Πολυτεχνείο Κρήτης
Τμήμα Ηλεκτρονικών Μηχανικών
και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

IoT management on the Cloud

Διαχείριση συσκευών «Διαδικτύου των Πραγμάτων» στο «Υπολογιστικό Νέφος»

Κωνσταντίνος Δούζης

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΠΤΡΟΠΗ
Ευριπίδης Πετράκης, Καθηγητής (Επιβλέπων)
Μίνως Γαροφαλάκης, Καθηγητής
Δρ. Στέλιος Σωτηριάδης, Επιστημονικός Συνεργάτης

ΧΑΝΙΑ 2015

Abstract

The Internet of Things was “born” in 2008 and since then is directly related to Cloud Computing technology. Many IoT application development environments motivate developers to create “smart” applications. FIWARE is a platform offering an environment for rapid IoT application development through modular components called Generic and Specific Enablers.

In this thesis we present the I.I.M (Intellicloud IoT Management) Specific Enabler. I.I.M is designed using SOA (Service Oriented Architecture) with the use of FIWARE Generic Enablers and is deployed on the Intellicloud infrastructure. The enabler offers management and storage services for data collected from different types of sensors. Also, I.I.M offers subscription and user notification capabilities. Services are provided through a REST API implementing sensor, user, permission, subscription and sensor data management functionality. The system aims to embed the functionality of the API to FIWARE in order to offer an entire IoT device management solution.

Περίληψη

Η ιδέα του Διαδικτύου των Πραγμάτων (Internet of Things) γεννήθηκε το 2008 και από τότε είναι άμεσα συνδεδεμένη με την τεχνολογία του Υπολογιστικού Νέφους (Cloud Computing). Πολλά περιβάλλοντα ανάπτυξης εφαρμογών του Διαδικτύου των Πραγμάτων δίνουν κίνητρα στους δημιουργούς λογισμικού για την ανάπτυξη «έξυπνων» εφαρμογών. Ένα από αυτά είναι το περιβάλλον του FIWARE, το οποίο επεκτείνεται συνεχώς και προσφέρει στην κοινότητά του υπηρεσίες γενικού (Generic Enablers) και ειδικού σκοπού (Specific Enablers) για την ανάπτυξη «έξυπνων» εφαρμογών στα πλαίσια του Διαδικτύου των Πραγμάτων.

Στην παρούσα διπλωματική εργασία αναπτύχθηκε η υπηρεσία ειδικού σκοπού I.I.M (Intellicloud IoT Management), η οποία στα πρότυπα της υπηρεσιο-κεντρικής αρχιτεκτονικής κάνει χρήση υπηρεσιών γενικού σκοπού που διατίθενται μέσω του FIWARE και της υποδομής Νέφους Intellicloud. Έχει τη δυνατότητα της διαχείρισης, αποθήκευσης και συνδρομής σε δεδομένα πολλών αισθητήρων διαφορετικού τύπου για την άμεση ενημέρωση των συνδρομητών. Η υπηρεσία διαθέτει ένα ολοκληρωμένο REST API, για λειτουργίες όπως η διαχείριση αισθητήρων, χρηστών, αδειών, συνδρομών και διαχείριση των δεδομένων των αισθητήρων. Στόχος του συστήματος είναι η ενσωμάτωση της λειτουργικότητας που προσφέρεται μέσω του API σε υπηρεσίες του περιβάλλοντος FIWARE για την σύνθεση μιας ολοκληρωμένης υπηρεσίας διαχείρισης συσκευών του Διαδικτύου των Πραγμάτων.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ.Πετράκη για την εμπιστοσύνη που έδειξε στις δυνατότητές μου, τις συμβουλές και τη καθοδήγησή του καθ' όλη την διάρκεια της διπλωματικής εργασίας. Επίσης όλα τα μέλη του εργαστηρίου προπτυχιακούς, μεταπτυχιακούς και συνεργάτες για την άψογη επικοινωνία μεταξύ μας που βοήθησε στην ολοκλήρωση της συγκεκριμένης διπλωματικής εργασίας. Τέλος την οικογένεια μου για την αμέριστη συμπαράσταση και στήριξή τους σε ολόκληρη τη διάρκεια των σπουδών μου.

Η διπλωματική εργασία είναι αφιερωμένη στην οικογένειά μου.

Κ.Δ.

Περιεχόμενα

1. **Εισαγωγή** 5

1.1 Υπολογιστικό Νέφος (Cloud Computing) 5

1.2 Διαδίκτυο των πραγμάτων (Internet of Things) 6

1.3 Το πρόβλημα 7

1.4 Περιγραφή της λύσης 10

1.5 Δομή της εργασίας 11

2. **Υπόβαθρο και Υπάρχουσες Τεχνολογίες** 12

2.1 Υπολογιστικό Νέφος (Cloud Computing) 12

2.1.1 Μοντέλα παροχής υπηρεσιών 14

2.1.2 Μοντέλα ανάπτυξης Υπολογιστικού Νέφους 15

2.1.3 Υπάρχουσες Τεχνολογίες 16

2.1.3.1 Υποδομή FIWARE και Intellicloud 16

2.1.3.2 Openstack 17

2.1.3.3 Υπηρεσίες Γενικού Σκοπού (Generic Enablers) και Ειδικού Σκοπού (Specific Enablers) 18

2.1.3.4 Υπηρεσιοκεντρική αρχιτεκτονική (Service Oriented Architecture) 18

2.1.3.5 Πρωτόκολλο επικοινωνίας HTTP και REST API 20

2.1.3.6 JSON (JavaScript Object Notation) 21

2.1.4 Εικονοποίηση πόρων σε Υπολογιστικό Νέφος 22

2.1.5 Πάροχοι Υπολογιστικού Νέφους 23

2.2 Διαδίκτυο των Πραγμάτων (Internet of Things) 24

2.2.1 Διαδίκτυο των Πραγμάτων (Internet of Things) και Υπολογιστικό Νέφος (Cloud Computing) 25

2.2.2 IoT περιβάλλοντα ανάπτυξης 26

2.2.3 Αισθητήρες και Διαδίκτυο των Πραγμάτων 26

3. **Σχεδιασμός και υλοποίηση του συστήματος** 29

3.1 Υπηρεσίες Γενικού Σκοπού και Ειδικού Σκοπού 31

3.1.1 KeyRock Identity Management GE 31

3.1.2 JSON Storage GE 31
3.1.3 Publish/Subscribe Context Broker-Orion Context Broker GE 33
3.1.4 Event Service Specific Enabler 35
3.2 Γενική Αρχιτεκτονική 37
3.2.1 Διεπαφή χρήστη (Front-End) 38
3.2.2 Διεπαφή διαχείρισης συστήματος (Back-End) 38
3.2.2.1 Αρχιτεκτονική του συστήματος 38
3.2.3 Χρήστες (Users) 40
3.3 Λειτουργικές απαιτήσεις 41
3.4 Παραμετροποίηση εικονικών μηχανών 44
3.4.1 Παραμετροποίηση KeyRock Identity Management GE 47
3.4.2 Παραμετροποίηση Publish/Subscribe Context Broker GE 48
3.4.3 Παραμετροποίηση JSON Storage GE 48
3.5 Υλοποίηση Ταυτοποίησης Χρήστη 50
3.6 Υλοποίηση Λογικής Συστήματος 52
3.6.1 Ανάλυση REST API 52
3.6.2 Υλοποίηση μεθόδων 57
3.6.3 Αιτήματα (Requests) και Απαντήσεις (Responses) 71
3.6.4 Παραδείγματα 72
3.6.5 Απαραίτητα εργαλεία 75
3.7 Πείραμα και αποτελέσματα 75

4. **Συμπεράσματα και Μελλοντική Δουλειά** 77
4.1 Συμπεράσματα 77
4.2 Περιορισμοί 78
4.3 Μελλοντική Δουλειά 79

Σχήματα

Σχήμα 1. Αναλογία των συσκευών που απαρτίζουν το Διαδίκτυο των Πραγμάτων με τον πληθυσμός της γης. 7

Σχήμα 2. Διεπαφή χρήστη «Front-End» και διεπαφή διαχείρισης συστήματος «Back-End» 9

Σχήμα 3. Υπηρεσιο-κεντρική αρχιτεκτονική 19

Σχήμα 4. REST API 21

Σχήμα 5. Η σχέση μεταξύ φυσικής υποδομής - hypervisor - εικονικών μηχανών 22

Σχήμα 6. Δομή JSON Storage GE 32

Σχήμα 7. Υποστηριζόμενο REST API του JSON Storage GE 33

Σχήμα 8. Παράδειγμα υπηρεσίας Context Broker GE 34

Σχήμα 9. Αρχιτεκτονική της υπηρεσίας ειδικού σκοπού Event Service 36

Σχήμα 10. Γενική αρχιτεκτονική συστήματος 38

Σχήμα 11. Αρχιτεκτονική του συστήματος και διαχωρισμός σε τμήματα 39

Σχήμα 12. Τοπολογία Δικτύου 45

Σχήμα 13. Παραμετροποίηση Στιγμιότυπου 46

Σχήμα 14. Ολοκληρωμένο στιγμιότυπο 46

Σχήμα 15. Αρχιτεκτονική Middleware 51

Σχήμα 16. Χρήση Authentication Middleware για την ταυτοποίηση του χρήστη 52

Σχήμα 17. Λειτουργία μεθόδου GET /contextNotifications 69

Σχήμα 18. Αίτημα και απάντηση του SE στην κλήση της μεθόδου GET /connected 73

Σχήμα 19. Αίτημα και απάντηση του SE στην κλήση της μεθόδου POST /subscription 74

Σχήμα 20. Αίτημα και απάντηση του SE στην κλήση της μεθόδου DELETE /subscription 74

Πίνακες

Πίνακας 1. Αρχική παραμετροποίηση του JSON storage με χρήστη το στιγμιότυπο 147.27.50.119 49
Πίνακας 2. Αρχική παραμετροποίηση του JSON storage με χρήστη public_user 50
Πίνακας 3. Μέθοδοι του REST API της κατηγορίας Διαχείριση Αισθητήρων 54
Πίνακας 4. Μέθοδοι του REST API της κατηγορίας Διαχείριση Χρηστών 55
Πίνακας 5. Μέθοδοι του REST API της κατηγορίας Διαχείριση Συνδρομών 56
Πίνακας 6. Μέθοδοι του REST API της κατηγορίας Διαχείριση Αδειών 56
Πίνακας 7. Μέθοδοι του REST API της κατηγορίας Διαχείριση Δεδομένων Αισθητήρων 57
Πίνακας 8. Απαραίτητες επικεφαλίδες στην κλήση μεθόδων του API 71
Πίνακας 9. Εξαιρέσεις και Κωδικοί Κατάστασης ως απαντήσεις σφάλματος στον χρήστη 72
Πίνακας 10. Πείραμα και αποτελέσματα 76

Κεφάλαιο 1ο

1. Εισαγωγή

1.1 Υπολογιστικό Νέφος (Cloud Computing)

Το Υπολογιστικό Νέφος είναι ένα μοντέλο που επιτρέπει εύκολη, ευέλικτη, κατά απαίτηση «on-demand» και απεριόριστη δικτυακή πρόσβαση σε μια συλλογή παραμετροποιήσιμων υπολογιστικών πόρων (δίκτυο, διακομιστές, αποθήκευση, εφαρμογές και υπηρεσίες), οι οποίοι μπορούν να δεσμευτούν και να απελευθερωθούν γρήγορα με την ελάχιστη δυνατή προσπάθεια και αλληλεπίδραση από το χρήστη. Αυτό το μοντέλο αποτελείται από τρία μοντέλα παροχής υπηρεσιών και τέσσερα μοντέλα ανάπτυξης. Τα μοντέλα παροχής υπηρεσιών που προσφέρει το Υπολογιστικό Νέφος είναι το **λογισμικό ως υπηρεσία (SaaS), πλατφόρμα ως υπηρεσία (PaaS)** και **υποδομή ως υπηρεσία (IaaS).** Επίσης, τα τέσσερα μοντέλα ανάπτυξης που χαρακτηρίζουν μια υποδομή Υπολογιστικού Νέφους είναι **το δημόσιο Νέφος (Public Cloud), ιδιωτικό Νέφος (Private Cloud), υβριδικό Νέφος (Hybrid Cloud), κοινοτικό Νέφος (Community Cloud)**.

Προσφέρει πολλά πλεονεκτήματα έτσι όλο και περισσότεροι καταναλωτές το προτιμούν. Μερικά από τα πλεονεκτήματα που προσφέρει σε σχέση με προγενέστερες τεχνολογίες είναι:

- **Χαμηλό κόστος υποδομής και συντήρησης**, δεν χρειάζεται οι καταναλωτές να αγοράσουν υποδομή για την φιλοξενία των εφαρμογών τους στο Νέφος. Το κόστος περιλαμβάνει μόνο τις συσκευές για την πρόσβαση στο Νέφος μέσω του διαδικτύου. Επίσης, δεν είναι αναγκαία η πρόσληψη εξειδικευμένου προσωπικού για την συντήρηση και την αναβάθμιση του λογισμικού που χρησιμοποιεί η επιχείρηση.

- **Ευελιξία**, καθώς οι καταναλωτές μπορούν να έχουν πρόσβαση στο Υπολογιστικό Νέφος από οπουδήποτε και αν βρίσκονται οποιαδήποτε χρονική στιγμή. Η μόνη προϋπόθεση είναι η κατοχή μιας συσκευής με δυνατότητα σύνδεσης στο διαδίκτυο.

- **Ελαστικότητα**, ο χρήστης έχει τη δυνατότητα να διαχειρίζεται τους πόρους που του παρέχει το Νέφος σύμφωνα με τις ανάγκες του. Μπορούν να αυξάνουν την υπολογιστική ισχύ (αποθηκευτικός χώρος, μνήμη κ.α) αν απαιτηθεί από την εφαρμογή που χρησιμοποιούν.

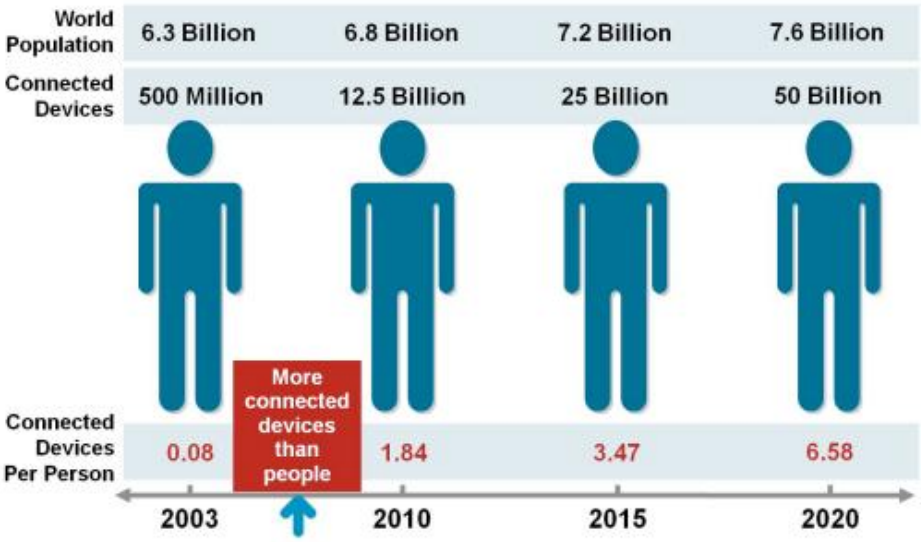
Μετά από μελέτες που έχουν γίνει έχει αποδειχθεί ότι είναι μια λύση πολύ συμφέρουσα για την ανάπτυξη εφαρμογών, χρήση υπηρεσιών, αξιοποίηση υπολογιστικών πόρων και όχι μόνο, σε σχέση με τις τεχνολογίες που μέχρι τώρα γνωρίζαμε. Η απομακρυσμένη διαχείριση δεδομένων, η εύκολη πρόσβαση και τα οικονομικά οφέλη που αυτό προσφέρει (μειωμένο κόστος υλικού και εξοπλισμού) είναι μερικοί από τους λόγους που το Νέφος προσελκύει το ενδιαφέρον μεγάλων εταιρειών αλλά και καταναλωτών. Εκτενής αναφορά στην τεχνολογία του Υπολογιστικού Νέφους θα γίνει στην ενότητα 2.1 του κεφαλαίου 2.

1.2 Διαδίκτυο των πραγμάτων (Internet of Things)

Οι αισθητήρες σήμερα έχουν διεισδύσει για τα καλά στην καθημερινότητα μας και βρίσκουν εφαρμογή σε πολλά επίπεδα της ζωής μας. Εταιρείες ανά τον κόσμο λόγο των πλεονεκτημάτων τους πειραματίζονται με καινοτόμες και έξυπνες εφαρμογές προάγοντας παράλληλα τις ήδη υπάρχουσες με σκοπό την βελτίωση της ποιότητας ζωής του ανθρώπου. Πρόνοια καταστροφών, περιβαλλοντικός έλεγχος, έξυπνα σπίτια, ιατρική, καταγραφή κινήσεων είναι μερικές από τις εφαρμογές που βασίζονται σε αισθητήρες οι οποίοι μέσω τον μετρήσεων τους παράγουν αποτέλεσμα που μπορεί να εκμεταλλευτεί ο άνθρωπος.

Ενδεικτικό της νέας τάσης αυτής είναι το γεγονός ότι έχουμε ήδη παγκοσμίως, περίπου 18 δισεκατομμύρια συσκευές που διεισδύουν στο διαδίκτυο και προβλέπεται ότι ο αριθμός αυτός να διπλασιαστεί. Από την ιδέα του Διαδικτύου των Πραγμάτων δεν θα μπορούσε να λείπει το Υπολογιστικό Νέφος. Η τεχνολογία Νέφους έρχεται να δώσει μια νέα διάσταση, καθώς εξυπηρετεί τους δημιουργούς τέτοιων εφαρμογών. Η υλοποίηση έξυπνων εφαρμογών μέσω των υπηρεσιών που προσφέρει το Νέφος (SaaS, PaaS, IaaS) γίνεται ευκολότερη και με οικονομικά οφέλη για τον καταναλωτή. Επιπλέον, η ταχύτητα αποστολής και ο μεγάλος όγκος δεδομένων που έχει την δυνατότητα κάποιος να αποθηκεύσει και να επεξεργαστεί στο Νέφος , το καθιστά ακόμα πιο ελκυστικό.

Figure 1. The Internet of Things Was "Born" Between 2008 and 2009



Source: Cisco IBSG, April 2011

Σχήμα 1. Αναλογία των συσκευών που απαρτίζουν το Διαδίκτυο των Πραγμάτων με τον πληθυσμός της γης.

