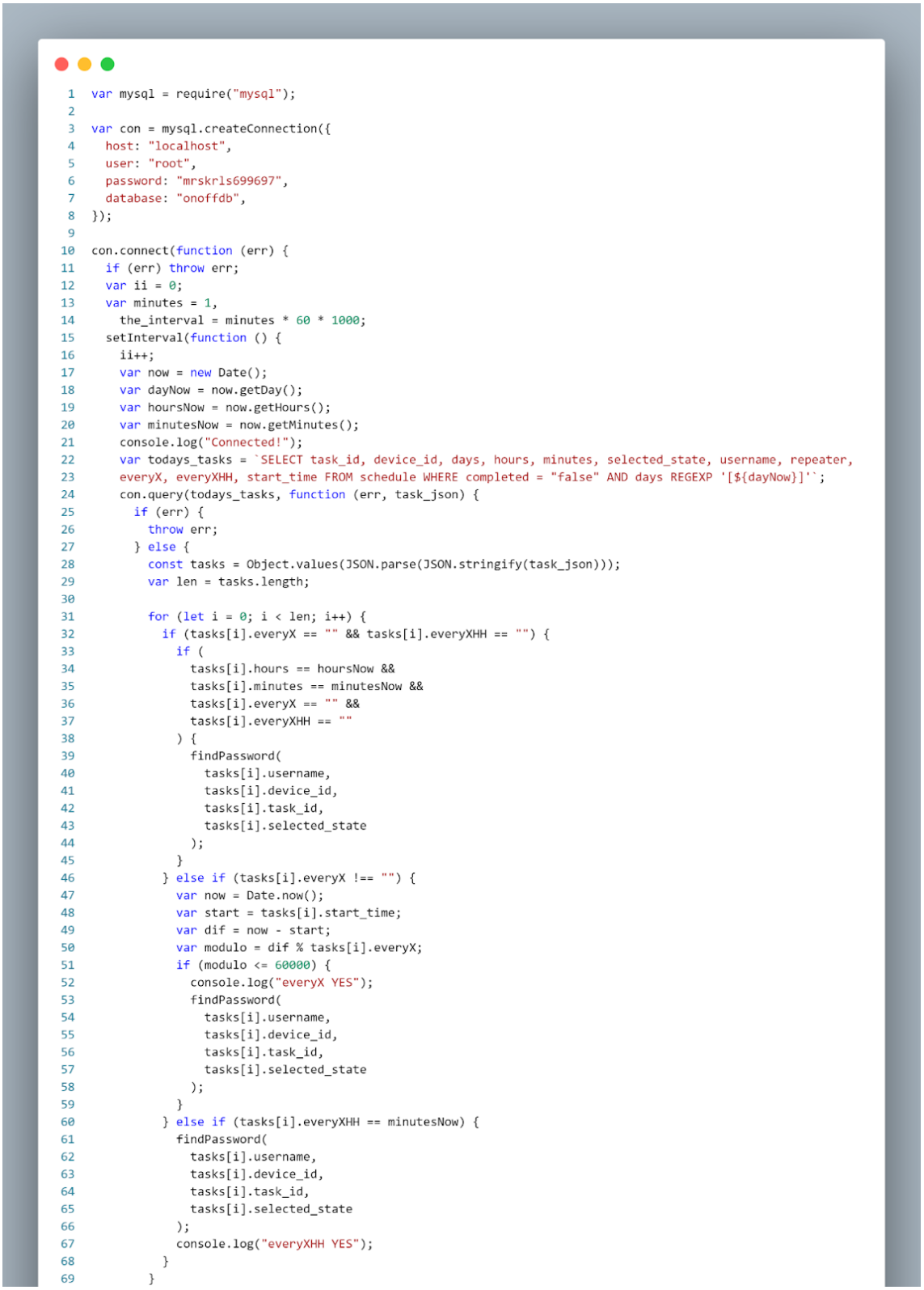
Κατά την επιλογή logout από το σχετικό button στη σελίδα Settings, καλείται η σχετική συνάρτηση, η οποία διαγράφει το token από το AsyncStorage για να μην πραγματοποιείται πλέον το auto login και οδηγεί το χρήστη στη Login σελίδα.

```
<Pressable style={styles.button} onPress={() => setShowDevices(true)}>
          <Text style={styles.text}>DEVICE RENAME</Text>
        </Pressable>
        <Pressable style={styles.button} onPress={logout}>
          <Text style={styles.text}>LOGOUT</Text>
        </Pressable>
```

```
const logout = async () => {
  await AsyncStorage.removeItem("token");
  navigation.navigate("Login");
};
```

4.3.21. Scheduler - Backend

Η συγκεκριμένη συνάρτηση τρέχει με interval ενός λεπτού και έχει σαν στόχο να εντοπίσει ποια από τα task που είναι καταχωρημένα στον πίνακα schedule πρέπει να εκτελεστούν, εκτελώντας τους απαραίτητους χρονικούς ελέγχους στα δεδομένα της βάσης δεδομένων.



```
var mysql = require("mysql");

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "mrskrls699697",
  database: "onoffdb",
});

con.connect(function (err) {
  if (err) throw err;
  var ii = 0;
  var minutes = 1,
    the_interval = minutes * 60 * 1000;
  setInterval(function () {
    ii++;
    var now = new Date();
    var dayNow = now.getDay();
    var hoursNow = now.getHours();
    var minutesNow = now.getMinutes();
    console.log("Connected!");
    var todays_tasks = `SELECT task_id, device_id, days, hours, minutes, selected_state, username, repeater,
    everyX, everyXHH, start_time FROM schedule WHERE completed = "false" AND days REGEXP '[${dayNow}]'`;
    con.query(todays_tasks, function (err, task_json) {
      if (err) {
        throw err;
      } else {
        const tasks = Object.values(JSON.parse(JSON.stringify(task_json)));
        var len = tasks.length;

        for (let i = 0; i < len; i++) {
          if (tasks[i].everyX == "" && tasks[i].everyXHH == "") {
            if (
              tasks[i].hours == hoursNow &&
              tasks[i].minutes == minutesNow &&
              tasks[i].everyX == "" &&
              tasks[i].everyXHH == ""
            ) {
              findPassword(
                tasks[i].username,
                tasks[i].device_id,
                tasks[i].task_id,
                tasks[i].selected_state
              );
            }
          } else if (tasks[i].everyX !== "") {
            var now = Date.now();
            var start = tasks[i].start_time;
            var dif = now - start;
            var modulo = dif % tasks[i].everyX;
            if (modulo <= 60000) {
              console.log("everyX YES");
              findPassword(
                tasks[i].username,
                tasks[i].device_id,
                tasks[i].task_id,
                tasks[i].selected_state
              );
            }
          } else if (tasks[i].everyXHH == minutesNow) {
            findPassword(
              tasks[i].username,
              tasks[i].device_id,
              tasks[i].task_id,
              tasks[i].selected_state
            );
            console.log("everyXHH YES");
          }
        }
```

Στη συνέχεια, και αφού εντοπιστούν τα task προς εκτέλεση, εκτελείται μία επανάληψη για κάθε task η οποία πραγματοποιεί:

1. Εύρεση των διαπιστευτηρίων εισόδου χρησιμοποιώντας το username και τον πίνακα users.
2. Εκτέλεση της ενέργειας που έχει καθοριστεί (on-off-toggle) στη συγκεκριμένη συσκευή.

Τέλος, ελέγχει αν το εκάστοτε task πρόκειται να εκτελεστεί ξανά, και ενημερώνει αντίστοιχα το πεδίο completed.

```
70
71          function findPassword(username, deviceID, task_id, selected_state) {
72            var password_finder = `SELECT password FROM users WHERE username = '${username}' `;
73            con.query(password_finder, function (err, password_json) {
74              if (err) {
75                throw err;
76              } else {
77                const passwords = Object.values(
78                  JSON.parse(JSON.stringify(password_json))
79                );
80                var pass = passwords[0].password;
81                var devices = [];
82                if (deviceID.indexOf(",") > -1) {
83                  devices = deviceID.split(",");
84                } else {
85                  devices = [deviceID];
86                }
87
88                const ewelink = require("ewelink-api");
89                (async () => {
90                  const connection = new ewelink({
91                    email: username,
92                    password: pass,
93                    region: "eu",
94                  });
95                  for (let i = 0; i < devices.length; i++) {
96                    const status = await connection.setDevicePowerState(
97                      devices[i],
98                      selected_state
99                    );
100                 }
101               })();
102
103               var set_complete = `UPDATE schedule SET completed = "true" WHERE task_id = '${task_id}' AND
104               repeater = "no repeat" AND everyX = "" AND everyXHH = ""`;
105               con.query(set_complete, function (err, result) {
106                 if (err) {
107                   throw err;
108                 }
109               });
110             }
111           });
112         }
113       }
114     });
115   }, the_interval);
116 });
117
```

4.3.22. Usage Query - Backend

Η συγκεκριμένη συνάρτηση εκτελείται σε επανάληψη με interval 30 λεπτών και εκτελεί την καταγραφή της λειτουργίας των συσκευών (on/off). Τα δεδομένα που καταγράφει, χρησιμοποιούνται από το front-end για την γραφική απεικόνιση της λειτουργίας των συσκευών στην σελίδα Graphs. Όπως φαίνεται, πραγματοποιείται σειριακή επανάληψη στον πίνακα users, ώστε να πραγματοποιηθεί καταγραφή της κατάστασης των συσκευών όλων των χρηστών μέσω εμφωλευμένης επανάληψης, η οποία πραγματοποιεί σειριακή κλίση του API της eWeLink για κάθε συσκευή.

```
var minutes = 30,
  the_interval = minutes * 60 * 1000;
setInterval(function () {
  console.log("starting...");
  var username = "";
  var password = "";
  var mysql = require("mysql");
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "mrskrls699697",
    database: "onoffdb",
  });

  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = `SELECT * FROM users`;
    con.query(sql, function (err, result) {
      if (err) {
        throw err;
      } else {
        const usrs = Object.values(JSON.parse(JSON.stringify(result)));
        var len = usrs.length;
        console.log(len);

        for (let i = 0; i < len; i++) {
          username = result[i].username;
          password = result[i].password;
          const ewelink = require("ewelink-api");
          var stat = [];

          (async () => {
            const connection = new ewelink({
              email: username,
              password: password,
              region: "eu",
            });
```

Με την ολοκλήρωση κάθε επανάληψης, συμπληρώνεται η σχετική εγγραφή στον πίνακα power και περιέχει τις εξής πληροφορίες:

1. **timestamp:** Η ακριβής τιμή ημερομηνίας και ώρας, χρησιμοποιείται για την χρονική ταξινόμηση των εγγραφών ώστε να χρησιμοποιηθούν σωστά στη δημιουργία των γραφημάτων.
2. **device id:** Χρησιμοποιείται κατά την επιλογή συσκευής στη σελίδα Graphs, ώστε να δημιουργηθεί το σωστό γράφημα της συγκεκριμένης συσκευής.
3. **device state:** Αφορά τις τιμές on-off που χρησιμοποιούνται στην γραφική απεικόνιση.
4. **username:** Το username του χρήστη στον οποίο ανήκει η συγκεκριμένη συσκευή.

```
39
40              /* get all devices */
41              const devices = await connection.getDevices();
42              for (device in devices) {
43                const status = await connection.getDevicePowerState(
44                  devices[device]["deviceid"]
45                );
46                var devstat = status.state;
47                if (devstat == null || devstat == "off") {
48                  devstat = 0;
49                } else if (devstat == "on") {
50                  devstat = 1;
51                }
52
53                var id = devices[device]["deviceid"];
54                var timestamp = new Date();
55                var mysql = require("mysql");
56                var con = mysql.createConnection({
57                  host: "localhost",
58                  user: "root",
59                  password: "mrskrls699697",
60                  database: "onoffdb",
61                });
62
63                con.connect(function (err) {
64                  if (err) throw err;
65                  console.log("Connected!");
66                  var sql = `INSERT INTO power (timestamp, device_id, device_state, username) VALUES (?, ?, ?, ?)`;
67                  con.query(
68                    sql,
69                    [timestamp, id, devstat, username],
70                    function (err, result) {
71                      if (err) throw err;
72                      console.log("...added", timestamp);
73                    }
74                  );
75                  con.end();
76                });
77              }
78            })();
79          }
80        }
81      });
82      con.end();
83    });
84  }, the_interval);
85
```

4.4. Κώδικας λειτουργιών της εφαρμογής smartwatch

Ακολουθούν στιγμιότυπα από την εφαρμογή Android Studio, όπου αναπτύχθηκε ο κώδικας της εφαρμογής που αφορά τα android smartwatch.

4.4.1. Login class

```
package com.example.postwatch;
import ...
public class Login extends AppCompatActivity {
    String username;
    String password;
    SharedPreferences sp;
    String namey, sid, status, connection ,ssid;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        ProgressBar pb = (ProgressBar) findViewById(R.id.progressBar);
        pb.setVisibility(View.GONE);
        Button btn = (Button)findViewById(R.id.button);
        sp = getSharedPreferences( name: "MyUserPrefs", MODE_PRIVATE);
        String usernameSP = sp.getString( s: "username",  s1: "");
        String passwordSP = sp.getString( s: "password",  s1: "");
        if (usernameSP != null && !usernameSP.equals("")) {
            Intent i = new Intent( packageContext: Login.this, MainActivity.class);
            i.putExtra( name: "username", usernameSP);
            startActivity(i);
        }
        btn.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View view) {
                pb.setVisibility(View.VISIBLE);
                EditText user = (EditText)findViewById(R.id.usernameText);
                EditText pass = (EditText)findViewById(R.id.passwordText);
                String username = user.getText().toString();
                String password = pass.getText().toString();
                SharedPreferences.Editor editor = sp.edit();
                editor.putString( s: "username", username);
                editor.putString( s: "password", password);
                editor.commit();
                RequestQueue queue = Volley.newRequestQueue( context: Login.this);
                    String url = "http://sonoff.ddns.net:3001/login/"+username+"/"+password;
                StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
                        new Response.Listener<String>() {
                            @Override
                            public void onResponse(String response) {
```

```
                    try {
                        JSONObject resJson = new JSONObject(response);
                        String res = resJson.getString( name: "message");
                        boolean b = res.endsWith("successful");
                        if (b) {
                            pb.setVisibility(View.GONE);
                            Toast.makeText( context: Login.this,  text: "Login successful", Toast.LENGTH_LONG).show();

                            Intent i = new Intent( packageContext: Login.this, MainActivity.class);
                            i.putExtra( name: "username", username);
                            startActivity(i);
                        } else {
                            pb.setVisibility(View.GONE);
                            Toast.makeText( context: Login.this,  text: "Wrong username/password. Please try again", Toast.LENGTH_LONG).show();
                            editor.clear();
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            pb.setVisibility(View.GONE);
            Toast.makeText( context: Login.this,  text: "Something went wrong, please try again.", Toast.LENGTH_LONG).show();
            editor.clear();
        }
    });
    queue.add(stringRequest);
}
});
}
}
```

Στην login class, πραγματοποιείται έλεγχος των στοιχείων που ο χρήστης εισάγει στα πεδία του login activity.

Στη συνέχεια μέσω ενός post request πραγματοποιείται επικοινωνία με το server για την επαλήθευση των διαπιστευτηρίων εισόδου που εισήγαγε ο χρήστης.

Αν τα στοιχεία επαληθευτούν η εφαρμογή θα φορτώσει το main activity και αφού πραγματοποιήθηκε επιτυχημένο login, θα αποθηκεύσει το username στα shared preferences της εφαρμογής ώστε να επιτυγχάνεται αυτόματη είσοδος στο μέλλον και να μην χρειαστεί νέα εκχώρηση των διαπιστευτηρίων.

Ο κώδικας xml για το login activity. Αφορά το εικαστικό κομμάτι της login σελίδας.

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000"
    tools:context=".Login">

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:orientation="horizontal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="@+id/linearLayout" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">


        <EditText
            android:id="@+id/usernameText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:gravity="center"
            android:hint="Username"
            android:inputType="textEmailAddress" />

        <EditText
            android:id="@+id/passwordText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:gravity="center"
            android:hint="Password"
            android:inputType="textPassword" />

        <Button
            android:id="@+id/button"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:gravity="center_horizontal"
            android:text="Login" />

    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ακολουθεί ο κώδικας του main activity class της εφαρμογής. Κατά την εκκίνηση του main activity, αντλείται από τα shared preferences η πληροφορία του username του χρήστη. Η συγκεκριμένη πληροφορία αποστέλλεται στον server μέσω post request και η απάντηση που λαμβάνει η εφαρμογή αποθηκεύεται σε ένα listview.

```
package com.example.postwatch;

import ...

public class MainActivity extends Activity {

    RequestQueue requestQueue;
    private ListView lv;
    String namey, sid, status, connection ,ssid;
    String username;
    ArrayList<Light> deviceList;
    ListAdapter adapter;
    ArrayList <String> newList;
    ArrayAdapter<Light> arrayAdapter;
    private String JSON_URL;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ProgressBar pb2 = (ProgressBar) findViewById(R.id.progressBar1);
        pb2.setVisibility(View.VISIBLE);
        SharedPreferences sp = getApplicationContext().getSharedPreferences( s: "MyUserPrefs", Context.MODE_PRIVATE);
        String username = sp.getString( s: "username",  s1: "");
        String postURL = "http://sonoff.ddns.net:3001/status?username=" + username  ;
        JSON_URL = postURL;
        deviceList = new ArrayList<>();
        lv = (ListView) findViewById(R.id.listview);
        GetData getData = new GetData();
        getData.execute();
        final ImageButton button = findViewById(R.id.imageButton);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) { recreate(); }
        });
        Button btn = (Button)findViewById(R.id.button0);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences sp = getSharedPreferences( name: "MyUserPrefs", Context.MODE_PRIVATE);
                SharedPreferences.Editor editor = sp.edit();
                editor.clear();
                editor.commit();
                startActivity(new Intent( packageContext: MainActivity.this, Login.class));
                finish();
            }
        });
    }
```

Κάθε γραμμή του listview αντιστοιχεί σε ένα button το οποίο μπορεί να ελέγξει τη λειτουργία της κάθε συσκευής, εμφανίζοντας επιβεβαιωτικό μήνυμα και αλλάζοντας την ένδειξη της λειτουργίας καταστάσεις στο εκάστοτε button.

```
@Override
protected void onPostExecute(String s) {
    try {
        JSONObject jsonObject = new JSONObject(s);
        JSONArray jsonArray = jsonObject.getJSONArray( name: "stat");
        for (int i = 0; i< jsonArray.length() ; i++) {
            JSONObject jsonObject1 = jsonArray.getJSONObject(i);
            namey = jsonObject1.getString( name: "name");
            sid = jsonObject1.getString( name: "id");
            connection = jsonObject1.getString( name: "connection");
            if(jsonObject1.length()>4) {
                status = jsonObject1.getString( name: "status");
            } else {
                status = "off";
            }
            ssid = jsonObject1.getString( name: "ssid");
            Light light = new Light(namey, sid, connection, status, ssid);
            deviceList.add(light);
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }

    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            arrayAdapter = new ArrayAdapter<Light>( context: MainActivity.this, android.R.layout.simple_list_item_1,deviceList);
            lv.setAdapter(arrayAdapter);
            ProgressBar pb2 = (ProgressBar) findViewById(R.id.progressBar1);
            pb2.setVisibility(View.GONE);
        }
    });
```

```
lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar5);
        final int oneMin = 1 * 3 * 1000; // 1 minute in milli seconds
        new CountDownTimer(oneMin,  countDownInterval: 1000) {
            public void onTick(long millisUntilFinished) {
                long finishedSeconds = oneMin - millisUntilFinished;
                int total = (int) (((float)finishedSeconds / (float)oneMin) * 100.0);
                progressBar.setProgress(total);
            }
            public void onFinish() {
                // DO something when 1 minute is up
            }
        }.start();

        Light device = deviceList.get(position);
        String url = "http://sonoff.ddns.net:3001/toggle/"+device.getId();
        StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
                response ->
                {
                    String st = device.getStatus();
                    if (st.equals("off")) {
                        device.SetStatus("off");
                        device.SetStatus("on");
                        st = "on";
                    } else if (st.equals("on")) {
                        device.SetStatus("on");
                        device.SetStatus("off");
                        st = "off";
                    }
                    lv.setAdapter(arrayAdapter);
                    Toast.makeText( context: MainActivity.this,  text: "Success, the device is turning " + st + ".", Toast.LENGTH_LONG).show();
                },
                error ->
                        Toast.makeText( context: MainActivity.this,  text: "Error", Toast.LENGTH_LONG).show());
        requestQueue = Volley.newRequestQueue( context: MainActivity.this);
        requestQueue.add(stringRequest);

    }
});
```

Ο κώδικας xml για το main activity. Αφορά το εικαστικό κομμάτι της main σελίδας.

