ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Smart Living

Ανθή Ν. Ζαφείρη
Αθανάσιος Σ. Πατάπης

Επιβλέποντες: **Αθανασία Αλωνιστιώτη,** Αναπληρώτρια Καθηγήτρια
Σωκράτης Μπαρμπουνάκης, Μεταδιδακτορικός Ερευνητής
Σουκαράς Δημήτρης, Μεταπτυχιακός Ερευνητής

ΑΘΗΝΑ

Οκτώβριος 2019

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Smart Living

Ανθή Ν. Ζαφείρη
Α.Μ.: 1115200700054

Αθανάσιος Σ. Πατάπης
Α.Μ.: 1115200700131

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Αθανασία Αλωνιστιώτη,** Αναπληρώτρια Καθηγήτρια
Σωκράτης Μπαρμπουνάκης, Μεταδιδακτορικός Ερευνητής
Σουκαράς Δημήτρης, Μεταπτυχιακός Ερευνητής

ΠΕΡΙΛΗΨΗ

Αυτή η πτυχιακής είναι η ανάπτυξη μίας εφαρμογής - στα πλαίσια των τεχνολογιών έξυπνης διαβίωσης (smart living) – που δίνει τη δυνατότητα στο χρήστη να παρακολουθεί και να διαχειρίζεται συγκεκριμένους πόρους (resources) με την βοήθεια αισθητήρων (sensors). Για την ακρίβεια, η παρακολούθηση (monitoring) γίνεται απομακρυσμένα και ο ενδιαφερόμενος μπορεί να ενημερώνεται για την κατάσταση και τη διαθεσιμότητα, αλλά και να παραμετροποιεί διάφορους παράγοντες της παρακολούθησης οποιαδήποτε στιγμή.

Για την επίτευξη του αντικειμένου χρησιμοποιούνται sensors (ή προσομοιώνονται sensors) που θεωρούμε ότι εγκαθίστανται σε μία φυσική δομή και οι οποίοι στέλνουνε μηνύματα σχετικά με τη διαθεσιμότητα και κατάσταση των resources σε έναν server (broker) σύμφωνα με το πρωτόκολλο mqtt. Το πρωτόκολλο αυτό αποτελεί ένα δίκτυο δημοσίευσης (publish) – εγγραφής (subscribe) που βοηθάει στην αποστολή μηνυμάτων μεταξύ δύο συσκευών, στη συγκεκριμένη περίπτωση χρησιμοποιείται για την αποστολή μηνυμάτων από τους sensors προς έναν server, τον broker. Στη συνέχεια, το σύστημά μας (που αποτελείται από εφαρμογές: μία προσομοιώνει τους sensors και μία αποτελεί τον server του συστήματος και είναι υπεύθυνος για όλες τις λειτουργίες του συστήματος) διαβάζει με ασύγχρονο τρόπο τα μηνύματα από τον broker και τα αποθηκεύει ή τα επεξεργάζεται. Έχει υλοποιηθεί ακόμη ένα mobile application το οποίο δείχνει την πληροφορία αυτή στον ενδιαφερόμενο, ο οποίος μπορεί να ενημερωθεί για τη διαθεσιμότητα ή την κατάσταση των resources που έχει επιλέξει να κάνει monitor ή και να παραμετροποιήσει τα χαρακτηριστικά της φυσικής δομής όπου βρίσκονται οι sensors. Αναφορικά κατά την εκπόνηση της πτυχιακής εργασίας θεωρούμε σαν ενδεικτικό παράδειγμα, για να γίνουν κατανοητά τα παραδείγματα και οι επεξηγήσεις, τη φυσική εγκατάσταση μιας κουζίνας, θεωρώντας ότι εγκαθιστούμε sensors σε όλα τα δυνατά τμήματα μιας ραφιέρας και παρακολουθούμε τα διάφορα προϊόντα που υπάρχουν στα τμήματα αυτά.

Τα αποτελέσματα της υλοποίησης του συστήματος που περιγράψαμε αφορούν τη δυνατότητα απομακρυσμένης ενημέρωσης από το χρήστη για τη διαθεσιμότητα και κατάσταση των προϊόντων που παρακολουθεί, τη δυνατότητα πρόληψης έλλειψης αποθεμάτων με την υλοποίηση αποστολής ενημερώσεων προς το χρήστη εάν το προϊόν κοντεύει να εξαντληθεί και τη δυνατότητα εξαγωγής στατιστικών δεδομένων ώστε γίνεται καλύτερη διαχείριση των resources. Το σύστημα που περιγράφουμε θεωρούμε ότι μπορεί να έχει πολλές εφαρμογές, όπως για έναν ιδιώτη (ενδεικτικα παραδείγματα: άνθρωποι που ενδιαφέρονται για την εισαγωγή της ιδέας του smart living στην καθημερινή ζωή, άνθρωποι με ειδικές ανάγκες, ηλικιωμένοι που δεν έχουν κάποιον που να μπορεί να τους φροντίζει, κλπ) ή για επαγγελματική χρήση (ενδεικτικά παραδείγματα: μεγάλες επιχειρήσεις με αποθήκες αποθεμάτων, ράφια σε σουπερ μάρκετ, και άλλα).

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Smart Living

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Sensors, mqtt, iot (internet of things), REST, web application

ABSTRACT

This thesis is the development of an application - in the context of smart living technologies - that enables the user to monitor and manage specific resources with the help of sensors. In fact, monitoring is done remotely and the stakeholder can be informed about the status and availability, but also can configure various monitoring factors at any time.

Sensors (or simulated sensors) are used to achieve the object which we consider to be installed in a physical structure and send messages about the availability and status of resources to a server (broker) according to the mqtt protocol. This protocol is a publish-subscribe network that helps send messages between two devices, in this case it is used to send messages from sensors to a server, the broker. Then, our system (which consists of applications: one simulates the sensors and the other is the server of the system and is responsible for all the functions of the system) reads asynchronously the messages from the broker and saves or processes them. Another mobile application has been implemented that shows this information to the stakeholder, who can be informed of the availability or status of the resources he has chosen to monitor or even customize the features of the physical structure where the sensors are located. When implementing the components of this thesis we consider as an illustrative example, in order to understand the examples and explanations used throughout this paper, the physical installation of a kitchen, considering that we install sensors in all possible sections of a rack and monitor the various products available in these sections.

The results of the implementation of the system that we have described relate to the ability of the user to be remotely informed about the availability and status of the products he is monitoring, the ability to prevent shortages by delivering updates to the user if the product is nearing depletion, and the ability to extract statistics data to help better manage resources. We consider that the system we are describing can have many applications, such as for an individual (examples include: people interested in introducing the idea of smart living into everyday life, people with disabilities, older people who do not have a caregiver, etc.) or for professional use (examples include: large businesses with warehouses, supermarket shelves, and more).

SUBJECT AREA: Smart Living

KEYWORDS: Sensors, mqtt, iot (internet of things), REST, web application

ΕΥΧΑΡΙΣΤΙΕΣ

Για την εκπόνηση της παρούσας πτυχιακής εργασίας, θα θέλαμε ιδιαιτέρως να ευχαριστήσουμε τους επιβλέποντες, αν. καθ. Αθανασία Αλωνιστιώτη, μεταδιδακτορικό ερευνητή Σωκράτη Μπαρμπουνάκη, μεταπτυχιακό ερευνητή Δημήτρη Σουκαρά, για τη συνεργασία, την πολύτιμη συμβολή και την αμέριστη υπομονή που υπέδειξαν για την ολοκλήρωση της.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΠΡΟΛΟΓΟΣ

Τα βασικά πλαίσια με τα οποία ασχολείται η πτυχιακή αυτή εργασία είναι το smart living και το internet of things.

Το Smart Living είναι ένα σύγχρονο concept που περιλαμβάνει εξελίξεις που δίνουν στους ανθρώπους την ευκαιρία να επωφεληθούν από νέους τρόπους ζωής. Περιλαμβάνει πρωτότυπες και καινοτόμες λύσεις που στοχεύουν να καταστήσουν τη ζωή πιο αποτελεσματική, ελεγχόμενη, οικονομική, παραγωγική, ολοκληρωμένη και βιώσιμη [1]. Αυτή είναι μια τάση που καλύπτει όλες τις πτυχές της καθημερινής ζωής, από κατοικίες, χώρους εργασίας καθώς και τον τρόπο με τον οποίο μετακινούνται οι άνθρωποι μέσα στις πόλεις. Εν ολίγοις, η έξυπνη διαβίωση (smart living) συνεπάγεται βελτίωση προτύπων σε διάφορες πτυχές της ζωής, ενώ παράλληλα προάγει την αποδοτικότητα, την οικονομία και τη μείωση του αποτυπώματος του άνθρακα [2].

Ακόμη, το διαδίκτυο των πραγμάτων (internet of things, short: IoT) είναι ένα σύστημα αλληλένδετων υπολογιστικών συσκευών, μηχανικών και ψηφιακών μηχανών, αντικειμένων, ζώων ή προσώπων που διαθέτουν μοναδικά αναγνωριστικά στοιχεία (UID) και τη δυνατότητα μεταφοράς δεδομένων μέσω δικτύου χωρίς να απαιτείται αλληλεπίδραση ανθρώπου με άνθρωπο ή ανθρώπου με υπολογιστή [3]. Η τεχνολογία IoT είναι περισσότερο συνώνυμη με τα προϊόντα που εμπίπτουν στην έννοια του "έξυπνου σπιτιού", καλύπτοντας συσκευές (όπως φωτιστικά, θερμοστάτες, συστήματα οικιακής ασφάλειας, κάμερες και άλλες οικιακές συσκευές) που υποστηρίζουν ένα ή περισσότερα κοινά οικοσυστήματα και μπορούν να ελέγχονται μέσω συσκευών που σχετίζονται με αυτό το οικοσύστημα, όπως τα smartphones και τα έξυπνα ηχεία. Στην περίπτωση μας οι συσκευές που χρησιμοποιούμε είναι οι αισθητήρες (sensors) και smartphones, καθώς και υπολογιστικά συστήματα για την υλοποίηση του συστήματος.

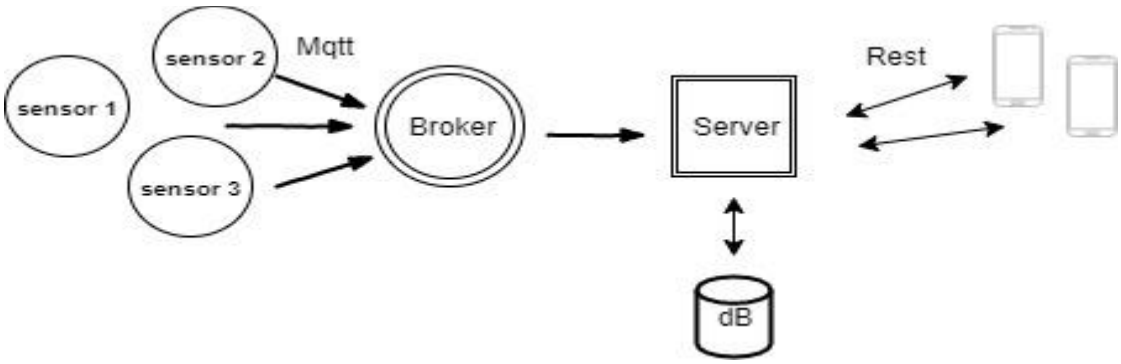
Στα δύο αυτά πλαίσια βασίζεται η πτυχιακή αυτή εργασία, χρησιμοποιώντας το internet of things για να προάγουμε την ιδέα του smart living, ελέγχοντας πόρους / προϊόντα με sensors που έχουν εγκατασταθεί σε μια φυσική δομή και επικοινωνούν απευθείας με το smartphone του ενδιαφερόμενου, χωρίς να είναι απαραίτητη η παρουσία του ανθρώπου, ώστε να επωφεληθούμε από τις δυνατότητες που μας προσφέρονται σε καθημερινή βάση καθιστώντας την καθημερινότητα πιο εύκολη. Μερικά παραδείγματα χρήσης που περιγράφουν τους ενδιαφερόμενους μιας τέτοιας τεχνολογίας αφορούν μεγάλες επιχειρήσεις που έχουν μεγάλες αποθηκευτικές ανάγκες, τα ράφια ενός super market ή ακόμη και ιδιώτες, άνθρωποι με ειδικές ανάγκες ή ηλικιωμένοι.