Στο σχήμα 1, παρουσιάζεται η ραγδαία ανάπτυξη της έννοιας του Διαδικτύου των Πραγμάτων και πως αυτή σχετίζεται με τον πληθυσμό της γης και τον αριθμό των συσκευών που συνδέονται στο διαδίκτυο. Όπως βλέπουμε, μέχρι το τέλος του 2015 θα υπάρχουν συνολικά 25 δισεκατομμύρια συσκευές του Διαδικτύου των Πραγμάτων και ο αντίστοιχος πληθυσμός της γης θα είναι 7.2 δισεκατομμύρια άνθρωποι. Επομένως, θα αντιστοιχούν 3.47 συνδεδεμένες συσκευές σε κάθε άνθρωπο μέχρι το τέλος του 2015. Εύκολα γίνεται αντιληπτό ότι ο όγκος της πληροφορίας που παράγουν όλες αυτές οι συσκευές είναι τεράστιος, οπότε πρέπει να βρεθούν ευέλικτες και αποτελεσματικές τεχνολογίες στην διαχείριση όλων αυτών των δεδομένων. Το συγκεκριμένο πρόβλημα λύνει η τεχνολογία του Υπολογιστικού Νέφους, η οποία με τα πλεονεκτήματα που μας προσφέρει κάνει ιδιαίτερα ελκυστική την ανάπτυξη εφαρμογών για συσκευές του Διαδικτύου των Πραγμάτων σε μια υποδομή Νέφους.

1.3 Το πρόβλημα

Το FIWARE[1] είναι μια μη εμπορική πλατφόρμα που προσφέρει δημόσια υπηρεσίες ακολουθούμενες από απλές διεπαφές προγραμματισμού εφαρμογών (APIs) ώστε να διευκολυνθεί η διαδικασία ανάπτυξης «έξυπνων» εφαρμογών. Δίνει πολλά κίνητρα

[1] http://www.fiware.org/

σε νέους δημιουργούς λογισμικού στην ανάπτυξη τέτοιων εφαρμογών στα πλαίσια της υγείας, του περιβάλλοντος και της «έξυπνης» πόλης γενικότερα.

Στα πλαίσια της «έξυπνης» πόλης (Smart City) πάρα πολλές συσκευές-αισθητήρες του Διαδικτύου των Πραγμάτων συνδέονται με υπηρεσίες Υπολογιστικού Νέφους για την αξιοποίηση των δεδομένων που παράγουν ώστε να αποφευχθούν φυσικές καταστροφές (πυρκαγιές, πλημμύρες κ.α), έλεγχος περιβαλλοντικών συνθηκών, εξοικονόμηση ενέργειας, έλεγχος της κατάστασης ασθενών κ.α. Ο τεράστιος όγκος δεδομένων που παράγουν οι αισθητήρες, έχει αναγκάσει την μεταφορά της ιδέας του Διαδικτύου των Πραγμάτων εξολοκλήρου στο Υπολογιστικό Νέφος.

Η πλατφόρμα του FIWARE έχει κατευθύνει τους δημιουργούς λογισμικού, που χρησιμοποιούν υπηρεσίες της, στην ανάπτυξη «έξυπνων» εφαρμογών ή υπηρεσιών στα πλαίσια της «έξυπνης πόλης» και κατ' επέκταση στην ιδέα του Διαδικτύου των Πραγμάτων. Ήδη τα τελευταία χρόνια, η κοινότητα του FIWARE έχει κάνει σημαντικά βήματα στην ανάπτυξη τέτοιων υπηρεσιών που βοηθούν στην δημιουργία πιο σύνθετων εφαρμογών. Σ' αυτή την κατεύθυνση, αναπτύχθηκε η υπηρεσία Sensor Data Collection[2], η οποία αν και βρίσκεται ακόμα σε δοκιμαστικό στάδιο υπόσχεται να λύσει το πρόβλημα σχετικά με την συλλογή των δεδομένων από διαφορετικές συσκευές-αισθητήρες.

Η υπηρεσία Sensor Data Collection αναπτύχθηκε για την δημιουργία μιας πύλης δικτύου (gateway) σε μια συσκευή με δυνατότητα σύνδεσης στο διαδίκτυο, ώστε να μπορεί η συσκευή να λαμβάνει τα σήματα διαφορετικών αισθητήρων και με κατάλληλους μηχανισμούς να τα στέλνει σε διάφορες άλλες υπηρεσίες ανεπτυγμένες στο περιβάλλον του FIWARE. Αρχικά δημιουργήθηκε, για την υποστήριξη ιατρικών συσκευών και η συσκευή που παίζει το ρόλο της πύλης δικτύου είναι μια android συσκευή. Με την συμμετοχή και άλλων δημιουργών λογισμικού θα μπορεί αυτή η υπηρεσία να επεκταθεί στην υποστήριξη πολλών αισθητήρων όπως περιβαλλοντικούς και κίνησης, και παράλληλα να μπορεί να υποστηρίξει και άλλες συσκευές ως πύλη δικτύου όπως συσκευές με λειτουργικό iOS. Σύμφωνα με το κείμενο τεκμηρίωσης που προσφέρεται [1], η υλοποίηση της υπηρεσίας βασίζεται σε υπηρεσίες που ονομάζονται Protocol Adapters. Αυτές οι υπηρεσίες αναπτύσσονται ειδικά για κάποιο πρωτόκολλο επικοινωνίας (Wi-Fi, ZigBee, Bluetooth κ.α) και επιπλέον για μια συγκεκριμένη συσκευή. Η υπηρεσία παρέχει διεπαφή προγραμματισμού εφαρμογών (API), σύμφωνα με το οποίο μπορούν να πραγματοποιηθούν λειτουργίες όπως συνδρομή σε μια συσκευή ώστε να υπάρξει ειδοποίηση σε περίπτωση που θέλει να στείλει δεδομένα, όταν μια συσκευή είναι έτοιμη να στείλει δεδομένα τότε ειδοποιείται η υπηρεσία. Επίσης, υπάρχει η δυνατότητα να ανακτηθούν η λίστα των χαρακτηριστικών μιας συσκευής και ανάκτηση της τελευταίας τιμής μιας ομάδας χαρακτηριστικών της συσκευής κ.α. Η απόκριση (response) που προκύπτει στην κλήση μιας μεθόδου του API είναι σε πρότυπο δεδομένων JSON (JavaScript Object Notation).

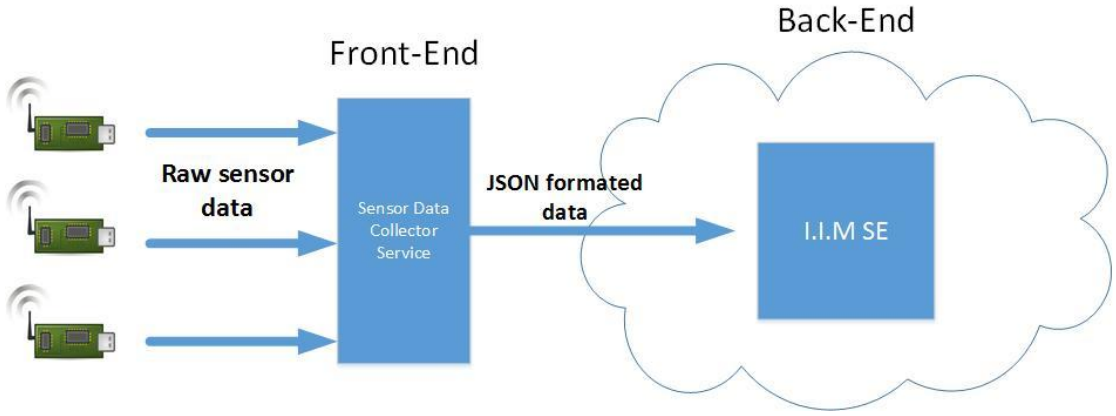
[2] http://catalogue.fi-star.eu/enablers/sensor-data-collection-service

Επομένως δύο κύρια προβλήματα που λύνει η υπηρεσία Sensor Data Collection είναι τα εξής:

- Το πρόβλημα της ύπαρξης πολλών διαφορετικών πρωτοκόλλων επικοινωνίας μεταξύ της συσκευής και μιας πύλης δικτύου. Τα κυριότερα πρωτόκολλα επικοινωνίας σε αισθητήρες του εμπορίου είναι το Wi-Fi, Bluetooth, ZigBee κ.α. Έτσι είναι επιβεβλημένη η ανάγκη μιας υπηρεσίας που θα υλοποιεί διεπαφές σύμφωνα με αυτά τα πρότυπα ώστε να γίνεται εύκολη η επικοινωνία μεταξύ της υπηρεσίας και του αισθητήρα.

- Επίσης ένα δεύτερο πρόβλημα που προκύπτει με τις πολλές διαφορετικές συσκευές είναι ότι κάθε εταιρία παρέχει ένα δικό της API για την συλλογή των δεδομένων από τους αισθητήρες, με αποτέλεσμα η υλοποίηση μιας υπηρεσίας για το σύνολο των αισθητήρων του εμπορίου μοιάζει αρκετά δύσκολη.

Κίνητρο αποτέλεσε η απουσία από το περιβάλλον του FIWARE μιας υπηρεσίας, η οποία θα έχει τη δυνατότητα της διαχείρισης, αποθήκευσης και διαμοιρασμού της πληροφορίας που προσφέρουν υπηρεσίες όπως ο Sensor Data Collection σ' ένα πλήθος χρηστών που έχουν κάνει συνδρομή στα δεδομένα κάποιου αισθητήρα. Οι χρήστες που θα χρησιμοποιούν αυτήν την υπηρεσία θα μπορεί να είναι φυσικά πρόσωπα, υπηρεσίες ή και εφαρμογές που αναπτύσσονται στο περιβάλλον του FIWARE και σε άλλα περιβάλλοντα ανάπτυξης.



Σχήμα 2. Διεπαφή χρήστη «Front-End» και διεπαφή διαχείρισης συστήματος «Back-End»

Διεπαφή χρήστη «Front-End» συγκεντρώνει όλα τα σήματα των διαφορετικών αισθητήρων και έπειτα από κατάλληλη επεξεργασία τα στέλνει μέσω διαδικτύου σε μια υπηρεσία στο Νέφος, όπου γίνεται η διαχείριση τους. Τον ρόλο αυτής της

υπηρεσίας μπορεί να πάρει η υπηρεσία Sensor Data Collection στο περιβάλλον FIWARE.

Διεπαφή διαχείρισης συστήματος «Back-End» αποθηκεύει, επεξεργάζεται και διαμοιράζει τα δεδομένα των αισθητήρων ανάλογα με τις συνδρομές των χρηστών της εφαρμογής.

Για την απλοποίηση του παραπάνω προβλήματος θεωρήσαμε τις δύο διεπαφές όπως παρουσιάστηκαν στο σχήμα 2 ως «μαύρα κουτιά», και στην συγκεκριμένη διπλωματική εργασία επικεντρωθήκαμε στην υλοποίηση της διεπαφής διαχείρισης συστήματος. Επομένως, θεωρούμε ότι είσοδος στο σύστημα μας είναι δεδομένα πολλών αισθητήρων, τα οποία υπακούν στο πρότυπο δεδομένων που ονομάζεται JSON. Εκτενής αναφορά στο πρότυπο δεδομένων JSON (JavaScript Object Notation) θα γίνει στο κεφάλαιο **2** στην υποενότητα **2.1.3.6.**

1.4 Περιγραφή της λύσης

Η λύση που προτείνουμε στο παραπάνω πρόβλημα είναι η υπηρεσία I.I.M (Intellicloud IoT Management), η οποία διαχειρίζεται τους χρήστες και τους αισθητήρες με στόχο την άμεση ενημέρωση συνδρομητών-χρηστών, σε ενημερώσεις των δεδομένων του εκάστοτε αισθητήρα.

Οι βασικές λειτουργίες που υποστηρίζονται:

- Προσθήκη/αφαίρεση/ανανέωση αισθητήρων από τον διαχειριστή.
- Προσθήκη/αφαίρεση/ανανέωση συνδρομών χρήστη.
- Προσθήκη/αφαίρεση/ανανέωση αδειών σε χρήστες για την συνδρομή αισθητήρων, από τον διαχειριστή.
- Άμεση ενημέρωση συνδρομητών σε ενημερώσεις αισθητήρων που έχουν συνδρομή.
- Ταυτοποίηση/διαγραφή χρηστών από τον διαχειριστή.
- Βάση δεδομένων με το ιστορικό των δεδομένων των αισθητήρων.

Η συνεισφορά της διπλωματικής εργασίας είναι η υλοποίηση του I.I.M SE, με τα κυριότερα πλεονεκτήματα να παρουσιάζονται παρακάτω:

- ***Επεκτάσιμο***, μπορούν να προστεθούν υπηρεσίες για την επέκταση των λειτουργιών του SE, όπως για παράδειγμα εργαλεία ανάλυσης δεδομένων.

- ***Εύκολο στην χρήση***, με ένα απλό και εύχρηστο REST (Representational State Transfer) API που προσφέρεται σε οποιονδήποτε θέλει να το χρησιμοποιήσει.
- ***Υποστήριξη πολλών διαφορετικών αισθητήρων και χρηστών/ελαστικότητα,*** προσαρμογή των υπολογιστικών πόρων της υποδομής νέφους σε περίπτωση απαίτησης από την εφαρμογή.
- ***Συμβατότητα,*** με υπηρεσίες που προσφέρουν περιβάλλοντα Νέφους όπως το FIWARE.

1.5 Δομή της εργασίας

Στο **2ο κεφάλαιο** γίνεται εκτενέστερη αναφορά στην τεχνολογία Υπολογιστικού Νέφους. Δίνεται ο ορισμός και αναλύονται τα πλεονεκτήματα και οι αδυναμίες της τεχνολογίας Νέφους, τί προσφέρει σήμερα στον κόσμο της τεχνολογίας αλλά και πού έχουν εντοπιστεί αδυναμίες. Επιπλέον, γίνεται ανάλυση όσων αφορά τα μοντέλα παροχής υπηρεσιών στο νέφος, τι είναι υπηρεσίες γενικού και ειδικού σκοπού. Αναφερόμαστε στην Υπηρεσιοκεντρική Αρχιτεκτονική και τις τεχνολογίες που χρησιμοποιούμε στην παρούσα διπλωματική εργασία (REST, HTTP, JSON κ.α). Στη συνέχεια, περιγράφεται η νέα τάση στο διαδίκτυο, Διαδίκτυο των Πραγμάτων (IoT), τι μας προσφέρει, πως συσχετίζεται άμεσα με το Υπολογιστικό Νέφος. Στο **3ο κεφάλαιο** παρουσιάζονται η αρχιτεκτονική του συστήματος και τα επιμέρους τμήματά της. Επίσης, αναλύεται η υλοποίηση του συστήματος μας και το πώς αυτό αναπτύχθηκε στο νέφος. Τέλος, διεξήχθη ένα πείραμα για να μπορέσουμε να μετρήσουμε την αποδοτικότητα του SE ως προς τον χρόνο απάντησης (response time). Τέλος, στο **κεφάλαιο 4** περιέχονται συμπεράσματα αλλά και προτάσεις για μελλοντική εξέλιξη του συστήματος.

Κεφάλαιο 2ο

2. Υπόβαθρο και Υπάρχουσες Τεχνολογίες

2.1 Υπολογιστικό Νέφος (Cloud Computing)

Το Υπολογιστικό Νέφος [2,3,4] είναι ένα μοντέλο που επιτρέπει εύκολη, ευέλικτη, κατά απαίτηση «on-demand» και απεριόριστη δικτυακή πρόσβαση σε μια συλλογή παραμετροποιήσιμων υπολογιστικών πόρων (δίκτυο, διακομιστές, αποθήκευση, εφαρμογές και υπηρεσίες), οι οποίοι μπορούν να δεσμευτούν και να απελευθερωθούν γρήγορα με την ελάχιστη δυνατή προσπάθεια και αλληλεπίδραση από το χρήστη. Αυτό το μοντέλο αποτελείται από τρία μοντέλα παροχής υπηρεσιών και τέσσερα μοντέλα ανάπτυξης.

Η αυξανόμενη ζήτηση του κοινού για το Υπολογιστικό Νέφος δείχνει ότι αυτή η νέα τεχνολογία προσφέρει πολλά οφέλη στους χρήστες και τις επιχειρήσεις.

- **Μειωμένο Κόστος**
  Οι χρήστες που επιλέγουν να χρησιμοποιήσουν εφαρμογές ή υπηρεσίες που παρέχονται από το Νέφος επωφελούνται οικονομικά. Η συντήρηση και οι αναβαθμίσεις του λογισμικού είναι αποκλειστική ευθύνη του πάροχου και όχι του χρήστη. Έτσι αποφεύγεται η πρόσληψη νέου εξειδικευμένου προσωπικού από τις επιχειρήσεις. Επίσης, ο εξοπλισμός που χρειάζεται μια εταιρεία μειώνεται, καθώς οι υπηρεσίες φιλοξενούνται σε απομακρυσμένους εξυπηρετητές και αρκεί οποιαδήποτε συσκευή με πρόσβαση στο διαδίκτυο ώστε αυτές να χρησιμοποιηθούν.

- **Ευελιξία**
  Η πρόσβαση σε υπηρεσίες που παρέχει το Νέφος πραγματοποιείται από οποιαδήποτε τοποθεσία ή χρονική στιγμή ζητήσει ο χρήστης, με δεδομένη την ύπαρξη σύνδεσης στο δίκτυο.

- **Ελαστικότητα**
  Ο χρήστης έχει τη δυνατότητα να διαχειρίζεται τους πόρους που του παρέχει το Νέφος σύμφωνα με τις ανάγκες του. Με αυτόν τον τρόπο οι καταναλωτές μπορούν να αυξάνουν υπολογιστική ισχύ (αποθηκευτικό χώρο, μνήμη κ.α.)

ανά τακτά χρονικά διαστήματα ή να απελευθερώνουν πόρους όταν πλέον δεν είναι χρήσιμοι.

- **Χρέωση ανά χρήση**
  Οι παρεχόμενοι πόροι είναι μετρήσιμοι και κοστολογούνται σύμφωνα με το χρονικό διάστημα χρήσης τους. Έτσι, η κατανάλωση και η χρέωσή τους ελέγχεται πλήρως από τον χρήστη. Για επιχειρήσεις που χρησιμοποιούν εφαρμογές με βάση τα ωράρια λειτουργίας τους αυτό το μοντέλο παροχής υπηρεσιών ενδείκνυται καθώς μειώνει τα έξοδα τους σε σημαντικό βαθμό.