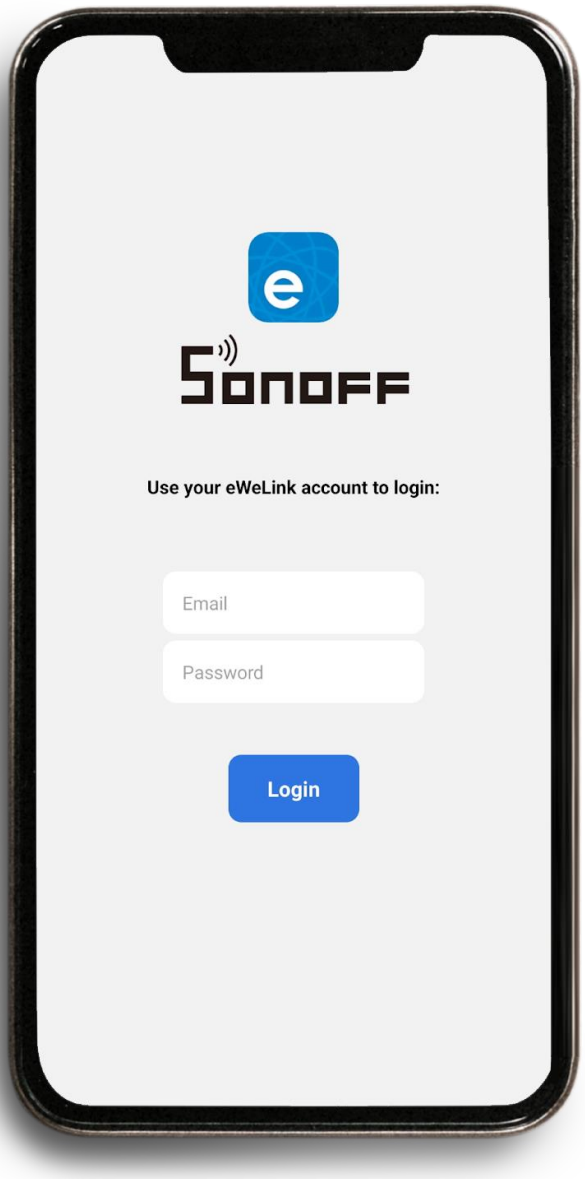
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.wear.widget.SwipeDismissFrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="0dp"
    tools:context=".MainActivity"
    tools:deviceIds="wear">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ProgressBar
            android:id="@+id/progressBar1"
            style="?android:attr/progressBarStyle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@+id/imageButton" />

        <ProgressBar
            android:id="@+id/progressBar5"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            tools:layout_editor_absoluteX="89dp"
            tools:layout_editor_absoluteY="104dp" />

        <ImageButton
            android:id="@+id/imageButton"
            android:layout_width="match_parent"
            android:layout_height="35dp"
            android:backgroundTint="#00000000"
            android:gravity="center_vertical"
            android:src="@android:drawable/ic_popup_sync" />

        <ScrollView
            android:layout_width="wrap_content"
            android:layout_height="177dp"
            android:layout_marginTop="43dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:orientation="vertical">

                <ListView
                    android:id="@+id/listview"
                    android:layout_width="172dp"
                    android:layout_height="177dp">

                </ListView>

                <Button
                    android:id="@+id/button0"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Logout" />