Τόπος διεξαγωγής της πτυχιακής εργασίας: Αθήνα, Εθνικό Καποδιστριακό Πανεπιστήμιο Αθηνών

1. ΔΟΜΗ (INSTALLATION)

1.1 Περιγραφή της υλοποίησης

Η δομή που έχει το σύστημά μας αποτελείται από οντότητες που λέγονται storage units. Ένα storage unit αντιπροσωπεύει μια φυσική εγκατάσταση η οποία αποτελείται από επίπεδα (levels) και κάθε ένα από αυτά μπορεί να έχει μία ή περισσότερες θέσεις αποθήκευσης (containers). Για να γίνει η δομή αυτή πιο κατανοητή, μπορούμε να την παρομοιάσουμε με έναν αποθηκευτικό χώρο, για παράδειγμα τα ράφια μιας κουζίνας. Τα ράφια αποτελούν ένα storage unit και μπορούν να είναι χωρισμένα σε επίπεδα (levels), καθώς επίσης κάθε επίπεδο μέσα στο storage unit μπορεί να είναι χωρισμένο σε μικρότερα μέρη (containers). Σε κάθε μέρος (container) του storage unit εγκαθιστούμε (στην πράξη προσομοιώνουμε με την εφαρμογή sensor που έχουμε υλοποιήσει) έναν sensor, ο οποίος παρακολουθεί το περιεχόμενο του container και στέλνει μηνύματα σε έναν server (mqtt broker) με τα χαρακτηριστικά του περιεχομένου που παρακολουθεί. Ο broker λαμβάνει τα μηνύματα αυτά και τα δρομολογεί στους ενδιαφερόμενους, δηλαδή στο server της εφαρμογής που έχουμε υλοποιήσει. Ο server αποθηκεύει και διαχειρίζεται τα μηνύματα αυτά και ενημερώνει την βάση δεδομένων και τον χρήστη για την κατάσταση και τα χαρακτηριστικά με την βοήθεια ενός REST API σε ένα mobile application που έχει υλοποιηθεί, όπου ο χρήστης μπορεί να πλοηγηθεί, να ενημερωθεί και να παραμετροποιήσει τα δεδομένα του συστήματος. Οι οντότητες της δομής που περιγράψαμε φαίνονται και στην παρακάτω εικόνα.



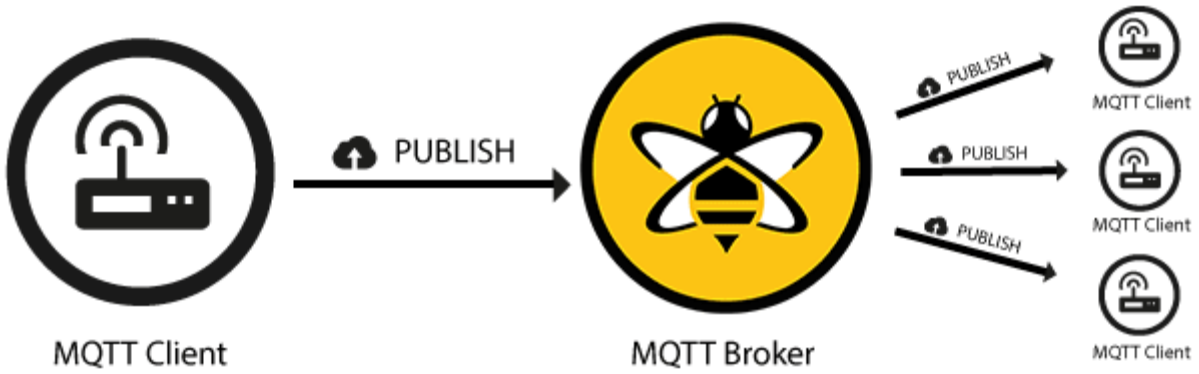
Εικόνα 1: Οντότητες της δομής που έχει υλοποιηθεί

2. ΠΡΩΤΟΚΟΛΛΟ MQTT

2.1 Περιγραφή του πρωτοκόλλου MQTT για την επικοινωνία των sensors με το server

Το πρωτόκολλο MQTT που χρησιμοποιήσαμε για την υλοποίηση της επικοινωνίας των sensors με τον server είναι ένα ελαφρύ σύστημα δημοσίευσης και εγγραφής(publish and subscribe) μέσω του οποίου μπορούμε να δημοσιεύσουμε και να λάβουμε μηνύματα ως πελάτης (client). Το MQTT είναι ένα απλό πρωτόκολλο μηνυμάτων, σχεδιασμένο για περιορισμένες συσκευές με χαμηλό εύρος ζώνης. Έτσι, είναι η τέλεια λύση για εφαρμογές του Internet of Things. Το MQTT επιτρέπει να στέλνονται μηνύματα για τον έλεγχο κάποιων resources, να διαβάζονται και να δημοσιεύονται δεδομένα από τους κόμβους των αισθητήρων και άλλα. Επομένως, είναι πολύ εύκολο να δημιουργηθεί μια επικοινωνία μεταξύ πολλών συσκευών.

Τα βασικά concepts που θα πρέπει να κατανοήσουμε που αφορούν το πρωτόκολλο MQTT είναι η δημοσίευση και εγγραφή, τα μηνύματα (messages), τα θέματα (topics) και ο “μεσίτης” (broker). Τα concepts αυτά, όπως φαίνονται και στην εικόνα [4], περιγράφονται πιο κάτω.



Εικόνα 2: Τα βασικά concepts του πρωτοκόλλου MQTT

2.1.1 Publish/Subscribe

Σε ένα σύστημα όπως αυτό όπου επιτρέπεται το publish και subscribe, ένας sensor μπορεί να κάνει publish ένα message σε ένα topic, δηλαδή να στέλνει μηνύματα για ένα συγκεκριμένο θέμα, ή ο subscriber που δημιουργείται από την εφαρμογή του server μπορεί να κάνει subscribe σε ένα topic και να δέχεται τα messages, δηλαδή να ακούει σε κάποιο ή και κάποια topics και να λαμβάνει μηνύματα όταν ο sensor στέλνει για το topic αυτό ένα μήνυμα.

2.1.2 Messages

Τα μηνύματα, δηλαδή η πληροφορία που θέλουμε να ανταλλάσσεται ανάμεσα στον sensor και τον server μας. Στην περίπτωσή μας, η πληροφορία αυτή αφορά χαρακτηριστικά του περιεχομένου του container, όπως το όνομα ή το βάρος του προϊόντος που παρακολουθείται μέσα στο container αυτό.

2.1.3 Topics

Τα topics είναι ο τρόπος με τον οποίο καταχωρούμε ενδιαφέρον για τα εισερχόμενα μηνύματα ή τον τρόπο με τον οποίο καθορίζουμε πού θέλουμε να δημοσιευθεί το μήνυμα. Τα topics είναι της μορφής storage-unit_storage-unitId/level_levelId/container_containerId και με τον τρόπο αυτό γνωρίζουμε ότι ο sensor που βρίσκεται για παράδειγμα στο container με containerId 1, στο level με levelId 1 και στο storage unit με storage-unitId 1 στέλνει μηνύματα στο topic: storage-unit_1/level_1/container_1. Θα μπορούσαμε να πούμε με αυτόν τον τρόπο για χάρη της γενίκευσης, ότι κατά την υλοποίησή μας ένα topic αντιστοιχεί σε ένα container.

2.1.4 Broker

Ο broker είναι κυρίως υπεύθυνος για τη λήψη όλων των μηνυμάτων, το φιλτράρισμα των μηνυμάτων, την επιλογή του ενδιαφερομένου και τη δημοσίευση του μηνύματος σε όλους τους εγγεγραμμένους πελάτες, δηλαδή τους subscribers. Με λίγα λόγια είναι αυτός που λαμβάνει όλα τα messages από τους sensors και τα δρομολογεί ξανά στα κατάλληλα topics και επομένως εν συνεχεία φτάνουν στους subscribers. Ο mqtt broker που χρησιμοποιούμε στην υλοποίηση της πτυχιακής εργασίας είναι ο Mosquitto broker, ο οποίος είναι ένας open source mqtt broker του eclipse foundation.

3. ΕΦΑΡΜΟΓΗ SENSOR

3.1.1 Περιγραφή της εφαρμογής που έχει υλοποιηθεί για τους αισθητήρες (sensors)

Παράλληλα με την εφαρμογή του server και την εφαρμογή του κινητού για το χρήστη και το διαχειριστή, έχει υλοποιηθεί και μια εφαρμογή για την προσομοίωση των sensors και τη λειτουργία τους στο σύστημα. Η μη δυνατότητα ύπαρξης και χρήσης φυσικών sensors καθ' όλη τη διάρκεια της υλοποίησης της πτυχιακής εργασίας οδήγησε στην επιτακτική ανάγκη υλοποίησης της εφαρμογής Sensors, ώστε να καλύπτεται, να ελέγχεται και να επιβεβαιώνεται η λειτουργία ολόκληρου του συστήματος. Στην εφαρμογή αυτή λοιπόν, καλύπτονται τρεις βασικές λειτουργίες των sensors:

- Η πρώτη αφορά την προσομοίωση αποστολής ενός μηνύματος προς τον mqtt broker ή αλλιώς ο sensor κάνει publish ένα mqtt message σε ένα topic.
- Η δεύτερη αφορά την προσομοίωση εξακολουθητικής αποστολής μηνυμάτων ανά ένα συγκεκριμένο χρονικό διάστημα προς τον mqtt broker για όσες φορές δηλωθεί να γίνει η αποστολή.
- Η τρίτη πρόκειται για τη δυνατότητα να σταματήσει ένας sensor να αποστέλλει μηνύματα προς τον mqtt broker, εφόσον έχει ξεκινήσει να στέλνει εξακολουθητικά μηνύματα αλλά δεν έχει φτάσει ακόμη το τέλος της λειτουργίας του, δεν έχουν τελειώσει οι φορές που θα πρέπει η αποστολή ενός μηνύματος να επαναληφθεί.

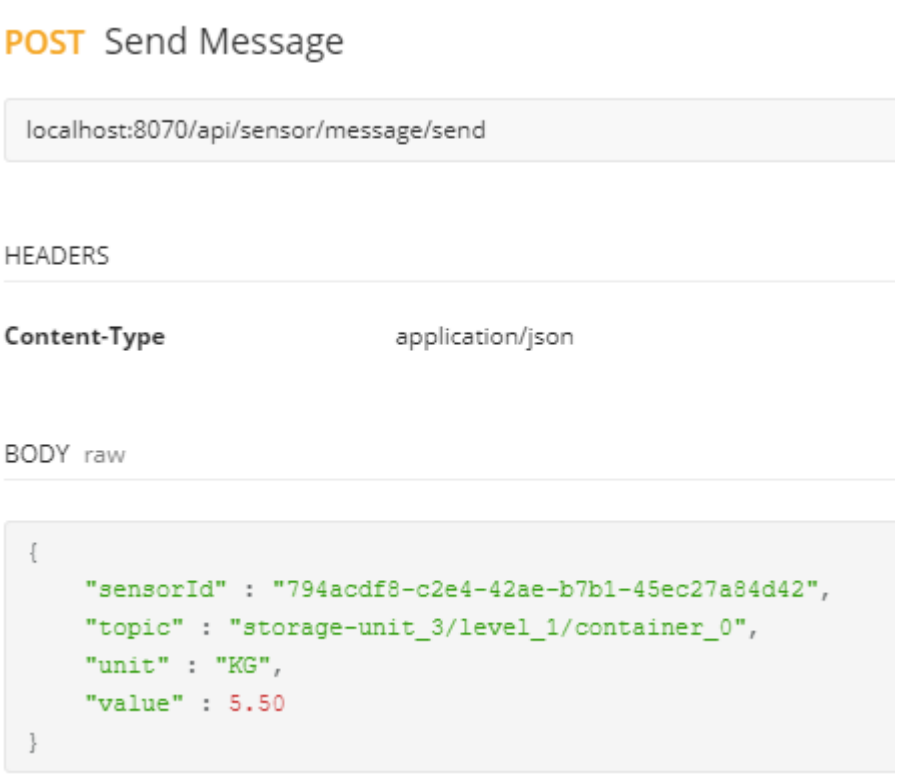
3.1.2 Περιγραφή των REST κλήσεων που έχουν υλοποιηθεί για την προσομοίωση της λειτουργίας των sensors

Όπως περιγράψαμε και πριν, υπάρχουν τρεις βασικές λειτουργίες που προσομοιώνονται στην εφαρμογή Sensor, οπότε και έχουν υλοποιηθεί αντίστοιχα τρεις κλήσεις REST για να καλύπτονται αυτές οι λειτουργίες.