- **Αύξηση παραγωγικότητας**
  Οι υπηρεσίες εξυπηρετούν παράλληλα και την ίδια χρονική στιγμή πολλούς χρήστες. Μια εταιρεία που απασχολεί μεγάλο αριθμό εργαζομένων έχει τη δυνατότητα να διαμερίσει αρχεία από ένα κεντρικό Η/Υ σε όλους τους εργαζόμενους ταυτόχρονα και όχι στον καθένα ξεχωριστά. Αυτό έχει σαν αποτέλεσμα την γρήγορη ενημέρωση των χρηστών και κατ' επέκταση την όσο το δυνατόν καλύτερη απόδοσή τους.

Όπως σε κάθε νέα τεχνολογία, έτσι και το Υπολογιστικό Νέφος διχάζει και προβληματίζει τους καταναλωτές, για το κατά πόσο είναι μια επαρκής και ασφαλής λύση γι' αυτούς. Πέρα από τα πλεονεκτήματα που προσφέρει το Νέφος υπάρχουν και αρκετά ερωτήματα για τα αρνητικά αποτελέσματα που είναι πιθανόν αυτό να προκαλέσει. Μερικά από αυτά είναι « Τα δεδομένα είναι ασφαλή; » , « Σε περίπτωση πτώσης του δικτύου μπορώ να έχω πρόσβαση στις υπηρεσίες; ». Θα προσπαθήσουμε παρακάτω να αναλύσουμε και να δώσουμε απαντήσεις στα κενά που έχουν παρατηρηθεί.

- **Ασφάλεια και Μυστικότητα**
  Όταν αναφερόμαστε στο Υπολογιστικό Νέφος, το ερώτημα που γεννάται είναι, αν τα δεδομένα που φιλοξενούνται σε αυτό είναι ασφαλή. Είναι γεγονός ότι οι χρήστες είναι ιδιαίτερα επιφυλακτικοί στο να εμπιστευτούν ευαίσθητα δεδομένα σε τρίτους, καθώς η ασφάλεια τους δεν διασφαλίζεται πλήρως από κανέναν πάροχο και δεν γίνονται γνωστές στους καταναλωτές τεχνικές λεπτομέρειες, όπως το πού βρίσκονται τα δεδομένα τους . Σύμφωνα με μελέτες που έχουν πραγματοποιηθεί [5,6] υπάρχει πάντα ο κίνδυνος ευαίσθητα δεδομένα να εκτίθενται δημόσια στο διαδίκτυο. Έτσι συμπεραίνουμε ότι σε μια εποχή που οι συνέπειες και τα πιθανά κόστη από πιθανές αστοχίες αυξάνονται, οι εταιρείες που διαχειρίζονται εμπιστευτικά δεδομένα πρέπει να αναπτύξουν καλύτερους τρόπους για την αξιολόγηση των πρακτικών ασφαλείας και μυστικότητας στις υπηρεσίες Υπολογιστικού Νέφους, ώστε να κερδίσουν την εμπιστοσύνη του κοινού.

- **Διαγραφή δεδομένων**
  Η ασφαλής και αποτελεσματική διαγραφή των δεδομένων κατά τον τερματισμό της συνεργασίας πελάτη-παρόχου δεν διασφαλίζεται. Δεν γίνεται γνωστή στον χρήστη η μέθοδος με την οποία διαγράφονται τα δεδομένα του και η ύπαρξη κάποιου αντίγραφου αυτών, που μπορεί να χρησιμοποιηθεί από μη εξουσιοδοτημένους χρήστες του Νέφους.

- **Νομικά ζητήματα**
  Η συμμόρφωση με το νόμο είναι ένα πολύ σημαντικό στοιχείο μεταξύ του καταναλωτή και του πάροχου από τη στιγμή που η νομοθεσία στο ζήτημα του Υπολογιστικού Νέφους θεωρείται προβληματική. Ο χρήστης, σήμερα, δεν γνωρίζει επαρκώς τα δικαιώματά του απέναντι στον πάροχο και ποιά είναι τα δικαιώματα του πάροχου πάνω στα δεδομένα του. Το ζήτημα αυτό εντείνεται από τη στιγμή που οι υποδομές Νέφους που φιλοξενούν τα δεδομένα δεν βρίσκονται σε ένα σημείο του πλανήτη αλλά διαμερίζονται ανά τον κόσμο, καθώς δεν υπάρχει ένα ενιαίο νομοθετικό πλαίσιο για την προστασία του καταναλωτή.

- **Χρήση δικτύου**
  Όπως έχουμε αναφέρει η πρόσβαση στις υπηρεσίες Νέφους πραγματοποιείται από οποιαδήποτε συσκευή έχει πρόσβαση στο διαδίκτυο, κάτι το οποίο διευκολύνει τον χρήστη. Το πλεονέκτημα αυτό, μετατρέπεται σε μειονέκτημα σε περίπτωση πιθανής διακοπής του δικτύου, καθώς γίνεται αδύνατη η πρόσβαση σε υπηρεσίες ή αρχεία που ο καταναλωτής επιθυμεί να προσπελάσει.

2.1.1 Μοντέλα παροχής υπηρεσιών

Το Υπολογιστικό Νέφος μπορεί να χαρακτηριστεί ως προς το μοντέλο της υπηρεσίας που παρέχει στους χρήστες. Τα τρία βασικά μοντέλα [2,3] είναι τα εξής: Λογισμικό ως Υπηρεσία (Software-as-a-Service), Πλατφόρμα ως Υπηρεσία (Platform-as-a-Service), Υποδομή ως Υπηρεσία (Infrastructure-as-a-Service).

- **Λογισμικό ως Υπηρεσία (SaaS)**
  Το Λογισμικό ως Υπηρεσία παρέχει πρόσβαση σε εφαρμογές-λογισμικό που φιλοξενούνται σε υποδομές Νέφους μέσω του διαδικτύου ως υπηρεσίες. Τέτοιες εφαρμογές, που σήμερα χρησιμοποιούνται κατά κόρον είναι το Gmail, Dropbox, Yahoo Mail κ.α. . Από τα τρία μοντέλα, το Λογισμικό ως Υπηρεσία προσελκύει περισσότερους χρήστες, καθώς εντοπίζονται αρκετά πλεονεκτήματα σε αυτό. Ο καταναλωτής, αποκτώντας μια τέτοια υπηρεσία

πληρώνει μια φορά την σχετική αμοιβή. Η υποστήριξη και οι ενημερώσεις παρέχονται από τον προμηθευτή, με αποτέλεσμα να μειώνονται τα έξοδα. Επιπλέον, υπάρχει η δυνατότητα πληρωμής της υπηρεσίας μέσω μιας συνδρομής, ώστε η κοστολόγηση να γίνεται μόνο για το διάστημα που αυτή χρησιμοποιείται. Τέλος, δεν δεσμεύονται πόροι στον υπολογιστή του χρήστη, όπως αποθηκευτικός χώρος, καθώς η πρόσβαση γίνεται μέσω ενός «φυλλομετρητή» (browser) χωρίς να εγκαθίσταται κάποιο επιπλέον λογισμικό.

- **Πλατφόρμα ως Υπηρεσία (PaaS)**
  Το μοντέλο αυτό παρέχει τις κατάλληλες υπηρεσίες προκειμένου κάποιος να μπορέσει να αναπτύξει, να δοκιμάσει, να διαθέσει και να συντηρήσει εφαρμογές και υπηρεσίες μέσα σε ένα ενιαίο περιβάλλον πλατφόρμας, το οποίο χαρακτηρίζεται για την ελαστικότητα και την ευελιξία που προσφέρει. Ο χρήστης δεν χρειάζεται να ανησυχεί για την συντήρηση του λογισμικού και της πλατφόρμας, καθώς ο πάροχος είναι αποκλειστικά υπεύθυνος. Με τέτοιες υπηρεσίες η ανάπτυξη εφαρμογών σε υποδομή Νέφους αυξάνεται καθώς είναι ένας έξυπνος και εύκολος τρόπος να δημιουργήσει ο καθένας ένα δικό του λογισμικό. Μεγάλες εταιρείες που προσφέρουν PaaS υπηρεσίες είναι : Microsoft (Windows Azure)[3], Google (App Engine)[4], Salesforce[5].

- **Υποδομή ως Υπηρεσία (IaaS)**
  Το μοντέλο Υποδομή ως Υπηρεσία είναι η παροχή υπολογιστικών και δικτυακών πόρων μέσω εικονικών μηχανών. Η εταιρεία ή ο ιδιώτης μπορεί να υπενοικιάσει υποδομή ανάλογα με τις απαιτήσεις εκείνης της χρονικής στιγμής με λογική, όπως και στο PaaS, αντί να προβεί στην αγορά εξοπλισμού (υπολογιστικού, δικτυακού, κλπ), το ενοικιάζει για όσο αυτό του είναι χρήσιμο. Παραδείγματα τέτοιων υποδομών είναι Windows Azure, Rackspace, Intellicloud.

2.1.2 Μοντέλα ανάπτυξης Υπολογιστικού Νέφους

Το Υπολογιστικό Νέφος, επιπλέον, χαρακτηρίζεται ως προς το μοντέλο ανάπτυξης [2,3] της υποδομής. Τα τέσσερα μοντέλα είναι τα Δημόσιο Νέφος (Public Cloud), Ιδιωτικό Νέφος (Private Cloud), Κοινοτικό Νέφος (Community Cloud), Υβριδικό Νέφος (Hybrid Cloud).

[3] http://www.microsoft.com/enterprise/microsoftcloud/
[4] https://cloud.google.com/appengine
[5] http://www.salesforce.com/

- **Δημόσιο Νέφος (Public Cloud)**
  Το Δημόσιο Νέφος αποτελεί ένα σύνολο από υπολογιστικούς πόρους, οι οποίοι διατίθενται μέσω του διαδικτύου και είναι διαθέσιμοι σε όλο το κοινό. Τα πλεονεκτήματα αυτής της υποδομής είναι η χρέωση με βάση το μοντέλο «Pay-per-use», η μεγάλη ευελιξία λόγω της άμεσης παροχής υπηρεσιών, η ύπαρξη κλιμάκωσης σε μεγαλύτερη ή μικρότερη χωρητικότητα σύμφωνα με τις ανάγκες του χρήστη.

- **Ιδιωτικό Νέφος (Private Cloud)**
  Οι υπηρεσίες παρέχονται σε χρήστες που ανήκουν σε έναν συγκεκριμένο οργανισμό σε αντίθεση με το δημόσιο νέφος που απευθύνεται σε όλους. Τέτοιου είδους λύσεις υιοθετούν εταιρείες που επιθυμούν τον πλήρη έλεγχο των υπηρεσιών, οι οποίες ακολουθούν ένα κοινό νομικό πλαίσιο παροχής ώστε να διασφαλίζεται η ασφάλειά τους. Σημαντικό χαρακτηριστικό είναι το μεγάλο κόστος δημιουργίας και λειτουργίας μιας τέτοιας υποδομής.

- **Κοινοτικό Νέφος (Community Cloud)**
  Η δομή αυτού του μοντέλου είναι κοινή για πολλούς οργανισμούς και υποστηρίζει μια συγκεκριμένη κοινότητα με κοινά ενδιαφέροντα (ασφάλεια, συμμόρφωση κ.α.) και ανάγκες.

- **Υβριδικό Νέφος (Hybrid Cloud)**
  Το Υβριδικό Νέφος είναι το μοντέλο που συνδυάζει όλα τα παραπάνω Νέφη, ώστε να δημιουργηθεί μια υποδομή. Η χρήση αυτού του μοντέλου παρέχει σε ορισμένες περιπτώσεις μεγαλύτερη ευελιξία σε επιχειρήσεις, δεδομένου ότι είναι εφικτή μια ποικιλία συνδυασμών των πόρων. Οι συνδυασμοί αυτοί επιτρέπουν την αποτελεσματικότερη αντιμετώπιση φόρτου εργασίας και μεγάλων απαιτήσεων υπολογιστικών πόρων.

2.1.3 Υπάρχουσες Τεχνολογίες

2.1.3.1Υποδομή FIWARE και Intellicloud

Το FIWARE είναι μια μη εμπορική πλατφόρμα που προσφέρει υπηρεσίες γενικού σκοπού (GEs) αλλά και ειδικού σκοπού (SEs) ακολουθούμενες από απλές διεπαφές προγραμματισμού εφαρμογών (APIs) ώστε να είναι δυνατή η υλοποίηση «έξυπνων» εφαρμογών. Οι Υπηρεσίες γενικού σκοπού (Generic Enablers) παρέχονται από

υποδομές Υπολογιστικού Νέφους ως SaaS για την ανάπτυξη εφαρμογών. Πολλές τέτοιες υπηρεσίες συνθέτουν μια υπηρεσία ειδικού σκοπού (Specific Enabler), η οποία συντελεί στην επίλυση ενός πιο σύνθετου προβλήματος. Επιπλέον, δίνεται η δυνατότητα στον χρήστη να αποκτήσει την κατάλληλη τεχνογνωσία ώστε να εξοικειωθεί με τις υπηρεσίες καθώς παρέχονται πειράματα και αποτελέσματα των υπηρεσιών αυτών από χρήστες που ήδη έχουν κάνει χρήση αυτών. Οι υπηρεσίες διανέμονται δωρεάν και είναι δημόσιες ώστε το κοινό να έχει πρόσβαση σε αυτές. Το γεγονός αυτό κάνει την πλατφόρμα ελκυστική στους χρήστες που θέλουν να δημιουργήσουν δικές τους εφαρμογές. Οι παροχές του FIWARE δεν σταματάνε στην διανομή υπηρεσιών με την μορφή λογισμικού αλλά δίνει την δυνατότητα σε προγραμματιστές να αποκτήσουν υπηρεσίες ως υποδομή (IaaS) δημιουργώντας εικονικές μηχανές και δεσμεύοντας υπολογιστικούς πόρους (μνήμη, επεξεργαστική ισχύ, αποθηκευτικό χώρο).

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 εξολοκλήρου στο Πολυτεχνείο Κρήτης και αυτή τη στιγμή περιλαμβάνει 128 πυρήνες επεξεργαστικής ισχύς, 284 GB RAM και 12 TB σε σκληρό δίσκο. Το λογισμικό που είναι υπεύθυνο για τη λειτουργία του Intellicloud είναι βασισμένο σε Openstack Grizzly.

2.1.3.2 Openstack

Το Openstack [7] είναι ένα λογισμικό ανοιχτού κώδικα, το οποίο επιτρέπει την δημιουργία μιας υποδομής Υπολογιστικού Νέφους. Οι θεμελιωτές της δημιουργίας αυτού του λογισμικού είναι οι Rackspace Hosting και η NASA, όταν το 2010 αποφάσισαν να δουλέψουν πάνω σε ένα κοινό έργο για την υλοποίηση υποδομής Υπολογιστικού Νέφους. Σήμερα, είναι διαδεδομένο σε όλο τον κόσμο καθώς πάνω από διακόσιες εταιρείες χρησιμοποιούν Openstack στα Νέφη τους όπως είναι η Yahoo[6], Intel[7], Oracle[8] κ.α. Τα συστήματα που είναι σχεδιασμένα στα πρότυπα του Openstack αποτελούνται από μια κεντρική αρχιτεκτονική και από επιμέρους μικρότερα κομμάτια κώδικα που είναι υπεύθυνα για τον έλεγχο του μεγάλου όγκου υπολογιστικών πόρων που διαχειρίζονται. Εύκολη και άμεση είναι και αλληλεπίδραση του χρήστη πάνω στο λογισμικό καθώς μέσω κάποιας κονσόλας τερματικού ή από APIs που παρέχονται μπορεί να έχει τον πλήρη έλεγχο τον φόρτου εργασίας της υποδομής και την ορθή διαμέριση των πόρων.

[6] http://www.yahoo.com
[7] http://www.intel.com/content/www/us/en/cloud-computing/intel-cloud-based-solutions.html
[8] https://cloud.oracle.com/home

2.1.3.3Υπηρεσίες Γενικού Σκοπού (Generic Enablers) και Ειδικού Σκοπού (Specific Enablers)

Οι Generic Enablers δεν είναι τίποτα άλλο από υπηρεσίες κοινού σκοπού οι οποίες διανέμονται μέσω διαδικτύου με τη μορφή SaaS από παρόχους Υπολογιστικού Νέφους για την δημιουργία εφαρμογών. Η αρχιτεκτονική στην οποία υπακούουν τέτοιου είδους υπηρεσίες και συμβαδίζουν με τους κανόνες της, είναι η REST και ακολουθείται από ένα API το οποίο επιτρέπει την πρόσβαση σε αυτές. Τα βασικά χαρακτηριστικά ενός Generic Enabler είναι η απλότητά του και η εύκολη χρήση του οποιαδήποτε στιγμή ζητηθεί από το χρήστη. Τέτοιες επαναχρησιμοποιήσιμες υπηρεσίες είναι πολύ σημαντικές σ' ένα Υπολογιστικό Νέφος καθώς διευκολύνουν τους προγραμματιστές στην ανάπτυξη πιο σύνθετων συστημάτων. Τέτοιες υπηρεσίες παρέχει η υποδομή FIWARE μέσω της σελίδας http://catalogue.fiware.org/enablers.

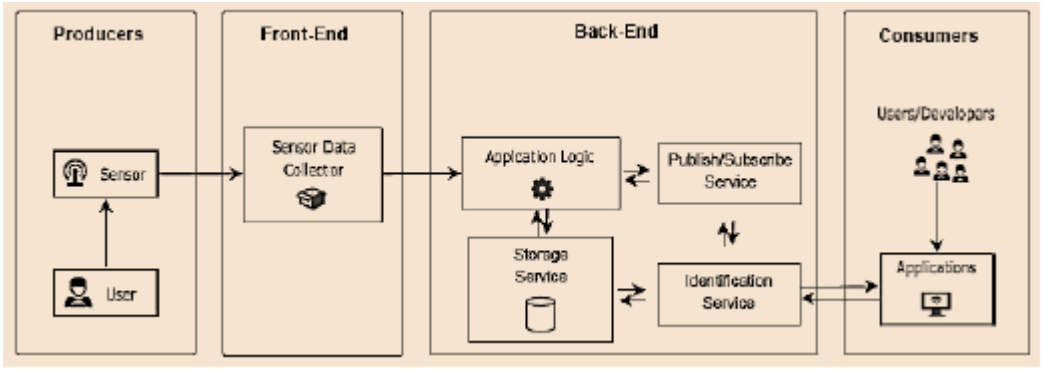
Η επικοινωνία και η σύνδεση παραπάνω του ενός Generic Enabler δημιουργεί έναν Specific Enabler που έχει ως στόχο την υλοποίηση μιας πιο σύνθετης λειτουργίας.. Σε περίπτωση, που ο προγραμματιστής του Specific Enabler θελήσει να εντάξει μια νέα υπηρεσία, αυτό γίνεται πολύ εύκολα χωρίς να χρειαστεί να επέμβει στα επιμέρους κομμάτια/Generic Enablers που απαρτίζουν τον SE. Σημαντικό επίσης πλεονέκτημα κρίνεται και η αντικατάσταση μιας επιμέρους υπηρεσίας/Generic Enabler αν για παράδειγμα έχει διατεθεί μια καινούρια βελτιωμένη έκδοση της υπηρεσίας προς αντικατάσταση. Με αυτόν τον τρόπο ο SE παραμένει ενημερωμένος χωρίς να χρειαστεί να αλλάξει όλο το σύστημα εξολοκλήρου.