            </LinearLayout>
        </ScrollView>

    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.wear.widget.SwipeDismissFrameLayout>
```

5. Παρουσίαση - χρήση της mobile εφαρμογής.

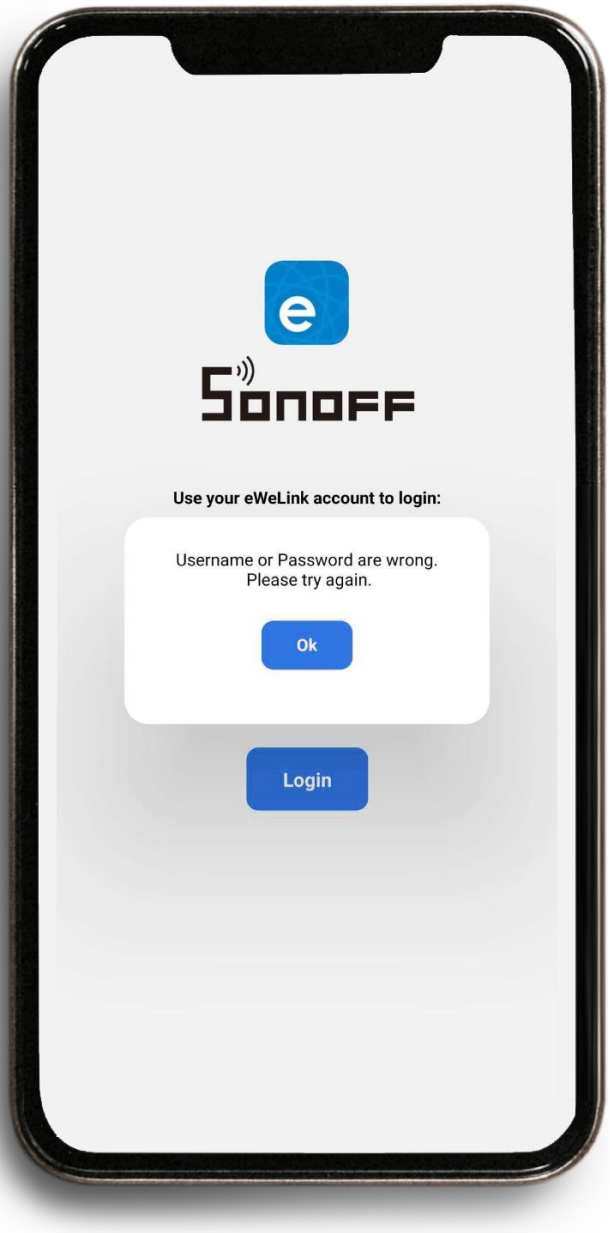
Εδώ θα παρουσιαστούν στιγμιότυπα από τη χρήση της εφαρμογής τόσο σε κινητό τηλέφωνο όσο και σε Smartwatch, τα οποία έχουν δημιουργηθεί μέσω emulator.

5.1. Login

Η σελίδα **Login** όπου ο χρήστης πρέπει να εισάγει τα διαπιστευτήρια εισόδου του eWeLink account του.



Στην περίπτωση εισαγωγής λανθασμένων δεδομένων, εμφανίζεται το ανάλογο μήνυμα:

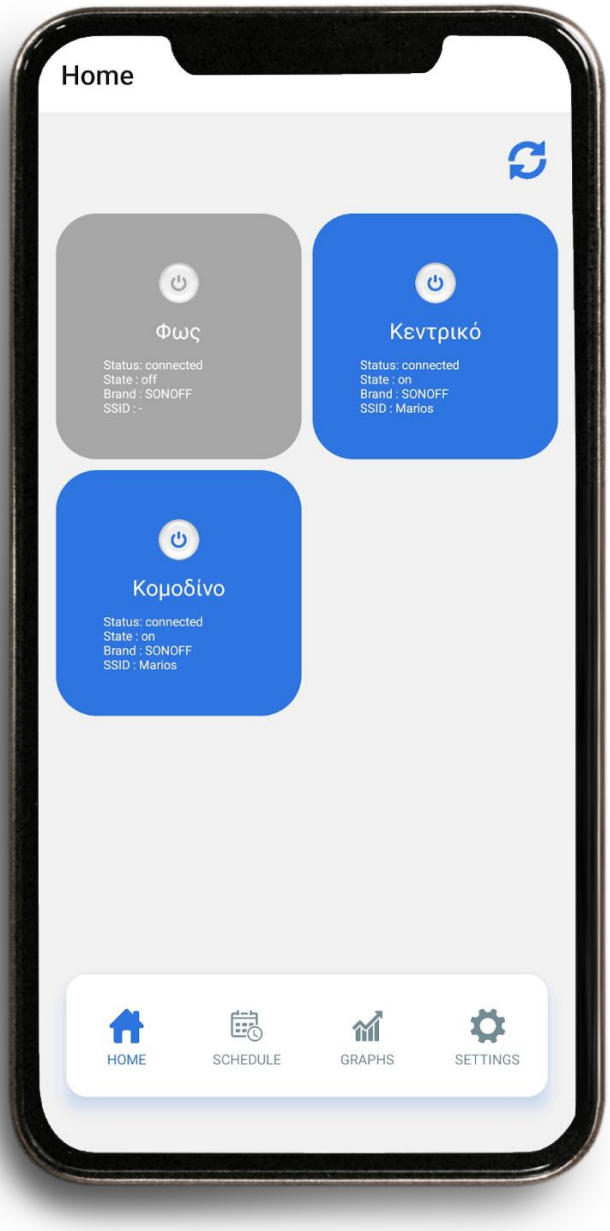


5.2. Home

Ύστερα από την επιτυχή εισαγωγή των διαπιστευτηρίων εισόδου (ή αυτόματα σε περίπτωση που έχει γίνει ήδη login στο παρελθόν), εμφανίζεται η **Home** σελίδα της εφαρμογής.

Εδώ, ο χρήστης βλέπει ένα button για κάθε συσκευή που έχει στο λογαριασμό του και τη σχετική πληροφορία για κάθε ένα (status, state, Brand, Network SSID). Αγγίζοντας ένα button, θα αλλάξει το state της συσκευής αντίστοιχα (κοκκινο: **on**, γκρι: **off**).

Επίσης, μπορεί να πραγματοποιήσει την ενημέρωση των στοιχείων της σελίδας χρησιμοποιώντας το **refresh** button στα δεξιά.

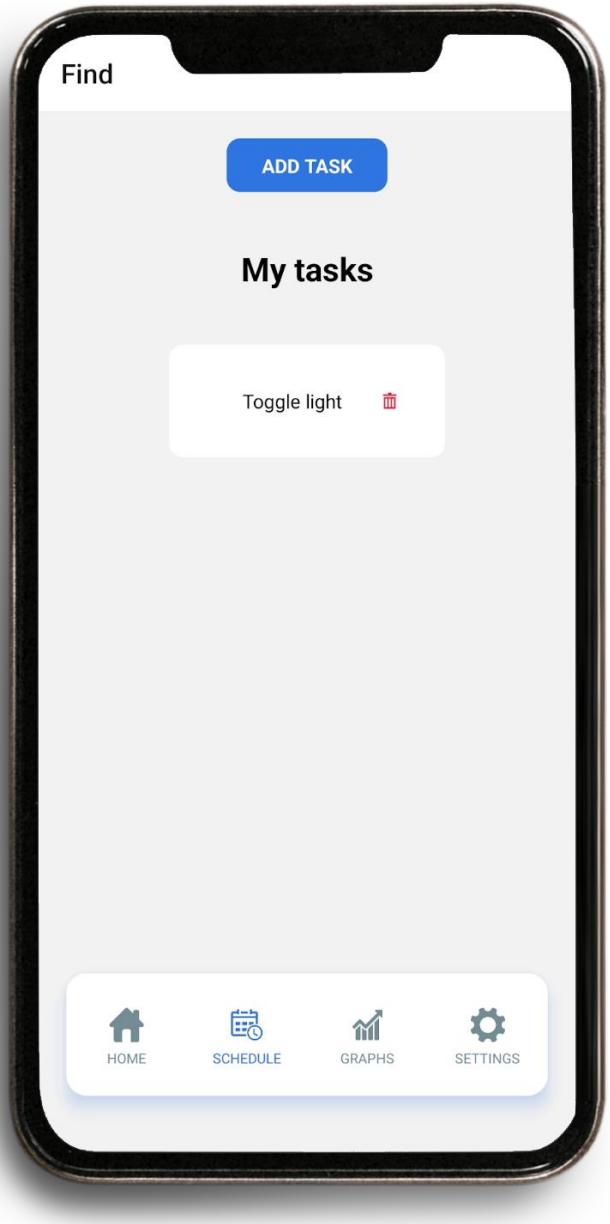


5.3. Schedule

Χρησιμοποιώντας την μπάρα πλοήγησης στο κάτω μέρος της εφαρμογής, ο χρήστης μπορεί να επισκεφθεί και τις υπόλοιπες σελίδες της εφαρμογής.

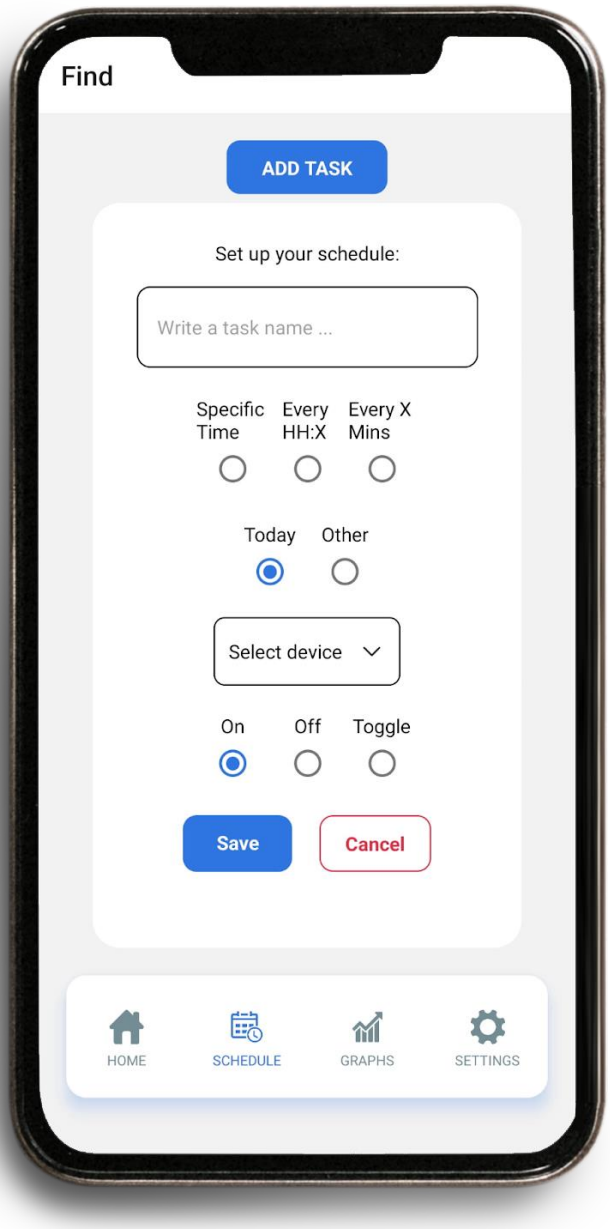
Η σελίδα **Schedule**, περιέχει τα task που μπορεί να δημιουργήσει ο χρήστης ώστε να προγραμματίσει στην επιθυμητή χρονική βάση τη λειτουργία των συσκευών του.

Χρησιμοποιώντας το **add task** button μπορεί να προσθέσει νέο task, ενώ το **εικονίδιο διαγραφής** θα αφαιρέσει το συγκεκριμένο task αφού πρώτα εμφανίσει επιβεβαιωτικό μήνυμα.

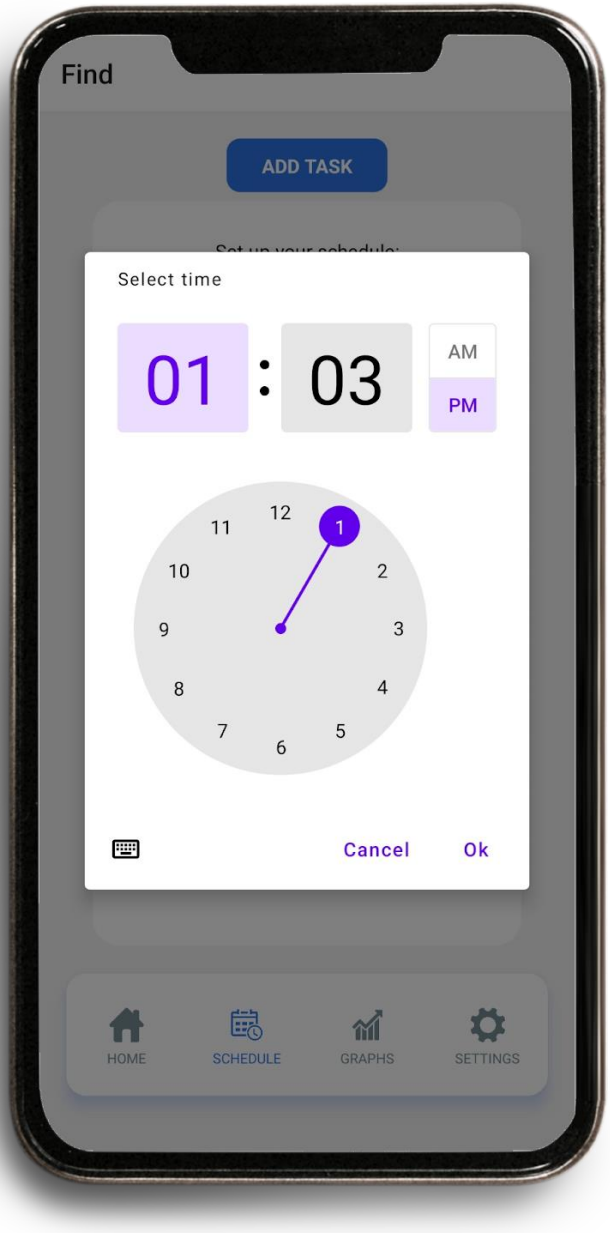


Στη σελίδα δημιουργίας νέου task ο χρήστης καλείται να εκτελέσει τις ακόλουθες ενέργειες:

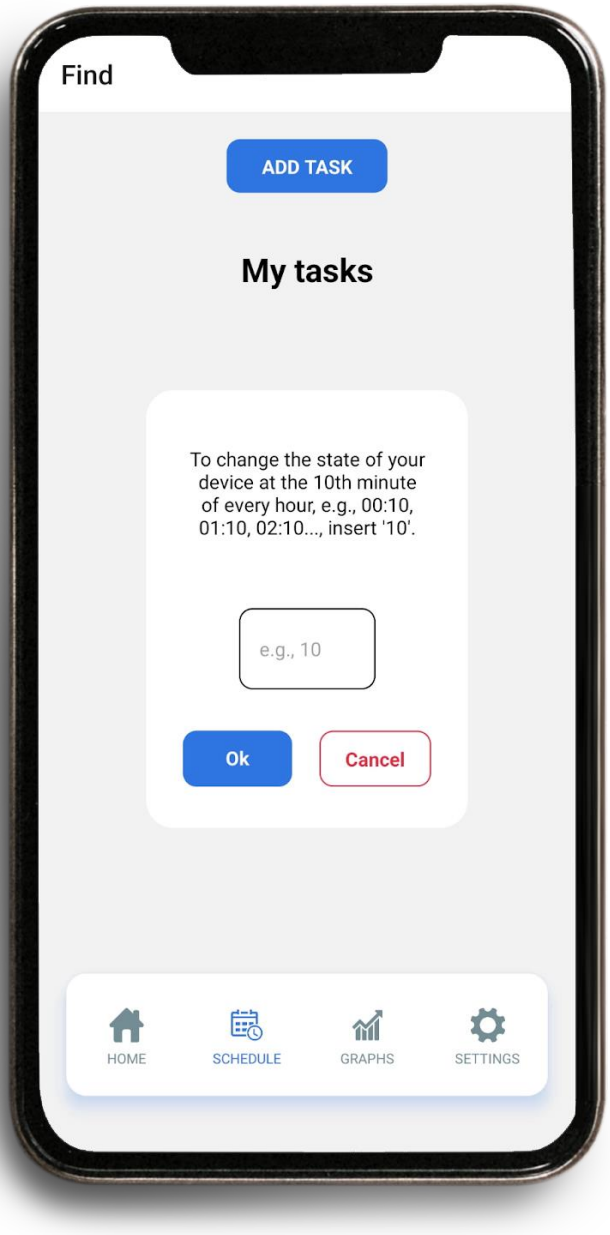
1. Εισαγωγή ονόματος
2. Επιλογή χρονικής περιόδου εκτέλεσης



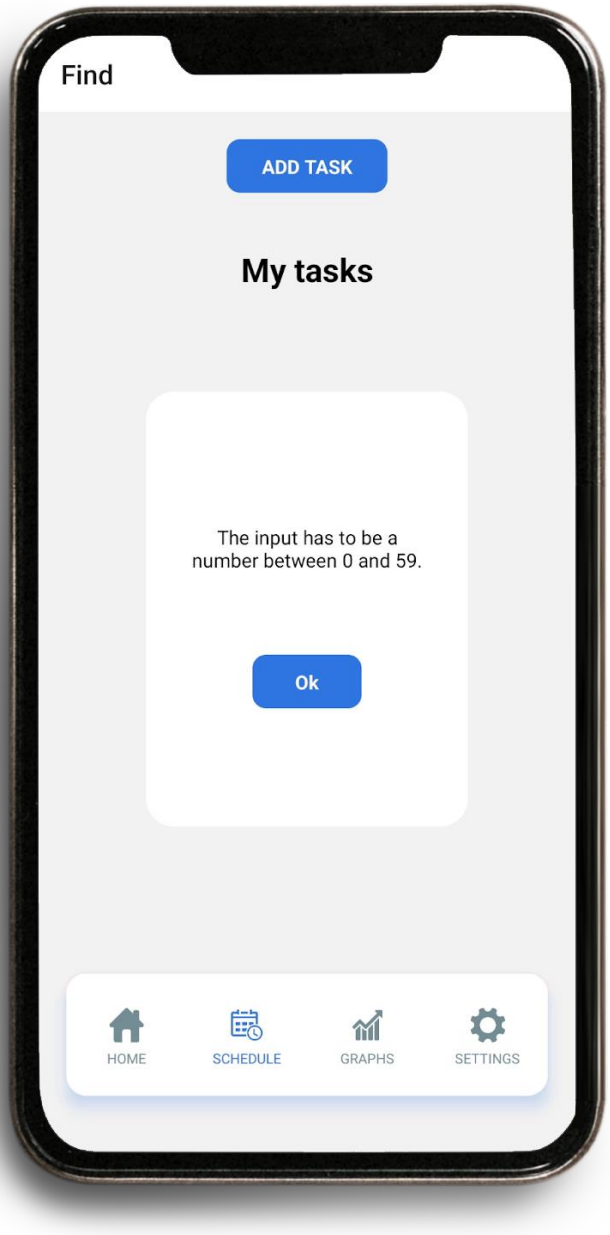
Η επιλογή specific time θα ανοίξει το σχετικό time picker όπου ο χρήστης μπορεί να επιλέξει την επιθυμητή ώρα εκτέλεσης.



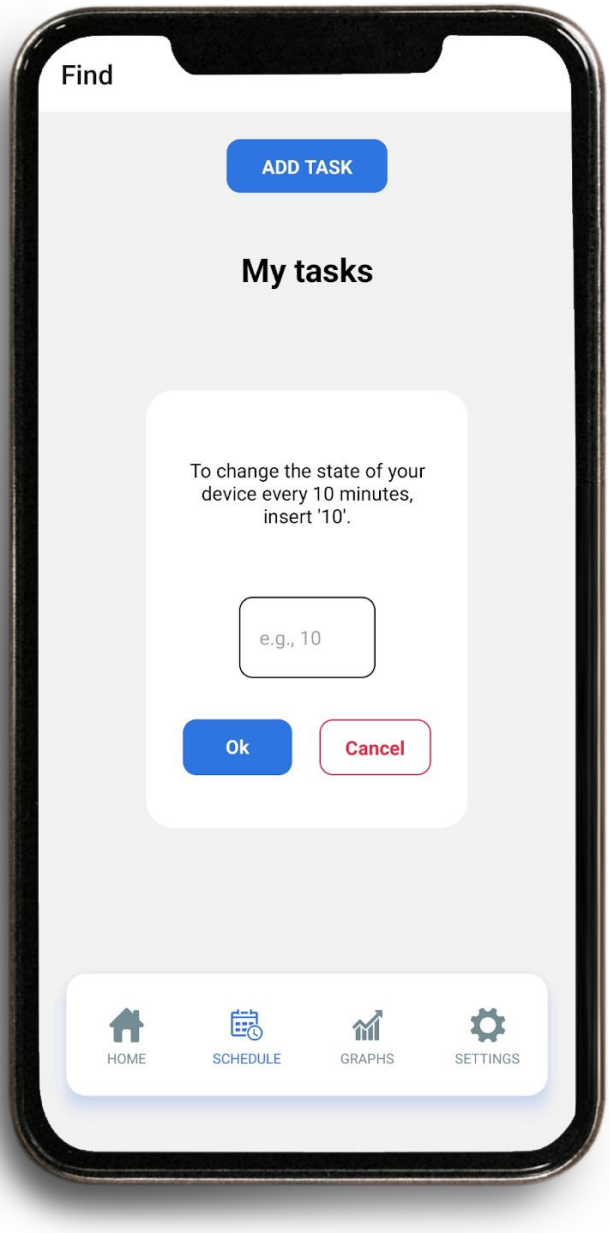