3.1.2.1 POST send message

Η μέθοδος αυτή αφορά όπως είπαμε την προσομοίωση αποστολής ενός μηνύματος από το sensor, δηλαδή της δυνατότητας να κάνει publish ο sensor ένα message σε ένα topic, το οποίο περνάμε σαν όρισμα στην κλήση της μεθόδου με το χαρακτηριστικό όνομα του topic, το οποίο αποτελείται από: storage-unit_storageUnitId/level_levelId/container_containerId. Το μήνυμα, όπως φαίνεται και στο body του json της rest κλήσης, αποτελείται από τα εξής χαρακτηριστικά:

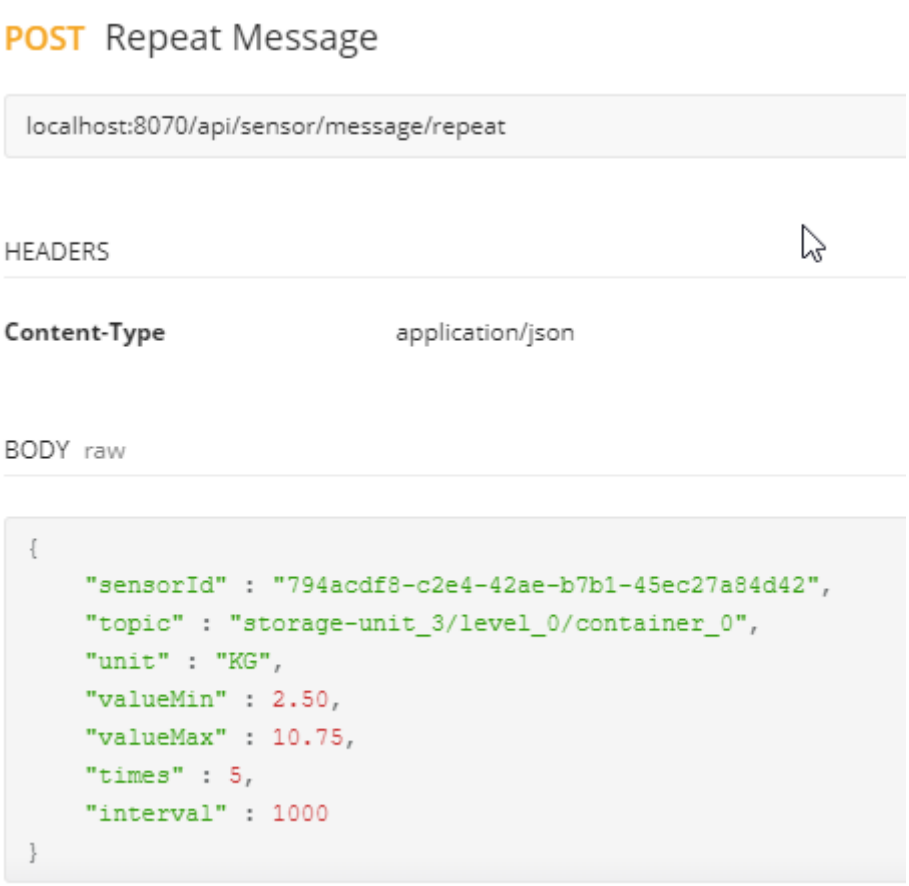
- Το διακριτό χαρακτηριστικό id του sensor ώστε να γνωρίζει η εφαρμογή ποιος sensor στέλνει το συγκεκριμένο μήνυμα.
- Το topic στο οποίο θα γίνει publish το μήνυμα αυτό και είναι της μορφής που ήδη αναφέρθηκε.
- Τη μονάδα μέτρησης (unit) του προϊόντος που παρακολουθεί ο συγκεκριμένος sensor.
- Την τιμή (value) τη χρονική αυτή στιγμή.



Εικόνα 3: Μέθοδος Send Message

3.1.2.2 POST repeat message

Όπως είδαμε και πιο πάνω, η μέθοδος αυτή αφορά την προσομοίωση της εξακολουθητικής αποστολής μηνυμάτων προς τον broker. Για να επιτευχθεί η προσομοίωση το body του json της κλήσης αυτής διαφέρει λίγο από την προηγούμενη μέθοδο. Για τη repeat μέθοδο το body αποτελείται, όπως φαίνεται και από την εικόνα παράδειγμα από κάτω, από: α) το id του sensor, β) το topic στο οποίο θα κάνει publish ο sensor, γ) το unit του προϊόντος, δ) το μικρότερο δυνατό value (valueMin) που μπορεί να πάρει το value του προϊόντος που παρακολουθούμε, ε) το μεγαλύτερο δυνατό value (valueMax) στ) τις φορές (times) που θέλουμε να επαναληφθεί η αποστολή του μηνύματος και ζ) το μεσοδιάστημα (interval) που θέλουμε να υπάρχει μεταξύ δύο συνεχόμενων αποστολών ενός μηνύματος σε δευτερόλεπτα. Για να επιτευχθεί λοιπόν η προσομοίωση, η εφαρμογή που έχει υλοποιηθεί μόλις λάβει μέθοδο repeat στέλνει μία τυχαία τιμή στον broker η οποία είναι ανάμεσα στο ανώτατο και κατώτατο value (valueMax – valueMin).



Εικόνα 4: Μέθοδος Repeat Message

3.1.2.3 PUT stop message

Ουσιαστικά η λειτουργία της μεθόδου αυτής είναι να σταματάει την προηγούμενη μέθοδο εάν δεν έχει προλάβει να τελειώσει τις φορές που θα πρέπει να επαναληφθεί. Σταματάει δηλαδή το sensor με sensorId που πήρε από την repeat μέθοδο να αποστέλλει μηνύματα προς τον broker.

4. ΕΦΑΡΜΟΓΗ SERVER

4.1.1 Περιγραφή της εφαρμογής server του συστήματος.

Κατά την αρχικοποίηση ενός storage unit, το id του δημιουργείται αυτόματα μέσω της sequence που αναφέραμε και κατά την περιγραφή της βάσης δεδομένων που χρησιμοποιούμε.

Έχουμε κάνει την παραδοχή ότι για κάθε φυσική εγκατάσταση έχουμε έναν μέγιστο αριθμό από storage units που μπορούν να εξυπηρετηθούν από το σύστημα, για παράδειγμα 10. Ο μέγιστος αυτός αριθμός ορίζεται και παραμετροποιείται στο αρχείο των properties του συστήματος (δηλαδή το application.yml), το οποίο διαβάζει ο server μόλις ξεκινήσει για να πάρει τα απαραίτητα στοιχεία που χρειάζεται για την αρχικοποίηση και παραμετροποίηση του συστήματος. (Για περισσότερες πληροφορίες για το αρχείο application.yml, δείτε στα References).

Τα levels/containers ορίζονται στο σύστημά μας ως μια δισδιάστατη λίστα, ή πιο απλά ως λίστες από λίστες. Δηλαδή τα levels αποτελούν λίστες που περιέχουν μέσα τους άλλες λίστες (τα containers), οι οποίες ακολούθως αποτελούν αντικείμενα τύπου Container.

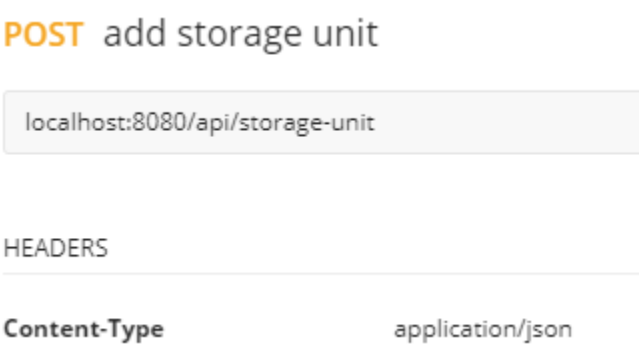
Μόλις ξεκινήσει ο server και αρχικοποιηθεί το storage unit που δημιουργούμε, τότε αποθηκεύεται στην βάση δεδομένων με τον τρόπο που περιγράψαμε πιο πάνω.

4.1.2 Περιγραφή των REST κλήσεων που διαχειρίζονται ένα storage unit στο σύστημα

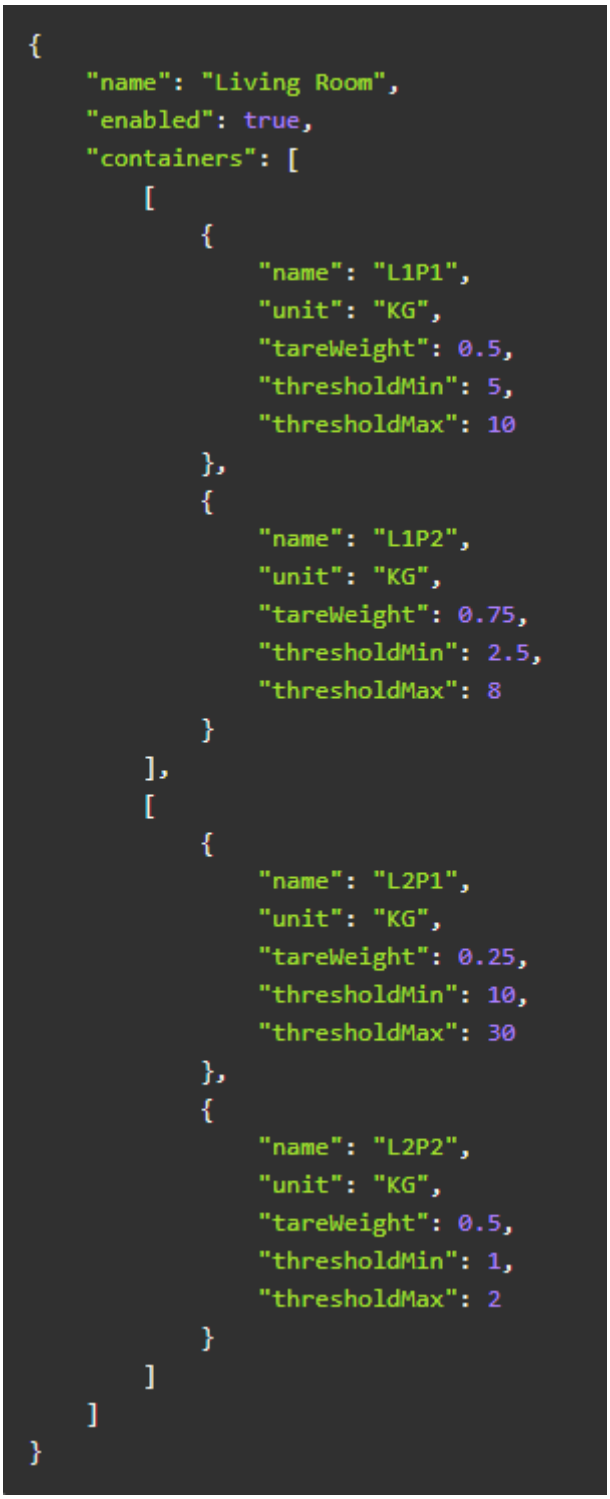
Στη συνέχεια θα αναλύσουμε τις REST κλήσεις που γίνονται ανάμεσα στο server και την εφαρμογή του χρήστη, για την ενημέρωση, ή του διαχειριστή, για την παραμετροποίηση του storage unit.

4.1.2.1 POST add storage unit

Με τη μέθοδο αυτή ο διαχειριστής πραγματοποιεί μία REST κλήση για την προσθήκη ενός νέου storage unit στο σύστημα. Το σώμα (body) της REST κλήσης είναι όπως φαίνεται στην παρακάτω εικόνα και αποτελεί την ακριβή περιγραφή ενός storage unit. Στο συγκεκριμένο παράδειγμα, προσθέτουμε στο σύστημα ένα storage unit, το οποίο ξεκινάει ως disabled (δηλαδή, δε θα παρακολουθείται από την εφαρμογή του χρήστη για ενημέρωση και παραμετροποίηση) και περιέχει 4 containers με τα χαρακτηριστικά τους (όνομα, μονάδα μέτρησης, απόβαρο, ανώτατο και κατώτατο όριο ή αντίστοιχα name, unit, tareWeight, thresholdMin, thresholdMax) και τις τιμές αυτών. Στην παρακάτω εικόνα μπορούμε να δούμε στο json ότι δημιουργούμε δύο δισδιάστατες λίστες containers, μία με τα δύο πρώτα blocks πληροφοριών και μία δεύτερη με τα δύο τελευταία blocks πληροφοριών, εννοώντας με αυτόν τον τρόπο ότι δημιουργούμε δύο levels, όπου το καθένα περιέχει δύο θέσεις. Με τον τρόπο αυτό δημιουργούμε και προσθέτουμε ένα καινούριο storage unit στο σύστημα.



