**Πολυτεχνείο Κρήτης**
**Τμήμα Ηλεκτρολόγων Μηχανικών**
**και Μηχανικών Υπολογιστών**

Διπλωματική Εργασία

# Σύστημα Ελέγχου Πρόσβασης σε Κτίρια Κατοικιών με χρήση Υβριδικού Υπολογιστικού Νέφους

# Access Control System for Residential Buildings using Hybrid Cloud Computing

Αντωνόπουλος Φίλιππος

Εξεταστική επιτροπή

Ευριπίδης Πετράκης, Καθηγητής (επιβλέπων)
Βασίλης Σαμολαδάς, Επίκουρος Καθηγητής
Δρ. Στέλιος Σωτηριάδης, Επιστημονικός Συνεργάτης

## Περίληψη

Αξιοποιώντας τεχνολογίες Υπολογιστικού Νέφους (Cloud Computing), αναπτύξαμε το Σύστημα Ελέγχου Πρόσβασης (ΣΕΠ) για την παρακολούθηση της κίνησης χρηστών σε μεγάλες κτιριακές υποδομές. Το ΣΕΠ παρέχει επίσης υπηρεσίες πληροφόρησης προς τους διαχειριστές των κτιρίων για ενδεχόμενη αυξημένη κίνηση και χρήση των υποδομών. Η χρήση των υποδομών παρέχεται στους χρήστες ως δικαίωμα με βάση συνδρομές χρηστών που έχουν συγκεκριμένη χρονική διάρκεια. Οι υπηρεσίες προς χρήστες - ενοίκους παρέχονται μέσω έξυπνων συσκευών όπως κινητά τηλέφωνα που συνδέονται με αισθητήρες πρόσβασης σε χώρους και με το διαδίκτυο μέσω WiFi. Επιπλέον μέσω του ΣΕΠ μπορούμε να παρέχουμε καλύτερη ενημέρωση προσωπικού και χρηστών σχετικά με την κατάσταση των κτιριακών υποδομών, ειδοποιώντας τους για αυξημένη ή μειωμένη κίνηση σε διάφορους χώρους. Επιπρόσθετα οι χρήστες θα είναι σε θέση να καλούν προσωπικό σε περίπτωση που επιθυμούν την άμεση εξυπηρέτησή τους. Τα δεδομένα που παράγονται σε κάθε ιδιωτικό νέφος θα αποστέλλονται σε ένα δημόσιο νέφος με στόχο την εξαγωγή συμπερασμάτων σχετικά με τις ανάγκες των χρηστών, οδηγώντας έτσι στη βελτίωση των υπηρεσιών που παρέχει το κτίριο κατοικίας τους.

## Abstract

Taking advantage of development in Cloud Computing technologies we developed IAM (Intelligent Access Management), an application for monitoring traffic and use of facilities in large residential infrastructures. IAM offers a variety of services informing the residential infrastructure manager of possible increase in activity and usage of the available facilities. Access privilege to facilities is granted to the users via limited time subscriptions. The services are provided as a Web application that runs on a smart device (e.g., mobile phone), which connects to sensors associated with specific areas of an infrastructure (e.g., residential building). IAM access and monitoring services are implemented to the cloud that can be a private cloud per infrastructure. IAM not only enables user monitoring and use of certain facilities of a building but also, allows users to interact with the management personnel in cases of emergency or when they need assistance. Anonymous use (log) data from separate private clouds are collected and transmitted to a public cloud for further analysis (e.g., big data analysis). This analysis can provide useful statistics about user preferences or uncover unknown data correlations that can be used to improve business operations and the quality of services (e.g. users' data may provide feedback for enhancing system functionality and users' acceptance).

## Ευχαριστίες

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Πετράκη για τη συμπαράσταση και το συμβουλευτικό του ρόλο καθ'όλη τη διάρκεια εκπόνησης της εργασίας. Εν συνεχεία, οφείλω να εκφράσω τις ευχαριστίες μου τόσο στα μέλη του εργαστηρίου, προπτυχιακούς,μεταπτυχιακούς όσο και στους συνεργάτες για την άριστη επικοινωνία και συνεργασία μας όλο αυτό το διάστημα. Εν κατακλείδι, δε θα μπορούσα να παραβλέψω τη στήριξη του οικογενειακού μου περιβάλλοντος από την έναρξη μέχρι και την επιτυχή ολοκλήρωση των σπουδών μου.

# Περιεχόμενα

# Κεφάλαιο 1

## 1.1 Κίνητρο

Η τεχνολογία του υπολογιστικού νέφους αποκτά συνέχεια όλο και περισσότερο ενδιαφέρον. Πλέον είναι πολύ εύκολο να αποκτήσει κάποιος πρόσβαση σε ένα υπολογιστικό νέφος ή να δημιουργήσει ένα ιδιωτικό υπολογιστικό νέφος ανάλογα με τις ανάγκες του. Υπάρχουν έτοιμες εμπορικές επιλογές που απλουστεύουν τις απαραίτητες διαδικασίες για την εγκατάσταση ιδιωτικής υποδομής υπολογιστικού νέφους (private cloud) σε εταιρείες που διστάζουν να χρησιμοποιήσουν υπηρεσίες που παρέχονται από δημόσιους παρόχους νέφους (π.χ. NuvlaBox[1]). Επομένως θεωρούμε ότι είναι η κατάλληλη χρονική στιγμή για να παρουσιάσουμε μία εφαρμογή που θα εκμεταλλεύεται τις δυνατότητες που παρέχει το υπολογιστικό νέφος. Το περιβάλλον που αφορά η εφαρμογή (κτιριακές υποδομές) είναι ένα κλειστό περιβάλλον που επιτρέπει με ευκολία στους χρήστες του να είναι συνδεδεμένοι με το Υπολογιστικό νέφος, χωρίς να υπάρχει ανάγκη ύπαρξης ακριβών υποδομών. Ακόμα λόγω της ανάπτυξης των κινητών συσκευών, ο κάθε χρήστης έχει σχεδόν συνέχεια πρόσβαση στο διαδίκτυο, με αποτέλεσμα να μας παρέχεται η δυνατότητα δημιουργίας μιας εφαρμογής με στόχο τη διαρκή ενημέρωση του.

## 1.2 Ορισμός προβλήματος

Το πρόβλημα που αντιμετωπίζουμε στην εργασία σχετίζεται με τον έλεγχο πρόσβασης και χρήσης υπηρεσιών σε μεγάλες κτιριακές υποδομές. Πολλές φορές επιθυμούμε να αποτρέπουμε σε χρήστες να έχουν πρόσβαση σε διάφορους χώρους, ή επιθυμούμε να παρέχουμε προσωρινή πρόσβαση σε ομάδες χρηστών. Η παραπάνω διαδικασία είναι συχνά χρονοβόρα και με υψηλό κόστος. Επιπλέον πολλές φορές ως χρήστες μίας κτιριακής υποδομής επιθυμούμε να γνωρίζουμε την κίνηση που επικρατεί στους διάφορους κοινόχρηστους χώρους ώστε να μπορούμε να προγραμματίζουμε καλύτερα το χρόνο μας. Ακόμα, δεν είναι λίγες οι φορές που τυχαίνει να βρεθούμε σε κάποιο κοινόχρηστο χώρο και να παρατηρούμε ότι υπάρχει συνωστισμός ή έλλειψη προσωπικού. Σε αυτές τις περιπτώσεις αναγκαζόμαστε να αναζητούμε μέλη του προσωπικού, που ενδεχομένως να βρίσκονται σε άλλο χώρο έχοντας άγνοια της υψηλής κινητικότητας που παρουσιάζεται στο συγκεκριμένο χώρο.

[1] http://sixsq.com/products/nuvlabox/

## 1.3 Προτεινόμενη λύση

Εξαιτίας των λόγων που αναφέραμε στην προηγούμενη ενότητα, οδηγηθήκαμε στη δημιουργία του Συστήματος Ελέγχου Πρόσβασης (ΣΕΠ). Η δημιουργία της εφαρμογής έχει ως στόχο την επίλυση των παραπάνω προβλημάτων, καθώς και τη γενική βελτίωση της διαχείρισης κτιριακών υποδομών. Η χρήση του βασίζεται στην εξής λογική. Κάθε κτιριακή υποδομή, εφοδιάζεται με υποδομή ιδιωτικού υπολογιστικού νέφους (private cloud). Παρόλα αυτά, σε περίπτωση που ο διαχειριστής (ή ιδιοκτήτης) του κτιρίου επιθυμεί, οι ίδιες υπηρεσίες μπορεί να παρέχονται το ίδιο καλά και μέσω μιας δημόσιας υποδομής υπολογιστικού νέφους (public cloud). Ωστόσο η χρήση ιδιωτικού νέφους παρέχει μεγαλύτερες εγγυήσεις προστασίας προσωπικών ή επιχειρηματικών δεδομένων καθώς αυτά παραμένουν στον έλεγχο της επιχείρησης ή του ιδιοκτήτη. Ανάλογα με τις απαιτήσεις της κτιριακής υποδομής, καθορίζεται και το μέγεθος του υπολογιστικού νέφους που απαιτείται. Για την πλειοψηφία των υποδομών ένα απλό υπολογιστικό σύστημα επαρκεί για να καλύψει της ανάγκες τους. Στη συνέχεια τοποθετούμε σε κάθε κοινόχρηστο ή ιδιωτικό χώρο, στον οποίο θέλουμε να ελέγχουμε την πρόσβαση, έναν αισθητήρα ανίχνευσης προσέγγισης με πρωτόκολλο Bluetooth Low Energy (BLE). Τέτοιοι αισθητήρες υπάρχουν στην αγορά και στην εργασία αυτή θα χρησιμοποιήσουμε αισθητήρες Beacon της εταιρείας Estimote[2]. Ο διαχειριστής της κτιριακής υποδομής, εγγράφει τους χώρους στο σύστημα. Όταν ένας χρήστης επιθυμεί να αποκτήσει πρόσβαση στην κτιριακή υποδομή, θα πρέπει να εγγράφεται στο σύστημα μέσω του υπολογιστικού νέφους. Στην εργασία, επιδεικνύουμε τη λειτουργία του συστήματος κάνοντας χρήστης του υπολογιστικού νέφους της υποδομής Fiware Lab [3]. Μετά την αποστολή αίτησης εγγραφής, ο διαχειριστής λαμβάνει την αίτηση, συμπληρώνοντας τα απαραίτητα στοιχεία χρήστη. Στη συνέχεια μεταβαίνει στην αντίστοιχη ενότητα του ΣΕΠ και ορίζει σε ποιους χώρους και για ποιες ημερομηνίες θα έχει δικαίωμα πρόσβασης ο χρήστης. Μετά το πέρας και αυτής της διαδικασίας ο χρήστης μπορεί να κάνει χρήση των υποδομών.

Μέσω μίας κινητής συσκευής ο χρήστης μπορεί να αλληλεπιδρά με τους αισθητήρες Beacon και να ενημερώνεται αν έχει δικαίωμα πρόσβασης ή όχι σε κάποιο χώρο. Στη συνέχεια, ανανεώνεται ο πληθυσμός του αντίστοιχου χώρου και θεωρούμε ότι ο χρήστης εισήλθε στο χώρο. Όταν ο χρήστης αλληλεπιδρά με κάποιο άλλο Beacon, θεωρούμε ότι έκανε μετάβαση σε άλλο χώρο και γίνονται οι αντίστοιχες ενημερώσεις στο σύστημα μας. Όταν το σύστημα παρατηρεί υψηλή κίνηση σε κάποιο χώρο (με βάση τα στοιχεία που έχει ορίσει ο διαχειριστής του συστήματος κατά τη δημιουργία του χώρου), θα αποστέλλεται μήνυμα προς το προσωπικό να μεταβεί στο συγκεκριμένο χώρο για την καλύτερη εξυπηρέτηση των

[2] http://estimote.com/
[3] https://www.fiware.org/lab/

χρηστών. Σε περίπτωση που δεν επαρκεί το προσωπικό για την κάλυψη των αναγκών των χρηστών, οι χρήστες θα μπορούν να στέλνουν αίτημα προς αυτό, ενημερώνοντας το για την έλλειψη. Το αίτημα αυτό στέλνεται στο ιδιωτικό νέφος και παραμένει ενεργό για κάποιο χρονικό διάστημα, με σκοπό να το αναλάβει κάποιο μέλος του προσωπικού.

Οι μεταβάσεις, οι υψηλές κινητικότητες σε χώρους, τα αιτήματα για αποστολή προσωπικού, καθώς και η αναφορά επιτυχημένης ή μη εξυπηρέτησής τους καταγράφονται στο σύστημα. Το ιδιωτικό νέφος επικοινωνεί με ένα δημόσιο νέφος με στόχο την αποστολή των παραπάνω ανώνυμων δεδομένων χρήσης των υποδομών. Αυτό αποσκοπεί στη δημιουργία ενός συστήματος διαχείρισης ενός δικτύου κτιριακών υποδομών (πχ κτίρια με κοινή διαχείριση ή ιδιοκτησία). Με τον τρόπο αυτό θα μπορούμε να γνωρίζουμε ποιοι χώροι παρουσιάζουν υψηλή κινητικότητα, καθώς και αν επαρκεί το προσωπικό για την κάλυψη των αναγκών των χρηστών. Έτσι θα είμαστε σε θέση να γνωρίζουμε καλύτερα τις ανάγκες των χρηστών για την καλύτερη εξυπηρέτησή τους.

## 1.4 Περιβάλλον εργασίας

Η προτεινόμενη λύση υλοποιήθηκε σε περιβάλλον υπολογιστικού νέφους OpenStack – Fiware. Πιο συγκεκριμένα οι υπηρεσίες παρέχονται από τον δημόσιο πάροχο Fiware Lab (που παρέχει υπηρεσίες νέφους σε χρήστες σε όλη την Ευρώπη) και το ιδιωτικό νέφος Intellicloud που έχει αναπτυχθεί για πειραματικούς σκοπούς από το Εργαστήριο Προγραμματισμού και Ευφυών συστημάτων του Πολυτεχνείου Κρήτης. Το ιδιωτικό νέφος που ανταποκρίνεται στην κτιριακή υποδομή υλοποιήθηκε στο Fiware, ενώ τα δεδομένα αποστέλλονται για επεξεργασία στο Intellicloud. Η μεταξύ τους επικοινωνία γίνεται μέσω αιτημάτων REST.

## 1.5 Δομή της εργασίας

Στο δεύτερο κεφάλαιο κάνουμε πιο αναλυτική περιγραφή του υπολογιστικού νέφους, καθώς και των άλλων τεχνολογιών που χρησιμοποιήσαμε. Στο τρίτο κεφάλαιο ασχολούμαστε κατά κύριο λόγο με το σχεδιασμό του συστήματος, καταγράφοντας τις απαιτήσεις του, καθώς και διαγράμματα που προηγήθηκαν της δημιουργίας του. Στο τέταρτο κεφάλαιο, περνάμε στην υλοποίηση του συστήματος, πρώτα από την πλευρά του Front-End, στη συνέχεια από την πλευρά του Back-End και τέλος παρουσιάζοντας παραδείγματα από τη χρήση του. Στο επόμενο κεφάλαιο ασχολούμαστε με την απόδοση του συστήματος κάτω από συνθήκες υψηλού

φόρτου, ενώ στο έκτο κεφάλαιο περιγράφουμε τα συμπεράσματα που προέκυψαν για την απόδοση του ΣΕΠ, καθώς και μελλοντικές επεκτάσεις του. Το έβδομο κεφάλαιο αποτελεί τη βιβλιογραφία.

# Κεφάλαιο 2

## 2.1 Υπολογιστικό Νέφος[4]

Το υπολογιστικό νέφος αποτελεί την τεχνολογία η οποία επιτρέπει την κατά ζήτηση παραχώρηση κατανεμημένων υπολογιστικών πόρων, δεδομένων και υπηρεσιών σε διάφορους χρήστες μέσω διαδικτύου.

Η ιδέα των διαμοιραζομένων πόρων, δεν αποτελεί κάτι το καινούριο καθώς υπήρχε από τα μέσα του 1950 με τη μορφή των κεντρικών υπολογιστών (mainframe computers). Ωστόσο η απόκτηση και συντήρηση ενός τέτοιου συστήματος αποτελούσε υψηλό έξοδο, αποτρέποντας έτσι την υιοθέτηση του μοντέλου από τις περισσότερες επιχειρήσεις. Περί τα 1970, έγινε εφικτή η δημιουργία εικονικών μηχανών, παρέχοντας έτσι τη δυνατότητα εκτέλεσης πολλαπλών λειτουργικών συστημάτων σε ένα υπολογιστικό σύστημα. Με αυτό τον τρόπο επεκτάθηκε η αρχική ιδέα του κεντρικού υπολογιστή, καθώς ένας κεντρικός υπολογιστής θα ήταν σε θέση να εξυπηρετεί ταυτόχρονα πολλούς χρήστες με διαφορετικές απαιτήσεις. Τη δεκαετία του 1990 οι εταιρίες τηλεπικοινωνιών ξεκίνησαν να προσφέρουν εικονικά ιδιωτικά δίκτυα (virtual private network). Με αυτό τον τρόπο κατάφεραν να εξισορροπήσουν το φόρτο των εξυπηρετητών.

Από το 2000 το υπολογιστικό νέφος απέκτησε τη μορφή με την οποία είναι γνωστό σήμερα. Το 2006 η Amazon παρουσίασε το Elastic Compute Cloud. Στις αρχές του 2008 είχαμε την κυκλοφορία του OpenNebula, λογισμικού ανοιχτού κώδικα με στόχο τη δημιουργία ιδιωτικών και υβριδικών νεφών. Το 2010 είχαμε την κυκλοφορία του Microsoft Azure(σήμερα γνωστό ως Windows Azure), ενός υπολογιστικού νέφους που παρέχεται από τη Microsoft. Αργότερα το ίδιο έτος είχαμε την κυκλοφορία του Openstack από τις Rackspace και NASA. Το Openstack αποτελεί ένα λειτουργικό σύστημα για υπολογιστικά νέφη, απλουστεύοντας έτσι τη διαδικασία δημιουργίας τους.

[4] https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/
https://en.wikipedia.org/wiki/Cloud_computing
https://www.ibm.com/blogs/cloud-computing/2014/03/a-brief-history-of-cloud-computing-3/

### 2.1.1 Πλεονεκτήματα και Μειονεκτήματα του Υπολογιστικού Νέφους

Σήμερα το υπολογιστικό νέφος έχει ενσωματωθεί σε μεγάλο βαθμό από πολλές υπηρεσίες και οργανισμούς, καθώς προσφέρει πολλά πλεονεκτήματα.

Μεταξύ άλλων:

- **Χαμηλό Κόστος**
  Μέσω του υπολογιστικού νέφους μπορούμε να απαλείψουμε σε μεγάλο βαθμό τα έξοδα για απόκτηση ακριβών υπολογιστικών μονάδων, αλλά να χρεωνόμαστε ανάλογα τη χρήση τους, χωρίς να υπάρχει η ανάγκη δαπάνης κεφαλαίου προκαταβολικά, όπως συμβαίνει με τα υπόλοιπα μοντέλα.

- **Ευελιξία (Elasticity)**
  Οι περισσότερες υπηρεσίες υπολογιστικού νέφους παρέχονται είτε αυτοεξυπηρετούμενες(self service) είτε κατά ζήτηση(on demand) με αποτέλεσμα να μπορούμε να έχουμε πρόσβαση σε μεγάλες ποσότητες υπολογιστικών πόρων άμεσα. Επιπλέον έχουμε τη δυνατότητα να προσαρμόζουμε τους πόρους που μας παρέχονται ανάλογα με τις απαιτήσεις μας.

- **Ελάχιστες απαιτήσεις συντήρησης από τον καταναλωτή**
  Ο καταναλωτής δεν μεταβαίνει σε εργασίες συντήρησης και αναβάθμισης, μειώνοντας έτσι αισθητά το χρόνο και τα έξοδα που θα υπήρχαν αν είχε δικό του υπολογιστικό σύστημα.

- **Απόδοση**
  Οι πάροχοι υπολογιστικών νεφών παρέχουν σύγχρονο υλικό που αναβαθμίζεται συνεχώς, επιτυγχάνοντας με αυτό τον τρόπο να έχουν υψηλή απόδοση.

- **Αξιοπιστία**
  Μέσω του υπολογιστικού νέφους η ανάκτηση αρχείων ύστερα από κάποιο σφάλμα, καθώς και η δημιουργία αντιγράφων ασφαλείας είναι εύκολα εφικτή.

Ωστόσο αξίζει να αναφερθούμε και στα μειονεκτήματα που προκύπτουν από τη χρήση του υπολογιστικού νέφους. Πιο συγκεκριμένα:

- **Ασφάλεια**
  Η ασφάλεια αποτελεί το μεγαλύτερο μειονέκτημα του υπολογιστικού νέφους. Οι τρόποι προστασίας των δεδομένων δε γνωστοποιούνται στον χρήστη, επομένως δεν είναι σε θέση να γνωρίζει πόσο ασφαλή είναι τα δεδομένα του.

- **Μειωμένη παραμετροποιησιμότητα**
  Εφόσον το υπολογιστικό νέφος παρέχεται από εξωτερικό πάροχο, περιοριζόμαστε ως προς τις διαθέσιμες τροποποιήσεις που αποσκοπούν στην καλύτερη προσαρμογή στις απαιτήσεις μας.

- **Προβλήματα νομικής φύσεως**
  Η χρήση του υπολογιστικού νέφους γίνεται ύστερα από σύναψη Συμφωνητικών Παροχής Υπηρεσιών (Service License Agreement). Ωστόσο πολλές φορές ο χρήστης δε γνωρίζει πλήρως τα δικαιώματα του, καθώς και τις υποχρεώσεις του παρόχου. Επιπλέον σε πολλές περιπτώσεις τα συμφωνητικά είναι ιδιαίτερα δεσμευτικά.

- **Πρόσβαση μέσω διαδικτύου**
  Για να έχει κανείς πρόσβαση στο υπολογιστικό νέφος, απαιτείται σύνδεση στο διαδίκτυο. Κάτι τέτοιο ενδεχομένως να δημιουργήσει προβλήματα σε οργανισμούς που θέλουν να χρησιμοποιήσουν το υπολογιστικό νέφος, ενώ αντιμετωπίζουν συχνές διακοπές στην παροχή του δικτύου ή που έχουν προγράμματα με περιορισμούς στη χρήση του.

- **Μεταφορά των δεδομένων**
  Σε πολλές περιπτώσεις δεν είναι εύκολη η μεταφορά των δεδομένων από το ένα υπολογιστικό νέφος στο άλλο. Κάτι τέτοιο οδηγεί πολλές φορές τον καταναλωτή να παραμείνει σε ένα υπολογιστικό νέφος παρόλο που δεν τον καλύπτει πλήρως.

### 2.1.2 Μοντέλα Υπολογιστικών Νεφών

- **Δημόσιο Νέφος (Public Cloud)**

Αποτελεί την πιο διαδεδομένη μορφή υπολογιστικού νέφους. Τα δημόσια νέφη παρέχονται και υλοποιούνται από έναν οργανισμό ή εταιρεία. Η πρόσβαση στις υπηρεσίες που βρίσκονται σε αυτά μπορεί να γίνεται χωρίς οικονομικό κόστος για τον τελικό χρήστη. Το μοντέλο αυτό χρησιμοποιείται κυρίως για παροχή υπηρεσιών

σε ευρύ κοινό. Ωστόσο δεν αποτελεί πάντα την ιδανική επιλογή, καθώς παρέχει χαμηλά επίπεδα ασφάλειας.

- **Ιδιωτικό Νέφος (Private Cloud)**

Η συγκεκριμένη μορφή υπολογιστικού νέφους, αναφέρεται σε συστήματα τα οποία χρησιμοποιούνται αποκλειστικά από έναν μόνο οργανισμό ή υπηρεσία. Χρησιμοποιούνται κυρίως για εφαρμογές στις οποίες είναι καθοριστικής σημασίας η ασφάλεια των δεδομένων. Επιπλέον, παρέχουν αρκετά μεγαλύτερο εύρος παραμετροποιησιμότητας για την κάλυψη των αναγκών των χρηστών. Ωστόσο η υλοποίηση τους είναι ιδιαίτερα ακριβή, κάνοντας πολλές φορές απαγορευτικό το κόστος, καθώς ο οργανισμός ή η υπηρεσία καλείται να καλύψει τα έξοδα δημιουργίας, διαχείρισης και συντήρησης.

- **Υβριδικό Νέφος (Hybrid Cloud)**

Αποτελεί μία ιδιαίτερη περίπτωση υπολογιστικού νέφους καθώς συνδυάζει τα δύο προαναφερθέντα νέφη. Η χρήση του γίνεται για να καλύψει αδυναμίες που παρουσιάζουν τα ιδιωτικά και δημόσια νέφη. Για παράδειγμα, πολλές φορές έχουμε παροδικές αυξήσεις στη ζήτηση υπολογιστικής ισχύς ενός ιδιωτικού νέφους. Στην περίπτωση αυτή κάνουμε συνδυασμό με ένα δημόσιο νέφος με σκοπό τον καταμερισμό της εργασίας. Εξίσου χαρακτηριστικό παράδειγμα αποτελεί η περίπτωση που μας αφορά η ασφάλεια των δεδομένων, επομένως χρησιμοποιούμε ένα ιδιωτικό νέφος για τη διαφύλαξη των δεδομένων σε συνδυασμό με ένα δημόσιο νέφος για τις υπολογιστικές ικανότητες του, με στόχο να αποφύγουμε την ανάγκη ύπαρξης ακριβού ιδιωτικού νέφους.

- **Κοινοτικό νέφος (Community Cloud)**

Παρόμοια λογική με το ιδιωτικό νέφος με τη διαφορά ότι χρησιμοποιείται από περισσότερους από έναν οργανισμούς. Το κοινοτικό νέφος ίσως στεγάζεται από πάροχο υπολογιστικών νεφών ή σε έναν από τους οργανισμούς. Ο λόγος που χρησιμοποιείται είναι για περιορισμό των δαπανών μέσω του διαμοιρασμού του νέφους, καθώς και για προστασία των δεδομένων, εφόσον οι οργανισμοί θεωρούνται αξιόπιστοι χρήστες.

### 2.1.3 Μοντέλα Παροχής Υπηρεσιών

*Εικόνα 1: Περιγραφή των μερών που χειρίζεται ο χρήστης ανάλογα με το μοντέλο παροχής Υπηρεσιών*[5]

Το υπολογιστικό νέφος διακρίνεται σε τρεις βασικές κατηγορίες με βάση το μοντέλο υπηρεσιών που παρέχει. Όπως φαίνεται και στην Εικόνα 1, ο πάροχος υπολογιστικού νέφους αναλαμβάνει τη διαχείριση των επιμέρους στοιχείων με σκοπό ο καταναλωτής να ασχολείται μόνο με όσα στοιχεία των ενδιαφέρουν άμεσα. Πέραν του έτοιμου λογισμικού διακρίνουμε τρεις κατηγορίες. Πιο συγκεκριμένα:

- **Υποδομή ως Υπηρεσία (Iaas: Infrastructure as a Service)**

  Αποτελεί την πιο απλή κατηγορία υπηρεσιών υπολογιστικού νέφους. Το συγκεκριμένο μοντέλο παρέχει στους χρήστες υποδομές καθώς και εικονικές μηχανές, μέσα αποθήκευσης, λειτουργικά συστήματα. Ο καταναλωτής θα προτιμήσει το συγκεκριμένο μοντέλο καθώς αποφεύγει τα έξοδα αγοράς υλικού και υπάρχει παραμετροποίηση ως προς τους πόρους που απαιτεί ο χρήστης. Έτσι μπορεί άμεσα να προσαρμόσει το υλικό στις απαιτήσεις του και να αποφύγει επιπλέον έξοδα.