2.1.3.4Υπηρεσιοκεντρική αρχιτεκτονική (Service Oriented Architecture)

Για την υλοποίηση συστημάτων στα πλαίσια του IoT είναι ευρέως γνωστή η υπηρεσιο-κεντρική αρχιτεκτονική [8], με βάση την οποία υλοποιήθηκε και το δικό μας σύστημα. Η Υπηρεσιο-κεντρική Αρχιτεκτονική βασίζεται στη λογική ότι οποιοδήποτε μεγάλο πρόβλημα μπορεί να λυθεί βέλτιστα και να αντιμετωπιστεί αποτελεσματικά η πολυπλοκότητά του εάν το διαιρέσουμε σε μικρότερα προβλήματα τα οποία το συνθέτουν. Κατά συνέπεια το ίδιο ισχύει και σε συστήματα τα οποία βασίζονται σε υπηρεσίες που φιλοξενούνται στο διαδίκτυο, όπως το δικό μας. Η αρχιτεκτονική αυτή είναι ένα ευέλικτο σύνολο αρχών σχεδιασμού και τεχνολογίας που χρησιμοποιείται στη φάση της ανάπτυξης των συστημάτων. Τέτοια συστήματα αποτελούνται από ένα σύνολο υπηρεσιών που επικοινωνούν μεταξύ τους και μπορούν να χρησιμοποιηθούν στο πλαίσιο πολλαπλών ,χωριστών συστημάτων από

διάφορους επιχειρηματικούς τομείς. Κάποια από τα πλεονεκτήματα που την κάνουν ιδιαίτερα γνωστή και ελκυστική είναι :

- Οι υπηρεσίες είναι επαναχρησιμοποιήσιμες και μπορούν να διατεθούν σε μεγάλη κλίμακα.
- Γρηγορότερη και αποτελεσματικότερη αποσφαλμάτωση.
- Μικρότερος χρόνος διάθεσης νέων προϊόντων και εφαρμογών.
- Οι υπηρεσίες αυτές δεν δεσμεύονται από το σύστημα, αλλά είναι δυνατή η αντικατάστασή τους.
- Σε ένα υπάρχον σύστημα η ένταξη και η ενσωμάτωση μιας νέας υπηρεσίας δεν απαιτεί αλλαγές στον μηχανισμό λειτουργίας της υπηρεσίας.



Σχήμα 3. Υπηρεσιο-κεντρική αρχιτεκτονική

Το σχήμα 3 χωρίζεται σε 4 τμήματα που αλληλεπιδρούν στην υλοποίηση μιας εφαρμογής στα πλαίσια του Διαδικτύου των Πραγμάτων. Αριστερά είναι οι «Παραγωγοί» (Producers), στους οποίους περιλαμβάνονται οι αισθητήρες που παράγουν την πληροφορία. Αυτοί οι αισθητήρες μπορεί να αλληλεπιδρούν με χρήστες όπως συμβαίνει στους ιατρικούς αισθητήρες. Στο τμήμα «Διεπαφή Χρήστη» (Front-End) περιλαμβάνεται η υπηρεσία που παίζει το ρόλο μιας πύλης δικτύου ανάμεσα στα δεδομένα που στέλνει ο αισθητήρας και τα δεδομένα που διαχειρίζεται η εφαρμογή μέσω της «Διεπαφής Διαχείρισης Συστήματος» (Back-End). Στην «Διεπαφή Διαχείρισης Συστήματος» περιλαμβάνονται όλες εκείνες οι υπηρεσίες γενικού σκοπού για την ταυτοποίηση του χρήστη, δημιουργία συνδρομής, αποθήκευση των δεδομένων της εφαρμογής. Όλες αυτές οι υπηρεσίες επικοινωνούν με την Λογική Συστήματος (Application Logic), η οποία κάνει χρήση κανόνων, ελέγχων και συνθηκών για την μεταφορά της πληροφορίας στις επιμέρους υπηρεσίες. Τέλος, στο τμήμα «Καταναλωτές» περιλαμβάνονται είτε τελικοί χρήστες, είτε άλλες εφαρμογές που επικοινωνούν με το API που παρέχεται.

2.1.3.5Πρωτόκολλο επικοινωνίας HTTP και REST API

Το **HTTP** (Hypertext Transfer Protocol) πρωτόκολλο επικοινωνίας αποτελεί το κύριο πρωτόκολλο που χρησιμοποιείται στους φυλλομετρητές του Παγκοσμίου Ιστού για να μεταφέρει δεδομένα ανάμεσα σε έναν δακομιστή (server) και έναν πελάτη (client). Οι τέσσερις βασικότερες μέθοδοι που ορίζονται από το πρωτόκολλο αυτό είναι τύπου CRUD (Create – Read – Update – Delete) και αναφέρονται παρακάτω:

- ***GET***, όταν ο πελάτης ζητάει ένα συγκεκριμένο πόρο (resource) από τον διακομιστή.
- ***POST***, όταν ο πελάτης στέλνει δεδομένα στον διακομιστή π.χ για εισαγωγή σε μια βάση δεδομένων.
- ***PUT***, όταν ο πελάτης στέλνει δεδομένα για ένα ήδη υπάρχοντα πόρο (resource), ώστε να ανανεωθεί η τιμή του.
- ***DELETE***, όταν ο πελάτης θέλει να διαγράψει έναν συγκεκριμένο πόρο από τον διακομιστή.

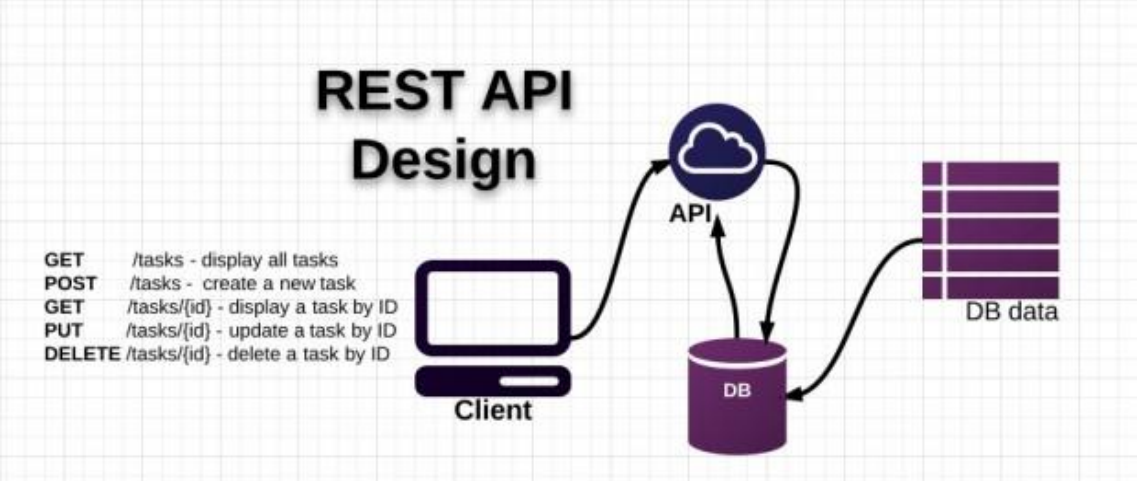
Αυτά όλα συμβαίνουν από την πλευρά του πελάτη όταν στέλνει αίτημα στον διακομιστή για μία συγκεκριμένη μέθοδο. Από την πλευρά του διακομιστή σε κάθε αίτημα που καλείται να επεξεργαστεί, αποκρίνεται στον πελάτη μ' έναν κωδικό κατάστασης ώστε να γνωρίζει αν για παράδειγμα βρέθηκε ο συγκεκριμένος πόρος που ζήτησε. Αυτοί οι κωδικοί αριθμοί χωρίζονται σε 5 μεγάλες κατηγορίες:

- Informational 1XX
- Successful 2XX
- Redirection 3XX
- Client Error 4XX
- Server Error 5XX

Το **REST** [9] είναι μια αρχιτεκτονική, η οποία αποτελείται από κανόνες και περιορισμούς για την δημιουργία λογισμικού και υπηρεσιών που φιλοξενούνται στο διαδίκτυο. Ο συγκεκριμένες υπηρεσίες επικοινωνούν μέσω του πρωτοκόλλου HTTP (Hypertext Transfer Protocol) με την χρήση των μεθόδων GET,POST,PUT,DELETE κ.α. που χρησιμοποιούνται για αιτήσεις από πελάτες σε εξυπηρετητές στο διαδίκτυο. Σε αυτό το μοντέλο, πελάτης-εξυπηρετητής, είναι βασισμένη η αρχιτεκτονική REST. Με την χρήση των μεθόδων οι πελάτες ξεκινούν αιτήματα (request) προ τους εξυπηρετητές, οι οποίοι με τη σειρά τους, επιστρέφουν τις ανάλογες απαντήσεις (response) κωδικοποιημένες συνήθως σε μορφή JSON ή XML.

Ένα **API** (Application Programming Interface) αποτελείται από ένα σύνολο οδηγιών και προτύπων προγραμματισμού για πρόσβαση σε υπηρεσίες που βρίσκονται στο

διαδίκτυο. Οι προμηθευτές των υπηρεσιών προσφέρουν τα λεγόμενα APIs στο κοινό, έτσι ώστε οι προγραμματιστές να μπορούν να σχεδιάσουν εφαρμογές που να βασίζονται σε αυτά. Επιπλέον, τα APIs επιτρέπουν σε ένα πρόγραμμα να αλληλεπιδρά με ένα άλλο, χωρίς να παρεμβάλλεται ο χρήστης. Για παράδειγμα, όταν μια διαδικτυακή εφαρμογή προσφέρει στον χρήστη την δυνατότητα εισαγωγής (login) του με λογαριασμό του Facebook, τότε η διαδικτυακή εφαρμογή κάνει χρήση του API που προσφέρει η εταιρεία Facebook για την ταυτοποίηση χρηστών. Με άλλα λόγια αναθέτουμε σε μια εξωτερική υπηρεσία (Facebook) την ταυτοποίηση των χρηστών της εφαρμογής μας για παροχή επιπλέον ασφάλειας, αξιοπιστίας και ευκολίας στον χρήστη.



Σχήμα 4. REST API

2.1.3.6JSON (JavaScript Object Notation)

Στις σύγχρονες διαδικτυακές υπηρεσίες το πρότυπο δεδομένων **JSON** [10] χρησιμοποιείται ως εναλλακτική της XML (Extensible Markup Language), για την επικοινωνία μεταξύ διακομιστή και πελάτη ή ενός διακομιστή με κάποιον άλλο διακομιστή. Τα πλεονεκτήματα της σε σχέση με την XML είναι πολλά μερικά από τα οποία αναφέρονται παρακάτω:

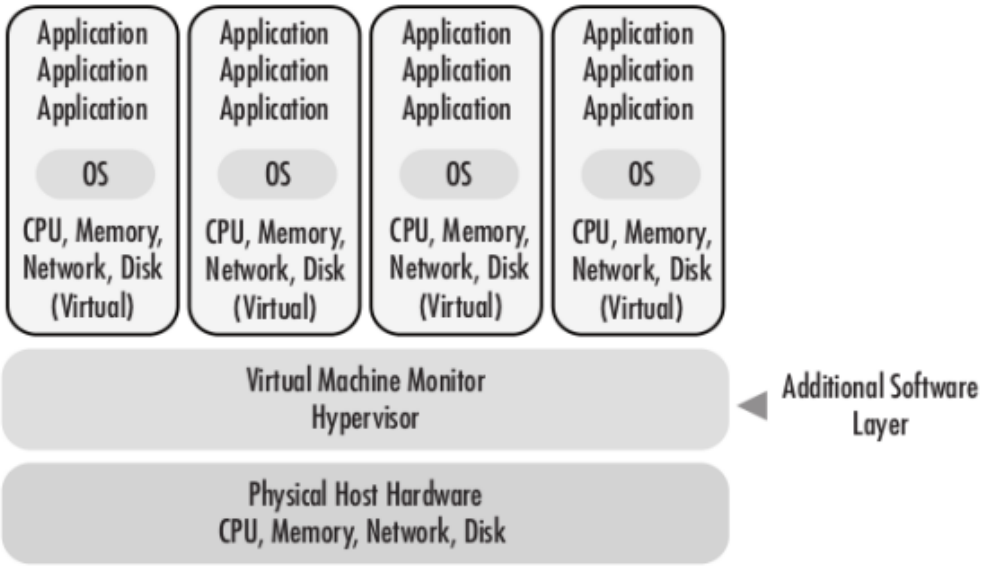
- Η ανάλυση ενός αρχείου JSON γίνεται πολύ πιο εύκολα από την ανάλυση ενός αρχείου XML.

- Στην XML σε αντίθεση με την JSON υπάρχει αρκετός πλεονασμός (redundancy), οπότε θέλει περισσότερη επεξεργασία.
- Είναι ευανάγνωστη στον άνθρωπο σε αντίθεση με την XML.

Από τα παραπάνω γίνεται σαφής η υπεροχή της περιγραφής JSON σε σχέση με την XML στην ανταλλαγή πληροφορίας στον Παγκόσμιο Ιστό τα τελευταία χρόνια.

2.1.4 Εικονοποίηση πόρων σε Υπολογιστικό Νέφος

Στο Υπολογιστικό Νέφος ιδιαίτερη βαρύτητα δίνεται από κάθε πάροχο στον διαμοιρασμό των υπολογιστικών πόρων, έτσι ώστε οι καταναλωτές να παραμένουν ευχαριστημένοι με τις παρεχόμενες υπηρεσίες. Το λογισμικό το οποίο είναι υπεύθυνο για το πώς θα διανεμηθούν οι διαθέσιμοι πόροι του Υπολογιστικού Νέφους ονομάζεται επιμελητής εικονικών μηχανών (hypervisor) [11].



Σχήμα 5. Η σχέση μεταξύ φυσικής υποδομής - hypervisor - εικονικών μηχανών

Το πρόσθετο στρώμα λογισμικού που διακρίνεται στο σχήμα 5 μεταξύ της φυσικής υποδομής (CPU, μνήμη, δίκτυο) και των εικονικών μηχανών αναφέρεται ως Επιμελητής Εικονικών Μηχανών – Hypevisor. Στην ουσία πρόκειται για ένα λειτουργικό σύστημα το οποίο εξυπηρετεί τις αιτήσεις που κάνουν οι εικονικές μηχανές προς τη φυσική υποδομή για τη χρήση συγκεκριμένων υπολογιστικών πόρων.

Το σχήμα 5 μοντελοποιεί την αλληλεπίδραση τριών οντοτήτων της Υποδομής Νέφους που παίζουν καθοριστικό ρόλο στην εξυπηρέτηση των καταναλωτών

- Οι ***Εικονικές Μηχανές*** αποτελούν ένα σύνολο από εικονικούς πόρους (εικονική επεξεργαστική ισχύ, εικονική μνήμη, εικονικός αποθηκευτικός χώρος κ.α), ένα Φιλοξενούμενο Λειτουργικό Σύστημα και τις εφαρμογές που εκτελούνται πάνω σ' αυτό το Λειτουργικό Σύστημα.
- Ο ***Επιμελητής Εικονικών Μηχανών***, ο οποίος διανέμει τους πόρους στις Εικονικές Μηχανές ανάλογα με τις απαιτήσεις τους.
- Οι ***Φυσικοί Πόροι***, αποτελούν την πραγματική επεξεργαστική ισχύ, μνήμη, αποθηκευτικό χώρο και δίκτυο που διαθέτη η Υποδομή Νέφους.

Η Υποδομή Νέφους του εργαστηρίου μας, Intellicloud είναι βασισμένη στην ανοιχτού κώδικα πλατφόρμα Openstack και ο Επιμελητής Εικονικών Μηχανών που βασίζεται η όλη λειτουργία της υποδομής είναι ο KVM Hypervisor[9].

2.1.5 Πάροχοι Υπολογιστικού Νέφους

Η τεχνολογία του Υπολογιστικού Νέφους έχει προσελκύσει το ενδιαφέρον μεγάλων και αναγνωρισμένων για το έργο τους στο χώρο της τεχνολογίας, εταιρείες. Η κάθε μια προσφέρει ξεχωριστές υπηρεσίες και οι υποδομές της υπακούουν σε διαφορετικές αρχιτεκτονικές. Παρακάτω γίνεται εκτενέστερη αναφορά σε μερικές από αυτές.

- **Amazon**
  Ένας από του πρώτους κολοσσούς που ανέπτυξαν την τεχνολογία Νέφους είναι η Amazon. Σήμερα με το πλήθος των υπηρεσιών που παρέχει βρίσκεται δικαίως στις πρώτες θέσεις προτίμησης από τους καταναλωτές. Έκτος των υπηρεσιών που διαθέτει στο κοινό, προσφέρει υπηρεσίες αποθήκευσης (Amazon Simple Storage Service[10]), πλατφόρμες (Amazon Elastic Compute Cloud[11]) και βάσεις δεδομένων (Amazon Simple DB[12]). Η Amazon έχει υλοποιήσει την υποδομή Νέφους πάνω σε μια δική της αρχιτεκτονική.