Εικόνα 5: Add Storage Unit



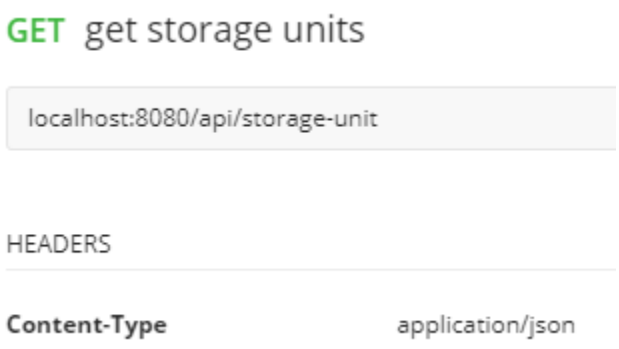
```
{
    "name": "Living Room",
    "enabled": true,
    "containers": [
        [
            {
                "name": "L1P1",
                "unit": "KG",
                "tareWeight": 0.5,
                "thresholdMin": 5,
                "thresholdMax": 10
            },
            {
                "name": "L1P2",
                "unit": "KG",
                "tareWeight": 0.75,
                "thresholdMin": 2.5,
                "thresholdMax": 8
            }
        ],
        [
            {
                "name": "L2P1",
                "unit": "KG",
                "tareWeight": 0.25,
                "thresholdMin": 10,
                "thresholdMax": 30
            },
            {
                "name": "L2P2",
                "unit": "KG",
                "tareWeight": 0.5,
                "thresholdMin": 1,
                "thresholdMax": 2
            }
        ]
    ]
}
```


Εικόνα 6: Add Storage Unit (JSON Body)

4.1.2.2 GET get storage units

Με τη μέθοδο Get storage units ο διαχειριστής ή ο χρήστης πραγματοποιεί μία REST κλήση για να ζητήσει από το σύστημα μία λίστα με τα διαθέσιμα storage units. Στην περίπτωση του διαχειριστή η μέθοδος αυτή επιστρέφει όλα τα storage units, είτε είναι enabled είτε όχι, ενώ

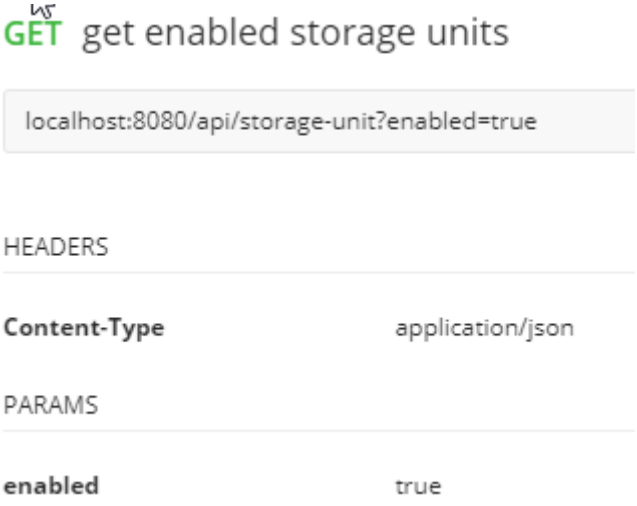
στην περίπτωση του χρήστη η παράμετρος enabled είναι πάντα true, δηλαδή μπορεί να δει μόνο τα storage units που είναι enabled.



Εικόνα 7: Get Storage Units

4.1.2.3 GET get enabled storage units

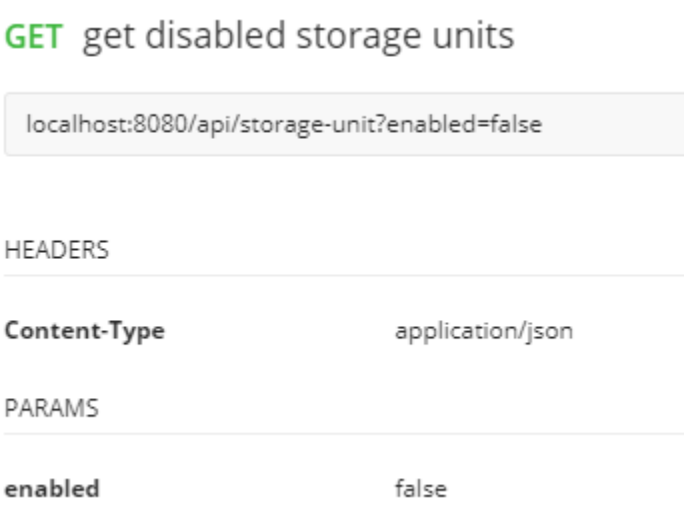
Αντίστοιχα η μέθοδος Get enabled storage units καλείται από το διαχειριστή και επιστρέφει όλα τα ενεργά storage units, δηλαδή αυτά που έχουμε επιλέξει να παρακολουθούμε.



Εικόνα 8: Get Enabled Storage Units

4.1.2.4 GET get disabled storage units

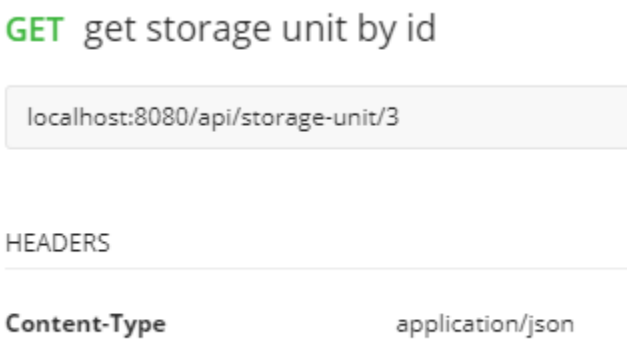
Με το ίδιο σκεπτικό, η μέθοδος Get disabled storage units επιστρέφει μία λίστα με τα ανενεργά storage units, αυτά που έχουμε επιλέξει να μην παρακολουθούμε.



Εικόνα 9: Get Disabled Storage Units

4.1.2.5 GET get storage unit by id

Η μέθοδος Get storage unit by id επιστρέφει σαν απάντηση (response) την ολοκληρωμένη δομή μιας φυσικής εγκατάστασης, ενός δηλαδή συγκεκριμένου storage unit. Ουσιαστικά στο response βλέπουμε τη δισδιάστατη λίστα που ανπιπροσωπεύει τα levels και τα containers με τα χαρακτηριστικά τους, όπως φαίνεται στην εικόνα Example Response από κάτω.



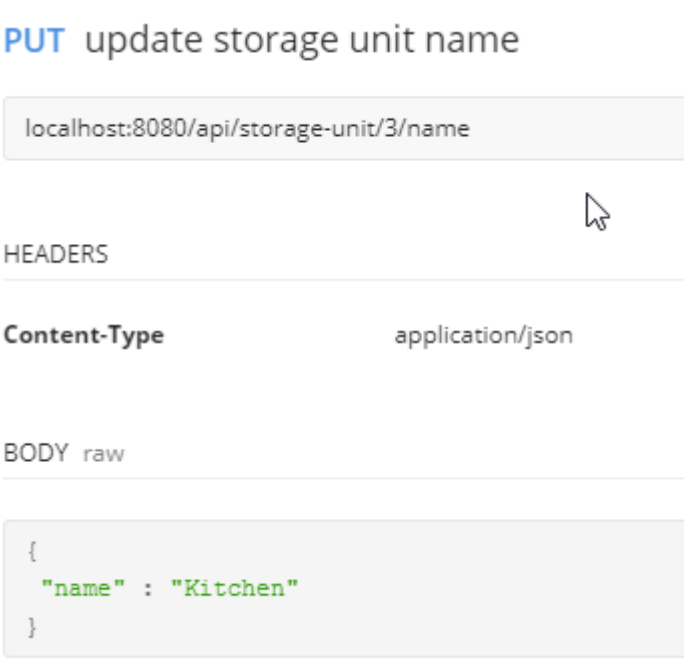
Εικόνα 10: Get Storage Unit By Id

```
{
  "id": 3,
  "name": "Living Room",
  "enabled": true,
  "containers": [
    [
      {
        "name": "L1P1",
        "unit": "KG",
        "tareWeight": 0.5,
        "thresholdMin": 5,
        "thresholdMax": 10
      },
      {
        "name": "L1P2",
        "unit": "KG",
        "tareWeight": 0.75,
        "thresholdMin": 2.5,
        "thresholdMax": 8
      }
    ],
    [
      {
        "name": "L2P1",
        "unit": "KG",
        "tareWeight": 0.25,
        "thresholdMin": 10,
        "thresholdMax": 30
      },
      {
        "name": "L2P2",
        "unit": "KG",
        "tareWeight": 0.5,
        "thresholdMin": 1,
        "thresholdMax": 2
      }
    ]
  ]
}
```


Εικόνα 11: Get Storage Unit By Id (Example Response)

4.1.2.6 PUT update storage unit name

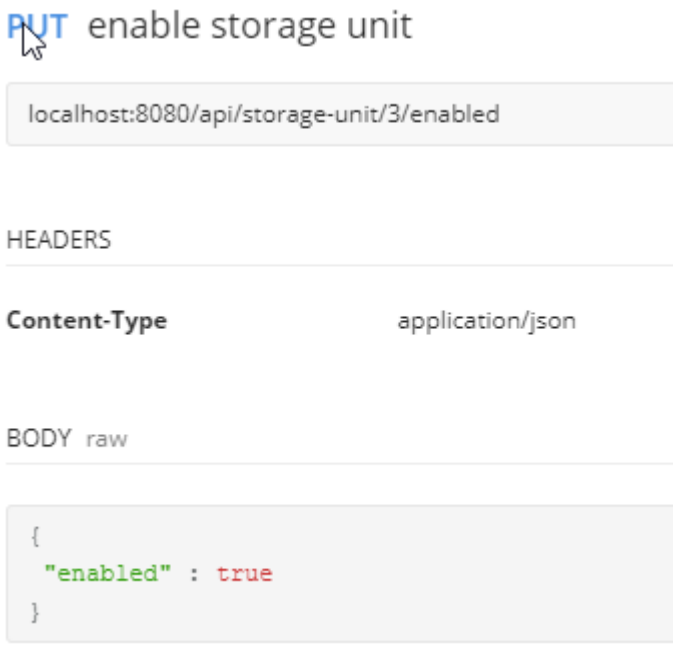
Με τη μέθοδο **αυτή** ο διαχειριστής ή ο χρήστης πραγματοποιεί μια REST κλήση για να παραμετροποιήσει το όνομα ενός storage unit.



Εικόνα 12: Update Storage Unit Name

4.1.2.7 PUT enable storage unit

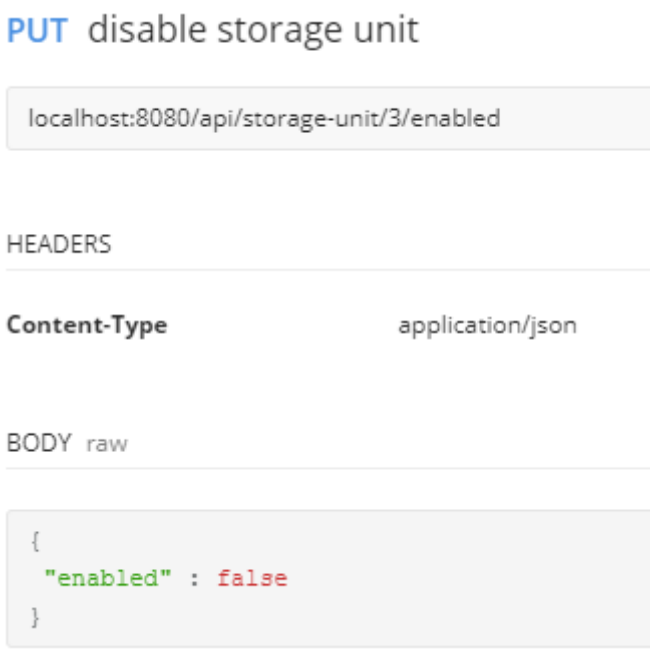
Η μέθοδος Enable storage unit επιτρέπει στο διαχειριστή να ενεργοποιήσει ένα storage unit. Το καθιστά δηλαδή enabled και επομένως θα μπορεί πλέον να το βλέπει ο χρήστης μέσω της εφαρμογής και να ενημερώνεται για αυτό ή να το παραμετροποιεί.



Εικόνα 13: Enable Storage Unit

4.1.2.8 PUT disable storage unit

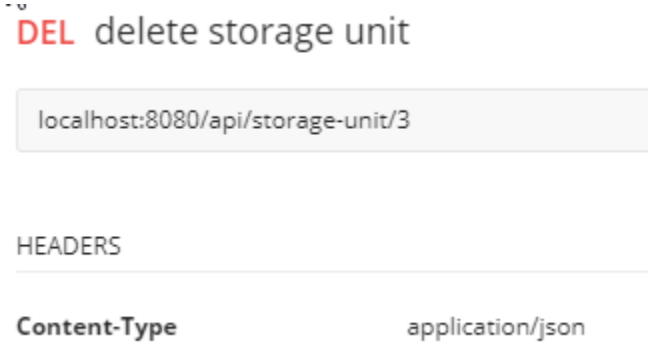
Παρομοίως η μέθοδος Disable storage unit επιτρέπει στο διαχειριστή να απενεργοποιήσει ένα storage unit και το καθιστά επομένως disabled, δηλαδή ο χρήστης δε θα μπορεί να το δει ή να ενημερωθεί για το storage unit αυτό μέσω της εφαρμογής.



Εικόνα 14: Disable Storage Unit

4.1.2.9 DELETE delete storage unit

Με τη μέθοδο Delete storage unit ο διαχειριστής πραγματοποιεί μία REST κλήση με σκοπό τη διαγραφή ενός storage unit από το σύστημα. Η πράξη αυτή επιφέρει και φυσική διαγραφή της δομής storage unit από την βάση δεδομένων.