Η επιλογή **Every HH:X**, σημαίνει ότι το συγκεκριμένο task θα εκτελείται τα X πρώτα λεπτά κάθε ώρας, οπότε πρέπει η τιμή που θα εισάγει ο χρήστης να είναι ένας ακέραιος αριθμός μεταξύ του 0 και του 59.



Στην περίπτωση που η τιμή την οποία θα καταχωρήσει ο χρήστης δεν ανταποκρίνεται στον παραπάνω περιορισμό εμφανίζεται το αντίστοιχο ενημερωτικό μήνυμα σφάλματος, που θα οδηγήσει το χρήστη πίσω στη σελίδα εισαγωγής τιμής.

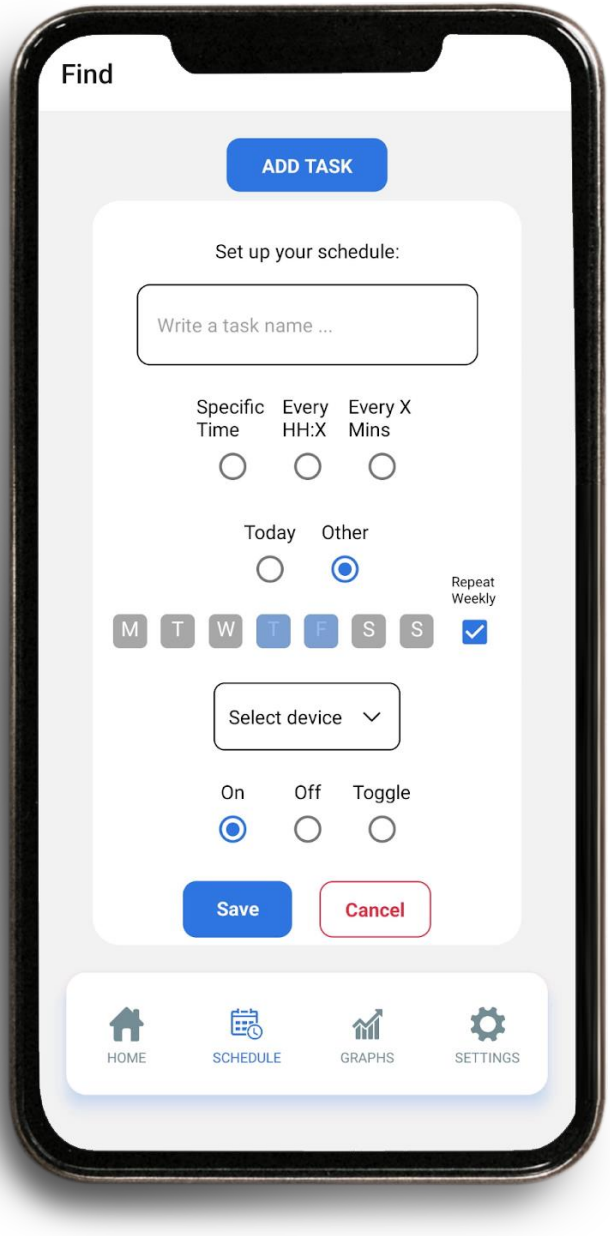


Η επιλογή **Every X**, αφορά την εκτέλεση του συγκεκριμένου task κάθε Χ λεπτά.
Παρομοίως ισχύει περιορισμός για την εισαγωγή ακέραιας τιμής με εμφάνιση μηνύματος σφάλματος στην περίπτωση εσφαλμένης εκχώρησης τιμής, οδηγώντας το χρήστη πίσω στη σελίδα εισαγωγής τιμής.

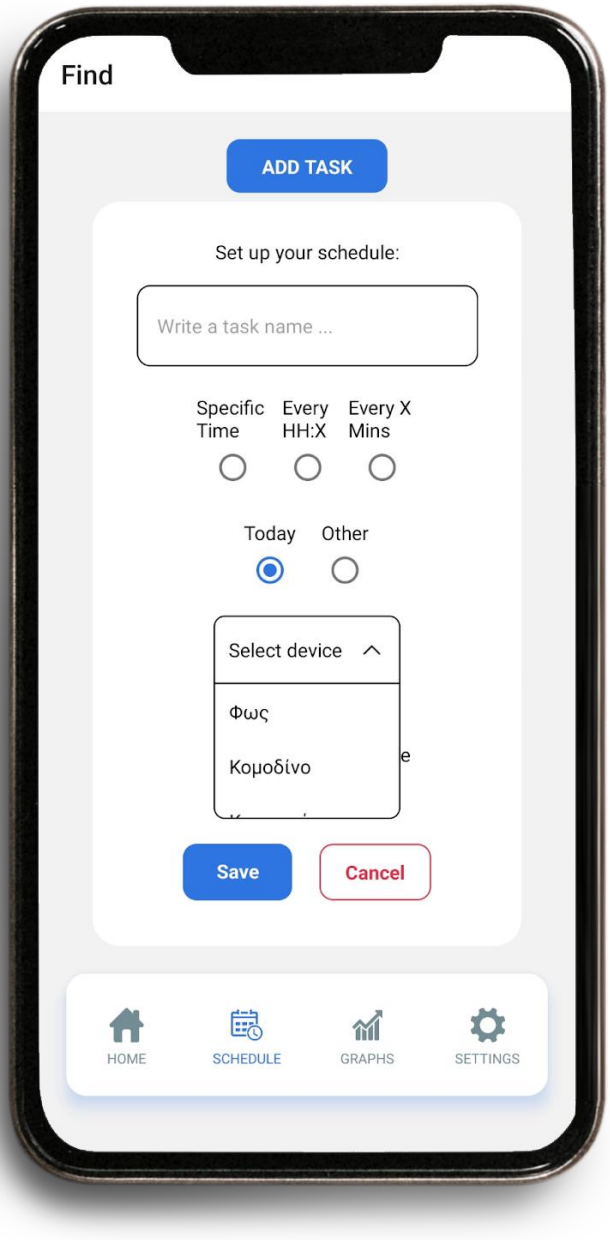


Κατά το βήμα επιλογής ημέρας εκτέλεσης ο χρήστης έχει τη δυνατότητα να επιλέξει μεταξύ των επιλογών του **Today** και **Other**. Η επιλογή **Today** θα καταχωρήσει ως ημερομηνία εκτέλεσης την σημερινή ημέρα. Η επιλογή **Other** δίνει στο χρήστη τη δυνατότητα να προγραμματίσει το συγκεκριμένο task για τις ημέρες της εβδομάδας που επιθυμεί.

Επίσης του δίνει τη δυνατότητα να προγραμματίσει την εβδομαδιαία επανάληψη του συγκεκριμένου task, μέσω του αντίστοιχου **Repeat Weekly** checkbox.

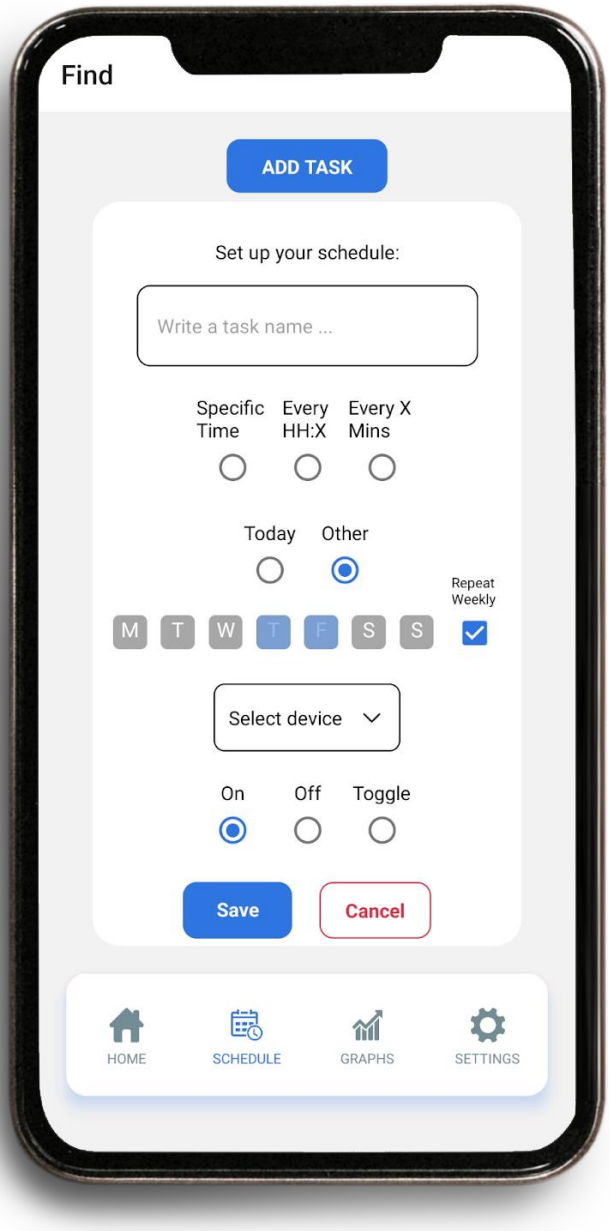


Στο επόμενο βήμα ο χρήστης καλείται να επιλέξει τη συσκευή την οποία αφορά το task που δημιουργεί. Στη **Select device** drop-down λίστα εμφανίζονται όλες οι συσκευές που είναι καταχωρημένες στο λογαριασμό του συγκεκριμένου χρήστη.

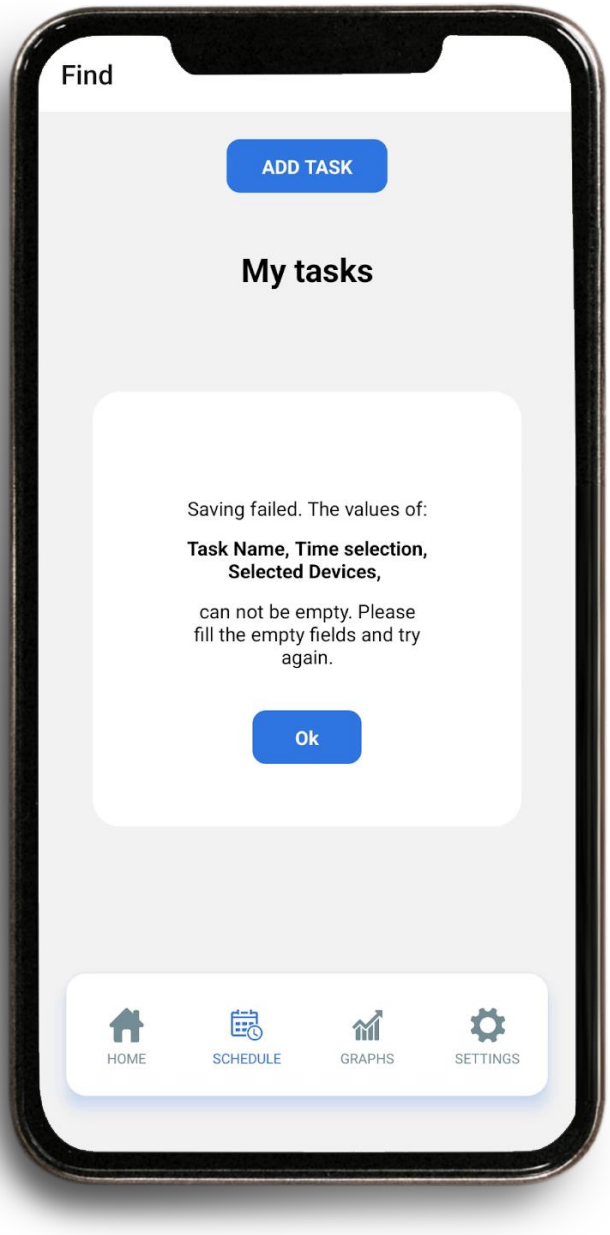


Τέλος ο χρήστης καλείται να επιλέξει τη λειτουργία στην οποία επιθυμεί να θέσει τη συσκευή του. Οι επιλογές που παρέχονται είναι οι εξής:

- **On:**
  Θέτει τη λειτουργία της συσκευής σε ενεργή κατάσταση
- **Off:**
  Θέτει τη λειτουργία της συσκευής σε ανενεργή κατάσταση
- **Toggle:**
  Θέτει τη λειτουργία της συσκευής στην αντίθετη από την τρέχουσα κατάσταση



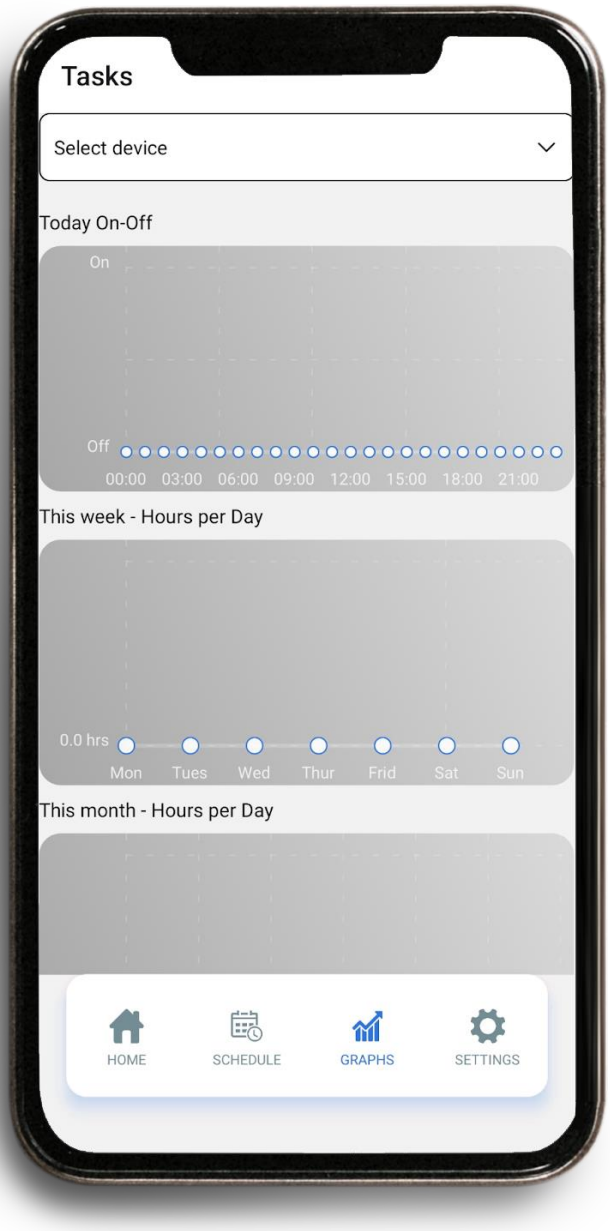
Στην περίπτωση που οποιαδήποτε από τις καταχωρημένες επιλογές είναι εσφαλμένη ή ελλιπής θα εμφανιστεί το αντίστοιχο ενημερωτικό μήνυμα σφάλματος. Εκεί, περιέχονται οι απαραίτητες πληροφορίες που θα οδηγήσουν το χρήστη στον εντοπισμό και τη διόρθωση του σφάλματος.



5.4. Graphs

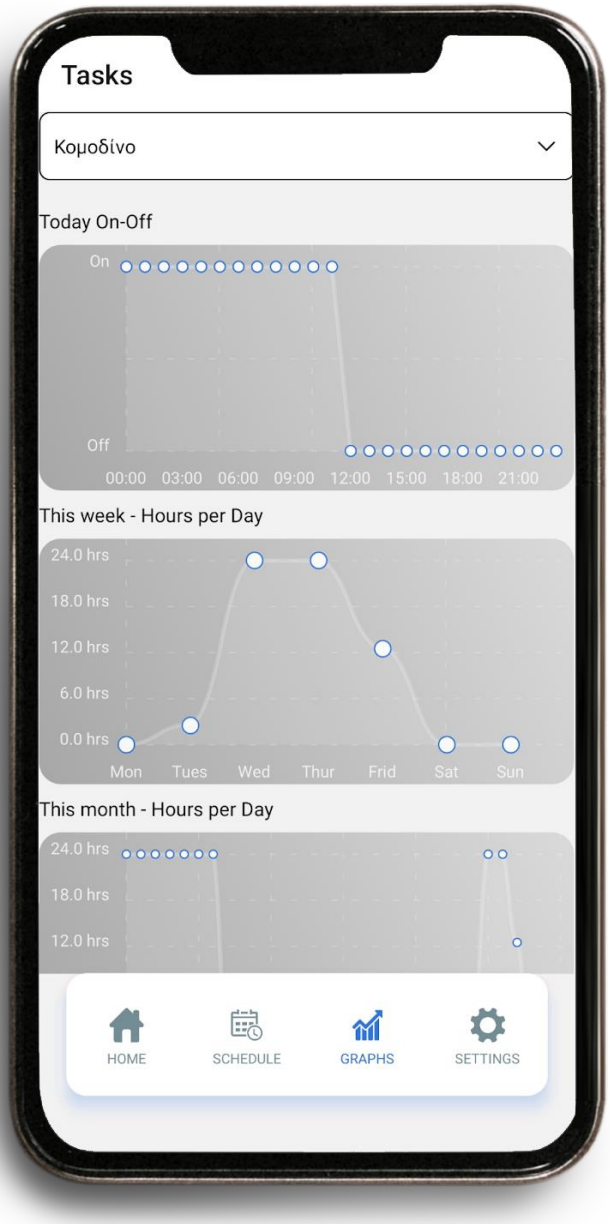