- **Πλατφόρμα ως υπηρεσία (Paas: Platform as a Service)**

  Το μοντέλο αυτό απευθύνεται σε όσους ενδιαφέρονται να αναπτύξουν, να δοκιμάσουν και να διαθέσουν εφαρμογές. Ο πάροχος υπολογιστικού νέφους παρέχει μία υπολογιστική μονάδα με δυνατότητα επιλογής λειτουργικού συστήματος, καθώς και τα απαραίτητα εργαλεία ανάπτυξης της εφαρμογής. Με αυτό τον τρόπο αποφεύγουμε το κόστος αγοράς εξοπλισμού για να αναπτυχθεί η εφαρμογή, καθώς μπορούμε άμεσα να δοκιμάσουμε πως συμπεριφέρεται η εφαρμογή σε διαφορετικά περιβάλλοντα. Το μοντέλο προσαρμόζεται στις απαιτήσεις χώρου και υπολογιστικής ισχύς του χρήστη διευκολύνοντας έτσι την ανάπτυξη.

- **Λογισμικό ως Υπηρεσία (Saas: Software as a Service)**

[5] http://www.silverlighthack.com/image.axd?picture=2011%2F2%2FCloudServiceHierarchy.png

Μέσω του συγκεκριμένου μοντέλου, ο χρήστης μπορεί να έχει πρόσβαση σε μία εφαρμογή η οποία βρίσκεται στο υπολογιστικό νέφος. Ο πάροχος του υπολογιστικού νέφους πραγματοποιεί τις εργασίες συντήρησης και ελέγχου για τη σωστή λειτουργία της εφαρμογής. Ο χρήστης μπορεί μέσω διαδικτύου να εκτελέσει την εφαρμογή, αποφεύγοντας έτσι την ανάγκη εγκατάστασής της στο σύστημα του, καθώς και τυχόν ασυμβατότητες ή ανάγκες ενημέρωσής της. Υπάρχει και σε αυτό το μοντέλο προσαρμοστικότητα στις υπολογιστικές απαιτήσεις, με τη διαφορά ότι ένα μηχάνημα μπορεί να εξυπηρετεί παραπάνω από ένα χρήστη, καθώς σε περίπτωση που το μηχάνημα δεν επαρκεί για όλους τους χρήστες, δημιουργείται ένα πανομοιότυπο αντίγραφο της εφαρμογής σε επιπλέον μηχάνημα, με σκοπό την εξυπηρέτηση των παραπάνω χρηστών. Η χρέωση σε αυτό το μοντέλο γίνεται κατά κανόνα σε μηνιαία βάση.

## 2.2 Fiware[6]

Το Fiware αποτελεί ένα υπολογιστικό νέφος ανοιχτού λογισμικού με βάση την Ευρώπη. Ως κύριο στόχο έχει τη δημιουργία μιας πλατφόρμας λογισμικού που θα διευκολύνει την ανάπτυξη έξυπνων εφαρμογών. Για αυτό το λόγο παρέχει διάφορες υπηρεσίες (enablers) που υλοποιούν μια συγκεκριμένη λειτουργία. Η κάθε υπηρεσία παρέχεται με τα αντίστοιχα έγγραφα (documentation) με αποτέλεσμα την εύκολη εκμάθηση της. Με αυτό τον τρόπο και δεδομένου ότι αποτελεί μία χωρίς χρέωση πλατφόρμα αποτελεί ιδανική επιλογή για εκμάθηση και εξοικείωση με τις τεχνολογίες υπολογιστικού νέφους. Παράλληλα προσφέρει και Iaas υπηρεσίες με αποτέλεσμα ο χρήστης να μπορεί να δημιουργήσει υπολογιστικές μηχανές ανάλογα με τις ανάγκες του.

### 2.2.1 Enablers[7]

Οι Enablers αποτελούν υπηρεσίες γενικού σκοπού προσβάσιμες μέσω διαδικτύου από το Fiware για τη δημιουργία εφαρμογών. Υπακούν στους κανόνες της REST αρχιτεκτονικής. Κατά κανόνα είναι απλοί στη χρήση, ενώ παρέχουν και API για ευκολία χρήσης.

Μερικοί από τους enablers του Fiware είναι οι εξής:

[6] https://www.fiware.org/

[7] https://catalogue.fiware.org/enablers/

- **BigData Analysis – Cosmos**

Υπηρεσία που δέχεται ως είσοδο δεδομένα από πολλές διαφορετικές πηγές, τα αναλύει και εξάγει συμπεράσματα όπως σχέσεις μεταξύ των δεδομένων που δεν είναι φανερές και στατιστικά στοιχεία.

- **Complex Event Processing (CEP) - Proactive Technology Online**

Ο συγκεκριμένος enabler δέχεται ως είσοδο δεδομένα για διάφορα γεγονότα (events). Τα δεδομένα αυτά υπόκεινται σε επεξεργασία με στόχο την εξαγωγή συμπερασμάτων με την εφαρμογή κανόνων που έχουν οριστεί εκ των προτέρων.

- **Identity Management – KeyRock**

Αποτελεί έναν enabler βασισμένο στο πρωτόκολλο OAuth2[8]. Στόχος του η δημιουργία ενός ενιαίου λογαριασμού για τις διάφορες υπηρεσίες του Fiware, καθώς και αύξηση της ασφάλειας των δεδομένων των χρηστών που κάνουν χρήση των αντίστοιχων υπηρεσιών. Μέσω της συγκεκριμένης υπηρεσίας, οι χρήστες συνδέονται με τα στοιχεία τους, με σκοπό να επαληθευτεί η ταυτότητα τους(authentication) και στη συνέχεια δίνουν εξουσιοδότηση σε υπηρεσίες που επιθυμούν να χρησιμοποιήσουν, για να έχουν πρόσβαση σε τμήμα των στοιχείων τους (authorization).

- **Publish/Subscribe Context Broker - Orion Context Broker**

Υπηρεσία με σκοπό τη διαχείριση εγγραφών. Ο χρήστης θα μπορεί να εγγράφεται σε διάφορες οντότητες (entities),δηλαδή ένα σύνολο αντικειμένων με κοινά χαρακτηριστικά, και να ενημερώνεται άμεσα μέσω ειδοποίησης για αλλαγές που συμβαίνουν σε αυτές.

## 2.3 Intellicloud

Η υποδομή υπολογιστικού νέφους Intellicloud, σχεδιάστηκε, υλοποιήθηκε και συντηρείται από το εργαστήριο Ευφυών Συστημάτων. Σκοπός της υποδομής αυτής είναι η παροχή υπολογιστικών πόρων, για την ανάπτυξη εφαρμογών στο νέφος. Η υποδομή φιλοξενείται εξ ολοκλήρου στο Πολυτεχνείο Κρήτης και αυτή τη στιγμή περιλαμβάνει 128 πυρήνες επεξεργαστικής ισχύς, 384 GB RAM και 12 TB σε σκληρό δίσκο(25 TB σε εικονικό σκληρό δίσκο). Το λογισμικό και η αρχιτεκτονική που είναι υπεύθυνο για τη λειτουργία του Intellicloud είναι βασισμένο σε Openstack Grizzly.

[8] https://oauth.net/2/

## 2.4 Υπηρεσιοκεντρική Αρχιτεκτονική

Η Υπηρεσιοκεντρική Αρχιτεκτονική (Service Oriented Architecture) αποτελεί ουσιαστικά το σχεδιασμό εφαρμογών κάνοντας χρήση προϋπαρχόντων υπηρεσιών. Ως υπηρεσία(service) ορίζουμε ένα τμήμα κώδικα διαθέσιμο μέσω διαδικτύου που εκτελεί μία συγκεκριμένη εργασία. Η επικοινωνία μεταξύ των υπηρεσιών γίνεται κάνοντας χρήση διαδεδομένων πρωτοκόλλων ανταλλαγής δεδομένων (XML, JSON). Ο χρήστης μπορεί να κάνει άμεσα χρήση μιας υπηρεσίας για την εκτέλεση μίας δραστηριότητας που επιθυμεί, γνωρίζοντας τις εντολές που απαιτούνται για επικοινωνία, χωρίς να κρίνεται αναγκαίο να γνωρίζει τον τρόπο λειτουργίας της υπηρεσίας.

Η υπηρεσιοκεντρική αρχιτεκτονική είναι ιδιαίτερα χρήσιμη, καθώς μειώνει σημαντικά το φόρτο εργασίας για τη δημιουργία μιας εφαρμογής. Επιπλέον οι υπηρεσίες όντας αυτόνομα τμήματα κώδικα, μπορούν να αναβαθμιστούν, ή σε περίπτωση που δε μας καλύπτουν απόλυτα να γραφούν πλήρως από την αρχή, ή και να αντικατασταθούν από κάποιες άλλες, χωρίς να επηρεάζεται ο υπόλοιπος κώδικας. Πέραν αυτού παρέχουν ευκολία λόγω της αυτοματοποιημένης μορφής τους, καθώς χρειάζονται λίγες γνώσεις ώστε να χρησιμοποιηθούν. Ακόμα, εφόσον μιλάμε για αυτόνομες υπηρεσίες, μπορούν να τοποθετηθούν σε ξεχωριστά μηχανήματα σε περίπτωση που μας απασχολεί ένας σχεδιασμός που να αξιοποιεί πολλές εικονικές μηχανές για βελτίωση της απόδοσης. Τέλος, χάρη στο υπολογιστικό νέφος, ο διαμοιρασμός των υπηρεσιών είναι εύκολα εφικτός, καθώς ενδέχεται ο πάροχος να παρέχει εικονικές μηχανές με προεγκατεστημένες κάποιες υπηρεσίες για την εύκολη χρησιμοποίησή τους.

## 2.5 Αρχιτεκτονική REST[9]

Η αρχιτεκτονική REST αποσκοπεί στη δημιουργία κάποιων κανόνων με σκοπό τη δημιουργία υπηρεσιών που θα είναι πιο εύκολο να επικοινωνούν μεταξύ τους. Ορίζοντας κανόνες στον τρόπο που στέλνονται και λαμβάνονται τα δεδομένα από μία υπηρεσία, επιτυγχάνουμε πιο εύκολη διασύνδεση των υπηρεσιών.

Οι κυριότεροι κανόνες της αρχιτεκτονικής REST είναι οι εξής:

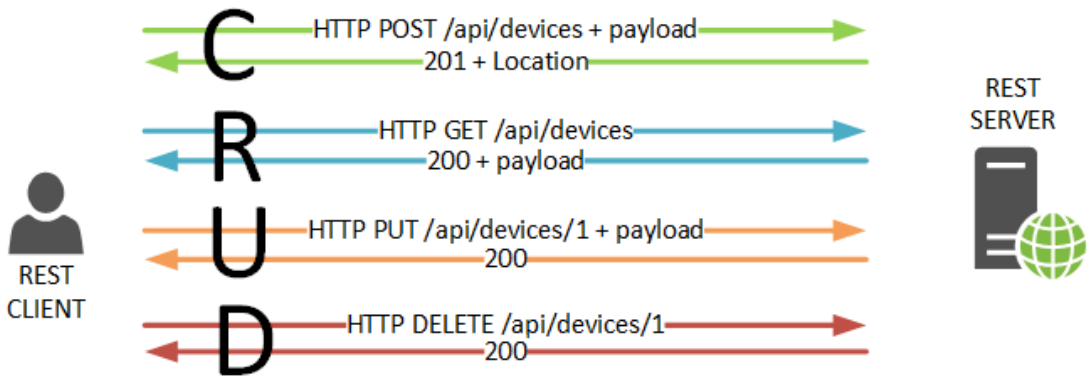
- **Χρήση διαφορετικών μεθόδων ανάλογα με τις απαιτήσεις μας**
  Η αρχιτεκτονική REST στηρίζεται στη χρήση διαφορετικών μεθόδων για την επικοινωνία μεταξύ των υπηρεσιών. Πιο συγκεκριμένα θεωρεί ότι πρέπει να

[9] http://www.ibm.com/developerworks/library/ws-restful/

κάνουμε κατ'ελάχιστο χρήση διαφορετικών μεθόδων για τη δημιουργία, την ανάκτηση, την επεξεργασία και τη διαγραφή των πόρων. Αυτό είναι γνωστό και ως CRUD(Create,Read,Update,Delete). Στην Εικόνα 2 παρουσιάζεται το εξής μοντέλο. Πιο αναλυτικά:

- GET, για ανάκτηση ενός πόρου
- POST, για δημιουργία ενός πόρου
- PUT, για ανανέωση των στοιχείων ενός πόρου
- DELETE, για διαγραφή ενός πόρου

Αντίστοιχα, ο εξυπηρετητής μετά την ολοκλήρωση του αιτήματος, θα πρέπει να επιστρέφει τον αντίστοιχο κωδικό απάντησης (response code).




*Εικόνα 2: Περιγραφή του CRUD*[10]

- 
- **Σχεδιασμός χωρίς καταστάσεις**
  Σε πολλές περιπτώσεις στο παρελθόν χρησιμοποιούσαμε καταστάσεις για την εξυπηρέτηση των διαφόρων αιτημάτων. Αυτό σήμαινε ότι ο εξυπηρετητής θα έπρεπε να κρατάει δεδομένα σχετικά με τα αιτήματα, να τα εξυπηρετεί, καθώς και να ανανεώνει την κατάσταση των δεδομένων αυτών. Κάτι τέτοιο δημιουργούσε προβλήματα καθώς ήταν απαιτητικό και υπήρχαν προβλήματα συγχρονισμού σε περίπτωση που κάποιο επόμενο αίτημα του ίδιου χρήστη έπρεπε να εκτελεστεί σε άλλον εξυπηρετητή. Για αυτό το σκοπό δεν κάνουμε χρήση καταστάσεων. Αντίθετα, στέλνουμε με κάθε αίτημα είτε στο URI, είτε στο σώμα του αιτήματος, όλη την απαραίτητη πληροφορία.

- **Χρήση απλών URI**

[10] http://networkop.co.uk/images/rest-crud.png

Συχνά, λόγω κακού σχεδιασμού καταλήγουμε με πολύπλοκα και δυσνόητα URI. Αυτό δυσκολεύει το διαμοιρασμό τους, καθώς και κάνει πιο δύσκολη την πρόσβαση σε αυτά. Μέσω του REST αποσκοπούμε σε ένα σχεδιασμό κατά τον οποίο τα URI θα αντιστοιχούν σε μιας μορφής ιεραρχίας. Πιο συγκεκριμένα, σε περίπτωση που θέλουμε να ζητήσουμε μία ιδιότητα ενός πόρου, ξεκινάμε από την κατηγορία του, στη συνέχεια μεταβαίνουμε στο όνομα του και τέλος στην ιδιότητα που θέλουμε να ανακτήσουμε. Μπορεί να γίνει και διαφορετικός διαχωρισμός, ανάλογα τις απαιτήσεις μας, ωστόσο καλό είναι να ακολουθείται ένας μόνο διαχωρισμός ώστε να γίνεται εύκολα κατανοητός.

- **Μεταφορά δεδομένων σε συγκεκριμένες μορφές**
  Για την ευκολότερη επικοινωνία μεταξύ των υπηρεσιών, η αρχιτεκτονική REST, προτείνει τη μεταφορά των δεδομένων σε μορφή JSON, ή σε μορφή XML. Με αυτό τον τρόπο οι περισσότερες υπηρεσίες θα είναι άμεσα συμβατές μεταξύ τους, χωρίς να χρειάζεται να μετατρέπουν τα δεδομένα σε άλλες μορφές.

## 2.6 Διαδίκτυο των πραγμάτων[11]

Το διαδίκτυο των πραγμάτων (Internet of things) είναι ένα όραμα που έχει ως στόχο τη διασύνδεση πολλών ηλεκτρονικών συσκευών καθημερινής χρήσης με σκοπό τη μεταξύ τους επικοινωνία ή την επικοινωνία με κάποιον άνθρωπο. Μέσω του δικτύου συσκευών, μπορούμε να συλλέγουμε δεδομένα από το περιβάλλον και να παρέχουμε κάποια υπηρεσία. Οι συσκευές, μπορούν να αλληλεπιδρούν μεταξύ τους, ή να ελέγχονται απομακρυσμένα μέσω διαδικτύου. Οι συσκευές μεταξύ άλλων μπορεί να είναι ιατρικές, αισθητήρες, αυτοκίνητα, οικιακές ηλεκτρικές και άλλα.

Αξίζει να αναφέρουμε ότι, αν και η ιδέα του διαδικτύου των πραγμάτων προσφέρει πολλά πλεονεκτήματα και δυνατότητες ανάπτυξης, εγκυμονεί κινδύνους λόγω του χαμηλού επιπέδου ασφαλείας που υπάρχει στις απλές συσκευές. Κάτι τέτοιο αποτελεί μεγάλο κίνδυνο καθώς υπάρχει επικοινωνία μεταξύ των συσκευών, όπως προαναφέραμε, με αποτέλεσμα το πρόβλημα να μπορεί εύκολα να επεκταθεί. Ωστόσο,το διαδίκτυο των πραγμάτων συνεχίζει να αναπτύσσεται ταχύτατα, με όλο και περισσότερες συσκευές να γίνονται «έξυπνες». Ανώτατος στόχος του διαδικτύου των πραγμάτων είναι η απλούστευση πολλών καθημερινών διαδικασιών.

[11] http://link.springer.com/chapter/10.1007/978-1-4419-8237-7_13#page-1

### 2.7 Bluetooth Low-Energy

Το Bluetooth Low-Energy(BLE) αποτελεί μία τεχνολογία δημιουργίας ασύρματου προσωπικού δικτύου. Η διαφορά του με το παραδοσιακό Bluetooth έγκειται στη χαμηλή κατανάλωση ενέργειας. Το γεγονός αυτό το κάνει ιδανικό για εφαρμογές Διαδικτύου των πραγμάτων. Ένα βασικό πλεονέκτημα του BLE αποτελεί το γεγονός ότι υπάρχει προεγκατεστημένο λογισμικό αναγνώρισης και αξιοποίησης του σε όλα σχεδόν τα σύγχρονα λειτουργικά συστήματα (iOS5, Windows Phone 8.1, Android 4.3, Windows 8, Linux 3.4 και μετέπειτα εκδόσεις τους). Εξαιτίας της υψηλής αυτονομίας του και τις ευρείας αποδοχής του από την αγορά, σήμερα χρησιμοποιείται εκτεταμένα σε εφαρμογές υγείας, σε διάφορους αισθητήρες καθώς και σε εφαρμογές παροχής ειδοποιήσεων.

## Κεφάλαιο 3$^{ο}$

### 3.1 Απαιτήσεις συστήματος

Προτού μεταβούμε στο σχεδιασμό της εφαρμογής είναι χρήσιμο να αναφερθούμε στις βασικές ομάδες χρηστών στις οποίες απευθύνεται το ΣΕΠ, καθώς και στις απαιτήσεις τους. Καταγράφοντας τις απαιτήσεις μπορούμε να αποφασίσουμε ποιες υπηρεσίες πρέπει να δημιουργηθούν για την κάλυψή τους.

Για την ευκολότερη καταγραφή δημιουργούμε τις εξής πέντε ομάδες χρηστών.

- Ο διαχειριστής του δημοσίου νέφους
- Ο διαχειριστής του ιδιωτικού νέφους
- Οι εργαζόμενοι της κτιριακής υποδομής
- Οι κάτοικοι της κτιριακής υποδομής
- Οι επισκέπτες της υποδομής

Στο πλαίσιο της εργασίας θα μας απασχολήσουν οι τέσσερις τελευταίες ομάδες.

Στη συνέχεια ορίζουμε τις απαιτήσεις συστήματος, κάνοντας το διαχωρισμό σε λειτουργικές και μη. Με τον όρο λειτουργικές απαιτήσεις, ορίζουμε τις διαδικασίες

τις οποίες πρέπει να υλοποιεί το σύστημα, δηλαδή είναι ο λόγος για τον οποίο το υλοποιήσαμε. Οι απαιτήσεις αυτές διαφέρουν ανάλογα με την ομάδα χρηστών, ωστόσο θα πρέπει να τηρούνται οι απαιτήσεις όλων των χρηστών.

Αναλυτικότερα οι λειτουργικές απαιτήσεις ανά κατηγορία χρήστη.

**Διαχειριστής ιδιωτικού νέφους**

1. Εισαγωγή/Επεξεργασία/Διαγραφή χρήστη
2. Εισαγωγή/Επεξεργασία/Διαγραφή κοινόχρηστου ή ιδιωτικού χώρου
3. Ορισμός μόνιμης θέσης εργασίας υπαλλήλου
4. Προβολή τωρινής θέσης υπαλλήλων στο κτίριο
5. Προβολή στατιστικών στοιχείων πληρότητας χώρων
6. Προβολή/Επεξεργασία/Διαγραφή δικαιωμάτων πρόσβασης
7. Προβολή ιστορικού χρηστών
8. Συσχετισμός χώρου με αισθητήρα

**Εργαζόμενος κτιριακής υποδομής**

1. Προβολή στατιστικών στοιχείων πληρότητας χώρων
2. Προβολή δικαιωμάτων πρόσβασης
3. Προβολή ατομικού ιστορικού προσβάσεων
4. Προβολή ενεργών συναγερμών
5. Χειρισμός ενεργών συναγερμών
6. Προβολή ενεργών αιτημάτων αποστολής προσωπικού
7. Χειρισμός ενεργών αιτημάτων αποστολής προσωπικού
8. Αίτηση για πρόσβαση σε τοποθεσία

**Κάτοικος/Επισκέπτης κτιριακής υποδομής**

1. Προβολή στατιστικών στοιχείων πληρότητας χώρων
2. Προβολή δικαιωμάτων πρόσβασης
3. Προβολή ατομικού ιστορικού προσβάσεων
4. Αίτηση για πρόσβαση σε τοποθεσία
5. Δημιουργία αίτησης για αποστολή προσωπικού

Η κύρια διαφορά μεταξύ ενός κατοίκου και ενός επισκέπτη μιας κτιριακής υποδομής βρίσκεται στο γεγονός ότι ο επισκέπτης δεν έχει δικαίωμα πρόσβασης σε ιδιωτικούς χώρους.

Στη συνέχεια θα αναλύσουμε τις μη λειτουργικές απαιτήσεις. Οι μη λειτουργικές απαιτήσεις, δεν είναι απαραίτητο να ικανοποιηθούν για τη δημιουργία του συστήματος. Ωστόσο, η μη ικανοποίησή τους θα οδηγήσει σε ένα κακής ποιότητας σύστημα, καθώς η κάλυψη τους καθορίζει την ποιότητα του.

Αναλυτικότερα:

- Απόδοση: Αφορά την ταχύτητα απόκρισης του συστήματος, υπό συνθήκες υψηλού φόρτου.
- Επεκτασιμότητα: Ορίζει πόσο εύκολο ή δύσκολο είναι να προστεθούν περισσότερες δυνατότητες και λειτουργίες στην εφαρμογή.
- Ασφάλεια: Ένα σύστημα με υψηλή ασφάλεια κάνει δύσκολη την υποκλοπή των δεδομένων των χρηστών του από άλλους χρήστες ή από τρίτους.
- Χρηστικότητα: Καθορίζει πόσο εύκολο στη χρήση του είναι το σύστημα.
- Χρόνος λειτουργικότητας: Ορίζει το χρόνο ανάμεσα σε δύο εργασίες συντήρησης του συστήματος. Ένα σωστά υλοποιημένο σύστημα χρειάζεται σύντομες και περιστασιακές εργασίες συντήρησης, σε αντίθεση με ένα πρόχειρα υλοποιημένο σύστημα, που ενδεχομένως να χρειάζεται απρογραμμάτιστες, συνεχείς εργασίες συντήρησης.

Για να ικανοποιήσουμε τις παραπάνω απαιτήσεις, αποφασίσαμε να αναπτύξουμε το ΣΕΠ με υπηρεσιοκεντρική αρχιτεκτονική, έτσι ώστε να είναι εύκολη η συντήρηση και η επέκταση του. Επιπλέον, η συγκεκριμένη αρχιτεκτονική επιτρέπει τον διαμοιρασμό του φόρτου εργασίας σε διαφορετικές εικονικές μηχανές, καθώς οι υπηρεσίες δεν είναι απαραίτητο να φιλοξενούνται στην ίδια εικονική μηχανή, αυξάνοντας έτσι την απόδοση του συστήματος. Παράλληλα, η εφαρμογή που θα παρέχεται στους χρήστες, δεν περιέχει στοιχεία σχετικά με τις λειτουργίες του διαχειριστή ιδιωτικού υπολογιστικού νέφους αυξάνοντας έτσι το επίπεδο ασφαλείας.

## 3.2 Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams)

Έχοντας περιγράψει στην Ενότητα 3.1 τις λειτουργικές απαιτήσεις χρηστών ανά κατηγορία, προσπαθούμε να τις ομαδοποιήσουμε. Με αυτό τον τρόπο προσπαθούμε να καθορίσουμε τις υπηρεσίες από τις οποίες θα αποτελείται το ΣΕΠ. Κάνουμε χρήση διαγραμμάτων περιπτώσεων χρήσης της UML[12](Use Case Diagrams) για την καλύτερη αποτύπωση τους.

Αναλυτικότερα, για το διαχειριστή ιδιωτικού νέφους παρατηρούμε ότι οι εξής απαιτήσεις έχουν να κάνουν με διαχείριση χρηστών.

[12] https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

- Εισαγωγή/Επεξεργασία/Διαγραφή χρήστη
- Ορισμός μόνιμης θέσης εργασίας υπαλλήλου
- Προβολή τωρινής θέσης υπαλλήλων στο κτίριο

Επομένως τις εντάσσουμε σε μία Υπηρεσία Διαχείρισης Χρηστών.