- **IBM**
  Η IBM προσφέρει υπηρεσίες Υπολογιστικού Νέφους για να βοηθήσει επιχειρήσεις να εκμεταλλευτούν αυτό το όλο και πιο ελκυστικό μοντέλο. Επιπλέον, προσφέρει υπηρεσίες SaaS, PaaS, IaaS. Το σημαντικό χαρακτηριστικό της είναι ότι παρέχει τη δυνατότητα υλοποίησης μιας λύσης Νέφους συνδυάζοντας τρία μοντέλα ανάπτυξης, ιδιωτικό, δημόσιο και

[9] http://www.linux-kvm.org/page/Main_Page
[10] http://aws.amazon.com/ses/
[11] http://aws.amazon.com/ec2/
[12] http://aws.amazon.com/simpledb/

υβριδικό. Η αρχιτεκτονική πάνω στην οποία έχει βασιστεί η υποδομή της IBM είναι, το ήδη γνωστό Openstack.

- **VMware**[13]
  Η VMware είναι ο γίγαντας της εικονοποίησης στο Νέφος. Παρέχει, ίσως την κορυφαία πλατφόρμα εικονοποίησης, VMware vSphere 5.1,την οποία μεγάλες εταιρείες, όπως η Stratogen[14] , χρησιμοποιούν ώστε να δημιουργήσουν το δικό τους υβριδικό μοντέλο Νέφους. Τα προϊόντα-υπηρεσίες της είναι συμβατά με τρία γνωστότερα λειτουργικά συστήματα, Windows,Linux,Mac OS X.

- **Microsoft**
  Η Microsoft διαθέτει υπηρεσίες Νέφους για επιχειρήσεις και προγραμματιστές. Ο ακρογωνιαίος λίθος των υπηρεσιών της Microsoft είναι η πλατφόρμα Azure Services Platform[15]. Η πλατφόρμα αυτή παρέχει μεγάλο εύρος εργαλείων για την ανάπτυξη, φιλοξενία και διαχείριση εφαρμογών από προγραμματιστές με την μορφή PaaS-IaaS. Επιπλέον, διαθέσιμη στο κοινό είναι η υπηρεσία Microsoft SQL Service, η οποία επιτρέπει την αποθήκευση δεδομένων, σχεσιακών ερωτημάτων πάνω σε αυτά.

2.2 Διαδίκτυο των Πραγμάτων (Internet of Things)

Διαδίκτυο των πραγμάτων (Internet of Things) [12] ονομάζεται το δίκτυο συσκευών που μεταδίδουν/αξιοποιούν δεδομένα από το φυσικό περιβάλλον για να παρέχουν κάποια υπηρεσία και μπορούν να επικοινωνούν (και) μέσω διαδικτύου. Π.χ. μπορεί αυτό να είναι ένα έξυπνο τηλέφωνο (smartphone) ή ένας αισθητήρας υγρασίας εδάφους σε ένα χωράφι, που στέλνει μετρήσεις σε κάποια διαδικτυακή υπηρεσία μέσω του GSM δικτύου. Έχουμε ήδη παγκοσμίως, περίπου 18 δισεκατομμύρια συσκευές που μπορούν να συνδεθούν στο διαδίκτυο και προβλέπεται να φτάσουν στις 50 δις, μέχρι το 2020. Το μεγαλύτερο μέρος από αυτές δεν είναι υπολογιστές με παραδοσιακή μορφή (smartphone/laptop/tablet), αλλά είναι "πράγματα" (things), δηλαδή κάποιο άλλο είδος συσκευής με αισθητήρες ενσωματωμένους.

Όλες αυτές οι συνδεδεμένες συσκευές που απαρτίζουν σήμερα το Διαδίκτυο των Πραγμάτων, χρησιμοποιούνται προκειμένου να συλλέξουν πληροφορίες από το περιβάλλον και την ανθρώπινη δραστηριότητα. Στόχος είναι η βελτίωση της ποιότητας ζωής των ανθρώπων και η αποφυγή απρόβλεπτων γεγονότων.

[13] http://www.vmware.com/
[14] http://www.stratogen.net/
[15] http://azure.microsoft.com/

Η συλλογή δεδομένων κρίνεται απαραίτητη στους παρακάτω τομείς:

- ***Παρακολούθηση περιβαλλοντικών συνθηκών***, για την αποφυγή ακραίων φυσικών καταστροφών, εξαγωγή συμπερασμάτων σχετικά με την ποιότητα του νερού και του αέρα κ.α.
- ***Κατασκευαστικό και βιομηχανικό τομέα***, για τον πλήρη αυτοματισμό και έλεγχο της κατασκευαστικής διαδικασίας.
- ***Διαχείριση ενεργειακών πόρων***, για την βελτιστοποίηση της ενεργειακής κατανάλωσης.
- ***Έλεγχος και επίβλεψη συστημάτων μέσων μεταφοράς***, όπως έξυπνα φανάρια, σύστημα αυτόματου παρκαρίσματος κ.α.
- ***Απομακρυσμένη ιατρική βοήθεια***, σε περιπτώσεις που ο ασθενής βρεθεί σε κρίσιμη κατάσταση ειδοποιείται άμεσα ο γιατρός.
- ***Αυτοματοποιημένα σπίτια***, για το έλεγχο όλων των ηλεκτρικών συσκευών ενός σπιτιού.

Συνεπώς η εύρεση μιας λύσης για την διαχείριση τόσο μεγάλων σε όγκο δεδομένων είναι επιβεβλημένη.

2.2.1 Διαδίκτυο των Πραγμάτων (Internet of Things) και Υπολογιστικό Νέφος (Cloud Computing)

Το πρόβλημα τις διαχείρισης και αποθήκευσης του τεράστιου όγκου δεδομένων που παράγουν αυτές οι συσκευές έρχεται να λύσει η τεχνολογία του Υπολογιστικού Νέφους. Όπως αναλύσαμε σε προηγούμενες παραγράφους του κεφαλαίου 2 τα οφέλη που μας προσφέρει το Υπολογιστικό Νέφος είναι πολλαπλά, οπότε οι δύο αυτές έννοιες δεν θα μπορούσαν παρά να είναι άμεσα συνδεδεμένες μεταξύ τους. «Έξυπνες » εφαρμογές που αναπτύσσονται στα πλαίσια μιας «έξυπνης» πόλης, θα πρέπει να έχουν την δυνατότητα της ευελιξίας και ελαστικότητας πλεονεκτήματα που αυτή τη στιγμή μόνο η τεχνολογία Υπολογιστικού Νέφους προσφέρει. Επίσης, η χρέωση ανά χρήση που προσφέρει στους καταναλωτές την κάνει ακόμα πιο ελκυστική στην χρήση της σε τέτοιου είδους εφαρμογές αφού μια συσκευή-αισθητήρας μπορεί να στέλνει συγκεκριμένες ώρες της ημέρας δεδομένα. Για παράδειγμα, αν ένας αισθητήρας παρακολουθεί αν οι περιβαλλοντικές συνθήκες ευνοούν την εκδήλωση πυρκαγιάς θα στέλνει δεδομένα σ’ ένα διάστημα που ο ήλιος είναι έντονος μόνο τους καλοκαιρινούς μήνες.

Πολλές εταιρίες που δραστηριοποιούνται στην παροχή υπηρεσιών Υπολογιστικού Νέφους ήδη έχουν αρχίσει την ανάπτυξη ειδικών εργαλείων για την αποθήκευση, επεξεργασία και ανάλυση δεδομένων που προέρχονται από συσκευές IoT. Η ανάπτυξη ειδικών υπηρεσιών σε περιβάλλοντα Υπολογιστικού Νέφους για την διαχείριση των συσκευών του Διαδικτύου των Πραγμάτων βρίσκεται ήδη σε

ανάπτυξη και αυτές οι υπηρεσίες είναι άμεσα διαθέσιμες σε δημιουργούς λογισμικού για την δημιουργία καινοτόμων εφαρμογών. Ειδικότερα, νεοσύστατες εταιρίες στον χώρο της τεχνολογίας (start-ups) έχουν την μεγαλύτερη ανάγκη από την τεχνολογία που προσφέρουν οι πάροχοι μέσω SaaS και PaaS για συσκευές του Διαδικτύου των Πραγμάτων λόγω του χαμηλού κόστους. Όλα αυτά τα εργαλεία για την αποθήκευση, επεξεργασία και ανάλυση των δεδομένων βρίσκονται όλα μαζί σε μια πλατφόρμα IoT και η ανάλυση τους γίνεται στην επόμενη ενότητα.

2.2.2 IoT περιβάλλοντα ανάπτυξης

Η ανάγκη των ανθρώπων να παρακολουθούν δεδομένα από πολλούς τύπους αισθητήρων ταυτόχρονα και να μπορούν να αναλύουν αυτά τα δεδομένα για την εξαγωγή κάποιων χρήσιμων συμπερασμάτων, οδηγεί πολλές εταιρίες στον χώρο του Υπολογιστικού Νέφους να υιοθετήσουν IoT πλατφόρμες. Είναι στην ουσία εργαλεία που παρέχουν οι πάροχοι Υπολογιστικού Νέφους στους καταναλωτές ώστε να μπορούν να επεξεργάζονται, αποθηκεύουν και αναλύουν πολλά διαφορετικά δεδομένα από πολλούς διαφορετικούς αισθητήρες της αγοράς.

Μεγαλύτερα βήματα προς την ανάπτυξη IoT πλατφόρμας στο Υπολογιστικό Νέφος έχει κάνει η εταιρία IBM, η οποία πρόσφατα επένδυσε 3 δις. δολάρια για να αναπτύξει τις πλατφόρμες **IoT Cloud Open Platform for Industries**, **IBM Bluemix IoT Zone** και **IoT Ecosystem.** Για την επίτευξη αυτού του στόχου θα συνεργαστεί με την The Weather Company, στην οποία ανήκουν τα Weather.com και The Weather Channel, και θα συλλέγει δεδομένα από τους αισθητήρες που έχει σε όλο τον κόσμο

Κάποιες ενδεικτικές IoT πλατφόρμες που υπάρχουν στο διαδίκτυο σε περιβάλλον Υπολογιστικού Νέφους, οι οποίες όμως πωλούν ειδικό υλικό (hardware) για την συλλογή των σημάτων από τους αισθητήρες είναι των εταιριών **Axeda**[16], **Thingwrox**[17], **Microstrain**[18] κ.α. [13].

2.2.3 Αισθητήρες και Διαδίκτυο των Πραγμάτων

Οι αισθητήρες σήμερα έχουν διεισδύσει για τα καλά στην καθημερινότητα μας και βρίσκουν εφαρμογή σε πολλά επίπεδα της ζωής μας. Εταιρείες ανά τον κόσμο λόγο των πλεονεκτημάτων τους πειραματίζονται με καινοτόμες και έξυπνες εφαρμογές προάγοντας παράλληλα τις είδη υπάρχουσες με σκοπό την βελτίωση της ποιότητας

[16] http://www.axeda.com
[17] http://www.thingwrox.com
[18] http://sensorcloud.com

ζωής του ανθρώπου. Πρόνοια καταστροφών, περιβαλλοντικός έλεγχος, έξυπνα σπίτια, ιατρική, καταγραφή κινήσεων είναι μερικές από τις εφαρμογές που βασίζονται σε αισθητήρες οι οποίοι μέσω τον μετρήσεων τους παράγουν αποτέλεσμα που μπορεί να εκμεταλλευτεί ο άνθρωπος.

Τα τελευταία χρόνια οι αισθητήρες έχουν μετακινηθεί στο διαδίκτυο ως Διαδίκτυο των Πραγμάτων (Internet of Things). Ειδικότερα, το Διαδίκτυο των Πραγμάτων αποτελείται από αισθητήρες-συσκευές που μεταδίδουν και αξιοποιούν τα δεδομένα από το φυσικό περιβάλλον για να παρέχουν κάποια υπηρεσία και μπορούν να επικοινωνούν μέσω διαδικτύου. Ενδεικτικό της νέας τάσης αυτής είναι το γεγονός ότι έχουμε ήδη παγκοσμίως, περίπου 18 δισεκατομμύρια συσκευές που διεισδύουν στο διαδίκτυο και προβλέπεται ότι ο αριθμός αυτός να διπλασιαστεί. Από την ιδέα του Διαδικτύου των Πραγμάτων δεν θα μπορούσε να λείπει το Υπολογιστικό Νέφος, καθώς είναι μέρος και αυτό του διαδικτύου. Η τεχνολογία Νέφους έρχεται να δώσει μια νέα διάσταση, καθώς εξυπηρετεί τους δημιουργούς τέτοιων εφαρμογών. Η υλοποίηση έξυπνων εφαρμογών μέσω των υπηρεσιών που προσφέρει το Νέφος (SaaS , PaaS, IaaS) γίνεται ευκολότερη και με σημαντική μείωση των εξόδων. Επιπλέον, η ταχύτητα αποστολής και ο μεγάλος όγκος δεδομένων που έχει την δυνατότητα κάποιος να αποθηκεύσει και να επεξεργαστεί στο Νέφος , το καθιστά ακόμα πιο ελκυστικό. Μερικοί αισθητήρες που χρησιμοποιούνται στο Διαδίκτυο των Πραγμάτων αναφέρονται παρακάτω :

Αισθητήρες κίνησης:

- **Kinect**[19]

  Μέχρι πρότινος, γνωρίζαμε ότι η καταγραφή κινήσεων ήταν δυνατή με την χρήση καμερών. Το μεγάλο κόστος για την αγορά και εγκατάσταση τέτοιων αισθητήρων ανάγκασε τους καταναλωτές να στραφούν σε άλλες, πιο οικονομικές λύσεις. Μια τέτοια είναι και ο αισθητήρας κίνησης Kinect, γνωστός για την εφαρμογή στην παιχνιδομηχανή Xbox 360. Το Kinect σήμερα, έχει ενσωματωθεί πλήρως στο Διαδίκτυο των Πραγμάτων καθώς η απλότητά του και η δυνατότητα να μεταφέρει δεδομένα στο διαδίκτυο το καθιστούν ελκυστικό. Έχουν υλοποιηθεί και συνεχώς εμφανίζονται νέες έξυπνες εφαρμογές στον τομέα της υγείας. Η πιο δημοφιλής είναι η Kinect Healthcare, μέσω της οποίας γίνεται δυνατή η απομακρυσμένη παρακολούθηση ασθενών με σοβαρά τραύματα από το γιατρό τους. Επιπλέον, πρακτικές εφαρμογές έχουν αναπτυχθεί και στον χώρο της τεχνολογίας, όπως είναι η ρομποτική. Πλέον, ο έλεγχος ρομπότ και γενικότερα αυτόνομων μηχανών επιτυγχάνεται μέσω της αναγνώρισης κινήσεων που παρέχει το Kinect.

[19] https://www.microsoft.com/en-us/kinectforwindows/

- **Leap Motion**[20]
  Ο πιο δημοφιλής αισθητήρας αναγνώρισης κινήσεων δαχτύλων και χεριών είναι ο Leap Motion Sensor. Συμβαδίζει πλήρως με το Διαδίκτυο των Πραγμάτων καθώς τα δεδομένα που παρέχει ταξιδεύουν στο διαδίκτυο μέσω λογισμικού που προσφέρει η ίδια η εταιρεία στους χρήστες. Όπως το Kinect, έτσι και το Leap έχει εφαρμογή σε υπηρεσίες που μπορούμε να χρησιμοποιούμε στην καθημερινότητα μας. Ο αριθμός των εφαρμογών αγγίζει τις διακόσιες, γεγονός που δείχνει την προτίμηση των προγραμματιστών στον συγκεκριμένο αισθητήρα, σε σχέση με το μικρό διάστημα ζωής του. Το Leap, ανοίγει τον δρόμο για τον χειρισμό οποιασδήποτε ηλεκτρονικής συσκευής με την κίνηση των δαχτύλων μέσω του διαδικτύου. Ο έλεγχος του Η/Υ μέσω του αισθητήρα είναι μια από της πιο διαδομένος εφαρμογές σήμερα. Επιπλέον, ο αισθητήρας Leap βρίσκει εφαρμογή και στην καθημερινότητα των ανθρώπων με ειδικές ανάγκες καθώς σήμερα είναι εφικτή η αναγνώριση της νοηματικής γλώσσας από οποιαδήποτε συσκευή είναι συνδεδεμένη ο αισθητήρας.

- **Netatmo**[21] **(*Περιβαλλοντικός αισθητήρας*)**
  Ο Netatmo είναι αισθητήρας παρακολούθησης κλιματικών αλλαγών (θερμοκρασία, υγρασία, πίεση κ.α.). Η κατασκευάστρια εταιρεία υποστηρίζει, μέσω υπηρεσιών που διαθέτει στο κοινό, την αποθήκευση και επεξεργασία των δεδομένων σε ιδιωτική υποδομή Νέφους. Τον αισθητήρα Netatmo τον συναντάμε σε εφαρμογές που είναι απαραίτητος ο απομακρυσμένος περιβαλλοντικός έλεγχος και η έγκαιρη πρόβλεψη φυσικών καταστροφών όπως πυρκαγιά ή παγετός.

- **Zephyr HxM Bluetooth Heart Rate Monitor (*Ιατρικός αισθητήρας*)**
  Η συγκεκριμένη ζώνη χρησιμοποιεί το πρωτόκολλο επικοινωνίας Bluetooth, ώστε να στέλνει σε μια συσκευή κινητού ιατρικά δεδομένα του ανθρώπου που την φοράει. Φοριέται στην περιοχή του στήθους και οι μετρήσεις που μπορεί να πάρει είναι καρδιακός παλμός, ταχύτητα, απόσταση, τον χρόνο που μεσολαβεί μεταξύ δύο κορυφών σ' ένα ηλεκτροκαρδιογράφημα.

[20] https://www.leapmotion.com/
[21] https://www.netatmo.com/

Κεφάλαιο 3ο

3. Σχεδιασμός και υλοποίηση του συστήματος

Η πλατφόρμα FIWARE μας δίνει τη δυνατότητα με υπηρεσίες που προσφέρει στην ανάπτυξη «έξυπνων» εφαρμογών. Η υπηρεσία Sensor Data Collection έχει τη δυνατότητα να συγκεντρώνει τα σήματα πολλών συσκευών-αισθητήρων και να τα στέλνει σε κάποια άλλη υπηρεσία ή εφαρμογή. Συγκεκριμένα, δύο προβλήματα που παρακάμπτονται με την χρήση της υπηρεσίας Sensor Data Collection είναι αρχικά το διαφορετικό πρωτόκολλο επικοινωνίας που έχει η κάθε συσκευή-αισθητήρας για την επικοινωνία με το Υπολογιστικό Νέφος και σε δεύτερη φάση η χρήση διαφορετικού API από την κάθε συσκευή για την εξόρυξη των δεδομένων της. Επομένως, η παρουσία της υπηρεσίας Sensor Data Collection, αν και βρίσκεται ακόμα σε δοκιμαστικό στάδιο, μπορεί να παίξει σημαντικό ρόλο στην ανάπτυξη εφαρμογών που επικοινωνούν απευθείας με IoT συσκευές.