Στη συγκεκριμένη σελίδα, μέσω γραφημάτων αναπαρίσταται η λειτουργία των συσκευών του χρήστη σε ημερήσια, εβδομαδιαία και μηνιαία βάση.

Μέσω της **Select device** drop down λίστας στο πάνω μέρος της εφαρμογής ο χρήστης μπορεί να επιλέξει μεταξύ των συσκευών του λογαριασμού του. Αυτό θα οδηγήσει στην αυτόματη ενημέρωση των δεδομένων των γραφημάτων



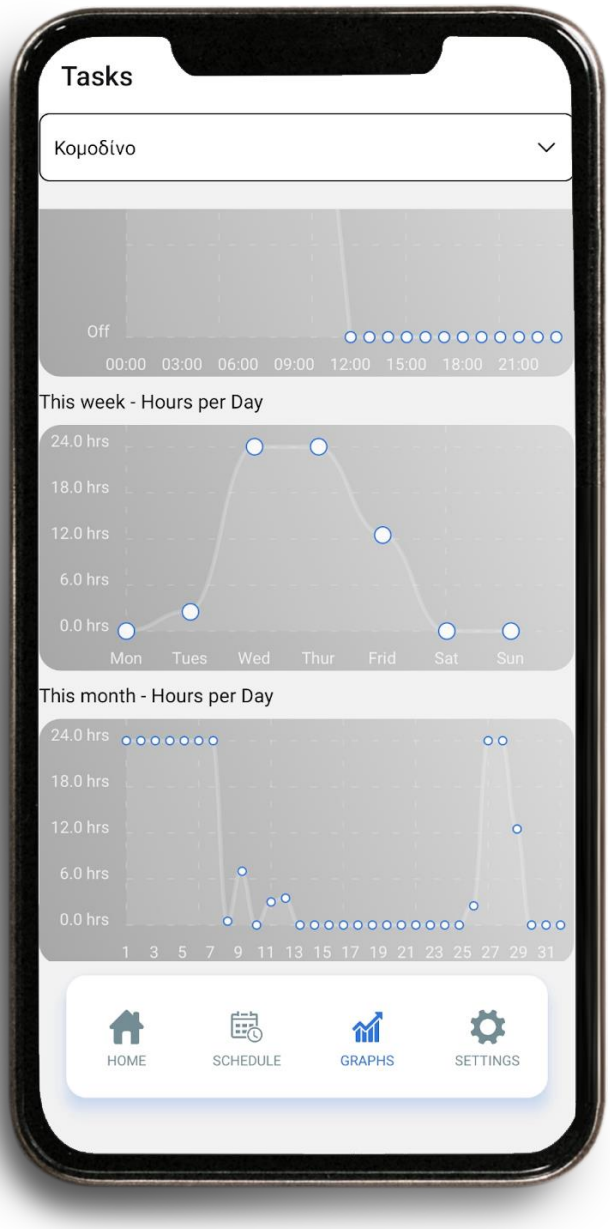
Το ημερήσιο πρόγραμμα λειτουργίας

Εδώ, οι τιμές του γραφήματος μπορούν να πάρουν μόνο τις τιμές on και off. Επομένως, χρησιμοποιώντας το συγκεκριμένο γράφημα, παρατηρούμε την εναλλαγή της λειτουργίας της εκάστοτε συσκευής στον ημερήσιο χρονικό άξονα .



Το εβδομαδιαίο και μηνιαίο πρόγραμμα λειτουργίας

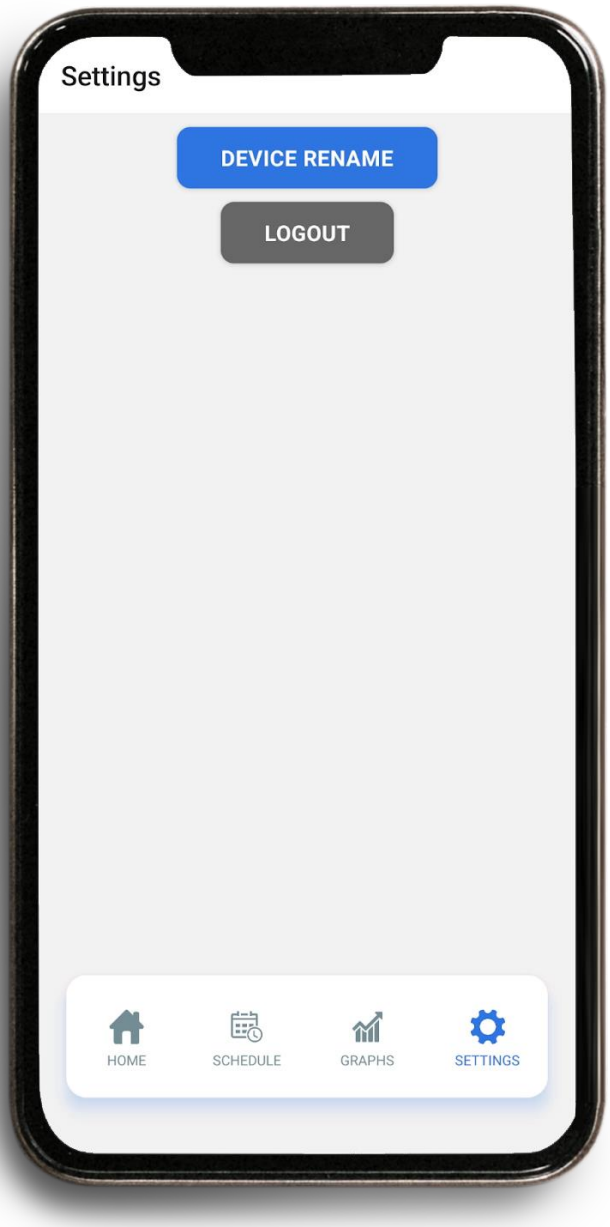
Στα συγκεκριμένα γραφήματα αναπαρίσταται η αθροιστική τιμή των ωρών λειτουργίας της εκάστοτε συσκευής ανά ημέρα.



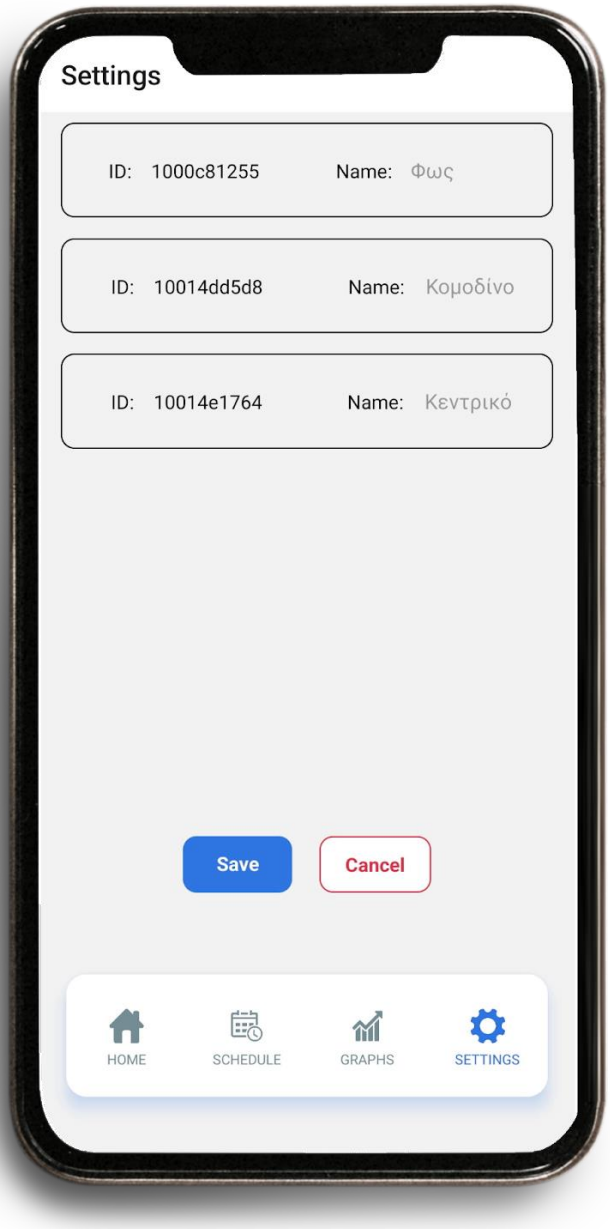
5.5. Settings

Η σελίδα **Settings**, παρέχει στο χρήστη τη δυνατότητα μετονομασίας των συσκευών του.

Το **logout** button θα οδηγήσει τον χρήστη σε αποσύνδεση από την εφαρμογή.



Κατά την επιλογή device rename, εμφανίζεται μία λίστα με τις καταχωρημένες συσκευές του λογαριασμού του συγκεκριμένου χρήστη. Η κάθε γραμμή περιλαμβάνει το device **ID** και το τρέχον **Name** της συσκευής. Ο χρήστης έχει τη δυνατότητα να εισάγει νέο όνομα για τη συσκευή και να αποθηκεύσει την επιλογή του.



6. Παρουσίαση - χρήση της smartwatch εφαρμογής.

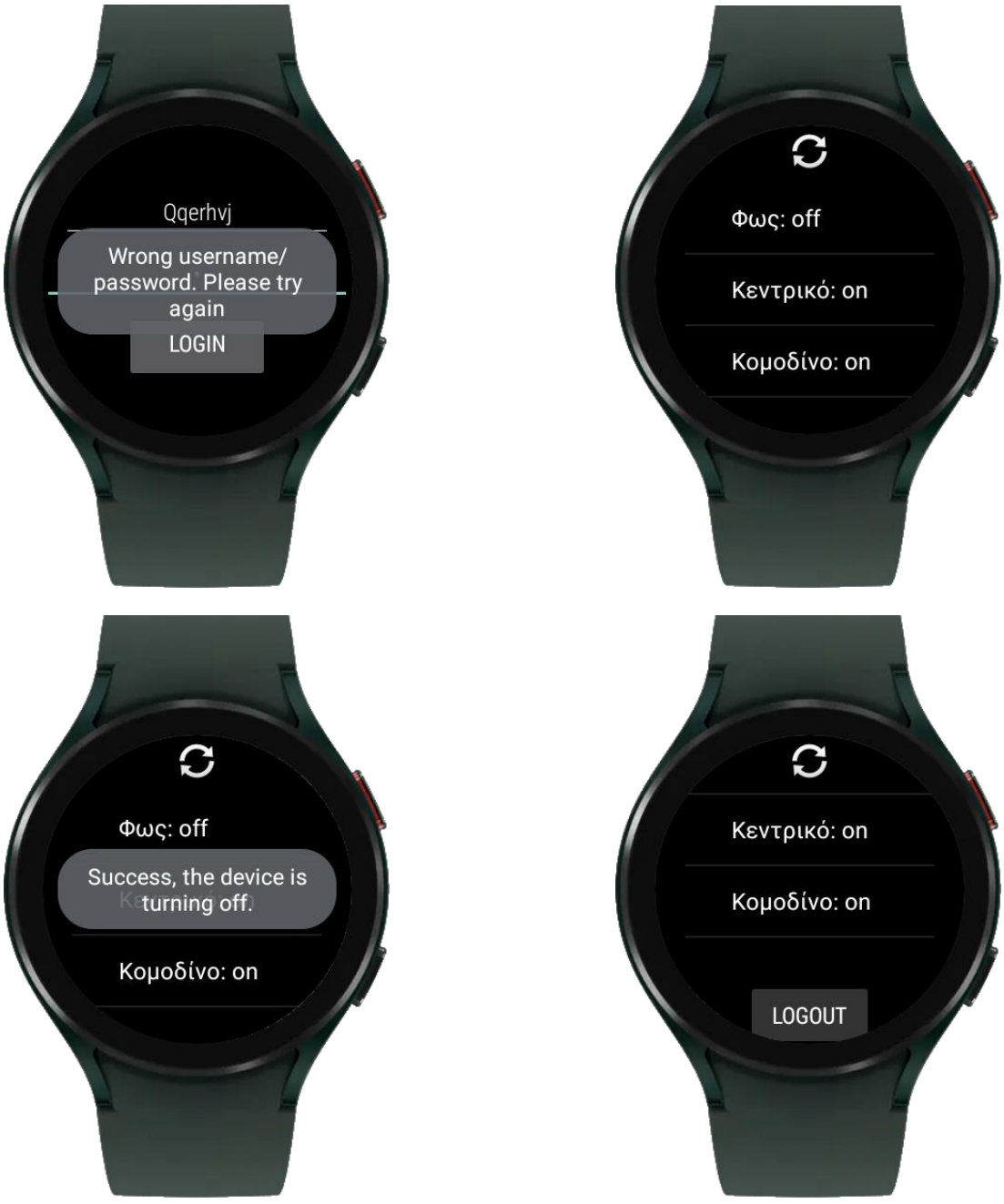
6.1. Login

Η σελίδα **login** όπου ο χρήστης πρέπει να εισάγει τα διαπιστευτήρια εισόδου του eWeLink account του.

Στη συνέχεια, και ενώ τα διαπιστευτήρια επιβεβαιωθούν, θα εμφανιστεί η **main** σελίδα της εφαρμογής, όπου θα εμφανιστεί η λίστα των συσκευών του. Στην περίπτωση που τα διαπιστευτήρια δεν επιβεβαιωθούν, θα εμφανιστεί σχετικό μήνυμα.

Επιλέγοντας κάποια συσκευή, πραγματοποιείται κλίση στον server για την αλλαγή της λειτουργίας του, και έπειτα ενημερώνεται το UI και ο χρήστης για την κατάσταση του αιτήματος.

Τέλος, στο κάτω μέρος της λίστας υπάρχει η επιλογή **Logout** για την αποσύνδεση από την εφαρμογή.



7. Συμπεράσματα

7.1. Σύνοψη

Ο στόχος που τέθηκε κατά την εκκίνηση της διαδικασίας ανάπτυξης της συγκεκριμένης εφαρμογής, ήταν να δημιουργηθεί ένα προϊόν το οποίο θα χαρακτηρίζεται από ένα μοντέρνο user interface, και η αλληλεπίδραση του χρήστη αυτό θα είναι απλή και εύκολη.