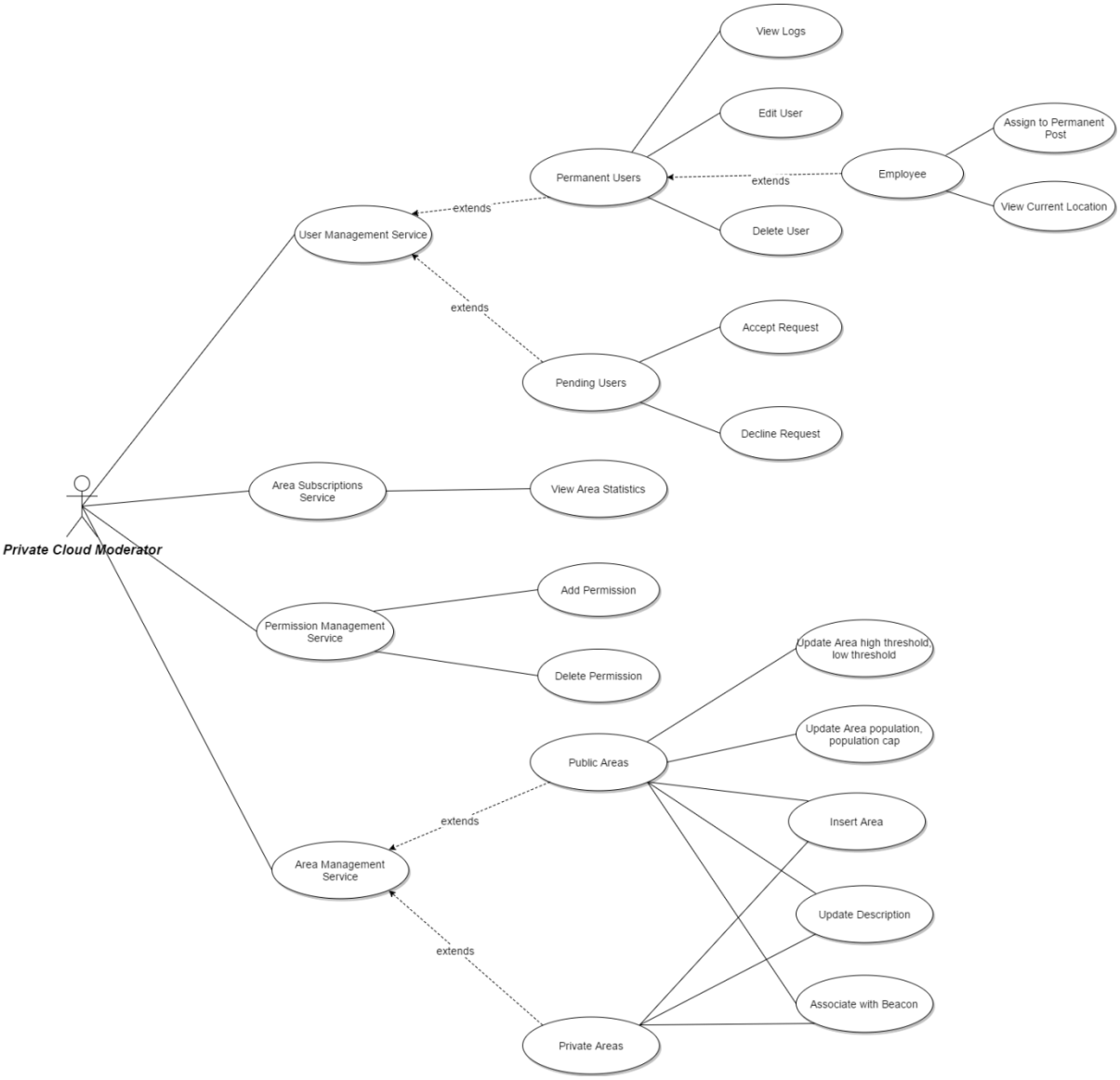
Αντίστοιχα αποφασίσαμε να έχουμε Υπηρεσία Διαχείρισης Χώρων που θα καλύπτει τις εξής απαιτήσεις

- Εισαγωγή/Επεξεργασία/Διαγραφή κοινόχρηστου ή ιδιωτικού χώρου
- Συσχετισμός χώρου με αισθητήρα

Μία τρίτη υπηρεσία που αποφασίσαμε να δημιουργήσουμε, αφορά στη Διαχείριση Δικαιωμάτων Πρόσβασης.

Για την προβολή των στοιχείων πληρότητας χώρων, δημιουργούμε μια υπηρεσία διαχείρισης εγγραφών. Ο διαχωρισμός από την Υπηρεσία Διαχείρισης Χώρων γίνεται για λόγους ασφάλειας και απόδοσης.

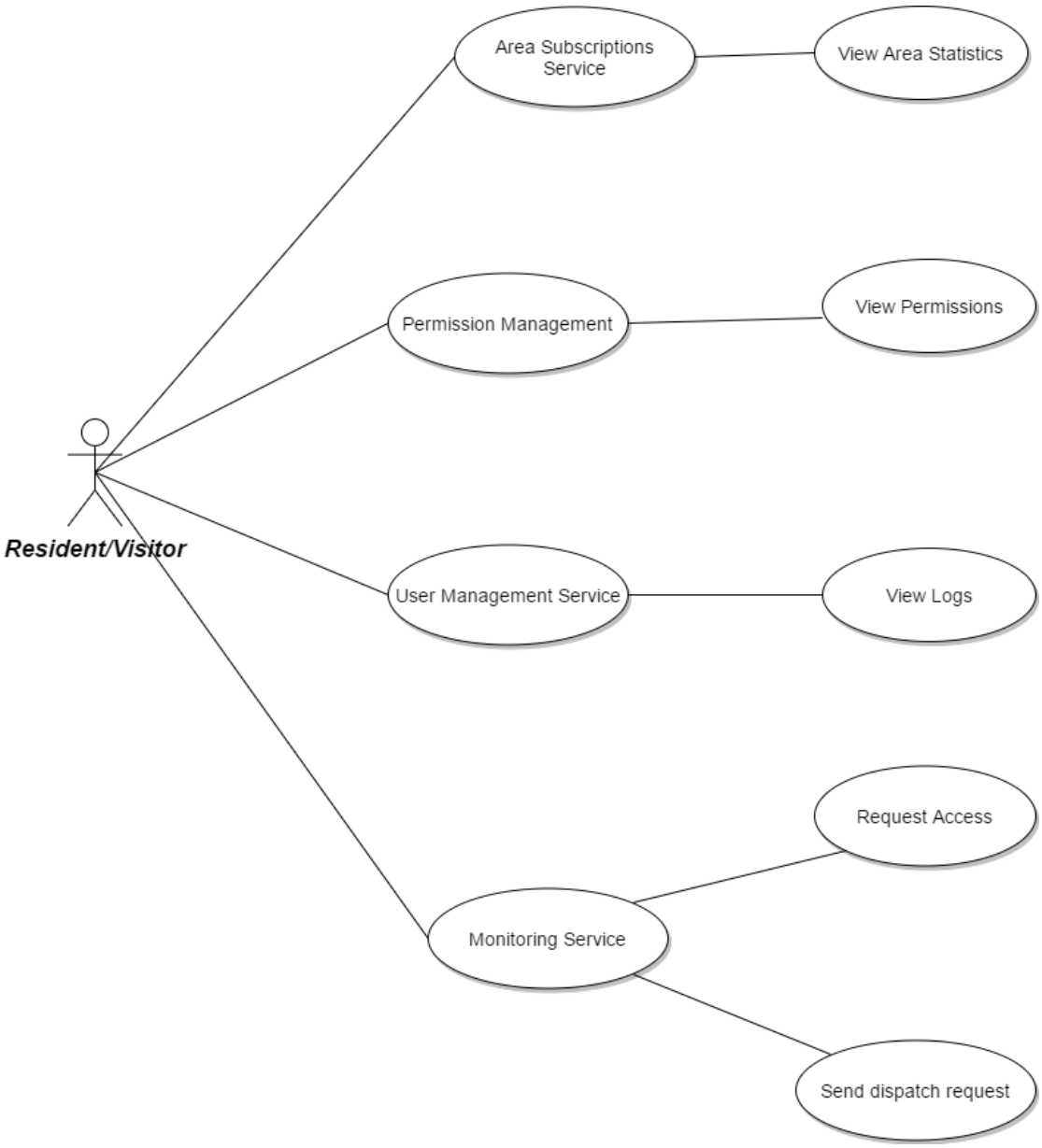
Συνεπώς καταλήγουμε στο διάγραμμα της Εικόνας 3.




*Εικόνα 3 Διάγραμμα περιπτώσεων χρήσης (Use case diagram) για το διαχειριστή ιδιωτικού νέφους*

Αντίστοιχη λογική εφαρμόζουμε και για τις λειτουργικές απαιτήσεις του κατοίκου/ επισκέπτη. Αποφασίσαμε τη δημιουργία μιας Υπηρεσίας Ελέγχου Προσβάσεων με στόχο τον έλεγχο πρόσβασης, καθώς και τη διαχείριση των αιτημάτων για αποστολή προσωπικού.

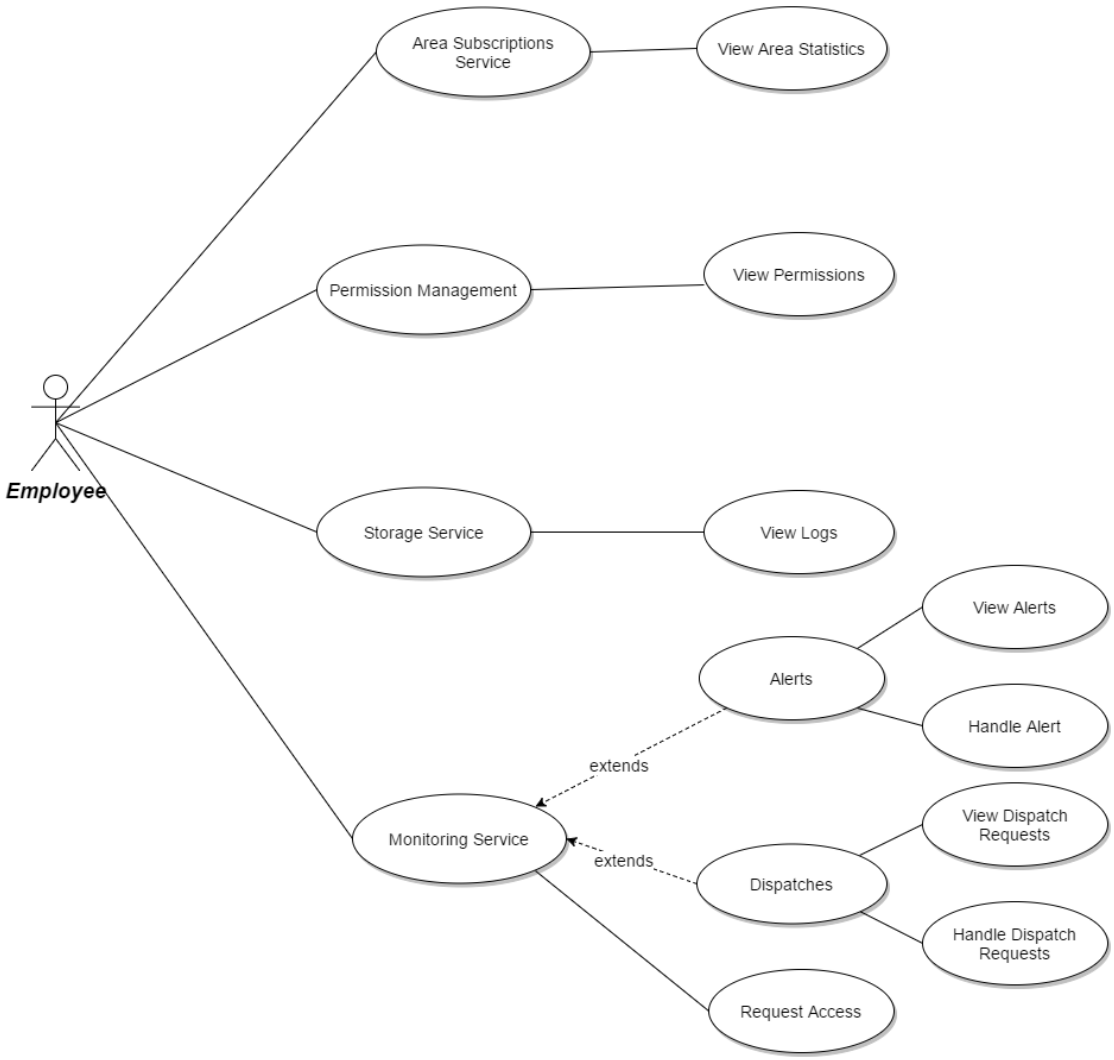
Στην Εικόνα 4 παρουσιάζουμε το αντίστοιχο διάγραμμα.




*Εικόνα 4 Διάγραμμα περιπτώσεων χρήσης (Use case diagram) για τους κατοίκους/επισκέπτες*

Στη συνέχεια παρουσιάζουμε το διάγραμμα που αντιστοιχεί στον εργαζόμενο της κτιριακής υποδομής. Οι διαφορές σε σύγκριση με το προηγούμενο διάγραμμα, βρίσκονται στην Υπηρεσία Ελέγχου Προσβάσεων, καθώς θα είναι υπεύθυνη και για

τη διαχείριση των διαφόρων συναγερμών, λόγω υψηλής κινητικότητας σε κάποιο χώρο.



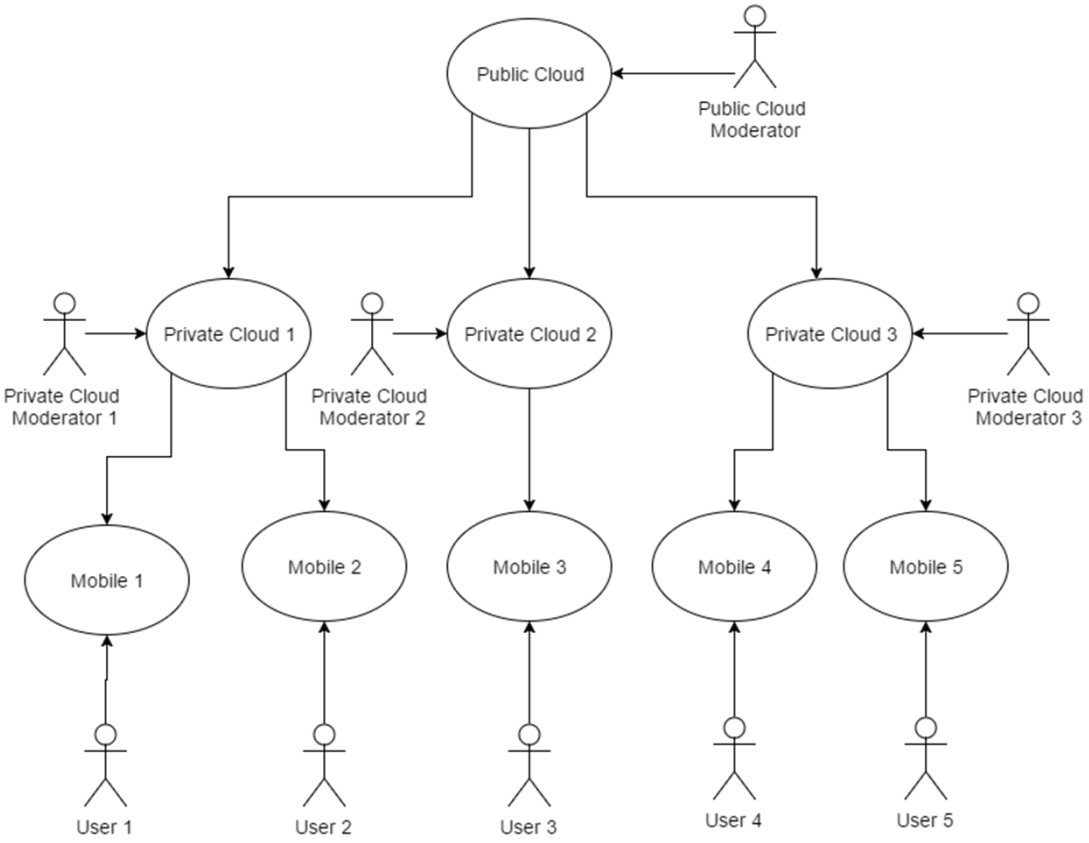

*Εικόνα 5 Διάγραμμα περιπτώσεων χρήσης (Use case diagram) για τους υπαλλήλους*

## 3.3 Διάγραμμα Τοπολογίας (Deployment Diagram)

Το επόμενο βήμα ήταν η κατασκευή ενός διαγράμματος τοπολογίας του ΣΕΠ (Deployment Diagram[13]). Στο διάγραμμα αυτό φαίνεται ότι έχουμε διάφορα Ιδιωτικά νέφη, που κάθε ένα αποτελεί μία κτιριακή υποδομή. Το κάθε νέφος έχει διάφορους χρήστες που έχουν πρόσβαση σε αυτό μέσω της χρήσης ενός κινητού τηλεφώνου. Οι χρήστες αυτοί μπορεί να είναι κάτοικοι, επισκέπτες, ή εργαζόμενοι. Επίσης κάθε ιδιωτικό νέφος έχει το διαχειριστή του, ενώ όλα τα νέφη επικοινωνούν με το δημόσιο νέφος, το οποίο με τη σειρά του έχει ένα διαχειριστή. Στην εργασία

[13] https://en.wikipedia.org/wiki/Deployment_diagram

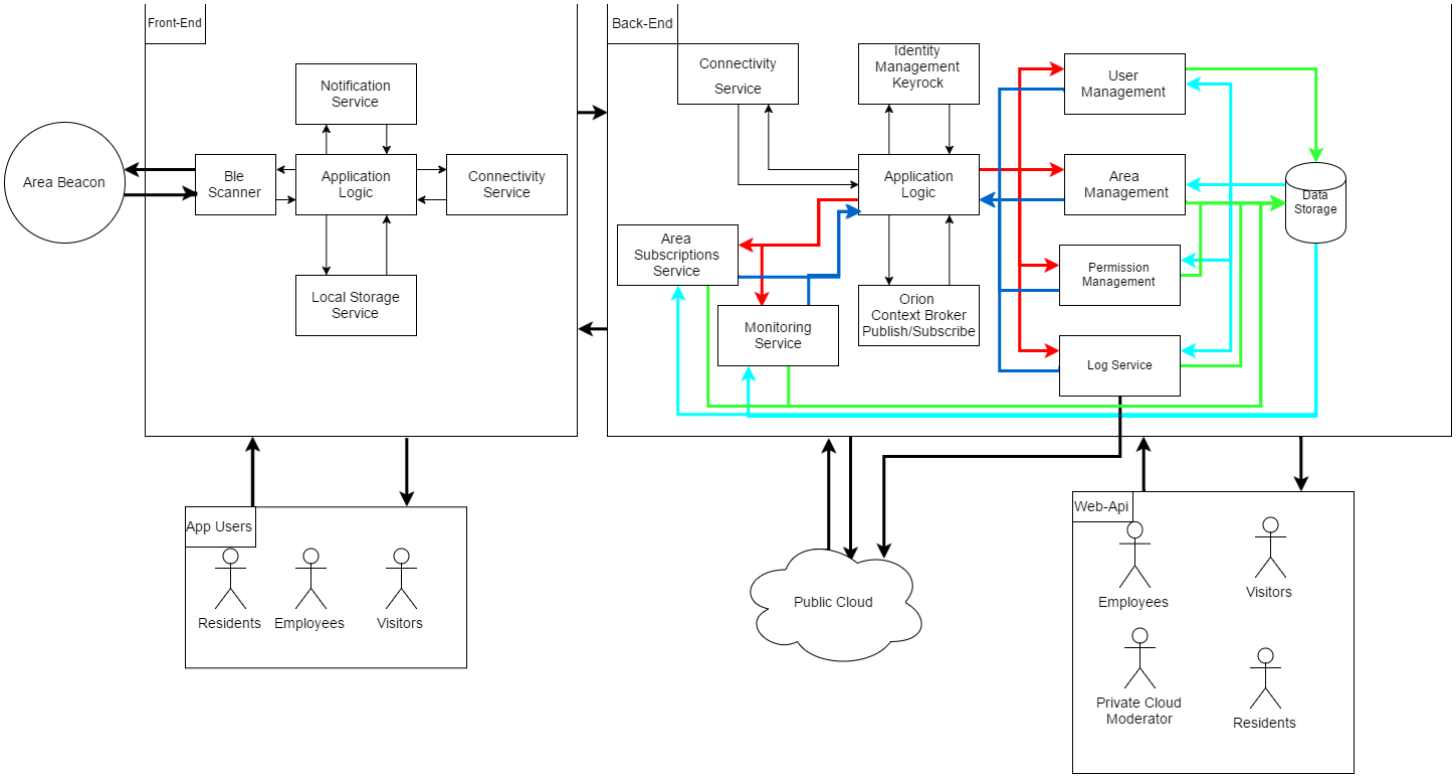
θα ασχοληθούμε κατά κύριο λόγο με ένα ιδιωτικό νέφος, και την πρόσβαση σε αυτό μέσω κινητών συσκευών ή φυλλομετρητή.




*Εικόνα 6 Διάγραμμα τοπολογίας συστήματος (Deployment diagram)*

## 3.4 Διάγραμμα  Αρχιτεκτονικής(Architectural Diagram)

Στη συνέχεια παρουσιάζουμε ένα εξειδικευμένο διάγραμμα της εφαρμογής μας που αφορά το κομμάτι με το οποίο θα ασχοληθούμε, δηλαδή ένα ιδιωτικό νέφος με τους χρήστες του, καθώς και η διασύνδεση του με το δημόσιο νέφος.




*Εικόνα 7 Διάγραμμα αρχιτεκτονικής συστήματος (Architectural Diagram)*

Το διάγραμμα της Εικόνας 7 περιγράφει την αρχιτεκτονική του ΣΕΠ. Ακολουθεί μία σύντομη περιγραφή του διαγράμματος, το οποίο θα μας απασχολήσει περισσότερο στο 4$^{ο}$ κεφάλαιο.

## Σύστημα Διεπαφής τελικού χρήστη (Front-End)

Το σύστημα διεπαφής τελικού χρήστη είναι το τμήμα του ΣΕΠ που καθιστά δυνατή την επικοινωνία του χρήστη με τον πυρήνα της εφαρμογής. Υπάρχουν δύο τρόποι με τους οποίους μπορούν να έχουν πρόσβαση στο ΣΕΠ οι χρήστες. Ο πρώτος είναι μέσω εφαρμογής για κινητά.

Η εφαρμογή περιγράφεται στο πάνω αριστερά τμήμα της εικόνας , στο πλαίσιο Front-End.

Αναλυτικότερα αποτελείται από τα εξής μέρη (εκτενέστερη ανάλυση στο Κεφάλαιο 4):

**Λογική εφαρμογής (Application Logic)**

Περιέχει τον κώδικα, ο οποίος χρησιμοποιείται για την επικοινωνία με τον πυρήνα του ΣΕΠ. Το τμήμα αυτό της εφαρμογής είναι υπεύθυνο για την ενορχήστρωση των επιμέρους μελών της.

**Τοπικός Χώρος Αποθήκευσης (Local Storage[14])**

Μέσω αυτής της υπηρεσίας, η εφαρμογή μας μπορεί να αποθηκεύσει τα απαραίτητα δεδομένα, ώστε να είναι αυτόματη η είσοδος στην εφαρμογή, την επόμενη φορά.

**Ανιχνευτής συσκευών BLE (BLE Scanner)**

Βιβλιοθήκη που επιτρέπει την αναγνώριση συσκευών BLE από την εφαρμογή.

**Υπηρεσία Διασύνδεσης (Connectivity Service)**

Υπηρεσία που ελέγχει τον τοπικό χώρο αποθήκευσης για δεδομένα σύνδεσης και μεταβαίνει σε προσπάθεια αυτόματης σύνδεσης του χρήστη.

**Υπηρεσία Ενημέρωσης (Notification Service)**

Υπηρεσία, μέσω της οποίας οι υπάλληλοι θα μπορούν να ενημερώνονται για τυχόν περιστατικά που συμβαίνουν.

Η εφαρμογή παρέχει την απαιτούμενη λειτουργικότητα για τους χρήστες που είναι υπάλληλοι, επισκέπτες ή κάτοικοι. Ο διαχειριστής ιδιωτικού νέφους δεν μπορεί να

[14] http://www.w3schools.com/html/html5_webstorage.asp

κάνει χρήση της εφαρμογής. Αυτό αποφασίστηκε για λόγους ευχρηστίας, καθώς και για λόγους ασφάλειας.

Ο διαχειριστής θα έχει πρόσβαση στην εφαρμογή μέσω φυλλομετρητή (Web Api). Αυτή τη δυνατότητα θα έχουν και οι υπόλοιπες ομάδες χρηστών σε περίπτωση που το επιθυμούν. Ωστόσο για τον έλεγχο εισόδου σε χώρους, απαιτείται χρήση της εφαρμογής για κινητά.

## Σύστημα επεξεργασίας αποθήκευσης δεδομένων και παροχής υπηρεσιών προς το Front-End (Back-End)

Αποτελεί τον πυρήνα του ΣΕΠ, καθώς σε αυτό υλοποιούνται οι απαραίτητες λειτουργίες του συστήματος. Στο σχήμα (Εικόνα 7) απεικονίζεται στο πάνω δεξιά μέρος.

Ακολουθεί σύντομη περιγραφή των στοιχείων του Back-End (αναλυτικότερη περιγραφή στο Κεφάλαιο 4):

**Λογική εφαρμογής (Application Logic)**

Αποτελεί τον πυρήνα του ΣΕΠ. Στόχος του η ενορχήστρωση των διάφορων υπηρεσιών και η σωστή μεταξύ τους επικοινωνία.

**Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών Keyrock (Keyrock Identity Manager Service)**

Υπηρεσία με στόχο την εύκολη εγγραφή και σύνδεση των χρηστών στο σύστημα.

**Υπηρεσία Διαχείρισης Χρηστών (User Management Service)**

Σκοπός της υπηρεσίας είναι η απαραίτητη λειτουργικότητα για τη δημιουργία, επεξεργασία, διαγραφή χρηστών, προβολή ιστορικού χρηστών, καθώς και παροχή στοιχείων σχετικά με την τωρινή τοποθεσία του χρήστη,.

**Υπηρεσία Διαχείρισης Χώρων (Area Management Service)**

Υπηρεσία που αποσκοπεί στην παροχή λειτουργικότητας για τη διαχείριση των χώρων, όπως δημιουργία, επεξεργασία, διαγραφή χώρων και συσχετισμός τους με έναν αισθητήρα beacon.

**Υπηρεσία Διαχείρισης Συνδρομών σε Χώρους (Area Subscriptions Service)**

Στόχος της υπηρεσίας, είναι η ενημέρωση των χρηστών σχετικά με την κίνηση στους χώρους της υποδομής, καθώς και η ανανέωση των στοιχείων πληρότητας σε περίπτωση εισόδου ή αποχώρησης χρηστών.

**Υπηρεσία Διαχείρισης Δικαιωμάτων (Permission Management Service)**

Υπηρεσία με στόχο την παροχή και διαχείριση δικαιωμάτων πρόσβασης των χρηστών.

**Υπηρεσία Ελέγχου προσβάσεων (Monitoring Service)**

Μέσω της συγκεκριμένης υπηρεσίας έχουμε ως στόχο να δίνουμε άδεια πρόσβασης στους διάφορους χώρους, καθώς και να ελέγχουμε τις καταστάσεις συναγερμού ή τα αιτήματα για επιπλέον προσωπικό.

**Υπηρεσία αποθήκευσης Δεδομένων (Storage Service)**

Υπηρεσία που μεταφέρει τα δεδομένα σχετικά με τις εισόδους χρηστών σε κοινόχρηστους χώρους, καθώς και τα στοιχεία που αφορούν την έναρξη,εξυπηρέτηση και λήξη ενός συναγερμού ή τα αιτήματα για επιπλέον προσωπικό, προς το δημόσιο νέφος.

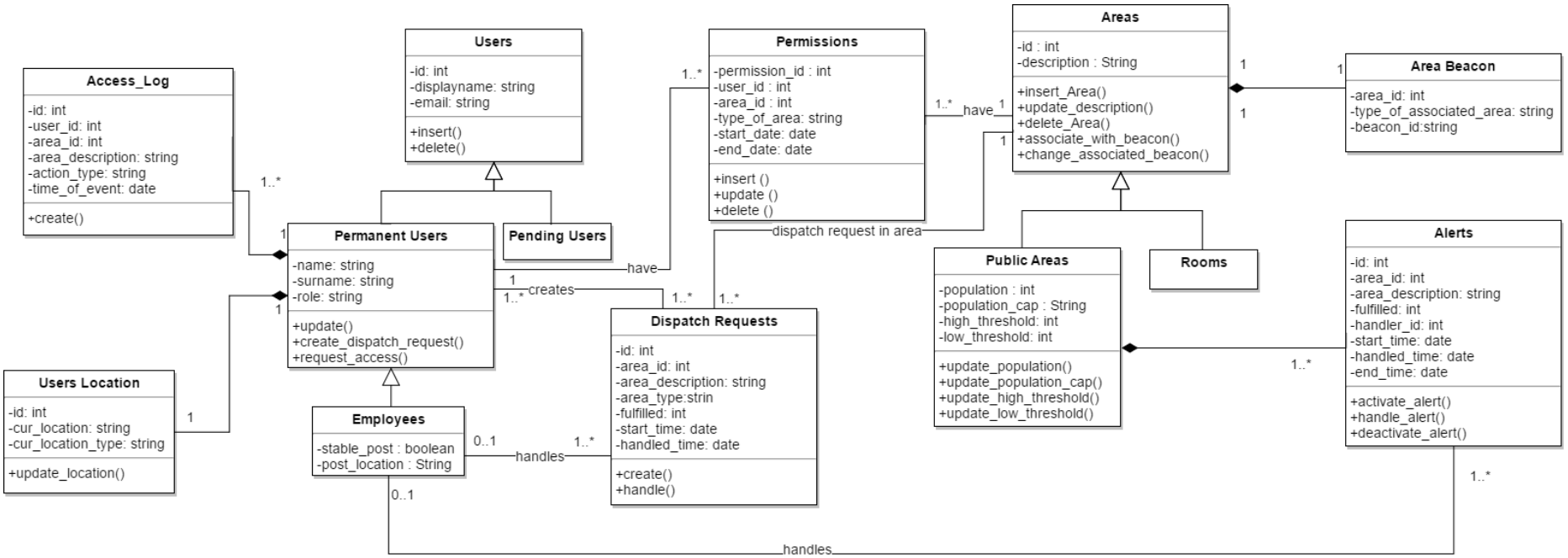
**Υπηρεσία Διαχείρισης Συνδέσεων (Connectivity Service)**

Υπηρεσία που δέχεται τα βασικά στοιχεία του χρήστη, όταν εισέρχεται στο ΣΕΠ, μέσω κινητού, ελέγχει τα στοιχεία του και παραχωρεί τα αντίστοιχα δικαιώματα, ανάλογα με το ρόλο του.

**Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Orion Context Broker Publish-Subscribe Service)**

Στόχος τις υπηρεσίας είναι η ενημέρωση του ΣΕΠ σχετικά με αλλαγές στον πληθυσμό των χώρων.

## 3.5 Διάγραμμα Κλάσεων (Class Diagram)




*Εικόνα 8 Διάγραμμα κλάσεων (Class Diagram)*

Το παραπάνω διάγραμμα αποτελεί ένα διάγραμμα κλάσεων που ικανοποιεί τους κανόνες τις UML[15]. Στόχος του συγκεκριμένου διαγράμματος αποτελεί η περιγραφή των διαφόρων κλάσεων από τις οποίες αποτελείται το σύστημα μας, καθώς και οι μεταξύ τους σχέσεις.

Πιο συγκεκριμένα:

**Κλάση Χρηστών (Users Class):** Αποτελεί μία υπερκλάση που περιέχει το σύνολο των χρηστών. Η κλάση έχει δύο υποκλάσεις ανάλογα με την κατηγορία του χρήστη. Σε αυτή την κλάση κρατούνται τα στοιχεία που είναι κοινά και για τις δύο κατηγορίες χρηστών. Το κάθε αντικείμενο χαρακτηρίζεται από το αναγνωριστικό του, το email του, καθώς και από το προβαλλόμενο όνομα. Η κλάση έχει δύο μεθόδους που αφορούν την εισαγωγή και διαγραφή χρηστών.

**Κλάση Μόνιμων Χρηστών (Permanent Users Class):** Περιέχει τα στοιχεία που έχουν μόνο οι μόνιμοι χρήστες, δηλαδή οι χρήστες που έχουν γίνει αποδεκτοί στο σύστημα από το διαχειριστή. Πιο συγκεκριμένα, το όνομα του χρήστη, το επίθετο του, καθώς και ο ρόλος του (υπάλληλος/επισκέπτης/μόνιμος κάτοικος). Τα υπόλοιπα στοιχεία κληρονομούνται από την υπερκλάση, συμπληρώνοντας έτσι όλα τα απαραίτητα στοιχεία. Η κλάση αυτή επιτρέπει την επεξεργασία των στοιχείων των αντικειμένων της, τη δημιουργία αιτήματος πρόσβασης, καθώς και τη δημιουργία αιτήματος για εξυπηρέτηση από το προσωπικό.

**Κλάση Προσωρινών Χρηστών (Pending Users Class):** Η κλάση αυτή αποτελείται από τους προσωρινούς χρήστες, χρήστες που ενώ έχουν κάνει αίτηση εγγραφής στο σύστημα η αίτηση τους δεν έχει γίνει ακόμα αποδεκτή. Το σύνολο των δεδομένων ενός προσωρινού χρήστη υπάρχει στην υπερκλάση, καθώς δεν έχει στοιχεία αποκλειστικά στην κατηγορία του.

**Κλάση Ιστορικού Προσβάσεων (Access Log Class):** Η κλάση αυτή σχετίζεται με την κλάση μόνιμων χρηστών. Κάθε μόνιμος χρήστης της εφαρμογής έχει και το αντίστοιχο ιστορικό πρόσβασης. Το κάθε αντικείμενο της κλάσης αυτής χαρακτηρίζεται από το αναγνωριστικό του, το αναγνωριστικό χρήστη στον οποίο αντιστοιχεί το αντικείμενο, το αναγνωριστικό του χώρου στον οποίο έγινε το αίτημα πρόσβασης, η περιγραφή της κατηγορίας του αιτήματος πρόσβασης (αν αποτελούσε είσοδο σε χώρο ή έξοδο από αυτόν), καθώς και η χρονική στιγμή στην οποία έγινε το συμβάν.

**Κλάση Τοποθεσίας Χρήστη (Users Location Class):** Στόχος της κλάσης είναι η παροχή πληροφορίας σχετικά με την τωρινή τοποθεσία του χρήστη. Πιο συγκεκριμένα έχουμε ένα μοναδικό αναγνωριστικό, το οποίο είναι ίδιο με το

[15] https://www.tutorialspoint.com/uml/uml_class_diagram.htm

αναγνωριστικό του χρήστη, τον κωδικό του χώρου στον οποίο βρίσκεται ο χρήστης και αν αυτός ο χώρος είναι κοινόχρηστος ή ιδιωτικός. Μέσω της μεθόδου ενημέρωσης τοποθεσίας, με κάθε αλλαγή θέσης του χρήστη ενημερώνουμε την κλάση.

**Κλάση Υπαλλήλων (Employees Class):** Αποτελεί υποκλάση των μόνιμων χρηστών. Πιο συγκεκριμένα σε αυτή την κλάση ανήκουν οι μόνιμοι χρήστες με το ρόλο υπαλλήλου. Στην κλάση αυτή κρατάμε πληροφορία για το αν ο υπάλληλος έχει σταθερή θέση στο κτίριο, και που βρίσκεται αυτή. Τα συγκεκριμένα δεδομένα δεν είναι ιδιαίτερα χρήσιμα στην παρούσα μορφή του ΣΕΠ, ωστόσο τοποθετήθηκαν για τυχόν μελλοντικές επεκτάσεις.

**Κλάση Χώρων (Areas Class):** Υπερκλάση των δύο κατηγοριών χώρων. Περιέχει τα κοινά στοιχεία που υπάρχουν στις δύο αυτές κατηγορίες, δηλαδή το αναγνωριστικό του χώρου και την περιγραφή του. Παράλληλα υπάρχουν οι μέθοδοι για προσθήκη καινούριου χώρου, ανανέωση της περιγραφής ενός χώρου, διαγραφή χώρου, συσχετισμός χώρου με αισθητήρα Beacon, καθώς και ανανέωση του συσχετισμένου αισθητήρα Beacon.

**Κλάση Κοινόχρηστων Χώρων (Public Areas Class):** Η κλάση αυτή περιέχει τους κοινόχρηστους χώρους, δηλαδή χώρους στους οποίους έχουν πρόσβαση πολλά άτομα και μας ενδιαφέρει να κρατάμε στατιστικά πρόσβασης, καθώς και να γνωρίζουμε πότε υπάρχει υψηλή κινητικότητα. Για το λόγο αυτό εισάγουμε παραμέτρους που ορίζουν τον τωρινό πληθυσμό του χώρου, τη χωρητικότητα του χώρου, ένα όριο πληρότητας που αν ξεπεραστεί θεωρούμε ότι υπάρχει ανάγκη αποστολής επιπλέον προσωπικού, καθώς και ένα όριο που ορίζει τη λήξη του περιστατικού. Η διαφοροποίηση γίνεται με σκοπό να μη δημιουργούνται άσκοπα πολλά αιτήματα με την αποχώρηση ή είσοδο ενός μόνο ατόμου. Αντίστοιχα με τις άλλες κλάσεις, υπάρχουν μέθοδοι για ανανέωση του πληθυσμού, της χωρητικότητας του χώρου, καθώς και για ανανέωση των δύο ορίων πληρότητας.

**Κλάση Ιδιωτικών Χώρων (Rooms Class):** Η συγκεκριμένη κλάση υπάρχει για να γίνεται διαχωρισμός μεταξύ των χώρων. Αποτελείται από ιδιωτικούς χώρους, κυρίως δωμάτια για τους οποίους δεν κρατάμε δεδομένα πρόσβασης, ή δεδομένα σχετικά με την πληρότητα τους.

**Κλάση Αισθητήρα χώρου (Area Beacons Class):** Σε αυτή την κλάση περιγράφουμε τον αισθητήρα που συσχετίζουμε με το χώρο. Πιο αναλυτικά, ορίζουμε το αναγνωριστικό του αισθητήρα, καθώς και το αναγνωριστικό και το είδος του συσχετιζόμενου χώρου.

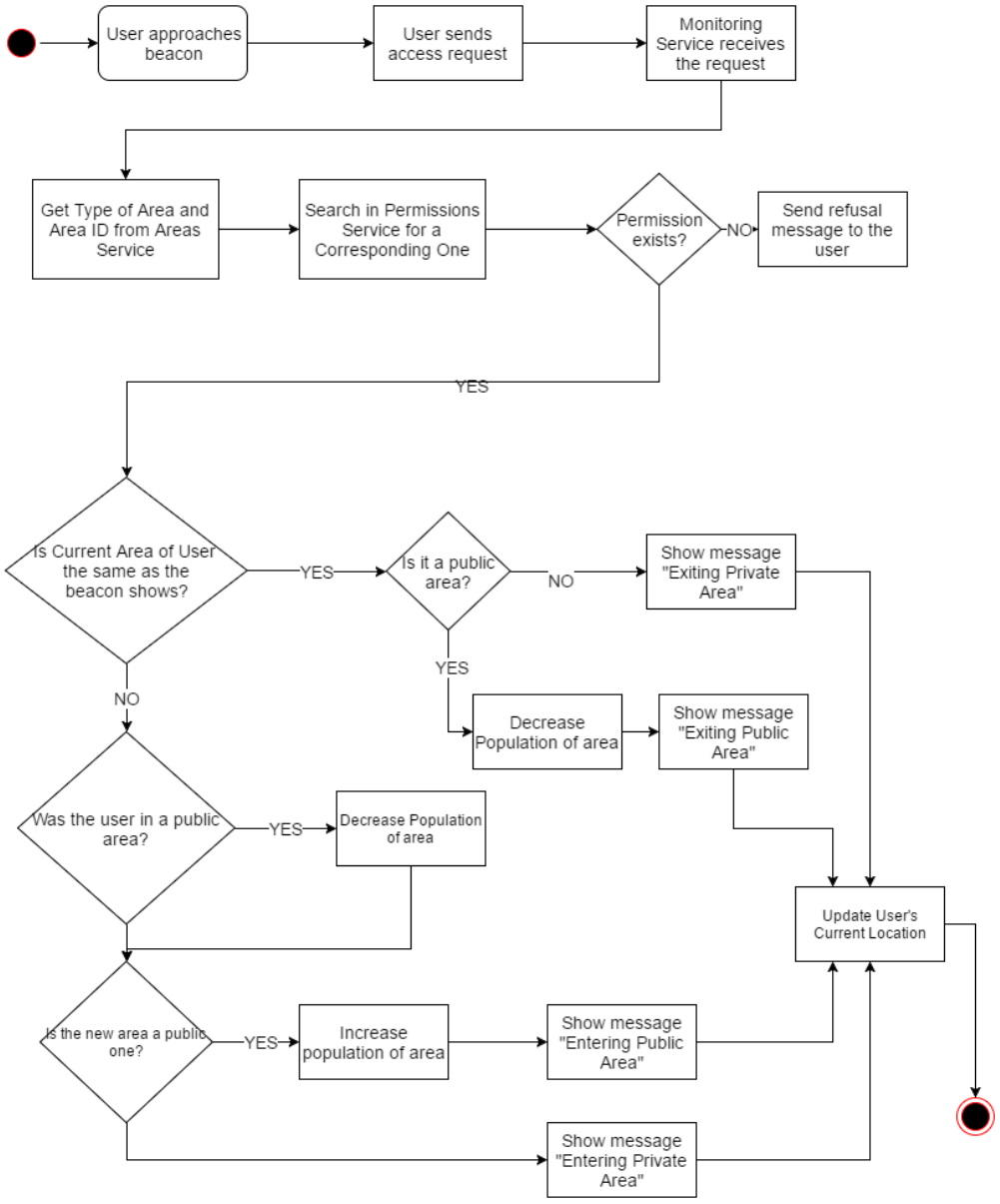
**Κλάση Δικαιωμάτων πρόσβασης (Permissions Class):** Προκειμένου να έχει πρόσβαση στους διάφορους χώρους της υποδομής ο χρήστης, είναι απαραίτητο να

κρατάμε μία λίστα με τα δικαιώματα πρόσβασής του. Το ρόλο αυτό αναλαμβάνει η συγκεκριμένη κλάση που αποτελείται από ένα αναγνωριστικό, ένα αναγνωριστικό μόνιμου χρήστη, το αναγνωριστικό και το είδους του χώρου, καθώς και τα όρια έναρξης και λήξης του δικαιώματος πρόσβασης. Επιπλέον υπάρχουν μέθοδοι για δημιουργία, επεξεργασία και διαγραφή ενός δικαιώματος πρόσβασης.

**Κλάση Αιτημάτων Αποστολής Προσωπικού (Dispatch Requests):** Πολλές φορές τυχαίνει ένας χρήστης να βρίσκεται σε ένα χώρο που δεν επαρκεί το προσωπικό για να τον εξυπηρετήσει άμεσα. Μέσω του ΣΕΠ μπορεί να στείλει αίτημα για να μεταβούν στο χώρο επιπλέον μέλη του προσωπικού. Η κλάση αυτή αποσκοπεί στην καταγραφή των στοιχείων που αποτελούν ένα συμβάν αυτού του είδους. Πιο συγκεκριμένα, κάθε αίτημα αποτελείται από ένα αναγνωριστικό, το αναγνωριστικό του χώρου από τον οποίο ο χρήστης έστειλε το αίτημα, την περιγραφή του χώρου, την κατηγορία του χώρου, το όνομα του υπαλλήλου σε περίπτωση που κάποιος ανέλαβε την εξυπηρέτηση του αιτήματος, καθώς και τη στιγμή δημιουργίας και εξυπηρέτησης του αιτήματος. Στην κλάση υπάρχουν και οι σχετικοί μέθοδοι για δημιουργία και χειρισμό του αιτήματος από μέλος του προσωπικού. Όπως γίνεται εμφανές η κλάση σχετίζεται άμεσα με την κλάση Χρηστών, την κλάση Χώρων καθώς και την Υπαλλήλων.

**Κλάση συμβάντων (Alerts Class):** Όπως αναφέραμε και προηγουμένως, όταν ο πληθυσμός ενός κοινόχρηστου χώρου ξεπεράσει κάποιο όριο, θεωρούμε ότι είναι χρήσιμο να σταλεί επιπλέον προσωπικό. Στην κλάση αυτή κρατάμε την απαραίτητη πληροφορία σχετικά με το γεγονός. Αναλυτικότερα, δημιουργούμε ένα αναγνωριστικό για το γεγονός και προσθέτουμε πληροφορία σχετικά με το αναγνωριστικό και την περιγραφή του χώρου στο οποίο πραγματοποιήθηκε, αν εκπληρώθηκε από μέλος του προσωπικού και το αναγνωριστικό του μέλους που ανέλαβε την επίλυσή του, τη χρονική στιγμή έναρξης, χειρισμού και λήξης του συμβάντος(όταν δηλαδή μειωθεί αισθητά ο πληθυσμός του χώρου). Αντίστοιχα υπάρχουν και οι σχετικοί μέθοδοι για δημιουργία, χειρισμό και λήξη του συμβάντος.

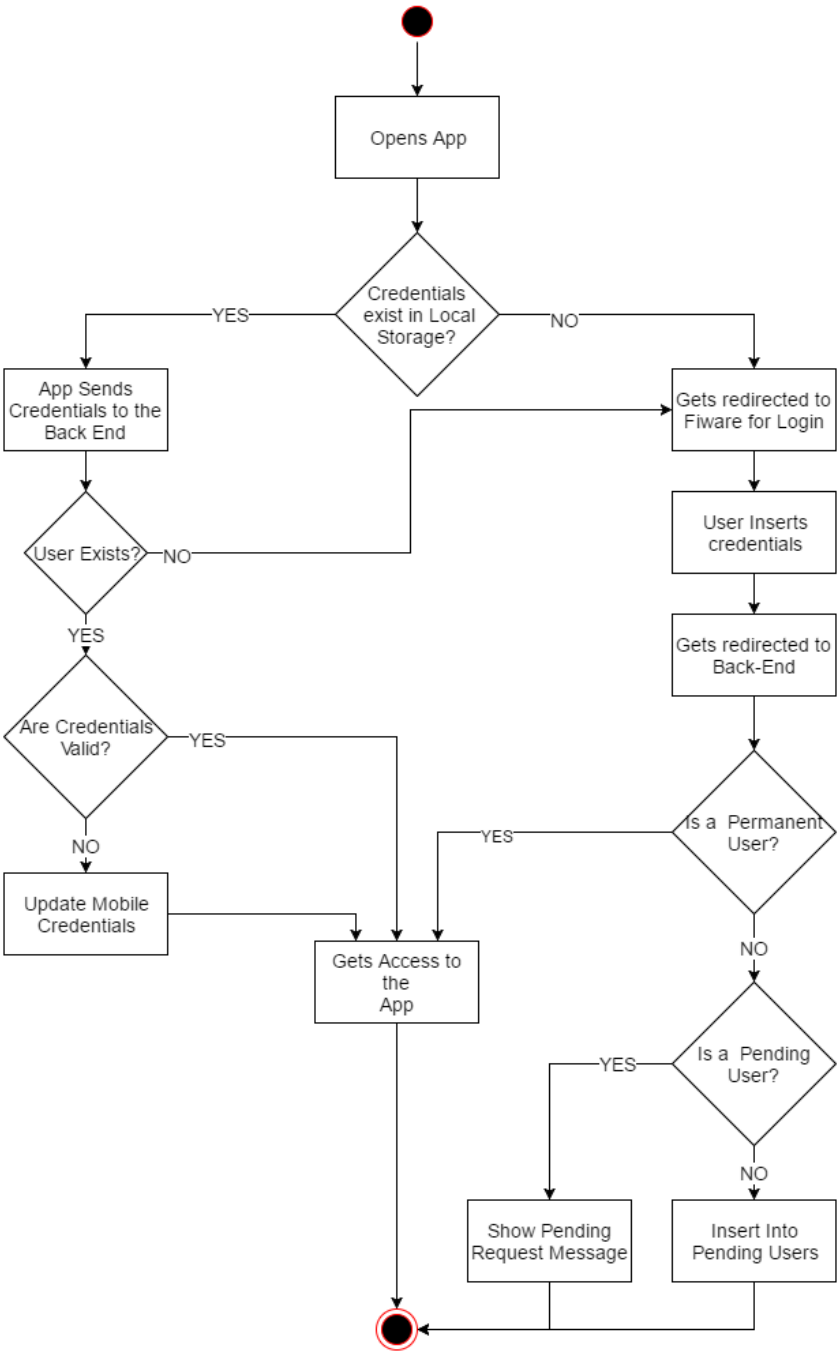
## 3.6 Διάγραμμα βασικών λειτουργιών της εφαρμογής (Activity Diagram)




*Εικόνα 9 Διάγραμμα διαδικασίας εισόδου σε χώρο*

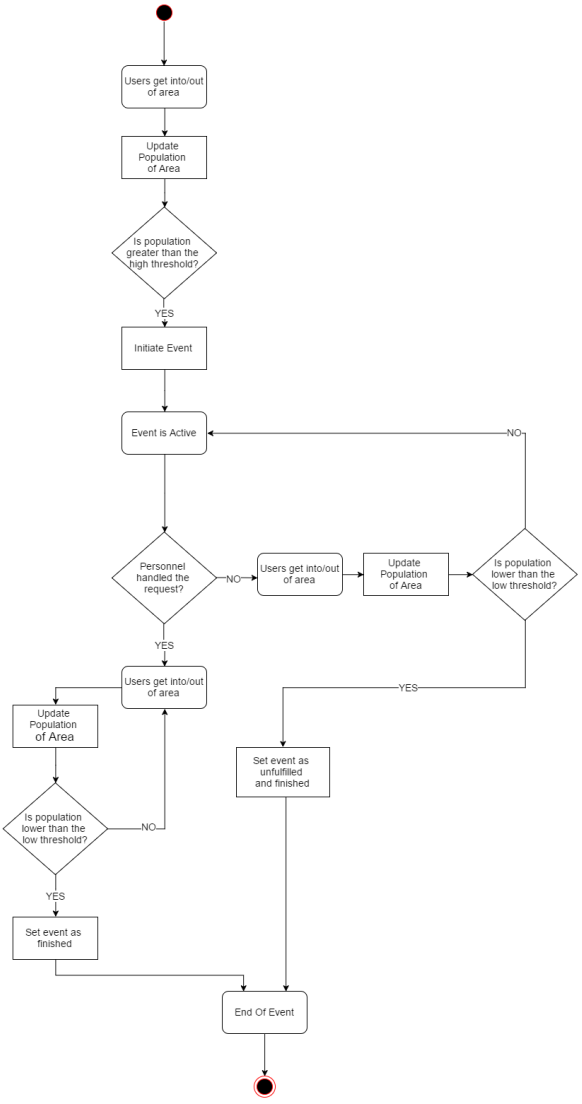
Στο παραπάνω διάγραμμα περιγράφουμε τη διαδικασία εισόδου/εξόδου ενός χρήστη από ένα χώρο. Αρχικά ο χρήστης αλληλεπιδρά με τον αισθητήρα Beacon που αντιστοιχεί στο χώρο. Μέσω του κινητού στέλνει στο ιδιωτικό νέφος τα στοιχεία του, καθώς και το αναγνωριστικό του αισθητήρα. Το αίτημα αυτό λαμβάνεται από την Υπηρεσία Ελέγχου Προσβάσεων. Στη συνέχεια, η συγκεκριμένη υπηρεσία θα ζητήσει από την Υπηρεσία Διαχείρισης Χώρων τα στοιχεία του χώρου, καθώς και τα από την Υπηρεσία Διαχείρισης Δικαιωμάτων τα δικαιώματα πρόσβασης που έχει ο χρήστης για το συγκεκριμένο χώρο. Σε περίπτωση που δεν

υπάρχει κάποιο έγκυρο δικαίωμα πρόσβασης, το αίτημα του χρήστη θα απορρίπτεται. Εναλλακτικά θα ελέγχουμε αν ο χώρος είναι ιδιωτικός ή κοινόχρηστος. Σε περίπτωση που ο χώρος στον οποίο ανταποκρίνεται ο αισθητήρας είναι ο ίδιος με την τοποθεσία στην οποία βρισκόταν ο χρήστης, θεωρούμε ότι ο χρήστης έκανε αίτηση εξόδου από το χώρο. Αν ο χώρος ήταν δημόσιος μειώνουμε τον πληθυσμό. Εναλλακτικά αν ο χρήστης βρισκόταν σε κοινόχρηστο χώρο και εισέρχεται σε διαφορετικό κοινόχρηστο χώρο, μειώνουμε τον πληθυσμό του πρώτου χώρου και αυξάνουμε τον πληθυσμό του δεύτερου χώρου. Αντίστοιχη διαδικασία ακολουθείται και για τις υπόλοιπες περιπτώσεις.




*Εικόνα 10 Διάγραμμα διαδικασίας εισόδου στην εφαρμογή μέσω κινητού*

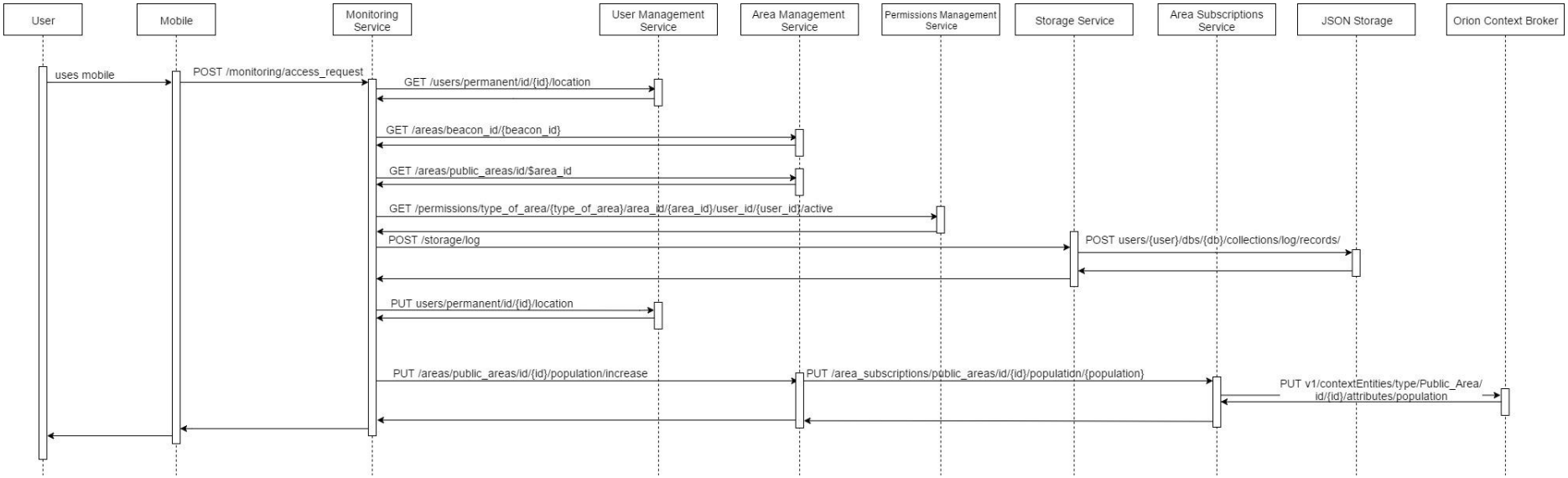
Στην Εικόνα 10 βλέπουμε τη διαδικασία που πραγματοποιείται για να εισέλθει ένας χρήστης στο ΣΕΠ μέσω κινητού. Ο χρήστης ανοίγει την εφαρμογή από το κινητό του. Σε περίπτωση που έχει συνδεθεί προηγουμένως , τότε στην εφαρμογή υπάρχουν αποθηκευμένα τα στοιχεία σύνδεσής του. Αποστέλλονται προς το νέφος για εξακρίβωση. Σε περίπτωση που είναι σωστά, τότε δίνεται δικαίωμα πρόσβασης στην εφαρμογή. Εναλλακτικά ενημερώνονται τα στοιχεία του χρήστη και ύστερα δίνουμε τα αντίστοιχα δικαιώματα που του αντιστοιχούν. Σε περίπτωση που δεν υπάρχουν αποθηκευμένα στοιχεία χρήστη, ή αυτά δεν αντιστοιχούν σε κάποιο χρήστη, τότε ο χρήστης στέλνεται μέσω της εφαρμογής στο Fiware για να συνδεθεί. Μετά τη σύνδεση του, στέλνονται στο κινητό τα στοιχεία του. Αν αποτελεί μόνιμο χρήστη, τότε έχει πρόσβαση στην εφαρμογή. Σε περίπτωση που αποτελεί προσωρινό χρήστη, ενημερώνεται πως δεν έχει εγκριθεί ακόμα το αίτημα του, ενώ αν δεν ανήκει σε κάποια από τις δύο κατηγορίες, ενημερώνεται με μήνυμα ότι στάλθηκε αίτημα για να εισέλθει στους χρήστες της εφαρμογής.