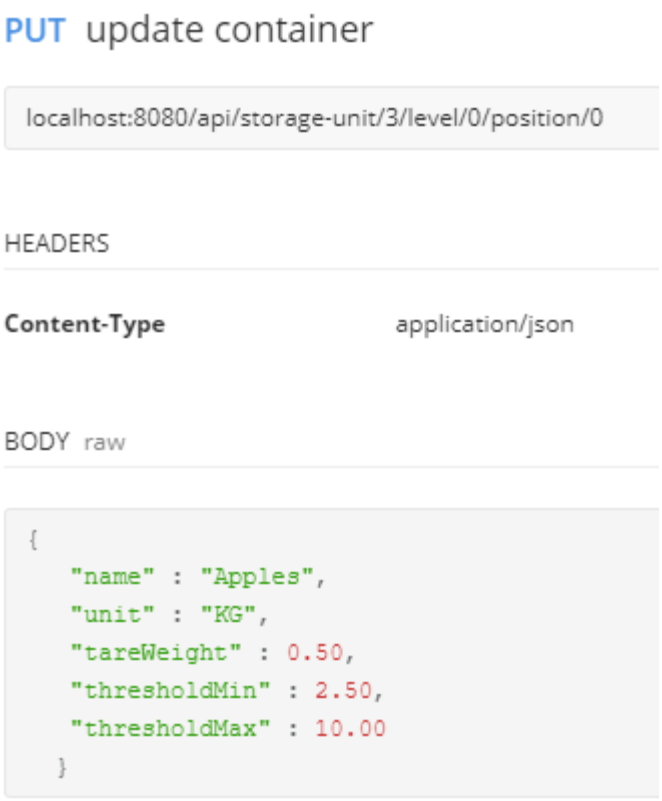


Εικόνα 15: Delete Storage Unit

4.1.2.10 PUT update container

Τέλος, η μέθοδος αυτή έχει δημιουργηθεί για να πραγματοποιεί την αλλαγή /

παραμετροποίηση του εσωτερικού ενός container. Ο χρήστης έχει τη δυνατότητα μέσω της εφαρμογής του κινητού να αλλάξει το όνομα (name) του container, τη μονάδα μέτρησης (unit) που χρησιμοποιούμε για το container αυτό, το απόβαρο (tareWeight) του container και το κατώτατο (thresholdMin) και ανώτατο (thresholdMax) όριο για τις μετρήσεις σε αυτό το container. Εάν ο χρήστης πραγματοποιήσει κάποια αλλαγή στο όνομα του container, τότε δημιουργείται ένα name event στην βάση δεδομένων, όπως θα αναφερθεί όταν περιγράψουμε τη λειτουργία και σημασία των events, στην περιγραφή της βάσης δεδομένων.



Εικόνα 16: Update Container

4.1.3 Περιγραφή της στρατηγικής παρακολούθησης ενός storage unit στο σύστημα

Για την παρακολούθηση (tracking) ενός storage unit έχει υλοποιηθεί μία στρατηγική (subscription strategy) σύμφωνα με την οποία υπάρχουν τρεις διαφορετικοί τρόποι δημιουργίας subscribers για ένα storage unit, που αφορούν στο είδος και την ποσότητα των topics στα οποία ακούνε).

1. Μπορεί να δημιουργηθεί ένας subscriber για ολόκληρο το storage unit. Ο subscriber αυτός ακούει στο topic storage-unit_{storageUnitId}/#, που στην ουσία πρόκειται για ένα σύνολο όλων των topics όλων των containers που υπάρχουν στο storage unit που θέλουμε να παρακολουθήσουμε.
2. Μπορεί να δημιουργηθεί ένας subscriber για κάθε level του storage unit που θέλουμε να παρακολουθήσουμε και επομένως μπορεί να ακούσει σε όλα τα topics του επιπέδου αυτού, για όσα containers και αν περιέχει αυτό το level. Το topic σε αυτήν

την περίπτωση είναι το storage-unit_{storageUnitId}/level_{levelId}/#.

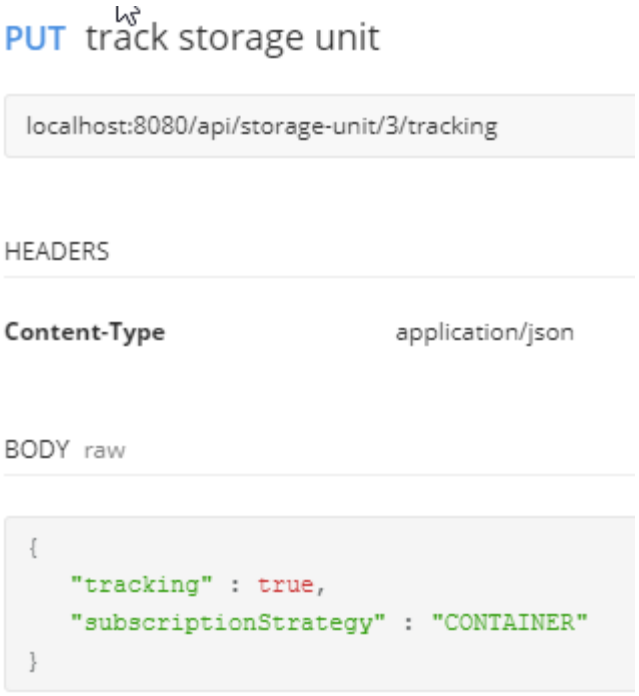
3. Μπορεί να δημιουργηθεί ένας subscriber για κάθε ξεχωριστό container, δηλαδή ο subscriber αυτός μπορεί να ακούσει μόνο σε ένα topic, στο topic του συγκεκριμένου container, το storage-unit_{storageUnitId}/level_{levelId}/container_{containerID}/#.

4.1.4 Περιγραφή των REST κλήσεων που απαιτούνται για την παρακολούθηση (tracking) ενός storage unit στο σύστημα

Στη συνέχεια θα αναλύσουμε τις REST κλήσεις που γίνονται ανάμεσα στο server και την εφαρμογή του διαχειριστή για τη διαχείριση του tracking ενός storage unit.

4.1.4.1 PUT track storage unit

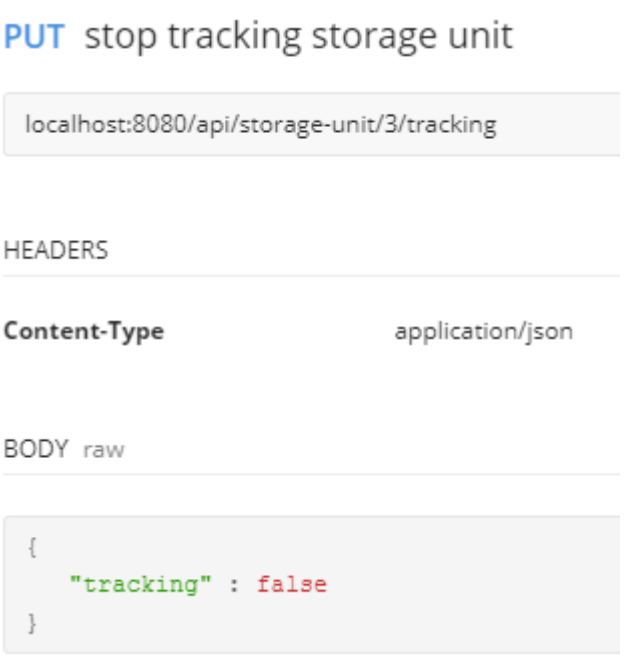
Η μέθοδος αυτή επιτρέπει στο διαχειριστή να ξεκινήσει την παρακολούθηση (tracking) ενός storage unit σύμφωνα με την subscription strategy που έχει επιλέξει (έναν απ' τους τρεις τρόπους που αναλύθηκαν προηγουμένως). Δηλαδή, ανάλογα με τη στρατηγική που επιλέγει, δημιουργούνται οι subscribers και στη συνέχεις ξεκινούν να ακούνε στα αντίστοιχα topics. Να σημειωθεί ότι εάν ένα storage unit παρακολουθείται ήδη, δηλαδή οι subscribers για το συγκεκριμένο storage unit υπάρχουν ήδη με την επιλεγμένη στρατηγική, τότε απλώς ξεκινούνε να ακούνε ξανά (με την προϋπόθεση ότι για κάποιο λόγο είχανε σταματήσει πιο πριν, για παράδειγμα μετά από εντολή του διαχειριστή).



Εικόνα 17: Track Storage Unit

4.1.4.2 PUT stop tracking storage unit

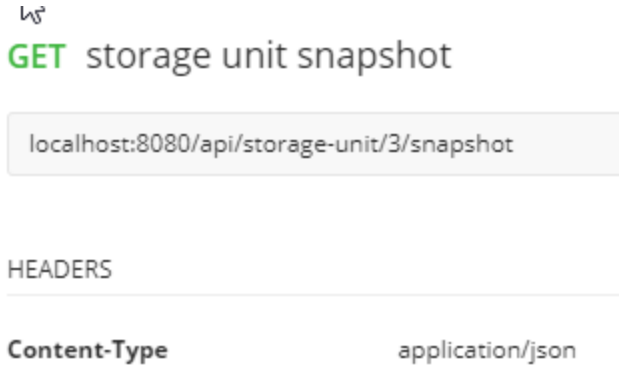
Η μέθοδος αυτή επιτρέπει στο διαχειριστή να δώσει εντολή στους subscribers να σταματήσουν το tracking για το συγκεκριμένο storage unit.



Εικόνα 18: Stop Tracking Storage Unit

4.1.4.3 GET storage unit snapshot

Αυτή η μέθοδος καλείται όταν ο χρήστης στην εφαρμογή του θέλει να ελέγξει την κατάσταση στην οποία βρίσκεται ένα storage unit μια συγκεκριμένη στιγμή. Μπορεί επομένως να δει τα χαρακτηριστικά όλων των containers μέσα σε ένα storage unit όπως ακριβώς είναι αυτά τη στιγμή που ζητάει την εικόνα του storage unit.



Εικόνα 19: Storage Unit Snapshot

```
{
  "id": 3,
  "name": "Living Room #2",
  "containers": [
    [
      {
        "name": "L1P1",
        "grossValue": 10.36,
        "unit": "KG",
        "tareWeight": 0.5,
        "thresholdMin": 5,
        "thresholdMax": 10
      },
      {
        "name": "L1P2",
        "grossValue": 0,
        "unit": "KG",
        "tareWeight": 0.75,
        "thresholdMin": 2.5,
        "thresholdMax": 8
      }
    ],
    [
      {
        "name": "L2P1",
        "grossValue": 5.5,
        "unit": "KG",
        "tareWeight": 0.25,
        "thresholdMin": 10,
        "thresholdMax": 30
      },
      {
        "name": "L2P2",
        "grossValue": 0,
        "unit": "KG",
        "tareWeight": 0.5,
        "thresholdMin": 1,
        "thresholdMax": 2
      }
    ]
  ]
}
```


Εικόνα 20: Storage Unit Snapshot (Example Response)

4.1.5 Περιγραφή του κύκλου ζωής (lifecycle) των subscribers και σχετικές λειτουργίες

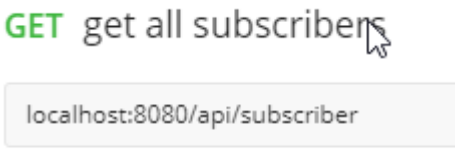
Η υλοποίηση του subscriber βρίσκεται στην κλάση: di.smartliving.server.web.mqtt.client.MqttSubscriber. Χρησιμοποιεί το mqtt client του eclipse paho: org.eclipse.paho.client.mqttv3.MqttClient με σκοπό να γίνει η πραγματική σύνδεση με τον broker. Η τοποθεσία του broker με τον οποίο κάνει τη σύνδεση ώστε να επικοινωνεί με τους sensors είναι γνωστή στο server από το properties file, γνωστό ως application.yml.

Η υλοποίηση των μεθόδων που αφορούν την διαχείριση των subscribers βρίσκεται στην κλάση: di.smartliving.server.service.SubscriberService, η οποία περιέχει μεθόδους όπως createSubscribers (για τη δημιουργία subscriber), getSubscribers (για να παίρνει όλους τους subscribers που αντιστοιχούν σε ένα storage unit), getSubscriberMap (για να παίρνει όλο το concurrent hashmap με όλους τους subscribers όλων των storage unit), deleteSubscribers (για τη διαγραφή subscriber), startSubscribers (για την εκκίνηση ενός subscriber ώστε να ακούει σε κάποιο topic) και stopSubscribers (για τη λήξη παρακολούθησης ενός topic από έναν subscriber).