Κίνητρο αποτέλεσε η απουσία από το περιβάλλον του FIWARE μιας υπηρεσίας, η οποία θα έχει τη δυνατότητα της διαχείρισης, αποθήκευσης και διαμοιρασμού της πληροφορίας που προσφέρουν υπηρεσίες όπως ο Sensor Data Collection σ’ ένα πλήθος χρηστών που έχουν κάνει συνδρομή στα δεδομένα κάποιου αισθητήρα. Οι χρήστες που θα χρησιμοποιούν αυτήν την υπηρεσία μπορούν να είναι φυσικά πρόσωπα, υπηρεσίες ή και εφαρμογές που αναπτύσσονται στο περιβάλλον του FIWARE και σε άλλα περιβάλλοντα ανάπτυξης.

Στόχος στην ανάπτυξη του I.I.M ήταν τα παρακάτω:

- Διατήρηση της ασφάλειας των δεδομένων των χρηστών με την χρήση ελεγμένης υπηρεσίας ταυτοποίησης χρηστών από την κοινότητα του FIWARE.
- Απομακρυσμένη αποθήκευση των δεδομένων των χρηστών, αισθητήρων και αδειών με την χρήση ειδικής υπηρεσίας αποθήκευσης για την ασφαλή και γρήγορη αποθήκευση και ανάκτηση των δεδομένων που αποθηκεύονται στη υπηρεσία.
- Να βασιστεί στην αρχιτεκτονική REST, για την εύκολη επικοινωνία με οποιεσδήποτε άλλες υπηρεσίες. Έτσι θα διευκολύνεται η ανάπτυξη πιο σύνθετων υπηρεσιών ή εφαρμογών με προσθήκη επιπλέον λειτουργικότητας.

Επίσης, η χρήση του προτύπου JSON για την ανταλλαγή της πληροφορίας μεταξύ των υπηρεσιών δίνει επιπλέον δυνατότητες στο σύστημα μας.

- Αξιοποίηση των πλεονεκτημάτων που προσφέρει το Υπολογιστικό Νέφος στην ανάπτυξη υπηρεσιών όπως ελαστικότητα, ευελιξία και χαμηλό κόστος υποδομής και συντήρησης.

Τα πλεονεκτήματα που προσφέρουμε:

- ***Επεκτασιμότητα***, μπορούν να προστεθούν υπηρεσίες για την επέκταση των λειτουργιών του SE, όπως για παράδειγμα εργαλεία ανάλυσης δεδομένων.
- ***Απλότητα στη χρήση***, με ένα απλό και εύχρηστο REST (Representational State Transfer) API που προσφέρεται σε οποιονδήποτε θέλει να το χρησιμοποιήσει.
- ***Υποστήριξη πολλών διαφορετικών αισθητήρων και χρηστών/ελαστικότητα,*** προσαρμογή των υπολογιστικών πόρων της υποδομής νέφους σε περίπτωση απαίτησης από την εφαρμογή.
- ***Συμβατότητα,*** με υπηρεσίες που προσφέρουν περιβάλλοντα Νέφους όπως το FIWARE.

Η υπηρεσία I.I.M, στα πλαίσια μιας «έξυπνης» πόλης μπορεί να χρησιμοποιηθεί σε διάφορες περιπτώσεις χρήσης (use-cases). Παρακάτω θα γίνει ανάλυση μιας περίπτωσης χρήσης στην οποία φαίνεται ξεκάθαρα η χρησιμότητα της υπηρεσίας I.I.M που αναπτύξαμε. Έστω ότι κάποιος ασθενής έχει διάφορα προβλήματα υγείας και πρέπει να παρακολουθείται ανά πάσα ώρα και στιγμή από έναν γιατρό. Έστω ότι είναι διασυνδεμένος με τρείς διαφορετικούς τύπους αισθητήρων(δύο ιατρικούς και μερικούς περιβαλλοντικούς για την κάλυψη όλου του χώρου), ένας εκ των οποίων μετράει με ακρίβεια τους παλμούς και την πίεση στο αίμα του, ένας δεύτερος το ποσοστό του οξυγόνου στο αίμα του, ενώ παράλληλα στους χώρους του σπιτιού του υπάρχουν και περιβαλλοντικοί αισθητήρες για την μέτρηση της θερμοκρασία, το ποσοστό υγρασίας στην ατμόσφαιρα και τα επίπεδα του διοξειδίου του άνθρακα στον αέρα. Οι περιβαλλοντικοί αισθητήρες τοποθετήθηκαν ώστε να διασφαλίζεται η καλή ποιότητα του αέρα μέσα στο σπίτι, ώστε να μην υπάρξουν απρόβλεπτες επιπλοκές στην υγεία του ασθενούς. Έστω ότι οι περιβαλλοντικοί αισθητήρες συνδέονται με ειδικό λογισμικό που ελέγχει την λειτουργία ενός αφυγραντήρα, της κεντρικής θέρμανσης και ειδικού συναγερμού που ειδοποιεί τον ασθενή σε περίπτωση αύξησης των επιπέδων διοξειδίου του άνθρακα στον αέρα του σπιτιού. Από την άλλη οι ιατρικοί αισθητήρες παρακολουθούνται από τον γιατρό του. Επομένως, ο γιατρός του μπορεί να ενημερώνεται όποτε αυτός επιλέξει ώστε να διασφαλιστεί η άμεση ενημέρωση του σε περίπτωση που κάτι δεν πάει καλά. Σε μια τέτοια περίπτωση χρήσης η υγεία του ασθενούς ελέγχεται πλήρως από την υπηρεσία που αναπτύξαμε σε συνεργασία με υπηρεσίες που ανακτούν τα δεδομένα των συσκευών που χρησιμοποιήσαμε.

3.1 Υπηρεσίες Γενικού Σκοπού και Ειδικού Σκοπού

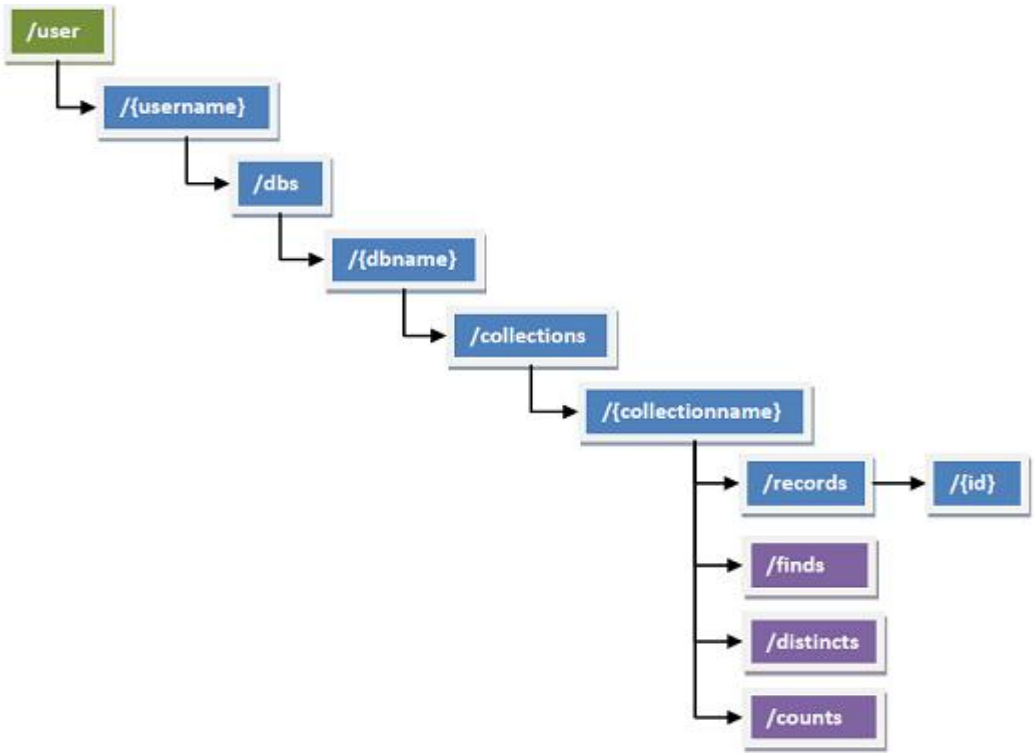
3.1.1 KeyRock Identity Management GE

Η υπηρεσία γενικού σκοπού KeyRock IDM [14] καλύπτει κάποιες πτυχές που αφορούν την πρόσβαση των χρηστών στα δίκτυα, υπηρεσίες και εφαρμογές. Επιπλέον, η συγκεκριμένη υπηρεσία χρησιμοποιείται για την έγκριση ξένων υπηρεσιών να αποκτούν πρόσβαση σε προσωπικά δεδομένα που είναι αποθηκευμένα σε ένα ασφαλές περιβάλλον. Ο KeyRock περιλαμβάνει διεπαφή REST API για την χρήση του από δημιουργούς εφαρμογών.

Στην περίπτωση της δικιάς μας υπηρεσίας χρησιμοποιούμε τον IDM για την ταυτοποίηση των χρηστών μας, οι οποίοι αρχικά πρέπει να είναι εγγεγραμμένοι στην υπηρεσία FIWARE. Ο IDM γνωρίζουμε από τις προδιαγραφές του ότι παρέχει ασφάλεια όσων αφορά τα προσωπικά δεδομένα των χρηστών, γι' αυτόν τον λόγο τον χρησιμοποιούμε. Γνωρίζουμε ότι κάνει χρήση του πρωτοκόλλου επικοινωνίας https, το οποίο ενδείκνυται για την ασφαλή μετάδοση προσωπικών δεδομένων (username, password) μέσω του διαδικτύου.

3.1.2 JSON Storage GE

Ο JSON Storage GE είναι η υπηρεσία που υποστηρίζει την αποθήκευση πληροφορίας σε μορφή JSON μέσω της πιο αφηρημένου τύπου βάσης Mongo DB. Η υπηρεσία συνοδεύεται από API σχεδιασμένο με βάση την αρχιτεκτονική REST. Οι βασικές λειτουργίες που υποστηρίζει η υπηρεσία είναι η δημιουργία, ανανέωση και διαγραφή χρηστών όπως επίσης και την δημιουργία βάσεων δεδομένων ,συλλογών και εγγραφών. Ο χρήστης δεσμεύει χώρο, έτσι ώστε να αποθηκεύει δεδομένα, δίνοντας όνομα χρήστη και κωδικό πρόσβασης, χωρίς να επιτρέπεται η προσπέλαση προσωπικών δεδομένων από τρίτους. Ο κάθε χρήστης δημιουργεί βάσεις δεδομένων που περιέχουν συλλογές και αυτές με τη σειρά τους περιλαμβάνουν εγγραφές. Η Εικόνα 7 παρουσιάζει την αναλυτική δομή της υπηρεσίας.



Σχήμα 6. Δομή JSON Storage GE

Η σχεδίαση και η υλοποίηση της συγκεκριμένης υπηρεσίας παρουσιάζει αρκετά πλεονεκτήματα σε σύγκριση με τους μέχρι τώρα γνωστούς τύπους βάσεων δεδομένων. Αναλυτικότερα, ο JSON Storage GE αποτελεί μια πιο ευέλικτη λύση για αποθήκευση δεδομένων καθώς δέχεται μεγαλύτερο όγκο δεδομένων ανταποκρινόμενος ικανοποιητικά και στην μεγάλη ταχύτητα με την οποία αυτά στέλνονται. Η ασφάλεια που παρέχει στο χρήστη για την προστασία των δεδομένων του συμπεριλαμβάνεται σε ένα ακόμα από τα πλεονεκτήματα της υπηρεσίας. Οι βασικές μέθοδοι που υποστηρίζει η υπηρεσία μέσω της διεπαφής παρουσιάζονται στον Πίνακα 2 :

Method	Base (/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.

Σχήμα 7. Υποστηριζόμενο REST API του JSON Storage GE

3.1.3 Publish/Subscribe Context Broker-Orion Context Broker GE

Ο Orion Context Broker [15] είναι μια εφαρμογή του Publish/Subscribe Context Broker GE, ακολουθούμενο από τις διεπαφές NGSI9 και NGSI10 βασισμένες στην αρχιτεκτονική REST. Η υπηρεσία αυτή διαχειρίζεται τον μεγάλο όγκο δεδομένων που λαμβάνονται από τους αισθητήρες και είναι υπεύθυνη για τις συνδρομές των χρηστών. Ο Context Broker GE παρέχει ένα πληροφοριακό σχήμα σε μορφή JSON, οντότητα (entity) όπως ονομάζεται στην βιβλιογραφία, στο οποίο αποθηκεύεται

προσωρινά πληροφορίες, όπως τα χαρακτηριστικά του αισθητήρα (sensorId) και τα χαρακτηριστικά (attributes) των δεδομένων του αισθητήρα (π.χ θερμοκρασία). Κατά την αποστολή δεδομένων τα χαρακτηριστικά που περιέχει το σχήμα ανανεώνεται αυτόματα. Οι λειτουργίες που προσφέρει ο Context Broker GE εμφανίζονται επιγραμματικά παρακάτω :

- Ερώτηση και ανάκτηση πληροφορίας που είναι αποθηκευμένη εκείνη τη στιγμή στην οντότητα.
- Διαγραφή οντότητας.
- Ανανέωση οντότητας (εισαγωγή νέων χαρακτηριστικών ή συσκευής).
- Αίτηση συνδρομής.

Η αίτηση συνδρομών αποτελεί τον βασικό λόγο χρησιμοποίησης της συγκεκριμένης υπηρεσίας στο σύστημά μας. Αναλυτικότερα, δημιουργούμε μία διαφορετική οντότητα για κάθε αισθητήρα που προστίθεται στο σύστημα. Στη συνέχεια οι χρήστες έχουν την δυνατότητα να κάνουν συνδρομή σε πολλά χαρακτηριστικά διαφορετικών αισθητήρων και να θέσουν ένα προκαθορισμένο χρονικό διάστημα ή σε κάθε αλλαγή των δεδομένων κάποιου χαρακτηριστικού που επιλέγουν αυτοί να λαμβάνονται απαντήσεις από τον Context Broker. Αφού πραγματοποιηθεί η συνδρομή, ο χρήστης λαμβάνει ως απάντηση από τον Context Broker ένα μοναδικό αναγνωριστικό συνδρομής subscription ID, ώστε να αποφεύγεται η πιθανή σύγχυση του συστήματος σε περίπτωση πολλών αιτήσεων από διαφορετικούς χρήστες και να είναι δυνατή η διαγραφή της συνδρομής.

Για την καλύτερη κατανόηση της υπηρεσίας γενικού σκοπού Context Broker ακολουθεί χαρακτηριστικό παράδειγμα χρήσης της.

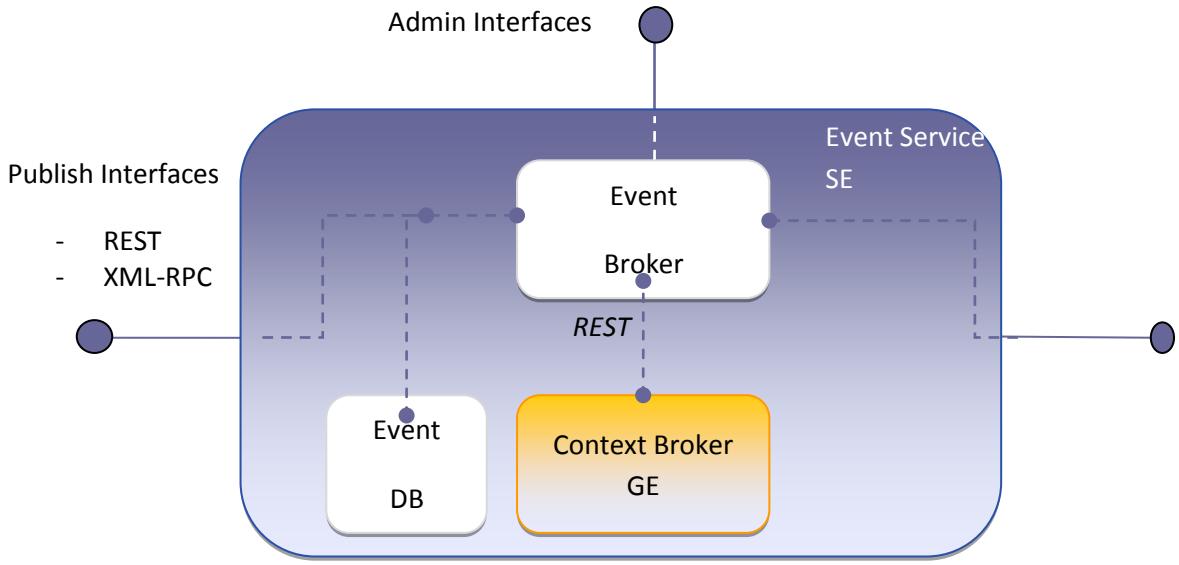
Σχήμα 8. Παράδειγμα υπηρεσίας Context Broker GE

Έστω ότι έχει αναπτυχθεί ένα σύστημα παρακολούθησης οχημάτων. Στην Εικόνα 6 παρουσιάζονται δυο αυτοκίνητα. Στο πρώτο καταγράφεται η ταχύτητα και στο δεύτερο η τοποθεσία. Τα οχήματα αντιπροσωπεύουν δυο οντότητες και η ταχύτητα και η τοποθεσία χαρακτηριστικά των οντοτήτων αυτών. Στην περίπτωση επιλογής παρακολούθησης του πρώτου αυτοκινήτου γίνεται αίτηση συνδρομής από τον χρήστη. Η υπηρεσία απαντάει στέλνοντας ένα σχήμα JSON το οποίο περιέχει την τιμή της ταχύτητας του αυτοκινήτου, η οποία εμφανίζεται στον χρήστη αφού αποκωδικοποιηθεί στο παρασκήνιο του συστήματος. Στην περίπτωση που ο δημιουργός του συστήματος έχει ορίσει ως κριτήριο για την λήψη της τιμής της ταχύτητας, την αλλαγή του συγκεκριμένου χαρακτηριστικού, οι απαντήσεις από την υπηρεσία θα λαμβάνονται αν και εφόσον η ταχύτητα μεταβληθεί. Στην περίπτωση ορισμού ενός χρονικού διαστήματος ως κριτήριο, η τιμή της ταχύτητας θα λαμβάνεται σύμφωνα με αυτό το χρονικό διάστημα. Με τον ίδιο τρόπο η υπηρεσία ανταποκρίνεται και σε αιτήσεις για το δεύτερο αυτοκίνητο.