*Εικόνα 11 Διάγραμμα χειρισμού συμβάντος συναγερμού*

Στην Εικόνα 11 περιγράφουμε τη διαχείριση ενός συμβάντος συναγερμού. Ως συναγερμό, ορίζουμε την κατάσταση στην οποία ο πληθυσμός ενός χώρου ξεπερνάει κάποιο όριο, επομένως κρίνεται χρήσιμη η ειδοποίηση του προσωπικού με στόχο τη μεταφορά κάποιου μέλους του στο χώρο στον οποίο παρατηρείται αυξημένη κινητικότητα. Αυτό γίνεται με τον εξής τρόπο. Χρήστες εισέρχονται και αποχωρούν από το χώρο. Είσοδο στο χώρο θεωρούμε ότι έχουμε όποτε ο χρήστης αλληλεπιδρά με τον αντίστοιχο αισθητήρα Beacon και το αίτημα του γίνει αποδεκτό. Έξοδο έχουμε όταν ο χρήστης αλληλεπιδρά ξανά με κάποιον αισθητήρα. Σε περίπτωση που είναι ο ίδιος αισθητήρας, θεωρούμε ότι ο χρήστης επιθυμεί να αποχωρήσει από το χώρο, ενώ σε περίπτωση που είναι διαφορετικός αισθητήρας, θεωρούμε ότι ο χρήστης προσπαθεί να εισέλθει σε άλλη τοποθεσία, οπότε έχει αποχωρήσει από το χώρο. Ο πληθυσμός του χώρου ανανεώνεται διαρκώς. Όταν ξεπεράσει το όριο που έχουμε ορίσει μεταβαίνουμε στην κατάσταση συναγερμού. Το προσωπικό λαμβάνει σχετικό μήνυμα. Αν κάποιο μέλος του προσωπικού απαντήσει ότι αναλαμβάνει το συμβάν, τότε θεωρούμε ότι το συμβάν είναι υπό έλεγχο. Σε περίπτωση που δεν αναλάβει κάποιος το συμβάν, ο συναγερμός εξακολουθεί να παραμένει ενεργός μέχρι την ανάληψή του, ή μέχρι τη μείωση του πληθυσμού κάτω από ένα δεύτερο, χαμηλότερο, όριο που σηματοδοτεί τη λήξη του συναγερμού. Ορίζονται διαφορετικά όρια για έναρξη και λήξη του συμβάντος με σκοπό να αποφεύγεται η συνεχής αλλαγή κατάστασης.

## 3.7 Διάγραμμα ακολουθίας (Sequence Diagram)




*Εικόνα 12 Διάγραμμα ακολουθίας για είσοδο σε δημόσιο χώρο*

Στο παραπάνω διάγραμμα περιγράφουμε τη διαδικασία που γίνεται προκειμένου να εισέλθει ένας χρήστης σε ένα χώρο. Αυτή τη φορά μας ενδιαφέρει η επικοινωνία που γίνεται μεταξύ των διαφόρων υπηρεσιών. Συνοπτικά η διαδικασία γίνεται με τον εξής τρόπο.

1. Ο χρήστης στέλνει αίτημα στην Υπηρεσία Ελέγχου Προσβάσεων για είσοδο σε ένα χώρο με το αναγνωριστικό του και το αναγνωριστικό του αισθητήρα Beacon, που αντιστοιχεί στο χώρο. Η υπηρεσία κάνει διαδοχικά αιτήματα όπως φαίνεται στο σχήμα.
2. Από την Υπηρεσία Διαχείρισης Χρηστών ενημερωνόμαστε για την τοποθεσία του χρήστη, προκειμένου να γνωρίζουμε αν αναφερόμαστε σε είσοδο ή έξοδο από χώρο.
3. Στη συνέχεια, κάνοντας χρήστης της Υπηρεσίας Διαχείρισης Χώρων, μαθαίνουμε το αναγνωριστικό του χώρου στον οποίο αντιστοιχεί ο αισθητήρας Beacon.
4. Ενημερωνόμαστε μέσω της Υπηρεσίας Διαχείρισης Χώρων, για τον τωρινό πληθυσμό του χώρου, καθώς και για τη χωρητικότητα του.
5. Από την Υπηρεσία Διαχείρισης Δικαιωμάτων, ελέγχουμε για την ύπαρξη ή όχι άδειας για διεκπεραίωση της συγκεκριμένης πρόσβασης.
6. Η Υπηρεσία Ελέγχου Προσβάσεων αποφασίζει αν ο χρήστης έχει δικαίωμα πρόσβασης στο χώρο ή όχι. Το διάγραμμα περιγράφει την περίπτωση που ο χρήστης όντως έχει δικαίωμα.
7. Στέλνουμε αίτημα στην Υπηρεσία Αποθήκευσης Δεδομένων να σταλεί στο δημόσιο νέφος εγγραφή σχετικά με την είσοδο ή έξοδο του χρήστη.
8. Γίνεται η αποστολή προς το δημόσιο νέφος, στην Υπηρεσία JSON Storage.
9. Στην Υπηρεσία Διαχείρισης Χρηστών, ανανεώνουμε την τωρινή θέση του χρήστη στο χώρο.
10. Σε περίπτωση που η κίνηση αφορά κοινόχρηστο χώρο, ανανεώνουμε τον πληθυσμό του μέσω της Υπηρεσία Διαχείρισης Χώρων.
11. Η Υπηρεσία Διαχείρισης Χώρων στέλνει αντίστοιχα αίτημα για αλλαγή του πληθυσμού προς την Υπηρεσία Διαχείρισης Εγγραφών Χώρων.
12. Η Υπηρεσία Διαχείρισης Εγγραφών Χώρων στέλνει αυτό το αίτημα προς το Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Orion Context Broker Publish-Subscribe Service), που ελέγχει με βάση τους κανόνες του αν δημιουργείται κάποιο περιστατικό.

Η απόδοση της συγκεκριμένης διαδικασίας υπολογίζεται στην Ενότητα 5.1.

# Κεφάλαιο 4ο

## 4.1 Σύστημα Διεπαφής τελικού χρήστη (Front-End)

Με τον όρο Front-End εννοούμε το σύστημα το οποίο χρησιμοποιεί ο χρήστης με σκοπό να έχει πρόσβαση στο ΣΕΠ. Μπορούμε να ισχυριστούμε ότι αποτελεί το μέσο, το οποίο παρέχει στο χρήστη πρόσβαση, στις υπηρεσίες και δυνατότητες του ΣΕΠ.

Όπως αναφέραμε και προηγουμένως το ΣΕΠ παρέχει δύο είδη front-end. Το πρώτο αποτελεί ένα Web-Api προσβάσιμο μέσω φυλλομετρητή, ενώ το δεύτερο αποτελεί μία εφαρμογή για κινητό. Ο σχεδιασμός έγινε κάνοντας χρήση HTML,CSS,Javascript καθώς και AngularJS[16].

Αναλυτικότερα, η HTML(Hypertext Markup Language) αποτελεί την καθιερωμένη γλώσσα σήμανσης υπερκειμένου που χρησιμοποιείται για την κατασκευή ιστοσελίδων και διαδικτυακών εφαρμογών.

Η CSS( Cascading Style Sheet) είναι μία γλώσσα που χρησιμοποιείται επιπρόσθετα με HTML, με στόχο την παροχή επιπλέον επιλογών σχετικά με τη μορφοποίηση του κειμένου και άλλες επιλογές που αφορούν την τελική εμφάνιση μίας σελίδας που έχει δημιουργηθεί με HTML.

Η Javascript, παρέχει στην HTML, τη δυνατότητα δημιουργίας συναρτήσεων που εκτελούνται τοπικά στο σύστημα του τελικού χρήστη, ενώ παρέχει δυνατότητα αποστολής δεδομένων προς τον εξυπηρετητή ασύγχρονα.

Η AngularJS, αποτελεί επέκταση της Javascript. Στην πραγματικότητα είναι ένα framework με σκοπό την απλούστευση πολλών διαδικασιών, κατά την κατασκευή μιας ιστοσελίδας, ή διαδικτυακής εφαρμογής.

Το Front-End λειτουργεί με την εξής λογική. Ο χρήστης διαλέγει μία από τις διαθέσιμες επιλογές που παρέχει η εφαρμογή ή ο φυλλομετρητής ( αναλυτικότερα ποιες είναι οι διαθέσιμες επιλογές στις Ενότητες 4.1.1 και 4.1.2). Σε περίπτωση που η συγκεκριμένη δραστηριότητα απαιτεί δυναμικό σχεδιασμό της απάντησης, στέλνουμε, μέσω της υπηρεσίας http της AngularJS[17], ένα ασύγχρονο αίτημα με το αναγνωριστικό του χρήστη και το ρόλο του, με σκοπό να λάβουμε τα προσαρμοσμένα δεδομένα. Ο χρήστης λαμβάνει την προσαρμοσμένη απάντηση. Ανάλογα με τις επιλογές του, κάνουμε το αντίστοιχο http αίτημα προς το Back-End, και στη συνέχεια περιμένουμε την απάντηση στο αίτημα.

[16] https://angularjs.org/
[17] https://docs.angularjs.org/api/ng/service/$http

### 4.1.1. Εφαρμογή για κινητά

Η εφαρμογή αναπτύχθηκε κάνοντας χρήση του Cordova[18], ενός μετατροπέα που δέχεται ως είσοδο τον κώδικα σε μορφή HTML παρέα με την Javascript και CSS και τον μετατρέπει σε εκτελέσιμο κώδικα για Android ή iOS. Στα πλαίσια της εφαρμογής που αναπτύξαμε, δοκιμάσαμε τη μετατροπή σε Android. Μέσω αυτής της μετατροπής καταφέραμε να εισάγουμε τη βιβλιοθήκη που χρειάζεται για το BLE Scanner.

Η εφαρμογή για κινητά αποτελείται από τα εξής χαρακτηριστικά.

**Application Logic:** Αποτελεί το κύριο τμήμα της εφαρμογής μας, με στόχο την ενορχήστρωση των διαφόρων υπηρεσιών. Μέσω αυτού γίνεται η επιλογή δραστηριότητας που θα διαλέξει ο χρήστης, καθώς και η επικοινωνία μεταξύ των διαφόρων υπηρεσιών.

**BLE Scanner**: Μία υπηρεσία, η οποία επιτρέπει τον εντοπισμό BLE συσκευών σε κοντινή εμβέλεια. Η υπηρεσία κάνει χρήση της βιβλιοθήκης cordova-plugin-ble. ΄Ετσι μπορούμε να εντοπίσουμε τους αισθητήρες Beacon που χρησιμοποιούμε για την αναπαράσταση των διαφόρων χώρων.

**Local Storage:** Μονάδα αποθήκευσης, στην οποία τοποθετούνται τα στοιχεία σύνδεσης του χρήστη, με στόχο την αυτόματη είσοδό του, σε επόμενες συνδέσεις του.

**Connectivity Service:** Υπηρεσία που αναλαμβάνει τη διαδικασία πιστοποίησης χρήστη μέσω αποστολής των δεδομένων χρήστη σε κωδικοποίηση Base64 προς το ιδιωτικό νέφος.

**Notification Service:** Η συγκεκριμένη υπηρεσία αφορά κυρίως τους εργαζομένους της υποδομής που κάνουν χρήση της εφαρμογής, καθώς μέσω αυτής μπορούν να ενημερώνονται για τυχόν ενεργά συμβάντα. Αξιοποιεί τη δυνατότητα των Server-Sent Events (Ενότητα 4.2.3). Με τον τρόπο αυτό στέλνονται ανά τακτά διαστήματα τα ενεργά γεγονότα στη συσκευή του χρήστη.

### 4.1.2 Διεπαφή χρήστη για κινητά

Η διεπαφή χρήστη για κινητές συσκευές είναι ιδιαίτερα απλή. Αν ο χρήστης δεν έχει συνδεθεί ήδη στην εφαρμογή μεταφέρεται στη σελίδα σύνδεσης του Keyrock

[18] https://cordova.apache.org/

Identity Manager με σκοπό να πραγματοποιήσει σύνδεση όπως περιγράψαμε στην Ενότητα 2.5. Σε περίπτωση που ο χρήστης είναι ήδη συνδεδεμένος, παρουσιάζονται οι αντίστοιχες επιλογές, ανάλογα με την ιδιότητα του.

Σε περίπτωση που ο χρήστης είναι κάτοικος ή επισκέπτης:

1. My Permissions: Προβολή των δικαιωμάτων πρόσβασης του χρήστη. (Στην περίπτωση επισκέπτη, εμφανίζονται δικαιώματα πρόσβασης μόνο για κοινόχρηστους χώρους).
2. Area Statistics: Προβολή της περιγραφής και πληρότητας του κάθε κοινόχρηστου χώρου.
3. Request Access: Το σημαντικότερο τμήμα της εφαρμογής. Στο συγκεκριμένο τμήμα, ο χρήστης ενεργοποιεί το BLE Scanner ώστε να εντοπίσει τους διαθέσιμους αισθητήρες Beacon. Στη συνέχεια επιλέγει έναν αισθητήρα και αντίστοιχα ελέγχει αν έχει ή όχι δικαίωμα πρόσβασης στον αντίστοιχο χώρο.
4. Request Personnel: Έχοντας εισέλθει ο χρήστης σε κάποιο χώρο μέσω της διαδικασίας που περιγράψαμε προηγουμένως, στη συνέχεια, μέσω αυτής της επιλογής μπορεί να στείλει αίτημα μετάβασης προσωπικού στην τοποθεσία του. Το αίτημα αυτό παραμένει ενεργό για ένα μικρό χρονικό διάστημα. Είτε εξυπηρετηθεί είτε όχι, καταγράφεται στο ιστορικό δραστηριοτήτων της υποδομής.
5. My Log: Προβολή ατομικού ιστορικού προσβάσεων.
6. Logout: Αποσύνδεση από την εφαρμογή και επιστροφή στην αρχική σελίδα.
7. Exit: Έξοδος από την εφαρμογή, χωρίς να γίνει αποσύνδεση, ώστε την επόμενη φορά να μη ζητηθούν τα στοιχεία σύνδεσης από το χρήστη.

Στην περίπτωση που έχουμε εργαζόμενο, πέραν από τις προαναφερθέντες επιλογές, υπάρχει και η επιλογή Alert Management. Η συγκεκριμένη επιλογή εμφανίζει τους ενεργούς συναγερμούς λόγω υψηλής πληρότητας χώρου, καθώς και τις αιτήσεις για αποστολή επιπλέον προσωπικού. Ο εργαζόμενος μπορεί να διαλέξει ένα από αυτά τα συμβάντα με σκοπό να ενημερώσει το σύστημα ότι θα μεταβεί στην τοποθεσία για επίλυση του συμβάντος. Στη συνέχεια το συμβάν θα καταγραφεί ως εκπληρωμένο από τον εργαζόμενο που το ανέλαβε.

### 4.1.3 Διεπαφή χρήστη μέσω φυλλομετρητή (Web-Api)

Η διεπαφή χρήστη μέσω φυλλομετρητή προσφέρει όλες τις δυνατότητες που περιγράφηκαν στην προηγούμενη Ενότητα, εκτός από αυτές που απαιτούν αλληλεπίδραση με κάποιο αισθητήρα Beacon. Ωστόσο,μόνο μέσω του Web-Api έχει πρόσβαση στο ΣΕΠ ο διαχειριστής ιδιωτικού νέφους.

Αναλυτικότερα, ο διαχειριστής ιδιωτικού νέφους έχει τις παρακάτω επιλογές:

1. Area Management: Παρέχει δυνατότητες σχετικά με τη διαχείριση των χώρων.
    a. Δημιουργία καινούριου κοινόχρηστου χώρου.
    b. Επεξεργασία περιγραφής, χωρητικότητας, ορίων έναρξης και λήξης συναγερμού κοινόχρηστου χώρου.
    c. Διαγραφή κοινόχρηστου χώρου.
    d. Επεξεργασία συσχετισμένου αισθητήρα Beacon σε κοινόχρηστο χώρο.
    e. Δημιουργία καινούριου ιδιωτικού χώρου.
    f. Επεξεργασία περιγραφής ιδιωτικού χώρου.
    g. Επεξεργασία συσχετισμένου αισθητήρα Beacon σε ιδιωτικό χώρο.
    h. Διαγραφή ιδιωτικού χώρου.
2. User Management: Παρέχει δυνατότητες σχετικά με τη διαχείριση μόνιμων χρηστών.
    a. Επεξεργασία ονόματος,επωνύμου,ρόλου ενός μόνιμου χρήστη.
    b. Διαγραφή μόνιμου χρήστη.
3. Request Management: Παρέχει δυνατότητες σχετικά με τη διαχείριση προσωρινών χρηστών.
    a. Συμπλήρωση στοιχείων και αποδοχή αιτήματος.
    b. Απόρριψη αιτήματος.
4. Employee Management: Παρέχει δυνατότητες σχετικά με τη διαχείριση εργαζομένων.
    a. Ορισμός ή μη μόνιμου πόστου εργαζομένου.
    b. Προβολή τωρινής θέσης εργαζομένου.
5. Permission Management: Παρέχει δυνατότητες σχετικά με τη διαχείριση των δικαιωμάτων πρόσβασης.
    a. Εισαγωγή δικαιώματος πρόσβασης σε κοινόχρηστο χώρο.
    b. Επεξεργασία δικαιώματος πρόσβασης σε κοινόχρηστο χώρο.
    c. Διαγραφή δικαιώματος πρόσβασης σε κοινόχρηστο χώρο.
    d. Εισαγωγή δικαιώματος πρόσβασης σε ιδιωτικό χώρο (ανενεργό στην περίπτωση που ο χρήστης είναι επισκέπτης).
    e. Επεξεργασία δικαιώματος πρόσβασης σε ιδιωτικό χώρο(ανενεργό στην περίπτωση που ο χρήστης είναι επισκέπτης).
    f. Διαγραφή δικαιώματος πρόσβασης σε ιδιωτικό χώρο(ανενεργό στην περίπτωση που ο χρήστης είναι επισκέπτης).
6. Logs: Προβολή του ιστορικού προσβάσεων κάποιου χρήστη.
7. Area Statistics: Προβολή της περιγραφής και πληρότητας του κάθε κοινόχρηστου χώρου.

## 4.2. Σύστημα επεξεργασίας δεδομένων και παροχής υπηρεσιών προς το Front-End (Back-End)

Το Back-End του ΣΕΠ αποτελείται από δύο ή περισσότερα υπολογιστικά νέφη. Ένα ή περισσότερα είναι ιδιωτικά και το κάθε ένα αντιστοιχεί σε μία κτιριακή υποδομή, ενώ υπάρχει ένα δημόσιο νέφος που λαμβάνει δεδομένα από τα επιμέρους νέφη. Το κύριο μέρος της εργασίας μας, ασχολείται με το τμήμα του ιδιωτικού νέφους, το οποίο και θα αναλύσουμε παρακάτω. Το Back-End σχεδιάστηκε σε PHP με χρήση της MySQL[19] βάσης δεδομένων. Επιπλέον έγινε χρήση του Slim-PHP[20] ενός framework που διευκολύνει τη δημιουργία κώδικα για χειρισμό αιτημάτων που βασίζονται στη REST αρχιτεκτονική.

Πιο συγκεκριμένα :

Η PHP αποτελεί μία γλώσσα δυναμικού σχεδιασμού ιστοσελίδων. Με τον τρόπο αυτό, αναλύοντας το αίτημα του χρήστη, έχουμε τη δυνατότητα δημιουργίας εξειδικευμένης σελίδας, ανάλογα με τις απαιτήσεις που έθεσε στο αίτημα του.

Η MySQL αποτελεί μία σχεσιακή βάση δεδομένων. (Αναλυτικότερα στην επόμενη Ενότητα)

Το Slim-PHP αποτελεί ένα micro-framework που βοηθάει στο σχεδιασμό δυναμικών ιστοσελίδων. Μέσω του Slim-PHP είναι ευκολότερη η διαχείριση των αιτημάτων REST.

### 4.2.1 Μονάδα αποθήκευσης του Back-End

Για να αποθηκεύσουμε τα δεδομένα που παράγονται από το Back-End κάναμε χρήση μίας MySQL βάσης. Η επιλογή της MySQL δεν έγινε τυχαία, καθώς είχαμε εγγραφές με σταθερά πεδία δεδομένων, ενώ ιδιαίτερα σημαντική ήταν και η δυνατότητα που μας παρέχει η συγκεκριμένη βάση δεδομένων για την υποβολή σύνθετων ερωτήσεων.

Τα δεδομένα που παράγονται από τη χρήση του ΣΕΠ αποστέλλονται προς το δημόσιο νέφος, ώστε να γίνει ανάλυση πληροφορίας που υπάρχει και δε φαίνεται σε πρώτη ανάγνωση με σκοπό να βελτιώσουμε τη χρήση του συστήματος και γενικά να πάρουμε καλύτερες επιχειρηματικές αποφάσεις. Αναλυτικότερα τα δεδομένα σχετίζονται με συμβάντα υψηλής κινητικότητας σε χώρους, καθώς και με αιτήσεις

[19] https://www.mysql.com/
[20] https://www.slimframework.com/

χρηστών για αποστολή προσωπικού και ο χρόνος που χρειάστηκε για την εξυπηρέτηση τους. Η αποστολή των δεδομένων προς το δημόσιο νέφος γίνεται σε μορφή JSON.

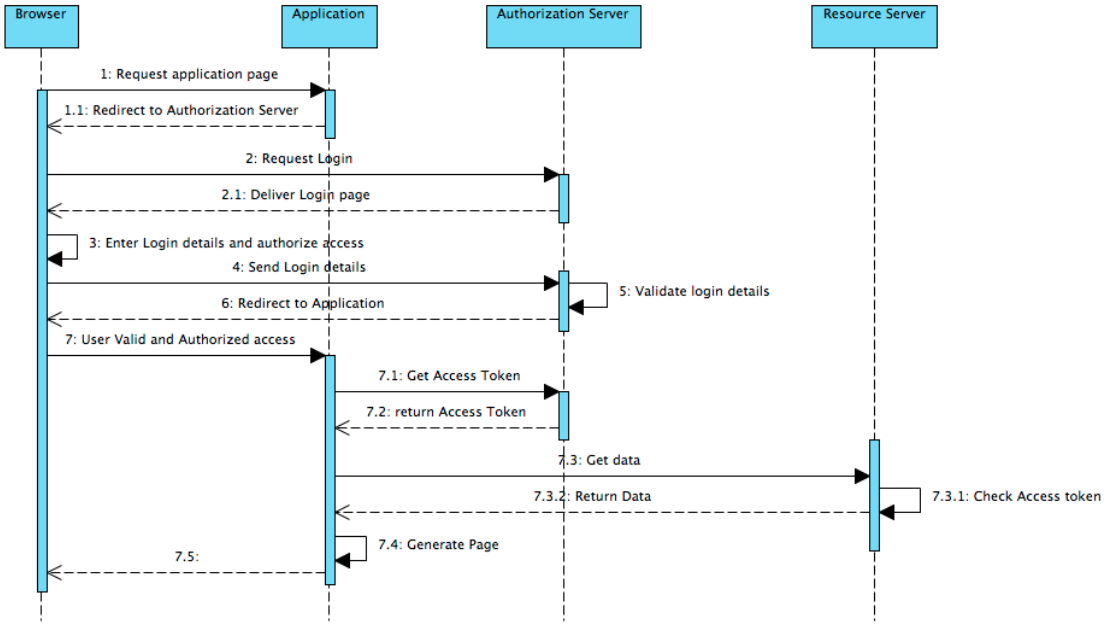
### 4.2.2 Υπηρεσίες Υπολογιστικού Νέφους (Back-End)

Στη συγκεκριμένη Ενότητα θα αναφερθούμε στις υπηρεσίες που χρησιμοποιήσαμε, καθώς και σε αυτές που δημιουργήσαμε, με σκοπό τη δημιουργία του ΣΕΠ.

#### 4.2.2.1 Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών Keyrock (Keyrock Identity Manager Service)

Η συγκεκριμένη υπηρεσία παρέχεται από το Fiware. Ο Keyrock Identity Manager είναι μία υπηρεσία ταυτοποίησης χρηστών που βασίζεται στο πρωτόκολλο Oauth2. Πιο συγκεκριμένα το OAuth2 λειτουργεί με την εξής λογική (η Εικόνα 13 περιγράφει σχηματικά τον τρόπο λειτουργίας).

1. Ο χρήστης εισέρχεται στην κεντρική σελίδα της εφαρμογής. Η εφαρμογή τον ανακατευθύνει στον εξυπηρετητή ταυτοποίησης, στην περίπτωση μας στο Keyrock.
2. Ο φυλλομετρητής του χρήστη, ζητάει από το Fiware τη σελίδα σύνδεσης χρήστη.
3. Ο χρήστης εισάγει τα στοιχεία του.
4. Γίνεται αποστολή των στοιχείων του χρήστη στο Keyrock.
5. Εξακριβώνονται τα στοιχεία του χρήστη από το Keyrock. Στο στάδιο αυτό έχει ολοκληρωθεί το Authentication (εξακρίβωση) του χρήστη. Αν είναι η πρώτη φορά που συνδέεται στην εφαρμογή, ο χρήστης έχει την επιλογή να επιτρέψει στην εφαρμογή να έχει πρόσβαση σε κάποια από τα στοιχεία του. Η διαδικασία αυτή είναι γνωστή ως Authorization (εξουσιοδότηση).
6. Σε περίπτωση που ο χρήστης δεχτεί να εξουσιοδοτήσει την εφαρμογή, γίνεται επιστροφή σε αυτή, παρέα με ένα κωδικό εξουσιοδότησης (authorization code).
7. Δίνεται πρόσβαση στην εφαρμογή σε περίπτωση έγκυρης ταυτοποίησης, ενώ λαμβάνεται μέσω του κωδικού εξουσιοδότησης του χρήστη ένα Access Token, μέσω του οποίου η εφαρμογή έχει πρόσβαση στα στοιχεία του χρήστη που παρέχονται από το Keyrock.




*Εικόνα 13 Διαδικασία σύνδεσης μέσω του πρωτοκόλλου OAuth2*[21]

Μετά την ολοκλήρωση της διαδικασίας, ανακτούμε τα υπόλοιπα στοιχεία του χρήστη μέσω της Υπηρεσίας Διαχείρισης Χρηστών και δημιουργούμε μία συνεδρία (session) με το αναγνωριστικό του χρήστη και το ρόλο του, με σκοπό να γνωρίζουμε τα δικαιώματα χρήσης των Υπηρεσιών που έχει ο κάθε χρήστης. Σε περίπτωση που ο χρήστης δεν έχει στοιχεία στην Υπηρεσία Διαχείρισης Χρηστών, θεωρούμε ότι αποτελεί καινούριο χρήστη, και τον ενημερώνουμε ότι το αίτημα εγγραφής του έχει σταλεί. Σε αυτή την περίπτωση δε δημιουργείται συνεδρία.