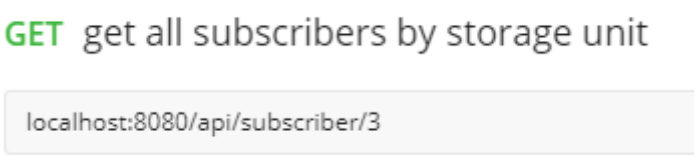
- ***Δημιουργία ενός subscriber:*** Η δημιουργία ενός subscriber γίνεται με την βοήθεια της κλάσης MqttSubscriberFactory. Σε αυτό περνάμε σαν ορίσματα το url του broker από το application.yml, το topic στο οποίο θα ακούει ο subscriber με τη δομή που αναφέρθηκε παραπάνω και την κλάση MqttSubscriberCallback, στην οποία ορίζουμε τι συμβαίνει κάθε φορά που λαμβάνουμε ένα νέο μήνυμα από το/τους sensor/sensors στο/α συγκεκριμένο/α topic/topics που έχουμε δηλώσει ότι ακούει ο subscriber. Κάθε φορά λοιπόν που λαμβάνουμε ένα νέο μήνυμα στο topic που τον ενδιαφέρει, δημιουργούμε ένα event, μία εγγραφή δηλαδή στον πίνακα events στην βάση δεδομένων. Υπενθυμίζουμε ότι η υλοποίηση των events αφορά δύο είδη event, τα value events και name events. Στην περίπτωση λήψης ενός νέου μηνύματος από το sensor και εφόσον η νέα τιμή είναι διαφορετική από την τελευταία εγγραφή που υπάρχει αποθηκευμένη στην βάση δεδομένων, μας αφορά μόνο η προσθήκη ενός value event. Η κλάση MqttSubscriberCallback εκτός από το χειρισμό ενός εισερχόμενου μηνύματος, διαχειρίζεται και την περίπτωση όπου η σύνδεση με τον broker έχει χαθεί ή την περίπτωση που η λήψη του νέου μηνύματος έχει γίνει σωστά.
- ***Αποθήκευση ενός subscriber:*** Οι MQTT subscribers που δημιουργούμε αποθηκεύονται στη μνήμη του συστήματος σε έναν concurrent hashmap. Στην αρχικοποίηση του συστήματος δημιουργείται αυτός ο πίνακας και αρχικά είναι κενός, ενώ προστίθενται σε αυτόν καινούρια subscribers όταν δημιουργούνται από το διαχειριστή. Ο πίνακας αυτός, ο προαναφερόμενος concurrent hashmap, περιέχει ζευγάρια από <κλειδί, τιμή> (<key, value>). Στην περίπτωσή μας, το key του concurrent hashmap είναι το διακριτό id του storage unit και το value είναι ένα set από subscribers. Με τον τρόπο αυτό μπορούμε να γνωρίζουμε ή να διαχειριστούμε τους subscribers ενός storage unit, αφού έτσι γνωρίζουμε πως το storage unit με id έστω 1 περιέχει το set από subscribers που δηλώνεται ως value στον concurrent hashmap. Ο πίνακας αυτός είναι διαθέσιμος στο σύστημα ως ένα Spring Bean, συγκεκριμένα περιγράφεται στο: di.smartliving.server.config.SmartLivingServerConfig.
- ***Πότε δημιουργείται ένας subscriber:*** Ένας subscriber δημιουργείται όταν καλεί ο διαχειριστής από το REST API τη μέθοδο PUT track storage unit. Όπως βλέπουμε αναλυτικότερα και στην περιγραφή της μεθόδου αυτής, ψάχνουμε το συγκεκριμένο storage unit στην βάση δεδομένων και ανάλογα με τη στρατηγική παρακολούθησης

(tracking strategy) που έχουμε επιλέξει, δημιουργείται ο subscriber και στη συνέχεια αποθηκεύεται στο concurrent hashmap που αναφέραμε με κλειδί το διακριτό id του storage unit όπως βρέθηκε από την εγγραφή στην βάση δεδομένων. Η δημιουργία ενός subscriber υλοποιείται στη μέθοδο createSubscribers της κλάσης SubscriberService.

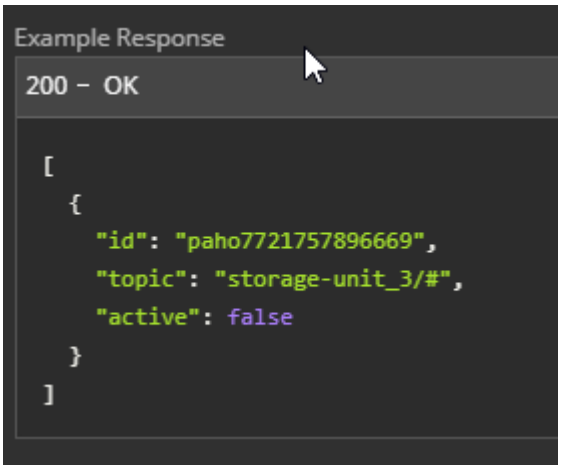
- ***Πότε διαγράφεται ένας subscriber:*** Ένας subscriber διαγράφεται όταν καλεί ο διαχειριστής από το REST API τη μέθοδο DELETE delete storage unit. Όπως βλέπουμε αναλυτικότερα και στην περιγραφή της μεθόδου αυτής, σταματάμε τους subscribers, εάν αυτοί τρέχουν ακόμη και ακούνε για νέα μηνύματα και τους διαγράφουμε από το concurrent hashmap. Η διαγραφή ενός storage unit και κατ' επέκταση ενός subscriber υλοποιείται στη μέθοδο deleteSubscribers της κλάσης SubscriberService.
- ***Πως κάνουμε monitor έναν subscriber όσο τρέχει ο server ή πως αποκτούμε πρόσβαση στην πληροφορία που περιέχει το topic – Περιγραφή των REST κλήσεων που έχουν υλοποιηθεί για την παρακολούθηση των subscribers:*** Το monitoring ενός subscriber γίνεται με τις REST κλήσεις ***GET get all subscribers*** και ***GET get all subscribers by storage unit***, οι οποίες μας επιστρέφουν όλους τους subscribers από όλα τα ενεργά storage units ή όλους τους subscribers από ένα συγκεκριμένο storage unit. Και τα δύο αυτά operations, διαβάζουν το concurrent hashmap και φέρνουν την απαραίτητη πληροφορία για κάθε storage unit ή για το συγκεκριμένο storage unit. Η πληροφορία αυτή αφορά το id, τη διακριτή τιμή δηλαδή του subscriber, το topic στο οποίο ακούει και μια boolean μεταβλητή active, η οποία μας ενημερώνει εάν ο subscriber είναι active, εάν ακούει στο συγκεκριμένο topic.



Εικόνα 21: Get All Subscribers



Εικόνα 22: Get All Subscribers By Storage Unit



Εικόνα 23: Get All Subscribers By Storage Unit (Example Response)

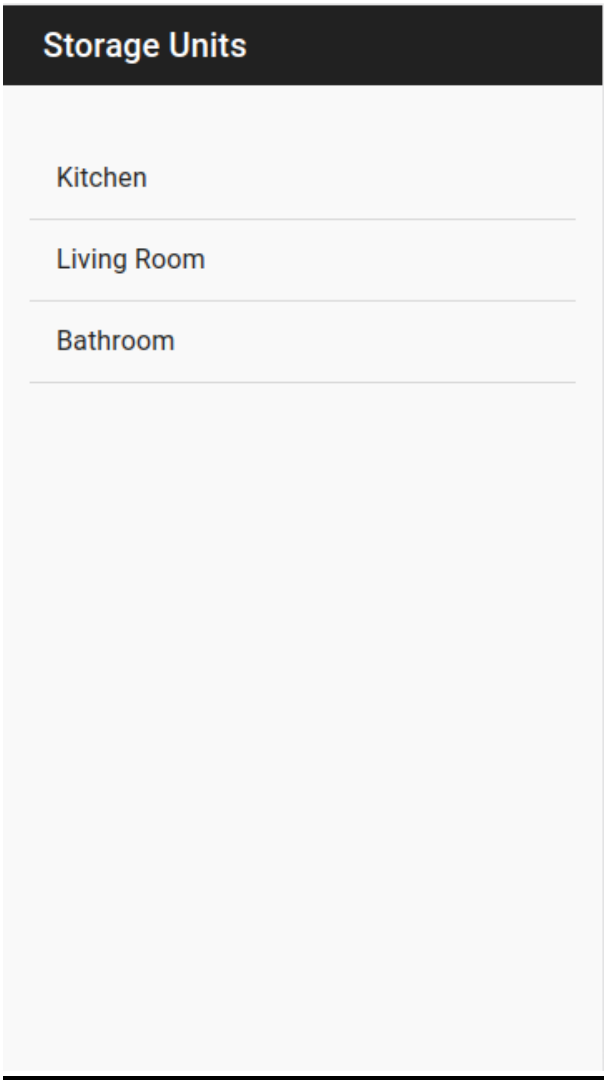
5. MOBILE APPLICATION

5.1 Περιγραφή του mobile application

Για το γραφικό περιβάλλον του χρήστη έχει υλοποιηθεί ένα mobile application ώστε να μπορεί ο χρήστης να κάνει monitor τα προϊόντα που τον ενδιαφέρουν, καθώς και να τα παραμετροποιεί ή να βλέπει στατιστικά δεδομένα. Το mobile application έχει γραφτεί σε React Native. Ενδεικτικά παραθέτουμε τις βασικές οθόνης πλοήγησης της εφαρμογής κινητού.

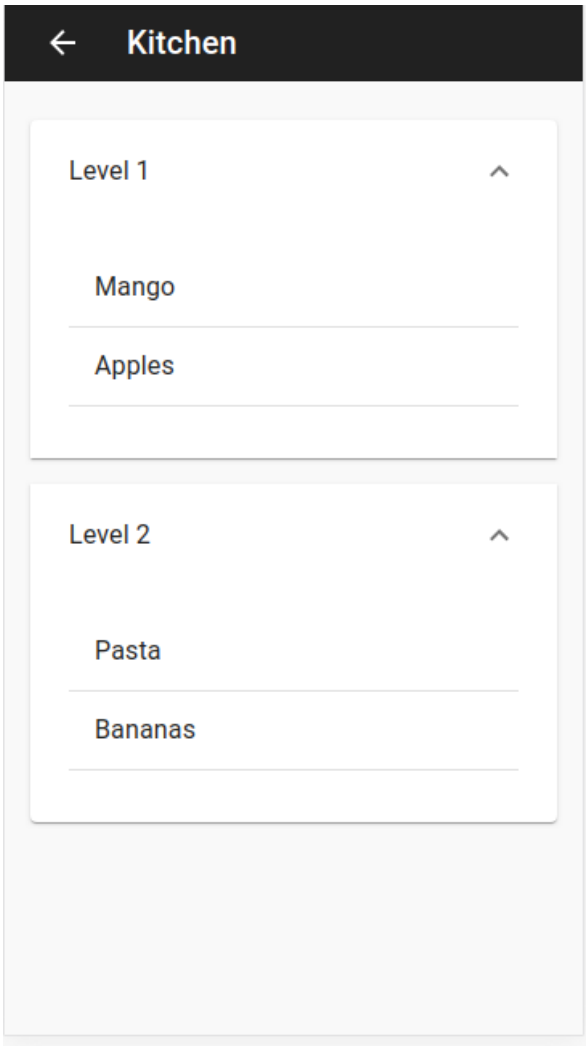
Η κάτω οθόνη δείχνει στο χρήστη τα διαθέσιμα storage units (kitchen, living room και bathroom) τα οποία είναι enabled και τα οποία μπορεί να κάνει monitor ή να παραμετροποιήσει. Μπορεί να επιλέξει ένα από αυτά και να πλοηγηθεί στα περιεχόμενά του.

Στη συγκεκριμένη οθόνη βλέπουμε το response της REST κλήσης GET enabled storage units, δηλαδή στην οθόνη αυτή εμφανίζονται μόνο τα storage units που είναι enabled και υπό παρακολούθηση.



Εικόνα 24: Storage Units

Έστω ότι επιλέγει το πρώτο storage unit, το kitchen. Η επόμενη οθόνη που εμφανίζεται ακριβώς από κάτω, στην οποία μπορούμε να δούμε από πόσα levels αποτελείται (Level 1, Level 2), καθώς και από πόσα containers (Mango, Apples για το Level 1 και Pasta, Bananas για το Level 2). Τα παράθυρα των levels είναι togglable (δηλαδή μπορούν να εμφανίσουν ή να αποκρύψουν τα περιεχόμενά τους – τα containers) και επιλέγοντας σε κάποιο container ο χρήστης μπορεί να πλοηγηθεί στα χαρακτηριστικά του container αυτού.