Από το παραπάνω παράδειγμα γίνεται εύκολα αντιληπτό ότι το κριτήριο χρόνου εξυπηρετεί εφαρμογές με γρήγορη ροή δεδομένων (π.χ. καταγραφή κίνησης, ταχύτητας) καθώς είναι αναγκαία η όσο τον δυνατών γρηγορότερη εμφάνιση των αποτελεσμάτων, ενώ το κριτήριο μεταβολής χαρακτηριστικού είναι πρακτικό για εφαρμογές που δεν αποστέλλουν κρίσιμη πληροφορία (π.χ. μέτρηση θερμοκρασίας).

3.1.4 Event Service Specific Enabler

Μέσω της πλατφόρμας FIWARE μας δίνεται η δυνατότητα στην πρόσβασης πολλών διαφορετικών Specific Enablers. Ένας SE ο οποίος στην λειτουργικότητα μοιάζει αρκετά με την υπηρεσία I.I.M που υλοποιήσαμε στην παρούσα διπλωματική εργασία είναι ο Event Service Specific Enabler.



Σχήμα 9. Αρχιτεκτονική της υπηρεσίας ειδικού σκοπού Event Service.

Σύμφωνα με τις προδιαγραφές του συγκεκριμένου Specific Enabler, η σχεδίαση του βασίζεται σε δύο οντότητες τους Event Senders και Event Receivers. Οι Event Senders είναι υπεύθυνοι για την αποστολή των γεγονότων στον SE και οι Event Receivers μπορούν να κάνουν συνδρομή στα γεγονότα που στέλνονται από τους Event Senders. Οι Event Senders όπως και οι Event Receivers μπορούν να είναι μια υπηρεσία που χρησιμοποιεί το API του Event Service SE ή κάποιο γραφικό περιβάλλον (GUI). Για την παραμετροποίηση του Specific Enabler παρέχεται ένα γραφικό περιβάλλον (GUI) που ονομάζεται Admin Interface. Ζητήματα ασφαλείας έχουν προκύψει στην ανάπτυξη της συγκεκριμένης υπηρεσίας καθώς δεν διασφαλίζεται με κάποιον τρόπο η ασφάλεια των ευαίσθητων ενδεχομένως δεδομένων που διαχειρίζεται και αποθηκεύει. Επίσης, βλέπουμε ότι χρησιμοποιεί μια βάση δεδομένων για την τοπική αποθήκευση των γεγονότων πράγμα που δεν διασφαλίζει απαραίτητα την ασφάλεια των δεδομένων λόγω μη χρήσης αλγορίθμων κρυπτογράφησης (AES) για την τοπική αποθήκευση. Αντιθέτως, για την ανάπτυξη της υπηρεσίας I.I.M κάναμε χρήση ειδικών υπηρεσιών που προσφέρονται από την πλατφόρμα FIWARE για την ταυτοποίηση των χρηστών και την αποθήκευση των δεδομένων ώστε να διασφαλίζεται κατά το δυνατόν η ασφάλεια του συστήματος.

Η χρήση του Context Broker GE διευκολύνει στην ανάπτυξη ενός συστήματος βασισμένο σε συνδρομές χρηστών. Για τον λόγο αυτόν ενσωματώσαμε και στο δικό μας σύστημα τις λειτουργίες που μας προσφέρει ο Context Broker GE μέσω του παρεχόμενου API. Επίσης, με την χρήση απομακρυσμένης αποθήκευσης μέσω του JSON Storage GE στο σύστημα I.I.M κατέστη δυνατή η δημιουργία μιας δημόσιας συλλογής, στην οποία μπορούν να έχουν πρόσβαση όλα τα στιγμιότυπα της υπηρεσίας, κάτι που δεν γίνεται στον Event Service GE λόγω της τοπικής

αποθήκευσης. Επιπλέον, η χρήση ειδικής υπηρεσίας για την ταυτοποίηση των χρηστών της εφαρμογής εντάσσεται στα πλεονεκτήματα του δικού μας συστήματος σε σχέση με την υπηρεσία Event Service GE.

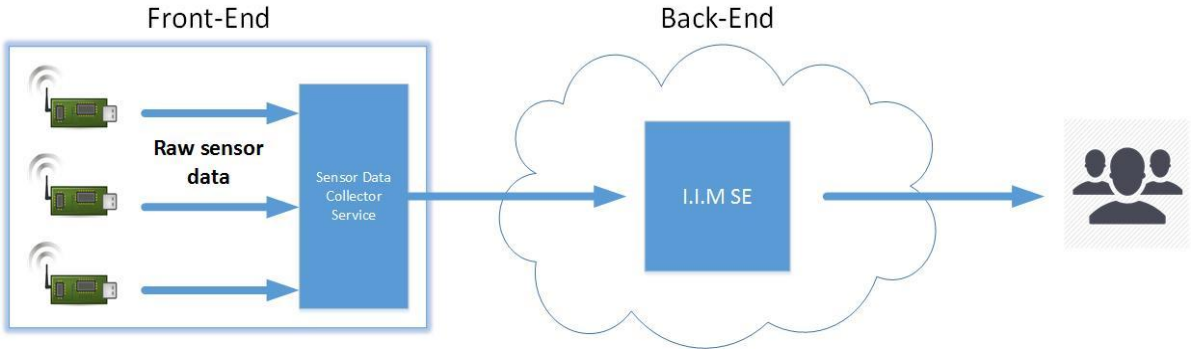
Επομένως, οι βασικές διαφορές της υπηρεσίας I.I.M με την υπηρεσία Event Service GE είναι:

- Ταυτοποίηση χρηστών που έχουν λογαριασμό στο FIWARE
- Απομακρυσμένη αποθήκευση των δεδομένων της υπηρεσίας με τη χρήση ειδικής υπηρεσίας αποθήκευσης
- Δυνατότητα προσθήκης αδειών σε αισθητήρες, ώστε να μην μπορεί ο κάθε χρήστης να κάνει συνδρομή σε συγκεκριμένους αισθητήρες που καθορίζει ο «διαχειριστής» του «στιγμιότυπου».

Στην παρακάτω ενότητα γίνεται εκτενής αναφορά στην χρήση της υπηρεσίας γενικού σκοπού για την ταυτοποίηση χρηστών, της υπηρεσίας γενικού σκοπού για την αποθήκευση των δεδομένων του I.I.M. Επίσης, ξεκαθαρίζεται τι είναι ο «διαχειριστής» ενός «στιγμιοτύπου», όπως επίσης και η λειτουργία προσθήκης αδειών σε αισθητήρες.

3.2 Γενική Αρχιτεκτονική

Για τον σχεδιασμό του συστήματος μας χρησιμοποιήσαμε την προσέγγιση **top-down**[22], δηλαδή θα αναφερθούμε αρχικά στην γενική αρχιτεκτονική του συστήματος και στην συνέχεια θα αναλύουμε με περισσότερη λεπτομέρεια τα υποσυστήματα που την απαρτίζουν. Η γενική αρχιτεκτονική απαρτίζεται από 3 διαφορετικά μεταξύ τους τμήματα (modules). Αρχικά είναι η διεπαφή χρήστη (**Front-End**) που συνδέεται άμεσα με τη διεπαφή διαχείρισης συστήματος (**Back-End**) και ακολουθούν οι χρήστες (**Users**) της εφαρμογής. Καθένα από αυτά τα τμήματα εξυπηρετούν συγκεκριμένους σκοπούς στο σύστημά μας, οι οποίοι αναλύονται παρακάτω.



[22] http://en.wikipedia.org/wiki/Top-down_and_bottom-up_design

Σχήμα 10. Γενική αρχιτεκτονική συστήματος

3.2.1 Διεπαφή χρήστη (Front-End)

Αυτό το τμήμα είναι υπεύθυνο για την σύνδεση των αισθητήρων και την εξαγωγή των δεδομένων τους. Όπως αναφέραμε στην εισαγωγή, επειδή το πρωτόκολλο επικοινωνίας που χρησιμοποιούν οι αισθητήρες για να επικοινωνήσουν με τον SE μας διαφέρει από αισθητήρα σε αισθητήρα, είναι απαραίτητο να χρησιμοποιηθεί μια υπηρεσία που θα κωδικοποιεί κατάλληλα τα δεδομένα σύμφωνα με τις προδιαγραφές που θέσαμε στον SE. Μια τέτοια υπηρεσία που λαμβάνει πολλά διαφορετικά δεδομένων από πολλούς διαφορετικούς αισθητήρες και θα βγάζει ως έξοδο τα δεδομένα τους σε μορφή JSON μπορεί να είναι η υπό δοκιμή υπηρεσία Sensor Data Collection Service που βρίσκεται στο περιβάλλον του FIWARE.

Στο τμήμα αυτό συμπεριλαμβάνονται λειτουργίες της υπηρεσίας που υλοποιήθηκαν όπως εισαγωγή/διαγραφή/ανανέωση περιγραφής αισθητήρων και σύνδεση/αποσύνδεση αισθητήρα. Όπως επίσης και η μέθοδος με την οποία στέλνονται τα δεδομένα των αισθητήρων με δικαιώματα διαχειριστή.

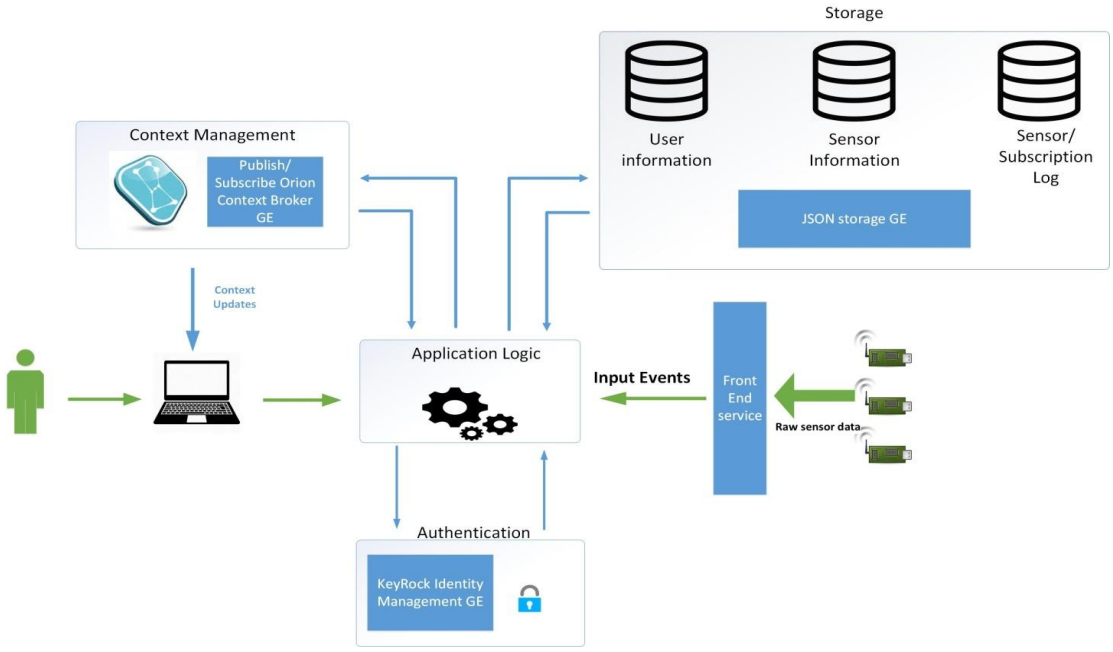
3.2.2 Διεπαφή διαχείρισης συστήματος (Back-End)

Το τμήμα «Διεπαφή διαχείρισης Συστήματος», αποτελείται από τις υπηρεσίες γενικού σκοπού που χρησιμοποιήσαμε για την επεξεργασία και την αποθήκευση των δεδομένων όλων των αισθητήρων που αλληλεπιδρούν με το σύστημά μας. Πιο συγκεκριμένα, οι υπηρεσίες είναι ο Publish/Subscribe Orion Context Broker GE και JSON Storage GE, οι οποίες είναι υπεύθυνες για την διαχείριση των συνδρομών του χρήστη και την αποθήκευση πληροφορίας και δεδομένων αντίστοιχα. Επιπλέον, το τμήμα αυτό περιέχει τον μηχανισμό ταυτοποίησης των χρηστών που εισέρχονται στην εφαρμογή, δηλαδή τον KeyRock Identity Management GE. Τέλος η λογική συστήματος (Application Logic) είναι το σημαντικότερο τμήμα που υλοποιήσαμε και στοχεύει στην μεταφορά της πληροφορίας στα κατάλληλα επιμέρους τμήματα (Αποθήκευση, Ταυτοποίηση, Διαχείριση πληροφορίας) ώστε το σύστημα να είναι λειτουργικό.

3.2.2.1Αρχιτεκτονική του συστήματος

Σημαντικό για την κατανόηση των λειτουργιών που μπορεί να διατελέσει η υπηρεσία μας είναι το παρακάτω σχήμα που δείχνει την αρχιτεκτονική του συστήματος

διαιρούμενη σε τέσσερις βασικές λειτουργίες, οι οποίες είναι υπεύθυνες για την πλήρη λειτουργικότητα του SE. Τα τέσσερα τμήματα τα οποία θα αναλύσουμε παρακάτω περιλαμβάνουν την Ταυτοποίηση χρηστών (***Authentication***), Διαχείριση Πλαισίου πληροφορίας (***Context Management***), Λογική του Συστήματος (***Application Logic***) και Αποθήκευση (***Storage***).



Σχήμα 11. Αρχιτεκτονική του συστήματος και διαχωρισμός σε τμήματα

- Η Ταυτοποίηση χρηστών (***Authentication***) πραγματοποιείται σε δύο επίπεδα, πρώτα μέσω του KeyRock Identity Management GE και έπειτα μέσω της υπηρεσίας, στην ουσία ο «διαχειριστής» έχει τη δυνατότητα να ταυτοποιήσει τον «καταναλωτή» ώστε να έχει πρόσβαση στις λειτουργίες του SE. Αυτό γίνεται επειδή ο «διαχειριστής» πρέπει να έχει τον πλήρη έλεγχο του «στιγμιότυπου» που έχει δημιουργήσει, δηλαδή δεν είναι δυνατόν να έχουν πρόσβαση στο συγκεκριμένο «στιγμιότυπο» «καταναλωτές», οι οποίοι δεν έχουν πάρει την έγκριση του «διαχειριστή».

- Η Διαχείριση Πλαισίου πληροφορίας (***Context Management***), πραγματοποιεί όλες τις απαραίτητες διεργασίες όπως δημιουργία οντότητας/αισθητήρα (entity creation), ανανέωση πλαισίου οντότητας (update context), δημιουργία/ανανέωση/διαγραφή συνδρομής χρήστη και διαγραφή οντότητας/αισθητήρα. Όλες αυτές οι λειτουργίες συμπεριλαμβάνονται στο Restful API που προσφέρει ο Publish/Subscribe Orion Context Broker GE.

- Η Λογική του Συστήματος (***Application Logic***) μπορεί να θεωρηθεί το σημαντικότερο τμήμα της Υπηρεσιοκεντρικής Αρχιτεκτονικής του συστήματός μας. Στο τμήμα αυτό περιέχονται έλεγχοι, κανονισμοί και οι απαραίτητες κλήσεις API ώστε το σύστημα να είναι λειτουργικό. Για να πραγματοποιηθεί οποιαδήποτε λειτουργία στον SE, τα δεδομένα περνούν από αυτό το τμήμα για να σταλούν σε κάποιον Generic Enabler ή να εμφανιστούν ως «απάντηση» (response) στον χρήστη.

- Η Αποθήκευση (***Storage***) χρησιμοποιείται στο σύστημα μας για την αποθήκευση πληροφορίας σχετικά με τους Χρήστες (διαχειριστής και καταναλωτές), τους Αισθητήρες και επιπλέον αποθηκεύονται Ιστορικό δεδομένων που έχουν σταλεί από κάποιον αισθητήρα και Ιστορικό συνδρομών των χρηστών.

3.2.3 Χρήστες (Users)

Στο τμήμα «Χρήστες» περιλαμβάνονται όλοι οι χρήστες που κάνουν είσοδο στο σύστημα μέσω οποιασδήποτε συσκευής με δυνατότητα σύνδεσης στο διαδίκτυο. Επίσης περιλαμβάνονται υπηρεσίες και εφαρμογές που θέλουν να ενσωματώσουν λειτουργίες του SE στην δικιά τους λειτουργικότητα μέσω του προσφερόμενου API.

Οι χρήστες που εισέρχονται στο σύστημα είναι δύο ειδών. Ο πρώτος χρήστης που εισέρχεται στο σύστημα λαμβάνει αυτόματα δικαιώματα «διαχειριστή» (**administrator**), δηλαδή είναι υπεύθυνος για την εισαγωγή των δεδομένων στο σύστημα, την διαχείριση των αισθητήρων αλλά και των χρηστών. Ο διαχειριστής επομένως είναι μοναδικός για κάθε «στιγμιότυπο» (instance) του SE. Ο λόγος που υπάρχει μόνο ένας διαχειριστής είναι για λόγους λειτουργικότητας του SE και επιπλέον οι περιορισμοί που αντιμετωπίσαμε λόγω της χρήσης του Publish/Subscribe Orion Context Broker GE. Για παράδειγμα αν ένας οποιοσδήποτε χρήστης έχει δύο αισθητήρες έναν ιατρικό που μετράει παλμούς και έναν επίσης ιατρικό που μετράει οξύτητα στο αίμα και πίεση, θα μπορεί μέσω του Intellicloud να ξεκινήσει το «στιγμιότυπο» (start instance), και στη συνέχεια να στέλνει δεδομένα που θα μπορεί να κάνει συνδρομή για παράδειγμα ο γιατρός του ώστε να αποφευχθεί κάποιο περιστατικό. Δεν υπάρχει η δυνατότητα οποιοσδήποτε άλλος χρήστης στο ίδιο «στιγμιότυπο» να έχει δικαιώματα διαχειριστή ώστε να στέλνει δεδομένα στο σύστημα.

Από την άλλη ο «καταναλωτής» (**consumer**) του συστήματος είναι σε θέση να κάνει συνδρομή σε δεδομένα πολλών αισθητήρων ταυτόχρονα και να μπορεί να τα ελέγχει

είτε όταν αλλάζουν κάποια χαρακτηριστικά (attributes) ή κατά ένα συγκεκριμένο χρονικό διάστημα πχ 2 δευτερόλεπτα.