Το REST της υπηρεσίας περιγράφεται στον παρακάτω πίνακα (Πίνακας 1)

*Πίνακας 1 REST Ταυτοποίησης και Εξουσιοδότησης Χρηστών Keyrock*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| GET | http://account.lab.fiware.org/oauth2/authorize?response_type=code&client_id=bbe6a9794a47462aa3295bc1cb5b44b9&state=xyz&redirect_uri=http://147.27.60.151/login.php | - | Μετάβαση στη σελίδα του Keyrock για εξακρίβωση χρήστη. Το client_id αποτελεί αναγνωριστικό της εφαρμογής μας. Το redirect_uri ορίζει τη σελίδα που θα μεταφερθεί ο χρήστης μετά τη σύνδεσή του. |
| GET | https://account.lab.fiware.org/oauth2/token | HEADERS:<br>'Content-Type: application/x-www-form-urlencoded',<br>'Authorization: Basic '.base64_encode("client_id:c | Αφού ο χρήστης αποδεχτεί να παραχωρήσει τα στοιχεία του στην εφαρμογή, ανακατευθύνεται στην εφαρμογή.<br>Μέσω αυτού του αιτήματος, |

[21] https://fhirblog.files.wordpress.com/2014/06/oauth2sequencediagram.png

| | | | |
|---|---|---|---|
| | | lient_secret")<br>URL-encoded:<br>grant_type=authorization_code&code="{code}"&redirect_uri="{redirect_uri} | η εφαρμογή στέλνει τον κωδικό εξουσιοδότησης που έλαβε από την εκτέλεση της προηγούμενης ενέργειας και τα στοιχεία της προς το Keyrock, με στόχο να επιστραφεί το Access Token. |
| GET | https://account.lab.fiware.org/user?access_token=$access_token | - | Το Access Token είναι ένα αναγνωριστικό που μας επιτρέπει να έχουμε πρόσβαση στα στοιχεία του χρήστη που βρίσκονται στο Keyrock, στέλνοντας αυτό το αίτημα. |

#### 4.2.2.2 Υπηρεσία Διαχείρισης Χρηστών (User Management Service)

Κύριος στόχος της υπηρεσίας αυτής είναι η δημιουργία, η επεξεργασία, καθώς και η διαγραφή των χρηστών. Κατά κύριο λόγο στην υπηρεσία αυτή έχει πρόσβαση ο διαχειριστής του συστήματος. Η βασική αρχή λειτουργίας της υπηρεσίας έχει ως εξής. Υπάρχουν δύο κατηγορίες χρηστών, οι μόνιμοι χρήστες και οι προσωρινοί χρήστες (Permanent Users και Pending Users). Όταν ένας χρήστης εισέρχεται για πρώτη φορά στην εφαρμογή, μέσω του Keyrock, εισάγεται στη βάση δεδομένων, στους προσωρινούς χρήστες με στοιχεία χρήστη το email που παρέχεται από τον Keyrock Identity Manager. Ο χειριστής του συστήματος, μέσω της υπηρεσίας, μπορεί να δει το αίτημα που δημιουργήθηκε, να το αποδεχτεί ή να το απορρίψει. Σε περίπτωση που αποδεχτεί το αίτημα, ο χρήστης θα διαγράφεται από τους προσωρινούς χρήστες και θα εισάγεται στους μόνιμους χρήστες, αφού συμπληρωθεί το ονοματεπώνυμο και ο ρόλος του χρήστη από το διαχειριστή. Τα στοιχεία αυτά είναι διαθέσιμα προς επεξεργασία στο διαχειριστή του συστήματος. Με την αλλαγή του ρόλου ενός χρήστη αντίστοιχα παραχωρούνται ή αφαιρούνται δικαιώματα. Ο διαχειριστής του συστήματος, καθορίζει αν ένας χρήστης είναι επισκέπτης, κάτοικος ή εργαζόμενος. Στην περίπτωση εργαζομένου, η υπηρεσία αυτή μας επιτρέπει, να βλέπουμε την τρέχουσα θέση του στο οικοδομικό συγκρότημα. Η υπηρεσία αυτή, κρατάει ιστορικό προσβάσεων του κάθε χρήστη. Το ιστορικό αυτό είναι διαθέσιμο προς ανάγνωση είτε από το διαχειριστή του ιδιωτικού νέφους είτε από τον ίδιο το χρήστη.

Επιπλέον η υπηρεσία αυτή σχετίζεται άμεσα με την Υπηρεσία Διαχείρισης Δικαιωμάτων, καθώς η Υπηρεσία Διαχείρισης Δικαιωμάτων επικοινωνεί μέσω REST με την Υπηρεσία Διαχείρισης Χρηστών προκειμένου να προβάλει μία λίστα με τους χρήστες για τους οποίους μπορούμε να δημιουργήσουμε εγγραφές για παροχή

δικαιωμάτων πρόσβασης. Αντίστοιχα σε περίπτωση διαγραφής κάποιου χρήστη, η Υπηρεσία Διαχείρισης Χρηστών ειδοποιεί την Υπηρεσία Διαχείρισης Δικαιωμάτων, ώστε να γίνει η διαγραφή των δικαιωμάτων που αφορούν το συγκεκριμένο χρήστη.

Στον Πίνακα 2 φαίνονται παρουσιάζεται η REST αρχιτεκτονική της υπηρεσίας.

*Πίνακας 2 REST Διαχείρισης Χρηστών*

| **Μέθοδος** | **URL Μεθόδου** | **Σώμα αιτήματος** | **Περιγραφή μεθόδου** |
|---|---|---|---|
| GET | /users/permanent | - | Ανάκτηση όλων των μονίμων χρηστών. |
| GET | /users/permanent/employees | - | Ανάκτηση όλων των μονίμων χρηστών με την ιδιότητα υπαλλήλου. |
| GET | /users/permanent/id/{id} | - | Ανάκτηση ενός συγκεκριμένου μόνιμου χρήστη. |
| GET | /users/permanent/email/{email} | - | Ανάκτηση ενός μόνιμου χρήστη με βάση το email του. |
| POST | /users/permanent | JSON-encoded:<br>displayname:string<br>email:string<br>name:string<br>surname:string<br>role:string | Δημιουργία μόνιμου χρήστη. |
| PUT | /users/permanent | URL-encoded:<br>id:string<br>name:string<br>surname:string<br>role:string | Ανανέωση μόνιμου χρήστη. |
| PUT | users/employees/employee_id/{id} | URL-encoded:<br>stable_post:string<br>stable_post_location:integer<br>status:string | Ανανέωση στοιχείων υπαλλήλου. |
| DELETE | /users/permanent/id/{id} | - | Διαγραφή μόνιμου χρήστη. |
| GET | /users/pending | - | Ανάκτηση όλων των προσωρινών χρηστών. |
| GET | /users/pending/email/{email} | - | Ανάκτηση ενός προσωρινού χρήστη με βάση το email του. |
| GET | /users/pending/id/{id} | - | Ανάκτηση ενός προσωρινού χρήστη με βάση το ID του. |
| POST | /users/pending | JSON-encoded:<br>displayname:string<br>email:string | Δημιουργία ενός προσωρινού χρήστη. |
| DELETE | /users/pending/id/{id} | - | Διαγραφή ενός προσωρινού χρήστη. |
| GET | /users/permanent/id/{id}/log | - | Ανάκτηση του ιστορικού ενός μόνιμου χρήστη. |

| | | | |
|---|---|---|---|
| GET | /users/permanent/id/{id}/location | - | Ανάκτηση στοιχείων σχετικά με την τοποθεσία ενός μόνιμου χρήστη. |
| PUT | /users/permanent/id/{id}/location | URL-encoded:id:integer cur_location:integer cur_location_type:string | Ανανέωση στοιχείων σχετικά με την τοποθεσία ενός μόνιμου χρήστη. |

### 4.2.2.3 Υπηρεσία Διαχείρισης Χώρων (Area Management Service)

Η συγκεκριμένη υπηρεσία, έχει ως στόχο την εύκολη δημιουργία και διαχείριση χώρων. Οι διάφοροι χώροι χωρίζονται σε δύο κατηγορίες, σε κοινόχρηστους και ιδιωτικούς (Public Areas και Rooms). Ο διαχωρισμός αυτός έγινε καθώς θέλουμε να κρατάμε στοιχεία σχετικά με τις προσβάσεις στους κοινόχρηστους χώρους, καθώς και να γνωρίζουμε ανά πάσα στιγμή την πληρότητα των συγκεκριμένων χώρων. Ωστόσο για λόγους προστασίας προσωπικών δεδομένων, θέλουμε να μη γίνονται οι παραπάνω διαδικασίες στους ιδιωτικούς χώρους.

Ο διαχειριστής ιδιωτικού νέφους είναι αυτός που έχει πρόσβαση στη συγκεκριμένη υπηρεσία. Μέσω αυτής μπορεί να εισάγει έναν καινούριο χώρο, δίνοντας του μία περιγραφή, καθώς και συσχετίζοντάς τον με έναν αισθητήρα Beacon. Ο συσχετισμός γίνεται καθώς η αλληλεπίδραση για είσοδο σε κάθε χώρο γίνεται μέσω του αισθητήρα. Τα στοιχεία αυτά θα είναι διαθέσιμα προς επεξεργασία σε περίπτωση που κριθεί απαραίτητο κάτι τέτοιο. Επιπλέον για τους κοινόχρηστους χώρους, ορίζουμε μία μέγιστη χωρητικότητα , καθώς και ένα υψηλό και χαμηλό όριο, που οριοθετούν τις ποσότητες που όταν ξεπεραστούν, θα καλείται, ή θα σταματάει να καλείται προσωπικό για την εξυπηρέτηση των χρηστών. Στον παρακάτω πίνακα φαίνεται η αρχιτεκτονική REST της υπηρεσίας.

*Πίνακας 3 REST Διαχείρισης Χώρων*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| GET | /areas/public_areas | - | Ανάκτηση όλων των κοινόχρηστων χώρων. |
| GET | /areas/public_areas/id/{id} | - | Ανάκτηση ενός κοινόχρηστου χώρου. |
| POST | /areas/public_areas | JSON-encoded: description:string population-cap:integer low-threshold:integer high-threshold:integer | Δημιουργία ενός κοινόχρηστου χώρου. |
| PUT | /areas/public_areas | URL-encoded: | Ανανέωση ενός |

| | | | |
|---|---|---|---|
| | | description:string<br>population-cap:integer<br>low-threshold:integer<br>high-threshold:integer | κοινόχρηστου χώρου. |
| DELETE | /areas/public_areas/id/{id} | - | Διαγραφή ενός κοινόχρηστου χώρου. |
| GET | /areas/rooms | - | Ανάκτηση όλων των ιδιωτικών χώρων. |
| GET | /areas/rooms/id/{id} | - | Ανάκτηση ενός συγκεκριμένου ιδιωτικού χώρου. |
| POST | /areas/rooms | JSON-encoded:<br>description:string | Δημιουργία ιδιωτικού χώρου. |
| PUT | /areas/rooms/id/{id}/description/{description} | - | Ανανέωση περιγραφής ενός ιδιωτικού χώρου. |
| DELETE | /areas/rooms/id/{id} | - | Διαγραφή ενός ιδιωτικού χώρου. |
| GET | /areas/public_areas/id/{id}/beacon | - | Ανάκτηση στοιχείων του αισθητήρα beacon ενός κοινόχρηστου χώρου. |
| GET | /areas/beacon_id/{beacon_id} | - | Αναζήτηση κοινόχρηστου χώρου με βάση το αναγνωριστικό του αισθητήρα Beacon. |
| PUT | /areas/public_areas/id/{id}/population/{operation} | - | Αυξομείωση του πληθυσμού ενός κοινόχρηστου χώρου. |
| GET | /areas/rooms/id/{id}/beacon | - | Αναζήτηση ιδιωτικού χώρου με βάση το ID του αισθητήρα Beacon. |
| PUT | /areas/public_areas/id/{area_id}/beacon/{beacon_id} | - | Ανανέωση του αισθητήρα Beacon για κοινόχρηστο χώρο. |
| PUT | /areas/rooms/id/{area_id}/beacon/{beacon_id} | - | Ανανέωση του αισθητήρα Beacon για ιδιωτικό χώρο. |

### 4.2.2.4 Υπηρεσία Διαχείρισης Συνδρομών σε Χώρους (Area Subscriptions Service)

Όταν δημιουργούμε ένα καινούριο κοινόχρηστο χώρο δημιουργείται αντίστοιχα και μία οντότητα στο Context Broker(αποτελεί υπηρεσία Publish/Subscribe, αναλυτικότερα στην Ενότητα 4.2.2.10). Η οντότητα αυτή περιέχει στοιχεία σχετικά με το όνομα του χώρου, τη μέγιστη χωρητικότητα του, καθώς και ανώτερα και κατώτερα όρια που όταν ξεπεραστούν καλούμε(ή σταματάμε να καλούμε) επιπλέον προσωπικό. Αντίστοιχα όταν υπάρχει ενημέρωση η διαγραφή ενός δημόσιου χώρου, μέσω της Υπηρεσίας Διαχείρισης Συνδρομών σε χώρους, ενημερώνεται και η οντότητα. Στόχος της συγκεκριμένης υπηρεσίας, είναι η αποστολή των παραπάνω αιτημάτων προς το Context Broker σε συμβατή με αυτό μορφή.

*Πίνακας 4 REST Διαχείρισης Συνδρομών σε χώρους*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| GET | /area_subscriptions | - | Επιστρέφει όλες τις οντότητες |
| POST | /area_subscriptions | JSON-encoded:<br>id:integer<br>description:string<br>population_cap:integer<br>high_threshold:integer<br>low_threshold:integer | Δημιουργία καινούριας οντότητας. |
| PUT | /area_subscriptions/public_areas/<br>id/{id}/description/{description} | - | Ανανέωση της περιγραφής μιας οντότητας. |
| PUT | /area_subscriptions/public_areas/<br>id/{id}/population_cap/{population_cap} | - | Ανανέωση του μέγιστου πληθυσμού μιας οντότητας. |
| PUT | /area_subscriptions/public_areas/<br>id/{id}/population/{population} | - | Ανανέωση του πληθυσμού της οντότητας. |
| PUT | /area_subscriptions/public_areas/<br>id/{id}/high_threshold/{high_threshold} | - | Ανανέωση ορίου έναρξης συναγερμού. |
| PUT | /area_subscriptions/public_areas/<br>id/{id}/low_threshold/{low_threshold} | - | Ανανέωση ορίου λήξης συναγερμού. |
| DELETE | /area_subscriptions/id/{id} | - | Διαγραφή μιας οντότητας. |

### 4.2.2.5 Υπηρεσία Διαχείρισης Δικαιωμάτων (Permission Management Service)

Η συγκεκριμένη υπηρεσία χρησιμοποιείται κατά κανόνα από το διαχειριστή του συστήματος. Ο διαχειριστής αφού διαλέξει ένα χρήστη, έχει τη δυνατότητα να προβάλλει, να επεξεργάζεται και να διαγράφει τα δικαιώματα πρόσβασης του χρήστη στους διάφορους χώρους της υποδομής. Σε περίπτωση που ο χρήστης αποτελεί επισκέπτη, τότε η υπηρεσία περιορίζει τα δικαιώματα πρόσβασης μόνο σε κοινόχρηστους χώρους. Επιπλέον, η υπηρεσία ελέγχει τις υπάρχουσες εγγραφές όταν εισέρχεται μία καινούρια με στόχο τη συνένωση, ή τη διαγραφή περιττών εγγραφών. Οι εγγραφές χαρακτηρίζονται από το χρήστη στον οποίο αναφέρονται, το χώρο τον οποίο αφορούν,το είδος του(ιδιωτικός ή κοινόχρηστος) καθώς και την ημερομηνία έναρξης και λήξης του δικαιώματος πρόσβασης. Η υπηρεσία επικοινωνεί με την Υπηρεσία Διαχείρισης Χώρων και την Υπηρεσία Διαχείρισης Χρηστών καθώς σε περίπτωση διαγραφής κάποιου χώρου ή χρήστη, μεταβαίνουμε και σε διαγραφή των αντίστοιχων δικαιωμάτων πρόσβασης.

*Πίνακας 5 REST Διαχείρισης Δικαιωμάτων Πρόσβασης*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| GET | /permissions | - | Ανάκτηση όλων των δικαιωμάτων πρόσβασης. |
| GET | /permissions/public_areas /user_id/{user_id} | - | Ανάκτηση όλων των δικαιωμάτων πρόσβασης σε κοινόχρηστους χώρους ενός χρήστη. |
| GET | /permissions/rooms/user_ id/{user_id} | - | Ανάκτηση όλων των δικαιωμάτων πρόσβασης σε ιδιωτικούς χώρους ενός χρήστη. |
| GET | /permissions/type_of_area /{type_of_area}/area_id/{ area_id}/user_id/{user_id} | - | Ανάκτηση όλων των δικαιωμάτων πρόσβασης ενός χρήστη, για ένα χώρο. |
| GET | /permissions/type_of_area /{type_of_area}/area_id/{ area_id}/user_id/{user_id} /active | - | Ανάκτηση όλων των ενεργών δικαιωμάτων πρόσβασης ενός χρήστη, για ένα χώρο. |
| POST | /permissions | JSON-encoded: area_id:integer user_id:integer start_date:date end_date:date type_of_area:string | Δημιουργία καινούριου δικαιώματος πρόσβασης. |
| DELETE | /permissions | URL-encoded: area_id:integer | Διαγραφή ενός δικαιώματος πρόσβασης. |

| | | | |
|---|---|---|---|
| | | user_id:integer<br>start_date:date<br>end_date:date<br>type_of_area:string | |

### 4.2.2.6 Υπηρεσία Ελέγχου Προσβάσεων (Monitoring Service)

Αποτελεί κεντρικό τμήμα του ΣΕΠ. Μέσω της συγκεκριμένης υπηρεσίας αποφασίζεται η παροχή πρόσβασης ή όχι προς ένα χρήστη. Για το σκοπό αυτό η συγκεκριμένη υπηρεσία επικοινωνεί και με τις υπόλοιπες ώστε να αποκτήσει τα στοιχεία των διαφόρων χρηστών, χώρων και δικαιωμάτων πρόσβασης (η διαδικασία περιγράφεται στην Ενότητα 3.6 καθώς και στο πρώτο διάγραμμα της Ενότητας 3.5). Ως είσοδο για τον έλεγχο παροχής πρόσβασης δέχεται το αναγνωριστικό του χρήστη, καθώς και τον κωδικό του αισθητήρα Beacon που αντιστοιχεί στο χώρο. Επίσης έχει ως στόχο πέραν της παροχής πρόσβασης, την ενημέρωση της Υπηρεσίας Διαχείρισης Συνδρομών σε Χώρους σχετικά με τις αυξομειώσεις που συμβαίνουν στον πληθυσμό ενός χώρου. Όταν κάποιος χώρος ξεπεράσει κάποιο όριο, μέσω αυτής της υπηρεσίας θα σταλεί το αίτημα έναρξης συναγερμού. Αντίστοιχα και για την περίπτωση που κάποιος χρήστης απαιτήσει κάποιο μέλος του προσωπικού να τον εξυπηρετήσει.

*Πίνακας 6 REST Ελέγχου Προσβάσεων*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| POST | /monitoring/access_request | JSON-encoded:<br>user_id:string<br>area_beacon:string | Δημιουργία καινούριου αιτήματος εισόδου. |
| POST | /monitoring/alert/start/area_id/{area_id} | - | Εισαγωγή καινούριου αιτήματος για έναρξη συναγερμού. |
| PUT | /monitoring/alert/handle/area_id/{area_id}/handler_id/{handler_id} | - | Αίτηση εξυπηρέτησης συναγερμού. |
| PUT | /monitoring/alert/stop/area_id/{area_id} | - | Αίτηση τερματισμού συναγερμού. |
| POST | /monitoring/dispatch/create | JSON-encoded:<br>user_id:string<br>area_id:string<br>area_type:string | Εισαγωγή καινούριου αιτήματος για αποστολή προσωπικού. |
| PUT | /monitoring/dispatch/handle/area_type/{area_type}/area_id/{area_id}/handler_id/{handler_id} | | Αίτηση εξυπηρέτησης αιτήματος για αποστολή προσωπικού. |

### 4.2.2.7 Υπηρεσία Αποθήκευσης Δεδομένων (Storage Service)

Η συγκεκριμένη υπηρεσία έχει ως κύριο στόχο την επικοινωνία με το δημόσιο νέφος. Για το σκοπό αυτό, αποστέλλει μηνύματα σε μορφή JSON προς την αντίστοιχη υπηρεσία του δημοσίου νέφους. Αυτο πραγματοποιήθηκε για την επικοινωνία της εφαρμογής με ένα δημόσιο νέφος το οποίο θα συγκεντρώνει στοιχεία και από άλλα στιγμιότυπα με σκοπό να εξάγει μεταδεδομένα και να γίνεται ανάλυση τους (Big Data Analysis). Στην παρούσα μορφή στέλνονται στοιχεία σχετικά με τις κινήσεις των διαφόρων χρηστών, καθώς και στοιχεία σχετικά με συναγερμούς και αιτήματα αποστολής προσωπικού. Τα δεδομένα αφού σταλούν από τη συγκεκριμένη υπηρεσία στο δημόσιο νέφος, αποθηκεύονται με χρήση της υπηρεσίας JSON Storage (περιγράφεται στην Ενότητα 4.2.2.9).

*Πίνακας 7 REST Αποθήκευσης Δεδομένων*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| **POST** | /storage/log | JSON-encoded:<br>user_id:integer<br>area_id:integer<br>action_type:string | Εισαγωγή καινούριας καταχώρησης στο ιστορικό χρήστη. |
| **POST** | /storage/start_alert | JSON-encoded<br>area_id: integer | Δημιουργία καταχώρησης για έναρξη συναγερμού. |
| **POST** | /storage/alert_handled | JSON-encoded<br>area_id: integer<br>handler_id:string | Αίτηση εξυπηρέτησης συναγερμού. |
| **POST** | /storage/stop_alert | JSON-encoded<br>area_id: integer | Δημιουργία καταχώρησης για λήξη συναγερμού. |
| **POST** | /storage/create_dispatch | JSON-encoded<br>user_id:integer<br>area_id:integer<br>area_type:string | Δημιουργία καταχώρησης αιτήματος αποστολής προσωπικού. |
| **POST** | /storage/handle_dispatch/handler_id/{handler_id} | JSON-encoded<br>handler_id:integer<br>area_id:integer<br>area_type:string | Αίτηση εξυπηρέτησης αιτήματος αποστολής προσωπικού. |

### 4.2.2.8 Υπηρεσία Διαχείρισης Συνδέσεων (Connectivity Service)

Η συγκεκριμένη υπηρεσία είναι ιδιαίτερα απλή στη χρήση της. Λαμβάνει ένα κωδικοποιημένο μήνυμα σε μορφή Base64[22] με τα στοιχεία ενός χρήστη, όταν ο χρήστης προσπαθεί να συνδεθεί στο ΣΕΠ μέσω της εφαρμογής για κινητό. Στη συνέχεια το αποκωδικοποιεί και ελέγχει αν τα συγκεκριμένα στοιχεία αντιστοιχούν σε κάποιο χρήστη, επικοινωνώντας με την Υπηρεσία Διαχείρισης Χρηστών. Σε

[22] https://en.wikipedia.org/wiki/Base64

περίπτωση επιτυχημένου ελέγχου στέλνεται μήνυμα επιτυχημένης εισόδου, καθώς και τα ανανεωμένα στοιχεία του χρήστη, αν έχει αλλάξει κάποιο από αυτά από την προηγούμενη σύνδεση του. Εναλλακτικά αποστέλλεται μήνυμα αποτυχημένης εισόδου στο χρήστη.

*Πίνακας 8 REST Διαχείρισης Συνδέσεων*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| **POST** | /connectivity | URL-encoded (Base 64 encoded):<br>id= integer<br>role= string | Αποστολή στοιχείων χρήστη με σκοπό την εξακρίβωσή του για αυτόματη είσοδο. |

### 4.2.2.9 JSON Storage GE

Ο JSON Storage GE αποτελεί μία υπηρεσία του υπολογιστικού νέφους Intellicloud. Τα δεδομένα αφού σταλούν από το ιδιωτικό νέφος μέσω της Υπηρεσίας Αποθήκευσης Δεδομένων προς το δημόσιο νέφος, λαμβάνονται από την υπηρεσία JSON Storage. Η συγκεκριμένη υπηρεσία αποτελεί ένα σύστημα αποθήκευσης δεδομένων βασισμένο σε μη σχεσιακές βάσεις δεδομένων (NoSQL Database System[23]) για αποθήκευση δεδομένων σε μορφή JSON. Ακολουθεί REST αρχιτεκτονική, ενώ παρέχει ένα πλήθος από διαθέσιμες επιλογές σχετικά με τη δημιουργία συλλογών για αποθήκευση δεδομένων. Στα πλαίσια της εργασίας χρησιμοποιήθηκε με τη λογική της αποστολής δεδομένων που προκύπτουν από τη χρήση του ΣΕΠ προς το δημόσιο νέφος. Επομένως έχουμε μόνο μία μονομερής επικοινωνία η οποία λειτουργεί με τον εξής τρόπο.

Δεδομένου ότι η υπηρεσία βρίσκεται σε μία κοινόχρηστη εικονική μηχανή, ο χρήστης συνδέεται εισάγοντας τα στοιχεία του σε μορφή Base64. Στη συνέχεια δημιουργούμε μία βάση δεδομένων η οποία περιέχει τρεις συλλογές. Η πρώτη αφορά τις διάφορες κινήσεις χρηστών (είσοδος και έξοδος από διάφορους χώρους). Η δεύτερη αφορά της περιπτώσεις που ο πληθυσμός ενός χώρου ξεπέρασε κάποιο όριο. Στις εγγραφές περιλαμβάνονται στοιχεία για το πότε έγινε αυτό, αν και πότε κάποιο μέλος του προσωπικού έκανε μετάβαση στο συγκεκριμένο χώρο και πότε έληξε το συγκεκριμένο περιστατικό. Η τρίτη συλλογή αφορά τις αιτήσεις χρηστών για επιπλέον προσωπικό και περιλαμβάνει στοιχεία σχετικά με το πότε εστάλη το αίτημα, αν εξυπηρετήθηκε και από ποιο μέλος του προσωπικού εξυπηρετήθηκε. Η αποστολή των στοιχείων γίνεται όποτε παράγονται καινούρια δεδομένα.

[23] https://en.wikipedia.org/wiki/NoSQL

Παρατίθεται ο Πίνακας 9 που περιγράφει αναλυτικά τις διαθέσιμες εντολές του JSON Storage.

*Πίνακας 9 REST JSON Storage*

| Method | BaseURL (/147.27.50.33:3000) | Data | Result |
|---|---|---|---|
| **POST** | /users | JSON Object that contains the username and password of the user. E.g., {"username":"test", "password":"1234"} | Creates a new user with the given credentials. Each user can access their private area (their own databases). |
| **GET** | /user/{username} | -- | Returns the user's credentials. |
| **PUT** | /user/{username} | JSON Object that contains the new password. E.g., {"password":"newPass"} | Updates the user password. The username cannot be changed as it is used as primary key. |
| **DELETE** | /user/{username} | -- | Removes the user and their private area (all databases and resources included). |
| **GET** | /users/{username}/dbs | -- | Returns the databases of the given user. |
| **POST** | /users/{username}/dbs | E.g., {"name":"myDb"} | Creates a new Database with the given name. |
| **DELETE** | /users/{username}/dbs/{dbname} | -- | Deletes the database and all its contents |
| **GET** | /users/{username}/dbs/{dbname}/collections | -- | Returns the collection info in the database. |
| **POST** | /users/{username}/dbs/{dbname}/collections | E.g., {"name":"myCollection"} | Creates a collection in the database with the given name. |
| **GET** | /users/{username}/dbs/{dbname}/collections /{collectionname} | -- | Returns the information about the specified collection |
| **PUT** | /users/{username}/dbs/{dbname}/collections /{collectionname} | E.g., {"name":"newName"} | Renames the collection. |
| **DELETE** | /users/{username}/dbs/{dbname}/collections /{collectionname} | -- | Removes the collection from the database and all the documents it contains. |
| **GET** | /users/{username}/dbs/{dbname}/collections /{collectionname}/records | -- | Returns all the records in the collection |
| **POST** | /users/{username}/dbs/{dbname}/collections /{collectionname}/records | JSON Object | Inserts the JSON Object into the collection. |
| **GET** | /users/{username}/dbs/{dbname}/collections /{collectionname}/records/{id} | -- | Returns the JSON Object specified by the given id. |
| **PUT** | /users/{username}/dbs/{dbname}/collections /{collectionname}/records/{id} | JSON Object | Updates the JSON Object specified by the given id. |
| **DELETE** | /users/{username}/dbs/{dbname}/collections /{collectionname}/records/{id} | -- | Deletes the JSON Object specified by the given id. |
| **POST** | /users/{username}/dbs/{dbname}/collections /{collectionname}/finds | JSON Object with the conditions that must be met. E.g., {"name":"John", "salary":1000} | Executes a query and returns a JSON array with the documents that satisfy the criteria specified in the posted data. The criteria should follow the MongoDB query syntax. |
| **POST** | /users/{username}/dbs/{dbname}/collections /{collectionname}/distinct | JSON Object with two attributes. 1) The value on which the distinction will take place ("key") and 2) the conditions ("query"). E.g., {"key":"name", "query":{"salary":1000}} | Finds the distinct values for a specified field across a single collection and returns the results in a JSON array. |
| **POST** | /users/{username}/dbs/{dbname}/collections /{collectionname}/counts | JSON Object with the criteria of the query. E.g., {"name":"John", "salary":1000} | Returns the count of documents that would match a *find* query. |

### 4.2.2.10 Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Orion Context Broker Publish-Subscribe Service)

Αποτελεί μία υπηρεσία που παρέχεται από το Fiware. Η συγκεκριμένη υπηρεσία παρέχει δυνατότητες Publish/Subscribe[24].