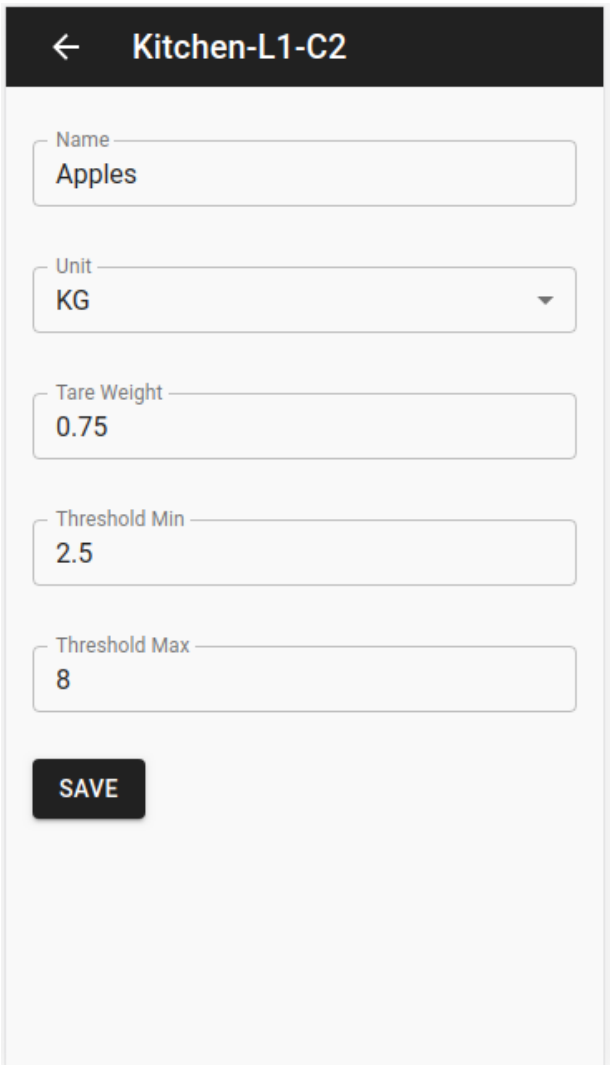


Εικόνα 25: Levels

Πατώντας λοιπόν σε κάποιο container ο χρήστης μεταφέρεται στην οθόνη που βλέπουμε στη συνέχεια. Παρατηρούμε από τον τίτλο ότι επέλεξε το δεύτερο container (C2) το οποίο βρίσκεται στη θέση:

Kitchen-L1-C2. Ο χρήστης μπορεί να αλλάξει τις τιμές που βλέπει στην οθόνη αυτή και να τις αποθηκεύσει, πατώντας το SAVE. Όταν αποθηκεύσει νέες τιμές γίνεται update στο container και στην βάση δεδομένων, για τα στοιχεία που έχει αλλάξει.

Στην οθόνη αυτή βλέπουμε το response που παίρνουμε όταν καλούμε τη μέθοδο GET storage units snapshot, δηλαδή μια λεπτομερή περιγραφή των χαρακτηριστικών του συγκεκριμένου container που έχουμε επιλέξει.



Εικόνα 26: Containers

6. ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

6.1 Περιγραφή της βάσης δεδομένων

Φανταζόμαστε λοιπόν ότι σε κάποια ραφιέρα της κουζίνας μας εγκαθιστούμε sensors για να παρακολουθούμε την ποσότητα των αντικειμένων τα οποία αποθηκεύουμε στα διάφορα μέρη (containers). Χρειαζόμαστε επομένως μια βάση δεδομένων για να μπορούμε να αποθηκεύουμε διάφορες χρήσιμες πληροφορίες, όπως:

1) Το ***storage unit*** που παρακολουθούμε. Στην dB πρόκειται για έναν πίνακα (storage_units) που περιέχει:
   - **id:** Έναν αριθμό που αποτελεί το primary key, έναν διακριτό αριθμό για το storage unit. Κάθε storage unit είναι υποχρεωτικό να έχει ένα id, το οποίο είναι διαφορετικό για κάθε storage unit και αυξάνεται κατά ένα (μέσω μιας ακολουθίας – sequence) κάθε φορά που προστίθεται ένα νέο storage unit στην βάση δεδομένων.
   - **enabled:** Πρόκειται για μια τύπου boolean μεταβλητή, η οποία μας ενημερώνει εάν το συγκεκριμένο storage unit θέλουμε να το παρακολουθούμε και επομένως να εμφανίζεται στην εφαρμογή του κινητού του χρήστη με τα ενεργά storage units.
   - **name:** Το όνομα του storage unit όπως το έχει ονομάσει ο χρήστης. Είναι προφανές ότι αυτή η πληροφορία δε χρειάζεται να είναι μοναδική, εφόσον ο χρήστης μπορεί να διακρίνει τα διάφορα storage units με το ίδιο όνομα.
2) Τις διάφορες θέσεις αποθήκευσης (***containers***). Αναπαριστούνται στην βάση δεδομένων από έναν πίνακα (containers) που περιέχει για κάθε container τα εξής:
   - **level:** Η πληροφορία που μας λέει σε ποιο επίπεδο βρίσκεται αυτή η θέση αποθήκευσης μέσα στο storage unit μας. Για κάθε container είναι υποχρεωτικό να γνωρίζουμε σε ποιο level βρίσκεται. Κάθε level είναι ένας αριθμός (int).
   - **position:** Η πληροφορία που μας λέει ουσιαστικά τις συντεταγμένες του container μέσα στο storage unit. Για κάθε container είναι επίσης υποχρεωτικό να γνωρίζουμε σε ποιο position βρίσκεται. Κάθε position είναι ομοίως ένας αριθμός (int). Για να γίνουμε κατανοητοί, έστω ότι χωρίζουμε το storage unit (έστω storage unit #1) σε 2 levels, στο level #1 (το πάνω ράφι) και level #2 (το κάτω ράφι). Το πρώτο level περιέχει δύο θέσεις, εκ των οποίων η πρώτη είναι το position #1 (αριστερά) και η δεύτερη το position #2 (δεξιά). Αντίστοιχα για το δεύτερο level, υποθέτουμε ότι χωρίζεται σε τρεις θέσεις, τα containers που βρίσκονται στα position #1, position #2 και position #3. Με αυτές τις πληροφορίες μπορούμε πλέον εύκολα να περιγράψουμε τη θέση του container που παρακολουθούμε λέγοντας απλά ότι το εν λόγω container βρίσκεται στη θέση 1.2.3, που σημαίνει ότι βρίσκεται στο storage unit #1 (ραφιέρα #1), στο level #2 (κάτω ράφι) και στο container #3 (δεξιά θέση). Αυτή η παραδοχή που κάνουμε για να περιγράψουμε την ακριβή θέση ενός container αποτελεί και το primary key του συγκεκριμένου πίνακα, που σημαίνει ότι η τριπλέτα αυτή είναι διαφορετική και μοναδική για κάθε container.
   - **creation_date:** Η ημερομηνία δημιουργίας του container, ή αναλυτικότερα, η ημερομηνία που μπαίνει η εγγραφή για το container αυτό στην βάση, ώστε να

αρχίσουμε να το παρακολουθούμε. Το πεδίο αυτό είναι υποχρεωτικό.

- **name:** Το όνομα του container όπως εμφανίζεται στην android εφαρμογή κινητού που χρησιμοποιεί ο χρήστης για ενημέρωση και παραμετροποίηση. Είναι προφανές ότι δε χρειάζεται να είναι μοναδικό, εφόσον ο χρήστης μπορεί να ξεχωρίσει τα διάφορα containers με το ίδιο όνομα. Το πεδίο αυτό δεν είναι υποχρεωτικό και μπορεί ο χρήστης να το αλλάξει μέσα από την android εφαρμογή.
- **tare_weight:** Στη συγκεκριμένη μεταβλητή κρατάμε το απόβαρο του container, μια πληροφορία που είναι χρήσιμη κατά την αρχική παραμετροποίηση της φυσικής μας εγκατάστασης, ώστε να μπορούμε να το αφαιρούμε από τους υπολογισμούς μας όταν κάνουμε μετρήσεις για το συγκεκριμένο container χρησιμοποιώντας τους sensors. Το πεδίο αυτό δεν είναι υποχρεωτικό, καθώς κατά την αρχικοποίηση του storage unit παίρνει μια default τιμή 0, μπορεί όμως ο χρήστη να αλλάξει την τιμή tare weight του container μέσα από την εφαρμογή, οποιαδήποτε στιγμή κατά τη διάρκεια λειτουργίας και παρακολούθησης του storage unit.
- **threshold_max:** Η συγκεκριμένη πληροφορία μας ενημερώνει για το πάνω όριο των μετρήσεων που γίνονται με τη βοήθεια των sensors, ώστε να γνωρίζουμε για παράδειγμα πόσο χωράει αυτό το container ή ώστε να δεχόμαστε ενημερώσεις από την εφαρμογή ότι το container αυτό είναι γεμάτο. Το πεδίο αυτό είναι παραμετροποιήσιμο από τον χρήστη μέσω της εφαρμογής.
- **threshold_min:** Αντίστοιχα, η πληροφορία αυτή μας ενημερώνει για το κατώτερο όριο των μετρήσεων που μπορούμε να δεχτούμε από τους sensors. Με αυτόν τον τρόπο μπορούμε για παράδειγμα να ενημερωνόμαστε εάν το προϊόν που παρακολουθούμε στο container αυτό τελειώνει και θα πρέπει ενδεχομένως να προβούμε σε αντικατάστασή του. Ομοίως με το ανώτατο όριο, το πεδίο αυτό μπορεί να αλλάξει από το χρήστη οποιαδήποτε στιγμή χρησιμοποιώντας την android εφαρμογή.
- **unit:** Πρόκειται για τη μονάδα μέτρησης που χρησιμοποιούμε κατά την παρακολούθηση του εκάστοτε container, για παράδειγμα – μιλώντας στην περίπτωσή μας αποκλειστικά και μόνο για βάρος – η μονάδα μέτρησης μπορεί να είναι κιλά kg, γραμμάρια g, λίτρα lt, και άλλα.
- **storage_unit_id:** Το id, ή διακριτός αριθμός του storage unit στο οποίο βρίσκεται το container. Το πεδίο αυτό αποτελεί και foreign key προς τον πίνακα storage_units και αντιστοιχίζεται με το πεδίο id του πίνακα αυτού.

3) Τα ***events***, δηλαδή τα διάφορα γεγονότα που έχουμε παραδεχτεί ότι συμβαίνουν και σχετίζονται με το container που έχουμε δημιουργήσει. Υπάρχουν δύο είδη event, τα **name events** και τα **value events**. Τα δύο αυτά events αντιπροσωπεύουν ένα ιστορικό των αλλαγών που έχουν να κάνουν με το όνομα και την τιμή που περιέχει ένα container. Δηλαδή, ένα name event συμβαίνει κάθε φορά που αλλάζει το όνομα ενός container και ένα value event συμβαίνει όταν αλλάζει η τιμή που περιέχει το container που παρακολουθούμε. Για κάθε συμβάν (event) κρατάμε στην βάση δεδομένων την ημερομηνία που αυτό έλαβε χώρα.
Για να κατανοήσουμε καλύτερα τη χρήση των events, ας απαντήσουμε σε δύο βασικές ερωτήσεις. Αρχικά, πως και πότε δημιουργείται ένα event; Κάθε φορά που ένας subscriber λαμβάνει ένα μήνυμα από τον broker που αφορά την τιμή που περιέχεται σε ένα container, η καινούρια αυτή τιμή συγκρίνεται με την προηγούμενη τιμή που

υπάρχει αποθηκευμένη στην βάση δεδομένων για το container αυτό, η προηγούμενη δηλαδή τιμή που στάλθηκε στο σύστημα. Εάν η νέα τιμή είναι διαφορετική, τότε δημιουργείται ένα νέο συμβάν, για την ακρίβεια ένα value event και αποθηκεύεται στην βάση δεδομένων.
Ακόμη, πότε και για ποιο λόγο χρησιμοποιούμε ένα event; Ο βασικός λόγος για τον οποίο έχει υλοποιηθεί η αποθήκευση των events που περιγράψαμε, είναι για να μπορεί ο χρήστης να ενημερώνεται οποιαδήποτε στιγμή για την κατάσταση κάθε container μέσω της android εφαρμογής. Να μπορεί δηλαδή να βλέπει οποιαδήποτε στιγμή ένα ενημερωμένο περιεχόμενο ενός container ή γενικότερα ενός storage unit σύμφωνα με τις τελευταίες μετρήσεις που έχουμε λάβει από τους sensors. Το ενημερωμένο αυτό περιεχόμενο το αποκαλούμε snapshot και αφορά τις τελευταίες διαφορετικές μετρήσεις που έχουμε λάβει. Ένας άλλος λόγος που αποφασίσαμε να αποθηκεύουμε τέτοιου είδους events είναι για στατιστικούς λόγους, για να μπορεί ο χρήστης να βλέπει ιστορικό για τα ονόματα ή τις τιμές που περιέχουν τα containers ή διαγράμματα αλλαγών των τιμών ή άλλα στατιστικά μέσω της εφαρμογής.
Σύμφωνα με την παραπάνω ανάλυση, τα ***events*** αποτελούν στην βάση δεδομένων έναν πίνακα (events) με τις παρακάτω κολώνες:

- **id:** Πρόκειται για τον διακριτό αριθμό ενός event, ο οποίος είναι υποχρεωτικός και αυξάνεται αυτόματα κατά ένα (με τη χρήση μιας ακολουθίας – sequence) κάθε φορά που λαμβάνει χώρα ένα νέο event με τον τρόπο που περιγράφηκε προηγουμένως. Το πεδίο αυτό αποτελεί επίσης το πρωτεύον κλειδί (primary key) του πίνακα, που σημαίνει ότι κάθε id είναι διαφοτερικό.
- **container_level:** Το πεδίο αυτό αντιπροσωπεύει το level του container για το οποίο αποθηκεύουμε το συμβάν.
- **container_position:** Αντίστοιχα, πρόκειται για το position του container για το οποίο δημιουργήθηκε το συμβάν.
- **storage_unit_id:** Ο διακριτός αριθμός ενός storage unit. Με τον τρόπο αυτό έχουμε για άλλη μια φορά την απαραίτητη πληροφορία για την ακριβή θέση του container για το οποίο αποθηκεύουμε ένα νέο συμβάν, την προηγουμένως αναφερόμενη ως τριπλέτα.
- **created_date:** Στο πεδίο αυτό αποθηκεύουμε την ημερομηνία που λαμβάνει χώρα το νέο συμβάν. Το πεδίο αυτό είναι υποχρεωτικό.
- **name:** Αναφερόμαστε στο νέο όνομα και επομένως το πεδίο αυτό παίρνει την τιμή του νέου ονόματος εάν αυτό είναι διαφορετικό από το προηγούμενο που υπάρχει αποθηκευμένο στην βάση δεδομένων και εφόσον λαμβάνει χώρα name event.
- **type:** Το πεδίο αυτό αντιπροσωπεύει τον τύπο του συμβάντος που θέλουμε να αποθηκεύσουμε στην dB και όπως είπαμε ο τύπος αυτός μπορεί να είναι δύο ειδών: name και value.
- **value:** Αναφερόμαστε στη νέα τιμή και επομένως το πεδίο αυτό παίρνει τη νέα τιμή εάν αυτή είναι διαφορετική από την προηγούμενη που υπάρχει αποθηκευμένη στην βάση δεδομένων και εφόσον λαμβάνει χώρα value event.