Επιπλέον, ήταν προϋπόθεση να μπορεί να προσφέρει τις υπηρεσίες που οι ανταγωνιστικές εφαρμογές διαθέτουν, τόσο σε επίπεδο αρχιτεκτονικής και σχεδίασης (auto login, χρήση loading animations κλπ.), όσο και στις κύριες λειτουργίες που αφορούν τις συσκευές του χρήστη (μετονομασία συσκευών, προγραμματισμός λειτουργίας κλπ.).

Εκτός αυτού, τέθηκε ο στόχος υλοποίησης λύσεων οι οποίες δεν προσφέρονται από τον ανταγωνισμό. Σε αυτή τη βάση, σχεδιάστηκε η σελίδα tasks με ορισμένες πρόσθετες δυνατότητες σε σύγκριση με έναν απλό scheduler, αλλά και η σελίδα γραφημάτων, μια δυνατότητα που λείπει από τις δημοφιλέστερες εφαρμογές του κλάδου.

Ένα ακόμη σημαντικό πρόσθετο της εφαρμογής, είναι η ανάπτυξη της συμβατής εφαρμογής με τα android smartwatches, η οποία επίσης σχεδιάστηκε με βάση την απλότητα και την ευκολία χρήσης. Η συγκεκριμένη λειτουργία, δεν είναι ιδιαίτερα διαδεδομένη στις εφαρμογές του χώρου αυτή τη στιγμή. Η δυνατότητα διαχείρισης της λειτουργίας των συσκευών του χρήστη μέσα από το ρολόι του, το οποίο αποτελεί ένα αξεσουάρ που φοράει καθ’ όλη τη διάρκεια της ημέρας, είναι πολύ χρήσιμη και έχει χώρο για πολλές προσθήκες, αφού τα σύγχρονα ρολόγια τείνουν να αποκτήσουν όμοιες δυνατότητες με τα κινητά τηλέφωνα.

Τέλος, η συμβατότητα της λύσης που αναπτύχθηκε ήταν εξίσου σημαντικό στοιχείο, ώστε να μην υπάρχουν περιορισμοί πρόσβασης για τον χρήστη. Για το λόγο αυτό, η ανάπτυξη πραγματοποιήθηκε με εργαλεία τα οποία επιτρέπουν τη δημιουργία μιας cross-platform εφαρμογής, ώστε να είναι προσβάσιμη από λειτουργικά συστήματα iOS και Android, αλλά και από οποιονδήποτε browser ως web app.

Οι στόχοι που αναφέρονται παραπάνω επιτεύχθηκαν και η OnOff είναι σε θέση να καλύψει σε πολύ μεγάλο βαθμό τις ανάγκες του χρήστη μιας τέτοιας εφαρμογής παρέχοντας λειτουργίες που χαρακτηρίζονται από αμεσότητα και ευκολία.
Επίσης παρέχονται καινοτόμες λειτουργίες, οι οποίες ανταποκρίνονται στα πρότυπα μιας σύγχρονης εφαρμογής.

7.2. Μελλοντικές επεκτάσεις

Είναι δεδομένο ότι οι έξυπνες συσκευές αποτελούν ένα φαινόμενο το οποίο αποκτά όλο και περισσότερους θαυμαστές με το πέρας του χρόνου. Μπορούμε με ασφάλεια να υποθέσουμε ότι οι λειτουργίες που θα παρέχονται στο μέλλον από τέτοιες συσκευές θα αυξηθούν σημαντικά, όπως και οι απαιτήσεις των χρηστών.