Πιο αναλυτικά με τον όρο Subscribe εννοούμε τη δυνατότητα των χρηστών να εγγραφούν σε μία οντότητα και να λαμβάνουν ενημερώσεις σε περίπτωση αλλαγής των παραμέτρων της. Ο όρος Publish αφορά τις δημοσιεύσεις που γίνονται σχετικά με δημιουργία ή επεξεργασία των διαφόρων οντοτήτων.

Μέσω του Context Broker έχουμε τη δυνατότητα δημιουργίας μιας οντότητας. Η κάθε οντότητα περιλαμβάνει διάφορες μεταβλητές. Ο κάθε χρήστης μπορεί να εγγραφεί στην οντότητα που δημιουργείται και να λαμβάνει ειδοποιήσεις όποτε γίνεται αλλαγή σε παράμετρο της. Η ειδοποίηση γίνεται με τη μορφή ενός REST αιτήματος που αποστέλλεται από την υπηρεσία προς ένα URL που ορίζει ο χρήστης κατά την εγγραφή του. Στα πλαίσια του ΣΕΠ θεωρούμε ως οντότητα τον κάθε κοινόχρηστο χώρο. Κατά τη διαδικασία δημιουργίας ενός χώρου, δημιουργούμε αυτόματα μέσω της Υπηρεσίας Διαχείρισης Συνδρομών σε Χώρους την αντίστοιχη οντότητα, που περιέχει την περιγραφή του χώρου, το μέγιστο πληθυσμό του, καθώς και τα όρια έναρξης και λήξης συμβάντων. Σε περίπτωση ανανέωσης της περιγραφής ή των ορίων απλώς στέλνουμε ένα αίτημα προς το Context Broker για ανανέωση των στοιχείων της οντότητας. Σε περίπτωση που αλλάξει ο πληθυσμός ενός χώρου, ο Orion Context Broker θα στείλει αίτημα προς την Υπηρεσία Ελέγχου Προσβάσεων με σκοπό να χειριστεί το αίτημα. Περισσότερα σχετικά με τις μεθόδους του Orion Context Broker βρίσκονται στο https://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker

Η θύρα (port) για επικοινωνία με το Context Broker είναι η 1026.

Στον Πίνακα 10, αναφέρουμε ενδεικτικά κάποια παραδείγματα που δημιουργήσαμε για την υλοποίηση του ΣΕΠ.

*Πίνακας 10 Μέθοδοι που χρησιμοποιήθηκαν από το Orion Context Broker*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| **POST** | /v1/contextEntities/ | id":"'{το id που χρησιμοποιήσαμε}'" ,"type": "Public_Area","attributes": [ { "name": "population", "type":"integer","value":"0"} ,{"name": "description", "type":"string","value":"'{description}'" | Δημιουργία οντότητας |

[24] https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern

| | | | |
|---|---|---|---|
| | | ,{"name": "population_cap", "type": "integer","value":"'population_cap}'" }, {"name": "high_threshold", "type": "integer","value":"'{high_threshold}'" }, {"name": "low_threshold", "type": "integer","value":"'{low_threshold}'" }]}'; | |
| **POST** | v1/subscribeContextv1/subscribeContext | {"entities": [{"type": "Public_Area","isPattern": "false","id": "'{id}'"} ], "attributes": [ ], "reference": "{το url στο οποίο θα λαμβάνουμε τα αιτήματα}","duration": "{διάρκεια εγγραφής}", "notifyConditions": [{"type": "ONCHANGE","condValues": ["population"]}], "throttling": "PT1S"} | Δημιουργία εγγραφής σε οντότητα. |
| **PUT** | /v1/contextEntities/type/Public_Area/id/{id}/attributes/description | JSON-encoded description: string | Επεξεργασία περιγραφής χώρου. |
| **PUT** | /v1/contextEntities/type/Public_Area/id/{id}attributes/population_cap | JSON-encoded area_id: integer | Επεξεργασία μέγιστης χωρητικότητας χώρου. |
| **PUT** | /v1/contextEntities/type/Public_Area/id/{id}attributes/population | JSON-encoded user_id:integer area_id:integer area_type:string | Επεξεργασία πληθυσμού χώρου. |
| **PUT** | /v1/contextEntities/type/Public_Area/id{id}/attributes/high_threshold | JSON-encoded handler_id:integer area_id:integer area_type:string | Επεξεργασία ορίου έναρξης συναγερμού. |
| **PUT** | /v1/contextEntities/type/Public_Area/id/{id}/attributes/low_threshold | - | Επεξεργασία ορίου λήξης συναγερμού. |
| **DELETE** | /v1/contextEntities/{id} | | Διαγραφή οντότητας |

### 4.2.3 Server-Sent Events

Όπως αναφέραμε και προηγουμένως, σε περίπτωση κάποιου συμβάντος, το προσωπικό της κτιριακής υποδομής πρέπει να ειδοποιείται άμεσα. Για το λόγο αυτό κάνουμε χρήση των Server-Sent Events.

Πιο αναλυτικά, το μέλος του προσωπικού θα συνδέεται στη σελίδα με τα ενεργά συμβάντα είτε από την εφαρμογή για κινητό είτε μέσω φυλλομετρητή. Κάνοντας χρήση αυτού του χαρακτηριστικού, όσο η σελίδα παραμένει ενεργή, θα λαμβάνει δεδομένα που στέλνονται από τον εξυπηρετητή ανά τακτά χρονικά διαστήματα (λίγα δευτερόλεπτα). Έτσι θα ενημερώνεται για συμβάντα χωρίς να υπάρχει ανάγκη ανανέωσης της σελίδας.

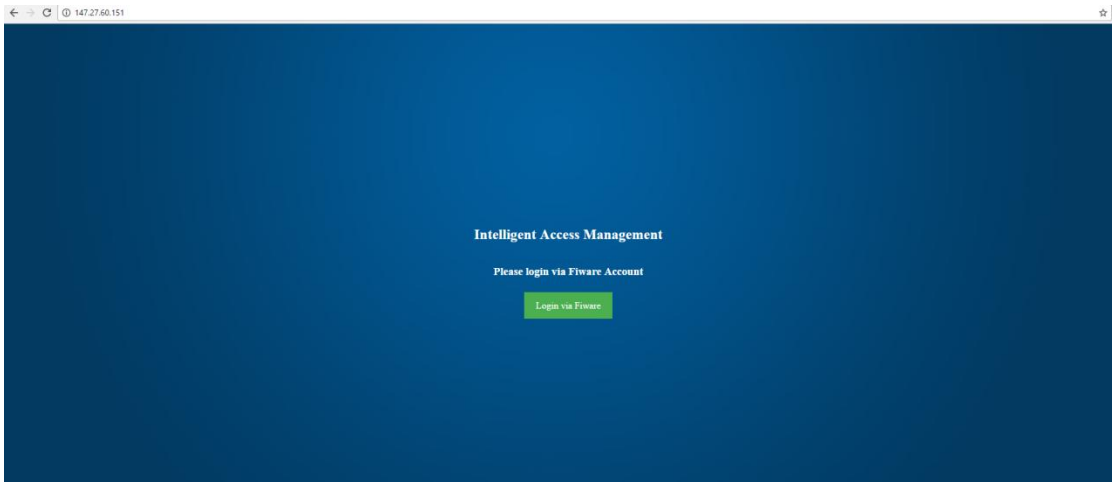
Η χρήση της συγκεκριμένης μεθόδου προτιμήθηκε έναντι του AJAX, καθώς με αυτό τον τρόπο ο χρήστης δε χρειάζεται να αλληλεπιδρά με τη σελίδα ώστε να ενημερώνεται το περιεχόμενό της. Εναλλακτικά μπορούσαμε να κάνουμε χρήση Web Sockets, όμως κάτι τέτοιο θα ήταν πιο πολύπλοκο και απαιτεί περισσότερους υπολογιστικούς πόρους συστήματος, σε περίπτωση σύνδεσης πολλών χρηστών.

Αξίζει να σημειωθεί ότι η συγκεκριμένη μέθοδος, δε γνωρίζει σε ποιους χρήστες έχει μεταδώσει προηγουμένως δεδομένα, με αποτέλεσμα να κρίνεται αναγκαίο να σταλούν όλα τα ενεργά συμβάντα ανεξαρτήτως αν έχουν σταλεί ή όχι στο παρελθόν σε ένα μέρος των χρηστών. Επομένως κρίνεται σκόπιμο να αποφεύγουμε την αποστολή συμβάντων που προκλήθηκαν πριν από μεγάλο χρονικό διάστημα, ή να προβαίνουμε στην αποστολή πολλών λεπτομερειών σχετικά με τα συμβάντα.

## 4.3 Παραδείγματα λειτουργίας του ΣΕΠ

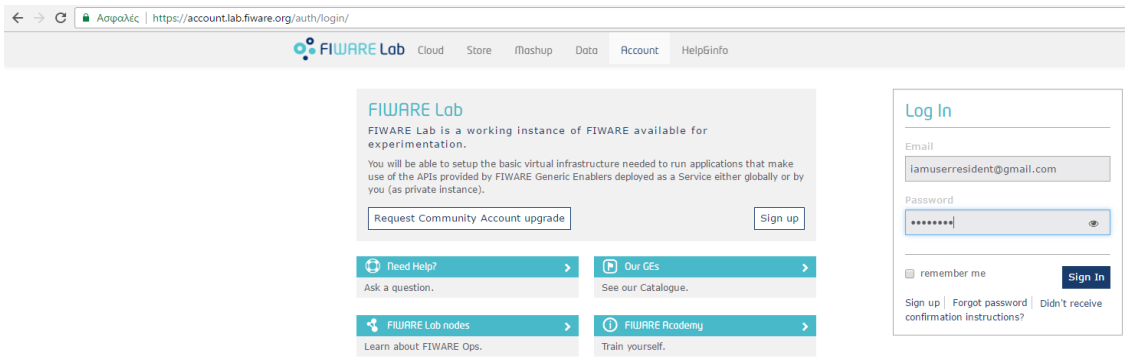
**Προσθήκη Χρήστη στο ΣΕΠ**

Ο χρήστης συνδέεται στη σελίδα του ΣΕΠ (http://147.27.60.151 ) όπως φαίνεται στην Εικόνα 14.



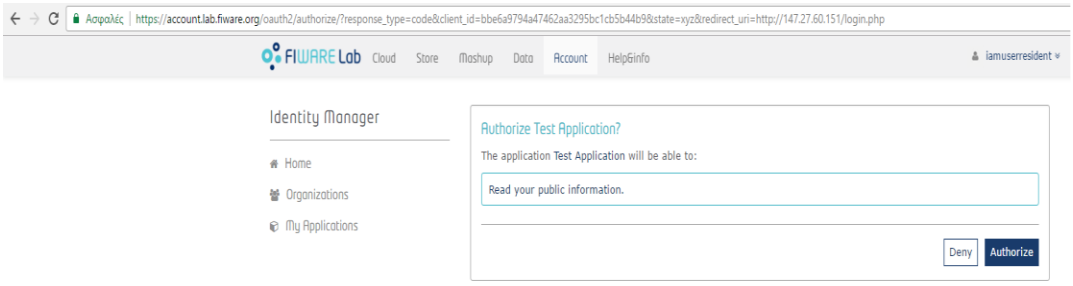

*Εικόνα 14 Κεντρική Σελίδα του ΣΕΠ*

Πατώντας το αντίστοιχο κουμπί μεταφέρεται στο Fiware για εισαγωγή των στοιχείων του. Η Εικόνα 15 παρουσιάζει τη σελίδα σύνδεσης του Fiware.



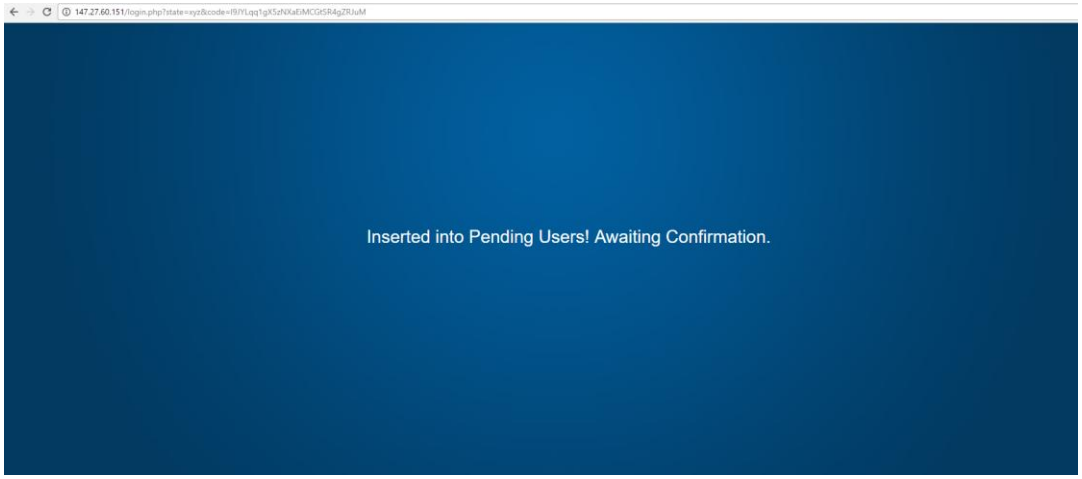

*Εικόνα 15 Σελίδα σύνδεσης Fiware Lab.*

Στη συνέχεια δίνουμε εξουσιοδότηση στην εφαρμογή, όπως στην Εικόνα 16.



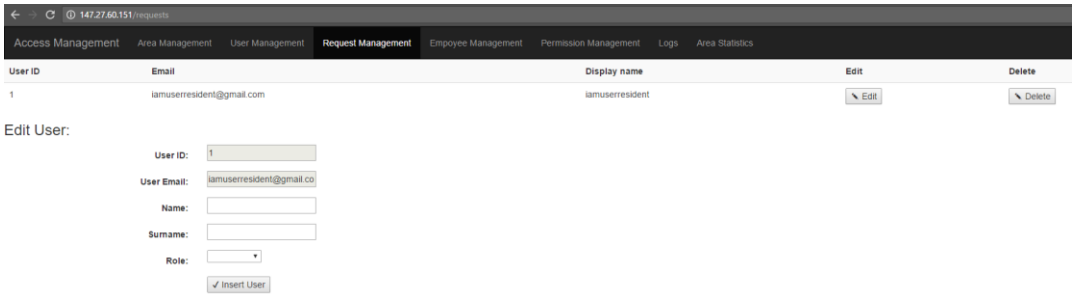

*Εικόνα 16 Παροχή Αδειών στην εφαρμογή με στόχο τη χρησιμοποίησή της*

Ο χρήστης ενημερώνεται μέσω μηνύματος ότι στάλθηκε το αίτημα του και περιμένει έγκριση από το διαχειριστή (Εικόνα 17).




*Εικόνα 17 Το μήνυμα που εμφανίζεται αμέσως μετά από μια εγγραφή χρήστη*

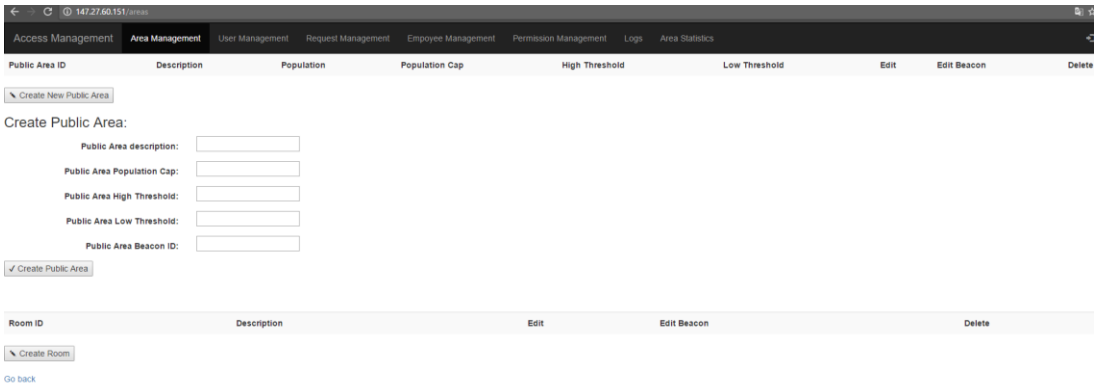
Στο σημείο αυτό μεταβαίνουμε στο λογαριασμό του διαχειριστή. Κάνουμε σύνδεση μέσω Fiware με αντίστοιχο τρόπο και μεταβαίνουμε στην καρτέλα Διαχείριση Αιτημάτων. Συμπληρώνουμε τα στοιχεία του χρήστη και πατάμε το κουμπί για εισαγωγή χρήστη (Insert User) όπως φαίνεται στην Εικόνα 18.




*Εικόνα 18 Επιλογές επεξεργασίας αιτημάτων χρηστών*

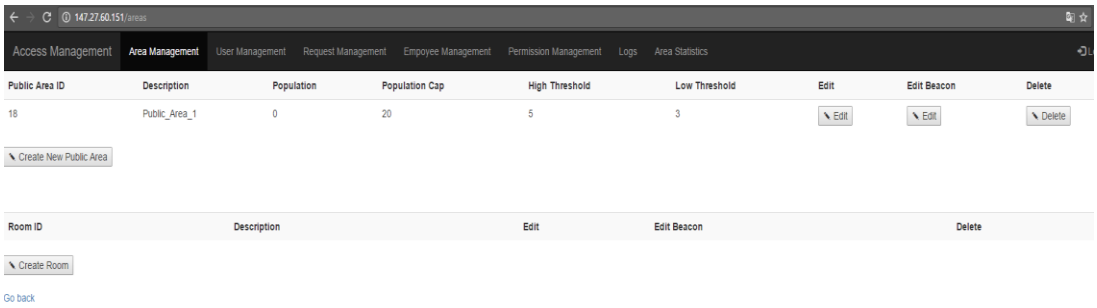
**Δημιουργία Χώρου στο ΣΕΠ**

Ως διαχειριστές μεταβαίνουμε στην καρτέλα εισαγωγής χώρου και εισάγουμε τα στοιχεία του χώρου. Εφόσον αναφερόμαστε σε κοινόχρηστο χώρο αυτά αφορούν την περιγραφή του χώρου, το μέγιστο πληθυσμό του, τα όρια έναρξης και λήξης συναγερμού, καθώς και το αναγνωριστικό του αισθητήρα Beacon που αντιστοιχεί στο χώρο. Η Εικόνα 19 περιγράφει την παραπάνω διαδικασία.




*Εικόνα 19 Επιλογές για δημιουργία κοινόχρηστου χώρου*

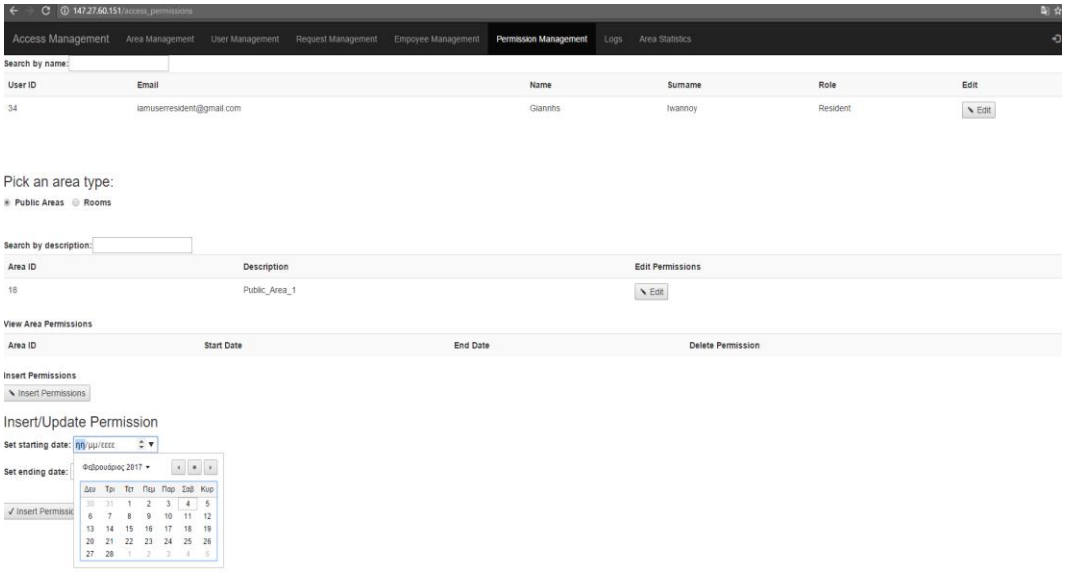
Μετά την εισαγωγή ο χώρος φαίνεται με τον εξής τρόπο στην εφαρμογή μας (Εικόνα 20).




*Εικόνα 20 Προβολή χώρων*

**Προσθήκη Δικαιωμάτων Πρόσβασης**

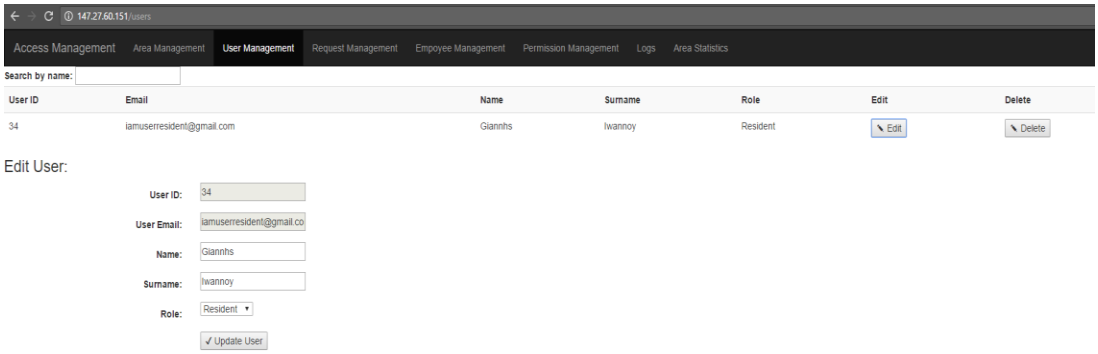
Για την προσθήκη δικαιωμάτων πρόσβασης,ως διαχειριστές, αρχικά διαλέγουμε το χρήστη στον οποίο θέλουμε να παραχωρήσουμε δικαιώματα πρόσβασης. Στη συνέχεια διαλέγουμε το είδος του χώρο. Έπειτα διαλέγουμε το χώρο που μας απασχολεί και εισάγουμε τις ημερομηνίες που θα καλύπτουν τα δικαιώματα πρόσβασης. Η διαδικασία περιγράφεται στην Εικόνα 21.




*Εικόνα 21 Διαδικασία προσθήκης δικαιώματος πρόσβασης χρήστη*

### Επεξεργασία Χρήστη

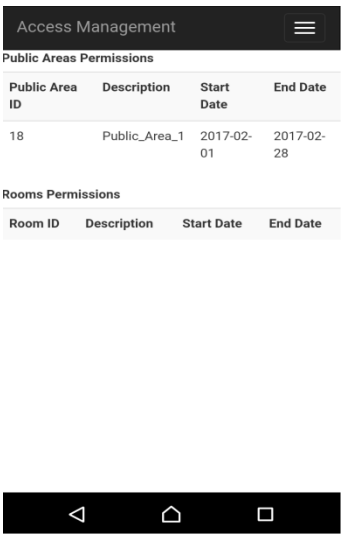
Αποτελεί ιδιότητα του διαχειριστή ιδιωτικού νέφους. Γίνεται μέσω της καρτέλας User Management (Διαχείριση Χρηστών). Οι επιλογές που είναι διαθέσιμες προς επεξεργασία παρουσιάζονται στην Εικόνα 22. Το αναγνωριστικό και το email του χρήστη θεωρούμε ότι δεν γίνεται να επεξεργαστεί.



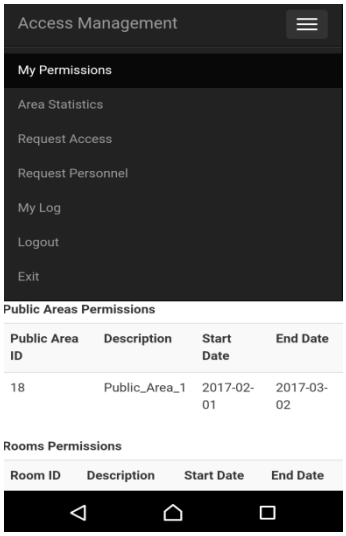

*Εικόνα 22 Επεξεργασία στοιχείων χρήστη*

### Περιγραφή βασικών λειτουργιών της εφαρμογής για κινητά

Ο χρήστης μετά από επιτυχημένη σύνδεση στην εφαρμογή μέσω κινητού, (παραλείπεται η περιγραφή καθώς είναι πανομοιότυπη με τη διαδικασία που περιγράφεται παραπάνω) μεταφέρεται στη σελίδα που αναφέρει τα δικαιώματα πρόσβασής του (Εικόνα 23).



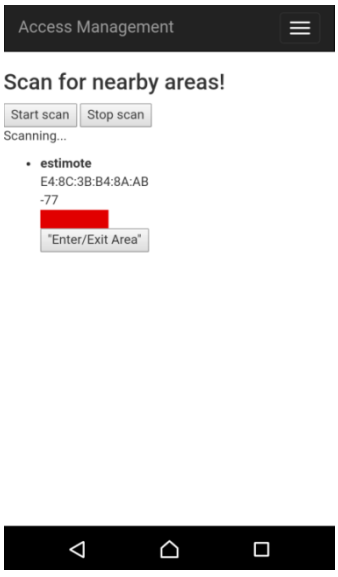

*Εικόνα 23 Προβολή δικαιωμάτων πρόσβασης χρήστη*



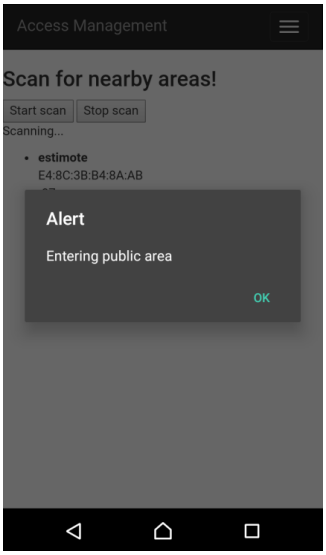

*Εικόνα 24 Διαθέσιμες επιλογές κατοίκου/επισκέπτη*

Στην Εικόνα 24 έχουμε μεταβεί στις διαθέσιμες επιλογές που υπάρχουν για το χρήστη πατώντας το πάνω δεξιά κουμπί.