7. ΣΤΑΤΙΣΤΙΚΑ

7.1.1 Περιγραφή της λειτουργίας των στατιστικών (statistics)

Κατά την υλοποίηση του συστήματος και όταν οι συνθήκες είναι ικανές αποθηκεύονται, υπολογίζονται ή αναζητούνται συγκεκριμένες τιμές στην βάση δεδομένων, με σκοπό τη συλλογή πληροφοριών για την αποθήκευση ιστορικού και προβολής στατιστικών δεδομένων στην εφαρμογή για την γραφική αναπαράσταση τιμών και την ενημέρωση του χρήστη. Για την ακρίβεια, έχουν υλοποιηθεί τριών ειδών στατιστικά δεδομένα και ιστορικού που αφορούν συγκεκριμένες τιμές που παίρνουν τα προϊόντα που παρακολουθούμε μέσα στα containers.

Τα στατιστικά αυτά δεδομένα αφορούν τα containers και τα πρώτα που επιδεικνύονται αφορούν την ιστορικότητα των αλλαγών των τιμών μέσα σε ένα container. Ο χρήστης μπορεί δηλαδή να δει κάθε πότε ή πόσο γρήγορα αλλάζει η τιμή ενός προϊόντος που παρακολουθεί και επομένως να γνωρίζει πόσο γρήγορα ή αργά εξαντλείται. Τα αποτελέσματα που επιδεικνύονται είναι paged μιας και μπορεί να είναι πάρα πολλά και ενδεικτικά φαίνονται τα πρώτα 10 σε κάθε page.

Ο δεύτερος τρόπος που χρησιμοποιούμε την εξαγωγή δεδομένων για τη συγκέντρωση στατιστικών αφορά τις τιμές που ένα προϊόν έφτασε στο μηδέν. Με τον τρόπο αυτό ο χρήστης μπορεί να ελέγξει πόσες φορές έχει ξεμείνει από ένα προϊόν χωρίς να το έχει αντικαταστήσει, ακόμα και αν έχει λάβει ενημέρωση από την εφαρμογή για την επερχόμενη εξάντληση του προϊόντος, θεωρώντας ότι το κατώτατο όριο (min threshold) είναι μία τιμή διαφορετική και μεγαλύτερη του μηδενός.

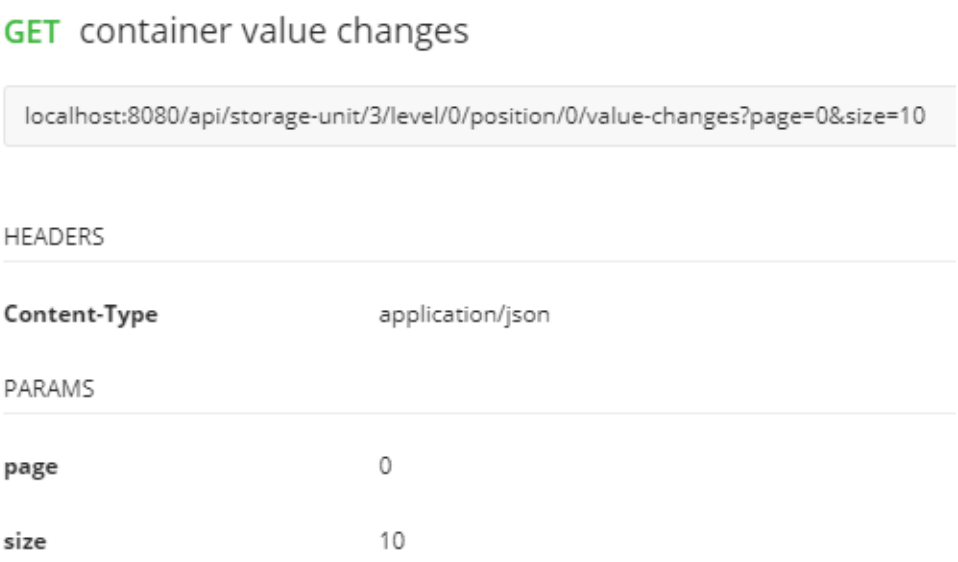
Τέλος, ο τρίτος τρόπος που υλοποιήθηκε για την επίδειξη στατιστικών δεδομένων αφορά το μέσο όρο των τιμών από την αρχή της παρακολούθησης έως και τη στιγμή που ζητείται να υπολογιστεί το συγκεκριμένο στατιστικό δεδομένο. Ο χρήστης μπορεί να ελέγξει το μέσο όρο των τιμών ενός container και μπορεί έτσι να γνωρίζει τη μέση κατανάλωση του προϊόντος αυτού. Τα αποτελέσματα βοηθούν το χρήστη να αποφασίσει εάν θέλει να συνεχίσει την παρακολούθηση του προϊόντος αυτού ή όχι - εάν όχι, μπορεί να αποφασίσει να αλλάξει προϊόν παρακολούθησης εφόσον κάποιο προϊόν βλέπει ότι δεν καταναλώνεται, ή εάν ναι, μπορεί να αποφασίσει ότι χρειάζεται ρύθμιση της κατανάλωσης του συγκεκριμένου προϊόντος στην περίπτωση που το προϊόν καταναλώνεται με πολύ γρήγορους ρυθμούς.

7.1.2 Περιγραφή των REST κλήσεων που έχουν υλοποιηθεί για τα statistics

Σύμφωνα με την αναλυτική περιγραφή των statistics που καλύφθηκε πιο πάνω, έχουν υλοποιηθεί τρεις REST κλήσεις:

7.1.2.1 GET container value changes

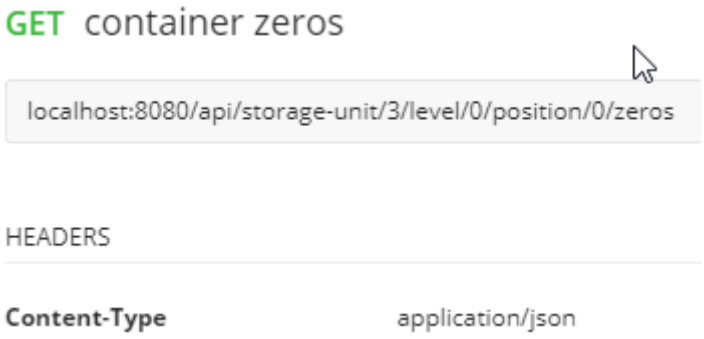
Η μέθοδος αυτή επιστρέφει το ιστορικό των αλλαγών στις τιμές ενός container για ένα προϊόν που παρακολουθείται. Τα αποτελέσματα, όπως φαίνεται και στο Example Response πιο κάτω, περιλαμβάνουν την τριάδα δεδομένων unit – value – timestamp, δηλαδή τη μονάδα μέτρησης που χρησιμοποιείται για το container αυτό, την τιμή value και τη χρονική στιγμή (timestamp) που λάβαμε από το sensor την τιμή αυτή. Η πρώτη τριάδα που βλέπουμε στο response αποτελεί και την πιο πρόσφατη αλλαγή στο value ενός container, ενώ η τελευταία την πρώτη αλλαγή στο value.



Εικόνα 27: Container Value Changes

7.1.2.2 GET container zeros

Η μέθοδος αυτή καλείται όταν ο χρήστης θέλει να λάβει τα στατιστικά δεδομένα που αφορούν την τιμή value ενός container όταν αυτή φτάνει στο μηδέν.



Εικόνα 28: Container Zeros

7.1.2.3 GET container average

Και τέλος, η μέθοδος αυτή αφορά τα statistics που λαμβάνουμε για την παρακολούθηση του μέσου όρου ενός value ενός container.

Εικόνα 29: Container Average

ΣΥΜΠΕΡΑΣΜΑΤΑ

Μετά την εκπόνηση της πτυχιακής αυτής εργασίας τα συμπεράσματα στα οποία έχουμε καταλήξει αφορούν τη χρήση της τάσης smart living στο σύγχρονο κόσμο, την αναγκαιότητά της και την ευρεία χρήση της στην καθημερινότητά μας. Ο σύγχρονος κόσμος κυριαρχείται από τη χρήση της τεχνολογίας δημιουργώντας προβλήματα και λύσεις που αφορούν το smart living, δηλαδή καινοτόμες ιδέες που προάγουν ένα καλύτερο, ευκολότερο και πιο οικολογικό τρόπο ζωής.

Παρατηρούμε για παράδειγμα τις χρήσεις (use cases) που μπορεί να έχει η ιδέα και υλοποίηση που περιγράφεται στην πτυχιακή αυτή εργασία, καθώς και τις συνέπειες που θα μπορούσε να επιφέρει στην καθημερινή ζωή εάν γινόταν στην πράξη.

Αναλυτικά, στο πλαίσιο μιας επιχείρησης θα μπορούσε να αντικαταστήσει μια δύσκολη, επίπονη και χρονοβόρα εργασία με ένα αυτοματοποιημένο σύστημα που ενημερώνει τον ενδιαφερόμενο για την κατάσταση των προϊόντων που παρακολουθούνται και την πιθανή εξάντλησή τους, καθώς και στατιστικά δεδομένα που μπορούμε να εξάγουμε από την παρακολούθηση.

Αντίστοιχα, στο περιβάλλον ενός σπιτιού, θα μπορούσαν να επωφεληθούν από το παρόν σύστημα διάφορα είδη χρηστών. Για παράδειγμα, άνθρωποι που έχουν κάνει το concept του smart living τρόπο ζωής, άτομα με ειδικές ανάγκες ή κινητικές δυσκολίες, καθώς και ηλικιωμένοι. Με τον τρόπο αυτό, θα μπορούσε κάποιος συγγενής που δεν είναι στον ίδιο χώρο με τους εν λόγω ενδιαφερόμενους να ενημερωθεί για την εξάντληση απαραίτητων προϊόντων καθημερινής χρήσης, ώστε να μπορέσει να προμηθευτεί τα προϊόντα αυτά πριν εξαντληθούν, ή και κατευθείαν το ίδιο το super market να ενημερώνεται για την παραγγελία που θα μπορούσε να γίνεται αυτόματα μέσω της εφαρμογής σε περίπτωση εξάντλησης.

Η ιδέα του smart living και γενικότερα του internet of things είναι δύο έννοιες που έχουν μεγάλη εξέλιξη στη σύγχρονη κοινωνία και στόχος τους είναι η βελτίωση της ανθρώπινης ζωής σε πολλές πτυχές της καθημερινότητας.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Installation	Δομή
Level	Επίπεδο
Container	Θέση
Sensor	Αισθητήρας
Broker	Μεσίτης
Topic	Θέμα
Message	Μήνυμα
Client	Πελάτης
Name	Όνομα
Unit	Μονάδα μέτρησης
Tare weight	Απόβαρο
Maximum threshold	Ανώτατο όριο
Minimum threshold	Κατώτατο όριο
Body	Σώμα
Response	Απάντηση
Tracking	Παρακολούθηση
Subscription strategy	Στρατηγική εγγραφής
Statistics	Στατιστικά
Timestamp	Χρονική Σημή
Publish	Δημοσίευση
Subscribe	Εγγραφή

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

REST	Representational state transfer
API	Application Program Interface
MQTT	MQ Telemetry Transport
ID	Document Identifier
dB	Database
IoT	Internet of Things
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΑΝΑΦΟΡΕΣ

[1] Laurent Probst, Erica Monfardini, Laurent Frideres, Daniela Cedola, PwC Luxembourg, “Smart Living, Smart construction products and processes”, European Union, February 2014. https://ec.europa.eu/docsroom/documents/13407/attachments/2/translations/en/renditions/native

[2] Internet of Things, https://en.wikipedia.org/wiki/Internet_of_things

[3] Carbon footprint, https://www.carbonfootprint.com/

[4] HiveMQ, Basic Concepts of MQTT protocol, https://www.hivemq.com/img/blog/publish_flow.gif