Αυτό αναπόφευκτα θα οδηγήσει στην ανάγκη προσθήκη νέων λειτουργιών στις εφαρμογές διαχείρισης έξυπνων συσκευών ώστε να μπορούν να ανταποκριθούν στις τρέχουσες απαιτήσεις της αγοράς.

Η συγκεκριμένη εφαρμογή αποτελεί μία πολύ καλή βάση πάνω στην οποία μπορούν στο μέλλον να προστεθούν νέες λειτουργίες για την κάλυψη των επικείμενων αναγκών. Ωστόσο, σκεπτόμενοι τις τρέχουσες ανάγκες της αγοράς, ακολουθεί μία λίστα από λειτουργίες οι οποίες θα μπορούσαν άμεσα να προστεθούν στις δυνατότητες της εφαρμογής και να παρέχουν στους χρήστες μία μεγαλύτερη γκάμα επιλογών.

Οι συγκεκριμένες δυνατότητες είναι:

1. **Υποστήριξη συσκευών από περισσότερες εταιρείες.**
   Η δυνατότητα υποστήριξης συσκευών από περισσότερες εταιρείες, αναμφίβολα θα κάνει την εφαρμογή να αποκτήσει μεγαλύτερη απήχηση στο ενδιαφερόμενο κοινό.

2. **Διαχείριση λειτουργίας συσκευών με περισσότερες επιλογές.**
   Ένα παράδειγμα της συγκεκριμένης δυνατότητας θα ήταν η διαχείριση ενός κλιματιστικού και η ενσωμάτωση λειτουργιών όπως ρύθμιση της θερμοκρασίας κλπ.

3. **Δημιουργία οικιών και δωματίων και ομάδων συσκευών**
   Η δυνατότητα δημιουργίας δωματίων, οικιών και ομάδων συσκευών, θα βοηθούσε τον χρήστη να οργανώσει καλύτερα τις συσκευές του και να πραγματοποιήσει ενέργειες μαζικής διαχείρισης.

4. **Ενσωμάτωση voice assistant**
   Η δυνατότητα ενσωμάτωσης ενός voice assistant ο οποίος θα καθιστούσε δυνατή τη διαχείριση των συσκευών μέσω φωνητικών εντολών, θα βελτίωνε αναμφίβολα την αλληλεπίδραση της εφαρμογής με το χρήστη σε επίπεδο ευκολίας και αμεσότητας.

5. **Αναβάθμιση server**
   Τέλος, η αναβάθμιση του server της εφαρμογής σε επίπεδο hardware, θα βοηθήσει στην ταχύτερη εξυπηρέτηση του αιτημάτων των χρηστών, βελτιώνοντας την εμπειρία χρήσης της εφαρμογής προσφέροντας άμεση ανταπόκριση των λειτουργιών της.

8. Βιβλιογραφία

https://www.statistics.gr/documents/20181/727080a4-4bb8-eb4d-af56-4aaaa8989b8c

https://support.google.com/chromecast/answer/7071794?hl=en&co=GENIE.Platform%3DAndroid

https://wiki.iteadstudio.com/EWeLink_Introduction

https://reactnative.dev/

https://nodejs.org/en/about/

https://developer.android.com/wear

https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html