Μεταβαίνουμε στην καρτέλα Request Access (Αίτηση Πρόσβασης-Εικόνα 25) για να ζητήσουμε πρόσβαση στο χώρο για τον οποίο δημιουργήσαμε το δικαίωμα πρόσβασης προηγουμένως. Πατώντας το πλήκτρο Start Scan (Εναρξη ανίχνευσης) εντοπίζουμε τους διαθέσιμους αισθητήρες Beacon που αντιστοιχούν στο χώρο. (Τα στοιχεία που παρουσιάζονται μπορούν να τροποποιηθούν μέσω της εφαρμογής της Estimote, ωστόσο διατηρήθηκαν ανέπαφα με σκοπό να τα εξηγήσουμε. Η πρώτη σειρά αποτελεί το όνομα του αισθητήρα, η δεύτερη αφορά το αναγνωριστικό της συσκευής, ενώ η τρίτη δείχνει την απόσταση του κινητού από τον αισθητήρα.) Πατάμε το κουμπί Enter/Exit Area (Είσοδος σε/Έξοδος από Χώρο). Πλέον έχει καταγραφεί η είσοδός μας στην περιοχή (Εικόνα 26).



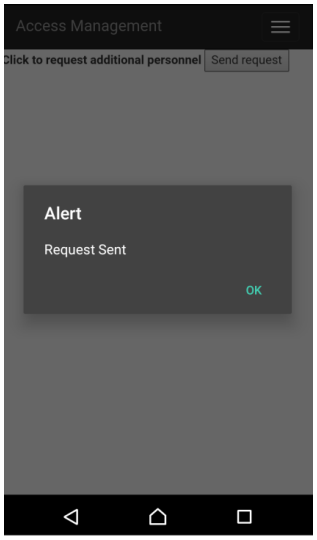

*Εικόνα 25 Ανίχνευση χώρων*




*Εικόνα 26 Μήνυμα επιτυχημένης πρόσβασης σε κοινόχρηστο χώρο*

Μεταβαίνουμε στην καρτέλα User Log(Ιστορικό Χρήστη-Εικόνα 27) για να δούμε την καταχώρηση και ύστερα μέσω του Request Personnel( Αίτηση Αποστολής προσωπικού-Εικόνα 28) κάνουμε αίτηση για επιπλέον προσωπικό.


*Εικόνα 27 Προβολή ιστορικού χρήστη*

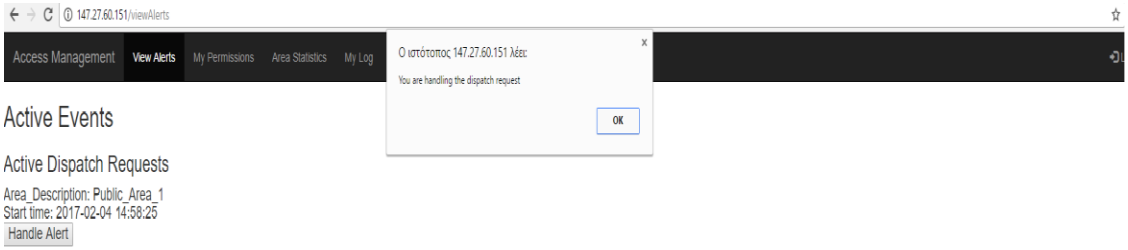



*Εικόνα 28 Αίτημα για αποστολή προσωπικού*

**Χειρισμός γεγονότων από το προσωπικό**

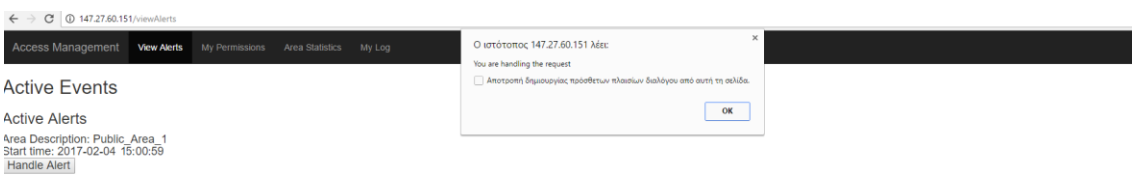
Για το χειρισμό γεγονότων από το προσωπικό κάναμε χρήση του Web-Api. Ωστόσο οι διαδικασίες αυτές θα μπορούσαν να εκτελεστούν και από την εφαρμογή για κινητά.

Αρχικά χειριστήκαμε το αίτημα για προσωπικό που στάλθηκε στο προηγούμενο βήμα όπως φαίνεται στην εικόνα (Εικόνα 29).




*Εικόνα 29 Προβολή ενεργών συμβάντων ζήτησης προσωπικού*

΄Υστερα, προκαλέσαμε ένα συμβάν συναγερμού, δηλαδή τοποθετήσαμε πολλούς χρήστες στον ίδιο χώρο για να παρουσιάσουμε ένα συμβάν συναγερμού (Εικόνα 30).




*Εικόνα 30 Προβολή ενεργών συμβάντων συναγερμού*

**Δεδομένα που στέλνονται στο δημόσιο νέφος**

Παραθέτουμε μία εικόνα που περιγράφει τη μορφή με την οποία αποστέλλονται τα δεδομένα κίνησης των χρηστών στο δημόσιο νέφος (Εικόνα 31).


*Εικόνα 31 Εγγραφή του δημοσίου νέφους για μία είσοδο χρήστη σε κοινόχρηστο χώρο*

# Κεφάλαιο 5ο

## 5.1 Ανάλυση Απόδοσης

Έχοντας ως στόχο να μετρήσουμε την απόδοση του συστήματος επιλέξαμε το σενάριο χρήσης που είναι πιο απαιτητικό από πλευράς υπολογιστικών πόρων. Το σενάριο αυτό περιγράφεται στο Ενότητα 3.6 και αφορά τη διαδικασία εισόδου χρήστη σε δημόσιο χώρο. Η διαδικασία ξεκινάει μέσω αποστολής αιτήματος στην εικονική μηχανή, στην οποία βρίσκεται η Υπηρεσία Ελέγχου Προσβάσεων. Όπως περιγράψαμε και προηγουμένως η συγκεκριμένη υπηρεσία επικοινωνεί με άλλες υπηρεσίες που βρίσκονται στην ίδια εικονική μηχανή, καθώς και με το Orion Context Broker, που βρίσκεται στη δεύτερη εικονική μηχανή και με το JSON Storage Service που φιλοξενείται στο Intellicloud.

Οι εικονικές μηχανές στις οποίες φιλοξενείται το ΣΕΠ είναι τρεις.

Οι δύο φιλοξενούνται στο Fiware και έχουν τα εξής χαρακτηριστικά:

- 1 εικονικός επεξεργαστής
- Αρχιτεκτονική επεξεργαστή x86_64
- Χρονισμός επεξεργαστή 2800Mhz
- Μέγεθος κρυφής μνήμης πρώτου επιπέδου 32KB

- Μέγεθος κρυφής μνήμης δευτέρου επιπέδου 4096KB
- Μέγεθος μνήμης τυχαίας πρόσβασης 2048MB
- Μέγεθος χωρητικότητας σκληρού δίσκου 20GB
- Λειτουργικό σύστημα Ubuntu 14.04
- Εξυπηρετητής Apache HTTP

Στην εικονική μηχανή που πραγματοποιήθηκαν οι μετρήσεις βρίσκονται όλες οι υπηρεσίες του ΣΕΠ. Η δεύτερη μηχανή του Fiware περιέχει το Orion Context Broker και η εικονική μηχανή που βρίσκεται στο Intellicloud, είναι κοινόχρηστη, οπότε δε γνωρίζουμε τα χαρακτηριστικά της και στεγάζει το JSON Storage Service (με μελλοντικό στόχο να λειτουργεί ως μονάδα ανάλυσης μεγάλων δεδομένων). Μετρήσαμε την απόδοση μόνο στη μία εικονική μηχανή, καθώς δε μπορούμε να εξάγουμε ασφαλή συμπεράσματα για την κοινόχρηστη μηχανή, ενώ και η πλειοψηφία των υπολογισμών γίνεται στην εικονική μηχανή που γίνεται το πείραμα. Στις μέτρήσεις των υπολογιστικών πόρων εμπεριέχεται και η κατανάλωση πόρων λόγω εκτέλεσης του πειράματος στην ίδια εικονική μηχανή. Οι μετρήσεις χρήσης υπολογιστικών πόρων έγινε μέσω της εντολής των Linux top[25], που ενεργοποιεί ένα ενσωματωμένο πρόγραμμα εποπτείας χρήσης υπολογιστικών πόρων της εικονικής μηχανής.

Μέσω του του ενσωματωμένου εργαλείου του Apache, ApacheBench (Apache HTTP server benchmarking tool)[26], μας δίνεται η δυνατότητα να στέλνουμε πολλά ταυτόχρονα αιτήματα. Μέσω του ApacheBench ορίζουμε το συνολικό αριθμό των αιτημάτων, καθώς και πόσα από αυτά θα εκτελούνται ταυτόχρονα. Ξεκινήσαμε με σειριακή αποστολή 200 και 2000 αιτημάτων. Αυτό γίνεται θέτοντας την παράμετρο concurrency ίση με 1. Τα αποτελέσματα παρουσιάζονται στους Πίνακες 11 και 12.

*Πίνακας 11 Απαιτούμενος χρόνος εκτέλεσης 200 αιτημάτων με concurrency 1*

| Ποσοστό αιτημάτων που εξυπηρετήθηκε | Χρόνος (σε ms) |
|---|---|
| 50% | 488 |
| 66% | 530 |
| 75% | 579 |
| 80% | 611 |
| 90% | 871 |
| 95% | 1055 |
| 98% | 1155 |
| 99% | 1539 |
| 100% | 1647 |

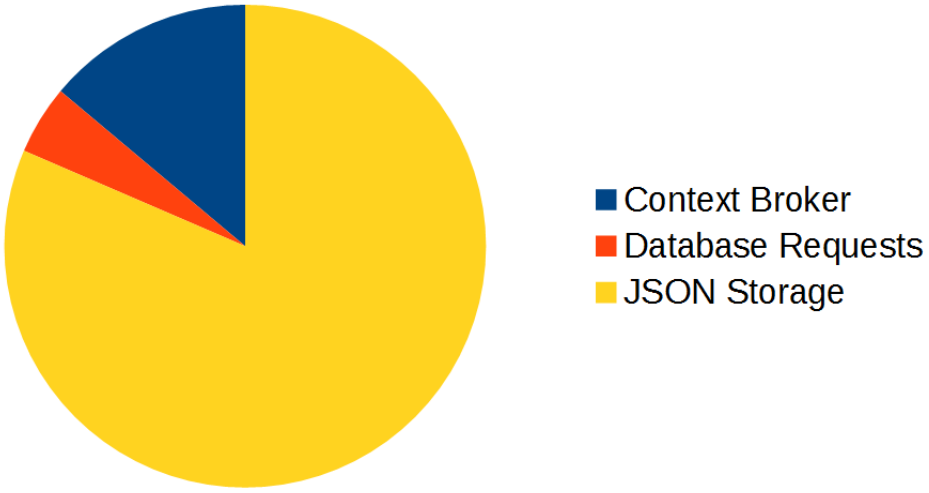
[25] https://linux.die.net/man/1/top
[26] https://httpd.apache.org/docs/2.4/programs/ab.html

*Πίνακας 12 Απαιτούμενος χρόνος εκτέλεσης 2000 αιτημάτων με concurrency 1*

| Ποσοστό αιτημάτων που εξυπηρετήθηκε | Χρόνος (σε ms) |
|---|---|
| 50% | 485 |
| 66% | 505 |
| 75% | 551 |
| 80% | 583 |
| 90% | 986 |
| 95% | 1095 |
| 98% | 1190 |
| 99% | 1254 |
| 100% | 1254 |

Επιπρόσθετα με τα αποτελέσματα, παρατηρήσαμε και πολύ χαμηλή χρήση πόρων κατά την εκτέλεση των παραπάνω μετρήσεων. Κάτι τέτοιο ήταν αναμενόμενο, καθώς μόνο ένα αίτημα εκτελούνταν κάθε χρονική στιγμή. Ο μέσος χρόνος ανά αίτημα ήταν ίσος με 551 ms, εκ των οποίων τα περισσότερα οφείλονται στο Storage Service, καθώς επικοινωνεί με το δημόσιο νέφος, που στεγάζεται στο Intellicloud, εισάγοντας έτσι καθυστερήσεις δικτύου. Ο ρυθμός μεταφοράς δεδομένων ήταν ίσος με 0.61 Kbytes/s.

Το ApacheBench δεν παρέχει πληροφορία για το χρόνο που χρειάστηκε η κάθε εικονική μηχανή να εκτελέσει τους υπολογισμούς τις. Επομένως δημιουργήσαμε ένα αυτοσχέδιο πείραμα για να διαπιστώσουμε που οφείλεται η καθυστέρηση στην εκτέλεση. Το πείραμα που δημιουργήσαμε, με σκοπό να μετρήσουμε το χρόνο που καταναλώνεται ανά εικονική μηχανή, αποτελεί αντίγραφο της Υπηρεσίας Ελέγχου Προσβάσεων, με την προσθήκη διαγνωστικών μηνυμάτων μετά από τις προσβάσεις στη βάση δεδομένων(Database Requests),για την επικοινωνία και εκτέλεση των διαδικασιών στην Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών(Context Broker), η οποία στεγάζεται στη δεύτερη εικονική μηχανή του Fiware, καθώς και για την επικοινωνία και εκτέλεση ενεργειών με την υπηρεσία JSON storage, που φιλοξενείται στο υπολογιστικό νέφος Intellicloud και αποτελεί κοινόχρηστη εικονική μηχανή. Στην Εικόνα 32 περιγράφεται το ποσοστό του χρόνου που δαπανήθηκε ανά εικονική μηχανή, συμπεριλαμβανομένου και του χρόνου που απαιτείται για τη μεταξύ τους επικοινωνία. Η διαδικασία ξεκινά από την πρώτη εικονική μηχανή που στεγάζει την Υπηρεσία Ελέγχου Προσβάσεων που με τη σειρά της καλεί τις υπόλοιπες υπηρεσίες. Η μέτρηση έγινε μόνο για σειριακή εκτέλεση αιτημάτων, λόγω αδυναμίας παράλληλης εκτέλεσης του πειράματος (concurrency>1).




*Εικόνα 32 Καταμερισμός χρόνου εκτέλεσης αιτήματος*

Είναι εμφανές ότι το μεγαλύτερο μέρος του χρόνου, καταναλώνεται στην επικοινωνία μεταξύ των διαφορετικών Υπολογιστικών Νεφών, καθώς δεν εκτελούνται πολύπλοκοι υπολογισμοί στο JSON Storage Service που να αιτιολογούν την καθυστέρηση.

Στη συνέχεια μετρήσαμε την απόδοση όταν έχουμε ταυτόχρονα αιτήματα. Για το επόμενο πείραμα εκτελέσαμε 2000 αιτήματα, ανά 40 ταυτόχρονα(concurrency=40) και καταλήξαμε στα αποτελέσματα που παρουσιάζονται στον Πίνακα 13.

*Πίνακας 13 Απαιτούμενος χρόνος εκτέλεσης 2000 αιτημάτων με concurrency 40*

| Ποσοστό αιτημάτων που εξυπηρετήθηκε | Χρόνος (σε ms) |
|---|---|
| 50% | 1908 |
| 66% | 2101 |
| 75% | 2238 |
| 80% | 2338 |
| 90% | 2708 |
| 95% | 2859 |
| 98% | 3301 |
| 99% | 3364 |
| 100% | 3478 |

Η χρήση του επεξεργαστή κατά μέσο όρο κυμαινόταν στο 50%. Η χρήση της RAM ανήλθε στα 150MB. Ο ρυθμός μεταφοράς δεδομένων ήταν ίσος με 7.04KB/s. Ο μέσος χρόνος ανά αίτημα ισούται με 1939ms. Παρατηρούμε ότι αρχίζει και παρουσιάζεται καθυστέρηση κατά την εκτέλεση του συστήματος μας, που όμως

κυμαίνεται ακόμα σε ανεκτά επίπεδα. Η καθυστέρηση οφείλεται στην επικοινωνία που πραγματοποιείται μεταξύ των υπηρεσιών, η οποία γίνεται με χρήση αιτημάτων CURL. Αυτό έχει ως αποτέλεσμα την απότομη αύξηση των απαιτούμενων υπολογιστικών πόρων. Σε συνδυασμό με την ανάγκη για ταυτόχρονη εξυπηρέτηση των αιτημάτων, ο εξυπηρετητής αναγκάζεται να μεταβεί σε ταυτόχρονη εκτέλεση πολλαπλών ενεργειών (multitasking).

Στη συνέχεια κάναμε περαιτέρω μετρήσεις για να μετρήσουμε πως αυξάνεται η χρήση του επεξεργαστή ανάλογα με την αύξηση των ταυτόχρονων αιτημάτων, καθώς και το μέγιστο αριθμό αιτημάτων που μπορούμε να εξυπηρετήσουμε.

*Πίνακας 14 Απαιτούμενος χρόνος εκτέλεσης 2000 αιτημάτων με concurrency 120*

| Ποσοστό αιτημάτων που εξυπηρετήθηκε | Χρόνος (σε ms) |
|---|---|
| 50% | 5766 |
| 66% | 6224 |
| 75% | 6576 |
| 80% | 6713 |
| 90% | 7183 |
| 95% | 10045 |
| 98% | 11578 |
| 99% | 12649 |
| 100% | 13621 |

Ο ρυθμός μεταφοράς δεδομένων ισούται με 7.61 Kbytes/s ενώ η χρήση του επεξεργαστή ήταν σχεδόν μέγιστη (~95%). Η χρήση της RAM ανήλθε στα 420MB. Από τα παραπάνω αποτελέσματα γίνεται άμεσα σαφές ότι το σύστημα δυσκολεύεται να αποκριθεί στα αιτήματα που στέλνονται διαρκώς.

Με διάφορες δοκιμές διαπιστώθηκε ότι αυτό οφείλεται στη χρήση πολλών CURL αιτημάτων,δηλαδή αιτημάτων επικοινωνίας μεταξύ των διαφόρων υπηρεσιών μέσω του πρωτοκόλλου HTTP (Εικόνα 12), με αποτέλεσμα να δημιουργείται υψηλός αριθμός από ενεργά αιτήματα. Το παραπάνω μειονέκτημα σε συνδυασμό με την αργή επικοινωνία με το δημόσιο νέφος(Εικόνα 32) σήμαινε ότι τα αιτήματα έπρεπε να είναι ενεργά για πολύ μεγάλο χρονικό διάστημα, δεσμεύοντας έτσι τους διαθέσιμους πόρους. Αυτό εξακριβώθηκε και από το γεγονός ότι απαιτούνταν αύξηση του μέγιστου αριθμού διαθέσιμων νημάτων εργασίας του Apache(MaxRequestsWorkers), με σκοπό την περαιτέρω αύξηση του αριθμού των παράλληλων αιτημάτων.

## 5.2 Αλλαγές που μπορούν να γίνουν για βελτίωση της απόδοσης

Στην εικονική μηχανή που χρησιμοποιήθηκε για τις μετρήσεις ήταν τοποθετημένες όλες οι υπηρεσίες που υλοποιήθηκαν (Υπηρεσία Διαχείρισης Χρηστών, Υπηρεσία Διαχείρισης Χώρων, Υπηρεσία Αποθήκευσης Δεδομένων, Υπηρεσία Ελέγχου Προσβάσεων, Υπηρεσία Διαχείρισης Δικαιωμάτων, Υπηρεσία Διαχείρισης Συνδρομών σε Χώρους). Αυτό δεν είναι απαραίτητο να γίνεται καθώς η κάθε υπηρεσία λειτουργεί ανεξάρτητα από την τοποθεσία ύπαρξης των άλλων υπηρεσιών. Επομένως μπορούμε να μειώσουμε το φόρτο της εικονικής μηχανής μοιράζοντας τις υπηρεσίες σε ξεχωριστές. Παράλληλα με αυτό, παρατηρούμε ότι η κύρια υπηρεσία που χρησιμοποιούμε είναι η Υπηρεσία Ελέγχου Προσβάσεων, καθώς όπως γίνεται εμφανές και από στην Ενότητα 3.6 τα αιτήματα στην πλειοψηφία τους ξεκινάνε από αυτή την υπηρεσία, ενώ παραμένει ενεργή καθ' όλη τη διάρκεια του σεναρίου χρήσης, σε αντίθεση με τις υπόλοιπες υπηρεσίες. Επομένως θα μπορούσαμε να έχουμε μία εικονική μηχανή με λιγότερους πόρους για τις υπόλοιπες υπηρεσίες και να αυξήσουμε τους πόρους στην εικονική μηχανή που στεγάζεται.

# Κεφάλαιο 6

## 6.1 Συμπεράσματα

Μέσω της ανάπτυξης του ΣΕΠ καταλήξαμε σε διάφορα συμπεράσματα σχετικά με τις τεχνολογίες που χρησιμοποιήσαμε, καθώς και αντιμετωπίσαμε διάφορα εμπόδια κατά τη δημιουργία του. Συνοπτικά αναφέρουμε μερικά από αυτά.

- Η ανάπτυξη εφαρμογών μέσω του υπολογιστικού νέφους, παρέχει πολλά πλεονεκτήματα, καθώς υπάρχουν υπηρεσίες που υλοποιούν ήδη βασικές λειτουργίες, όπως η Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Orion Context Broker), η Υπηρεσία JSON storage, καθώς και η Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών (Keyrock Identity Manager Service).
- Μέσω της χρήσης προεγκατεστημένων και ρυθμισμένων εικονικών μηχανών, μπορούμε άμεσα να ξεκινήσουμε την ανάπτυξη της εφαρμογής μας, αποφεύγοντας τυχόν προβλήματα ασυμβατότητας.

- Η επικοινωνία μεταξύ διαφορετικών υπολογιστικών νεφών εισάγει σημαντικές καθυστερήσεις, λόγω δικτύου, επομένως είναι χρήσιμο να αποφεύγεται από διαδικασίες στις οποίες ο χρήστης περιμένει άμεσα απάντηση.
- Η ελαστικότητα είναι μία ιδιαίτερα σημαντική δυνατότητα του υπολογιστικού νέφους για την ανάπτυξη εφαρμογών, καθώς όπως παρατηρήσαμε οι απαιτήσεις σε υπολογιστικούς πόρους αλλάζουν σημαντικά ανάλογα με το πόσους ταυτόχρονους χρήστες έχουμε. Το Fiware δεν παρέχει ελαστικότητα, εκτός από την αρχική επιλογή του μεγέθους των υπολογιστικών πόρων της εικονικής μηχανής.
- Η χρήση υπηρεσιοκεντρικής αρχιτεκτονικής, καθώς και η χρήση του REST κάνει ιδιαίτερα εύκολη την επικοινωνία μεταξύ των υπηρεσιών.
- Η χρήση της AngularJS διευκόλυνε ιδιαίτερα στη δημιουργία ενός απλού περιβάλλοντος διεπαφής χρήστη, απλουστεύοντας πολλές από τις διαδικασίες σχεδιασμού του Συστήματος Διεπαφής τελικού χρήστη (Front-End).
- Αν και ο διαχωρισμός της εφαρμογής σε πολλές υπηρεσίες παρουσιάζει αρκετά θετικά χαρακτηριστικά, είναι σκόπιμο να ομαδοποιούμε τις υπηρεσίες που χρειάζονται για μία διαδικασία, καθώς σε διαφορετική περίπτωση αυξάνει κατά πολύ το φόρτο της εικονικής μηχανής, λόγω της συνεχής ανάγκης επικοινωνίας μεταξύ των διαφόρων υπηρεσιών.

## 6.2 Μελλοντικές Επεκτάσεις

Ως μελλοντική επέκταση θα μπορούσαμε να υλοποιήσουμε ένα φυσικό μηχανισμό για την παροχή ή απόρριψη πρόσβασης σε χώρους. Αυτό θα μπορούσε να υλοποιηθεί με ένα σύστημα πόρτας που θα άνοιγε όταν δεχόταν το αντίστοιχο αίτημα μέσω χρήσης NFC[27].

Επιπλέον η δημιουργία ενός πλήρους δημοσίου νέφους που θα επεξεργάζεται τα δεδομένα που δέχεται και θα εξάγει αντίστοιχα συμπεράσματα, ενημερώνοντας για παράδειγμα το χρήστη για τυχόν δικαιώματα πρόσβασης που λήγουν με σκοπό να του υπενθυμίσει να τα ανανεώσει, ή να του υποδείξει ότι δεν έκανε χρήση κάποιου δικαιώματος αποτελεί μελλοντικό στόχο.

Τέλος μπορούμε να υλοποιήσουμε ένα σύστημα χρεώσεων με βάση τον αριθμό των προσβάσεων, καθώς και τη διάρκεια τους.

[27] http://nearfieldcommunication.org/

# Κεφάλαιο 7

## Βιβλιογραφία


[1] NuvlaBox The smart cloud-in-a-box | SixSq

http://sixsq.com/products/nuvlabox/

[2] Estimote Beacons — real world context for your apps

http://estimote.com/

[3] FIWARE lab, the open innovation Lab

https://www.fiware.org/lab/

[4] What is cloud computing? A beginner's guide | Microsoft Azure

https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

Cloud computing - Wikipedia

https://en.wikipedia.org/wiki/Cloud_computing

A brief history of cloud computing - Cloud computing news

https://www.ibm.com/blogs/cloud-computing/2014/03/a-brief-history-of-cloud-computing-3/

[6] FIWARE

https://www.fiware.org/

[7] Generic Enablers | FIWARE Catalogue

https://catalogue.fiware.org/enablers/

[8] OAuth 2.0

https://oauth.net/2/

[9] RESTful Web services: The basics

http://www.ibm.com/developerworks/library/ws-restful/

[11] Internet of Things - Springer

http://link.springer.com/chapter/10.1007/978-1-4419-8237-7_13#page-1

[12] UML - Use Case Diagrams

https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

[13] Deployment diagram - Wikipedia

https://en.wikipedia.org/wiki/Deployment_diagram

[14] HTML5 Web Storage

http://www.w3schools.com/html/html5_webstorage.asp

[15] UML - Class Diagram

https://www.tutorialspoint.com/uml/uml_class_diagram.htm

[16] AngularJS — Superheroic JavaScript MVW Framework

https://angularjs.org/

[17] AngularJS: API: $http

https://docs.angularjs.org/api/ng/service/$http

[18]Apache Cordova

https://cordova.apache.org/

[19] MySQl

https://www.mysql.com/

[20] Slim Framework

https://www.slimframework.com/

[22] Base64 - Wikipedia

https://en.wikipedia.org/wiki/Base64

[23] NoSQL - Wikipedia

https://en.wikipedia.org/wiki/NoSQL

[24] Publish–subscribe pattern - Wikipedia

https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern

[25] top(1): tasks - Linux man page

https://linux.die.net/man/1/top

[26] ab - Apache HTTP server benchmarking tool - Apache HTTP Server Version 2.4

https://httpd.apache.org/docs/2.4/programs/ab.html

[27] Near Field Communication: What is Near Field Communication?

http://nearfieldcommunication.org/

## Βιβλιογραφία Εικόνων

[5][Εικόνα 1] Περιγραφή των μοντέλων παροχής Υπηρεσιών:

http://www.silverlighthack.com/image.axd?picture=2011%2F2%2FCloudServiceHierarchy.png

[10][Εικόνα 2] Περιγραφή του CRUD μοντέλου:

http://networkop.co.uk/images/rest-crud.png

[21][Εικόνα 13] Περιγραφή του OAuth2:

https://fhirblog.files.wordpress.com/2014/06/oauth2sequencediagram.png