3.3 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις του συστήματος διαχωρίζονται σε πρώτη φάση ανάλογα με τους χρήστες της εφαρμογής (Διαχειριστής-Καταναλωτής) και ανάλογα με την λειτουργικότητα που θέλει να επιτύχει ο εκάστοτε χρήστης. Αρχικά, αναλύουμε κάποια ειδικά χαρακτηριστικά (ο «Διαχειριστής» είναι μοναδικός σε κάθε στιγμιότυπο του I.I.M) που έχει ο κάθε χρήστης της εφαρμογής «Διαχειριστής» ή «Καταναλωτής» και ύστερα για κάθε χρήστη αναλύονται οι ειδικές λειτουργίες (Δημιουργία ενός νέου σχήματος αισθητήρα) που μπορεί να χρησιμοποιήσει πάνω σε μια οντότητα (Αισθητήρες, Χρήστες, Συνδρομές, Άδειες, Δεδομένα Αισθητήρων).

1. ***«Διαχειριστής»***
   - Πρέπει να είναι εγγεγραμμένος στο FIWARE LAB[23]
   - Είναι ο πρώτος χρήστης που συνδέεται με username και password στον SE
   - Είναι μοναδικός σε κάθε «στιγμιότυπο» του SE
   - Είναι υπεύθυνος για την ταυτοποίηση των χρηστών αφού πρώτα έχουν ταυτοποιηθεί οι χρήστες από τον KeyRock IDM GE. Αν ο χρήστης δεν ταυτοποιηθεί από τον «διαχειριστή» δεν έχει πρόσβαση σε καμία λειτουργία του API
   - Έχει τη δυνατότητα καθορισμού αδειών στους αισθητήρες, ώστε να επιτρέπει σε συγκεκριμένους χρήστες να κάνουν συνδρομή σε συγκεκριμένους αισθητήρες. Η άδεια κάθε αισθητήρα μπορεί να περιλαμβάνει πολλούς χρήστες ταυτόχρονα και είναι μοναδική. Σε περίπτωση που δεν έχει δοθεί άδεια σε κάποιον αισθητήρα τότε αυτόματα μπορούν όλοι οι χρήστες να κάνουν συνδρομή στα δεδομένα του
   - Έχει τη δυνατότητα μιας συνδρομής σε όσους αισθητήρες επιθυμεί. Επίσης μπορεί να ανανεώσει ή να διαγράψει την συνδρομή του. Σε περίπτωση της ανανέωσης μπορεί μόνο να αλλάξει τον τύπο της συνδρομής και όχι τα δεδομένα των αισθητήρων που θέλει να ενημερώνεται
   - Είναι υπεύθυνος για την αποστολή των δεδομένων των αισθητήρων στο σύστημα

   Διαχείριση Αισθητήρων (Sensors Management)

   - Σύνδεση αισθητήρων στο σύστημα
   - Αποσύνδεση αισθητήρων από το σύστημα

[23] https://account.lab.fiware.org/sign_up/

- Ανάκτηση των σχημάτων όλων των συνδεδεμένων αισθητήρων
- Ανάκτηση του σχήματος ενός συγκεκριμένου συνδεδεμένου αισθητήρα
- Δημιουργία ενός νέου σχήματος αισθητήρα
- Διαγραφή ενός συγκεκριμένου σχήματος αισθητήρα
- Ανανέωση ενός συγκεκριμένου σχήματος αισθητήρα
- Ανάκτηση ενός συγκεκριμένου σχήματος αισθητήρα
- Ανάκτηση όλων των σχημάτων των αισθητήρων
- Ανάκτηση του ιστορικού ενός συγκεκριμένου αισθητήρα

Διαχείριση Χρηστών (Users Management)

- Ανάκτηση όλων των χρηστών του συγκεκριμένου «στιγμιότυπου»
- Ταυτοποίηση ενός συγκεκριμένου χρήστη, ώστε να έχει πρόσβαση στον SE ως «καταναλωτής»
- Διαγραφή ενός συγκεκριμένου «καταναλωτή» του SE
- Ανάκτηση όλων των ταυτοποιημένων «καταναλωτών»
- Ανάκτηση όλων των μη ταυτοποιημένων «καταναλωτών»
- Εξουσιοδότηση ενός «καταναλωτή» με δικαιώματα «διαχειριστή» και αλλαγή των δικαιωμάτων του «διαχειριστή» με δικαιώματα «καταναλωτή»

Διαχείριση Συνδρομών (Subscriptions Management)

- Ανάκτηση όλων των συνδρομών που έχουν γίνει από όλους τους χρήστες
- Ανάκτηση μιας συγκεκριμένης συνδρομής ενός «καταναλωτή»
- Διαγραφή μιας συγκεκριμένης συνδρομής ενός «καταναλωτή»
- Δημιουργία μιας νέας δικιά του συνδρομής
- Ανάκτηση της δικής του συνδρομής
- Διαγραφή της δικής του συνδρομής
- Ανανέωση της δικής του συνδρομής

Διαχείριση Αδειών (Permissions Management)

- Δημιουργία άδειας για έναν συγκεκριμένο αισθητήρα
- Ανάκτηση άδειας ενός συγκεκριμένου αισθητήρα
- Διαγραφή άδειας ενός συγκεκριμένου αισθητήρα
- Ανανέωση άδειας ενός συγκεκριμένου αισθητήρα

Διαχείριση Δεδομένων Αισθητήρων (Sensor Data Management)

- Ταυτοποιώντας το username και το password του «διαχειριστή», υπάρχει η δυνατότητα να στείλει δεδομένα αισθητήρων στο σύστημα
- Ανάκτηση των δεδομένων των αισθητήρων ανάλογα με την συνδρομή που έχει κάνει

2. «Καταναλωτής»

- Πρέπει να είναι εγγεγραμμένος στο FIWARE LAB
- Για να έχει πρόσβαση σε οποιαδήποτε από τις παρακάτω λειτουργίες πρέπει να ταυτοποιηθεί αρχικά από τον KeyRock Identity Mangement GE με το username του και τον κωδικό, και σε δεύτερη φάση από τον «διαχειριστή» του «στιγμιότυπου» (instance).
- Για να κάνει συνδρομή στα δεδομένα κάποιου αισθητήρα θα πρέπει πρώτα να του έχει δοθεί άδεια από τον «διαχειριστή»
- Έχει τη δυνατότητα μιας συνδρομής σε όσους αισθητήρες επιθυμεί. Επίσης μπορεί να ανανεώσει ή να διαγράψει την συνδρομή του. Σε περίπτωση της ανανέωσης μπορεί μόνο να αλλάξει τον τύπο της συνδρομής και όχι τα δεδομένα των αισθητήρων που θέλει να ενημερώνεται
- Είναι ο χρήστης που έχει δικαίωμα στην συνδρομή αισθητήρων, για του οποίους του έχει δοθεί άδεια από τον «διαχειριστή». Δεν έχει τη δυνατότητα της αποστολής των δεδομένων των αισθητήρων στο σύστημα

Διαχείριση Αισθητήρων (Sensors Management)

- Ανάκτηση των σχημάτων όλων των συνδεδεμένων αισθητήρων
- Ανάκτηση του σχήματος ενός συγκεκριμένου συνδεδεμένου αισθητήρα
- Ανάκτηση ενός συγκεκριμένου σχήματος αισθητήρα
- Ανάκτηση όλων των σχημάτων των αισθητήρων
- Ανάκτηση του ιστορικού ενός συγκεκριμένου αισθητήρα

Διαχείριση Συνδρομών (Subscriptions Management)

- Δημιουργία μιας νέας δικιάς του συνδρομής
- Ανάκτηση της δικής του συνδρομής
- Διαγραφή της δικής του συνδρομής
- Ανανέωση της δικής του συνδρομής

Διαχείριση Αδειών (Permissions Management)

- Ανάκτηση αδειών για έναν συγκεκριμένο αισθητήρα

Διαχείριση Δεδομένων Αισθητήρων (Sensor Data Management)

- Ανάκτηση των δεδομένων των αισθητήρων ανάλογα με την συνδρομή που έχει κάνει

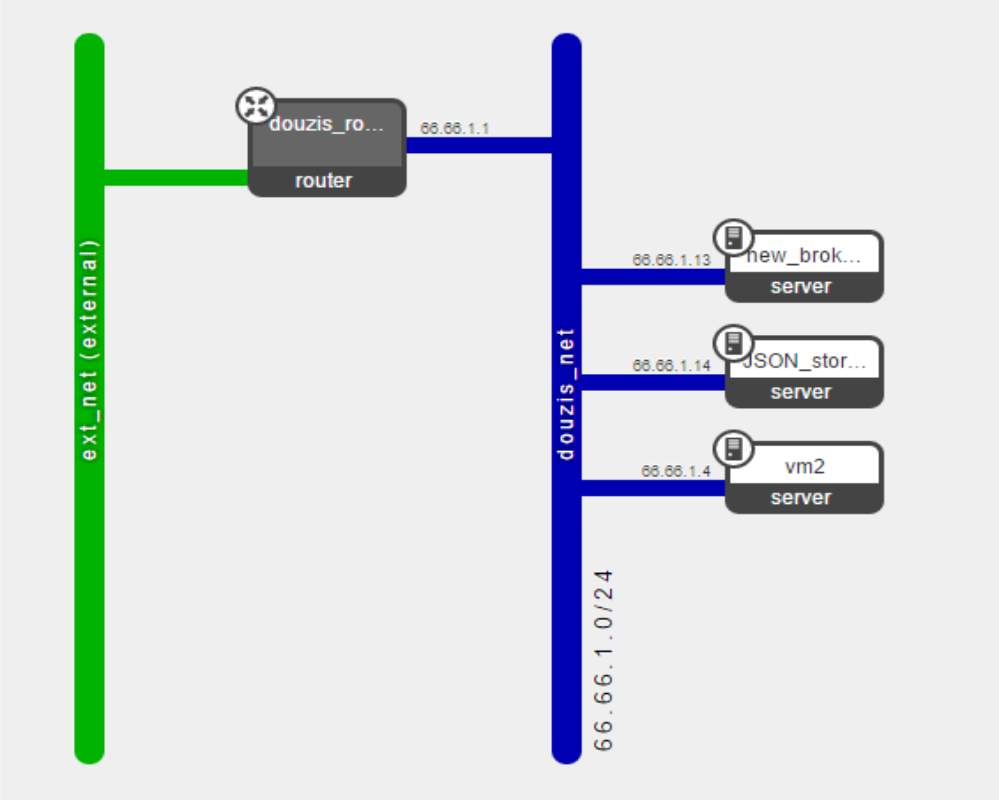
Η περεταίρω ανάλυση των λειτουργιών του SE, θα πραγματοποιηθεί σε επόμενη υποενότητα **3.6.2** όπου θα περιγράφονται αναλυτικά τα βήματα που ακολουθούνται για την υλοποίηση της κάθε λειτουργίας.

3.4 Παραμετροποίηση εικονικών μηχανών

Η υποδομή Νέφους Intellicloud μέσω τον υπηρεσιών σε μορφή IaaS που προσφέρει μας έδωσε τη δυνατότητα να φιλοξενήσουμε το σύστημα μας στο Νέφος, δημιουργώντας εικονικές μηχανές. Αρχικά, για να αποκτήσουμε πρόσβαση στη συγκεκριμένη ιδιωτική υποδομή μας δόθηκαν τα κατάλληλα αναγνωριστικά (όνομα χρήστη και κωδικός χρήστη) από τον διαχειριστή της υποδομής Νέφους. Για την δημιουργία μιας εικονικής μηχανής ήταν απαραίτητο να ακολουθηθούν κάποια συγκεκριμένα βήματα, όπως αυτά περιγράφονται παρακάτω.

1. **Δημιουργία Τοπολογίας Δικτύου**

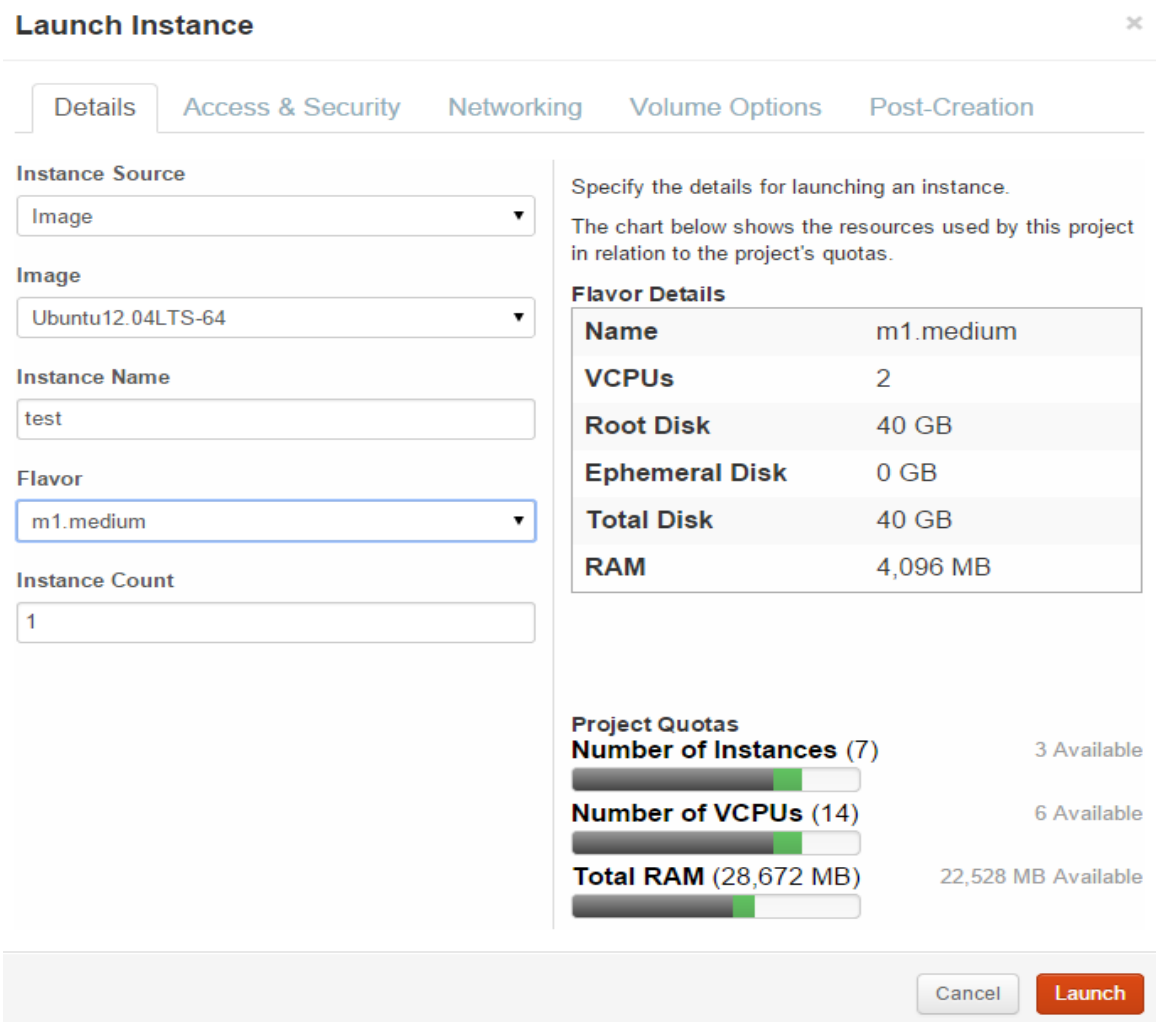
   Αρχικά, η δημιουργία τοπολογίας δικτύου πραγματοποιήθηκε με τον ορισμό ενός εσωτερικού εικονικού δικτύου (virtual network) και ενός εικονικού δρομολογητή (virtual router) έτσι ώστε να υπάρχει επικοινωνία με εξωτερικά δίκτυα. Η επικοινωνία αυτή είναι βασισμένη σε κανόνες και πρωτόκολλα τα οποία εμείς ορίσαμε σύμφωνα με τις ανάγκες και τις απαιτήσεις των υπηρεσιών που αλληλεπιδρούν με το σύστημά μας. Η τοπολογία παρουσιάζεται στο σχήμα 12.



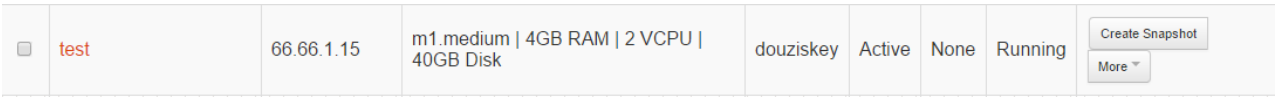
Σχήμα 12. Τοπολογία Δικτύου

2. Δημιουργία Στιγμιότυπου (Instance)

Με την δημιουργία στιγμιότυπου δεσμεύουμε μια Εικόνα (Image) μέσα στο οποίο περιλαμβάνεται η λογική της εφαρμογής μας. Την Εικόνα την παραμετροποιήσαμε σύμφωνα με τις ανάγκες μας και την υπολογιστική ισχύ που χρειαζόμασταν. Ειδικότερα, επιλέξαμε λειτουργικό σύστημα Ubuntu 12.04 LTS-64 bit, υπολογιστική ισχύ medium με μνήμη RAM 4GB, επεξεργαστή δύο πυρήνων, σκληρό δίσκο 40GB (Σχήμα 13). Στη συνέχεια, για την πρόσβαση στο διαδίκτυο δώσαμε εξωτερική και εσωτερική διεύθυνση (IP) ώστε να ολοκληρωθεί το στιγμιότυπο (Σχήμα 14).



Σχήμα 13. Παραμετροποίηση Στιγμιότυπου



Σχήμα 14. Ολοκληρωμένο στιγμιότυπο

3. Εργαλεία ανάπτυξης εφαρμογής και Εικονική Μηχανή

Αφού δημιουργήσαμε την εικονική μηχανή, εγκαταστήσαμε τα εργαλεία που ήταν απαραίτητα για την υλοποίηση της λογικής του συστήματος. Για την εγκατάσταση και την πρόσβαση στην εικόνα χρησιμοποιήθηκε το εργαλείο Putty.

Αρχικά, χρησιμοποιήσαμε το λογισμικό Apache ώστε η εικονική μηχανή να έχει τη δυνατότητα προσφοράς υπηρεσιών στον διακομιστή (server). Επιπλέον, εγκαταστάθηκε η βιβλιοθήκη της PHP που μας ήταν χρήσιμη στην επικοινωνία με τον διακομιστή και περιείχε τα εργαλεία με τα οποία πραγματοποιούνται οι κλήσεις στις υπηρεσίες. Επιπλέον εγκαταστάθηκαν οι βιβλιοθήκες cURL, για την δημιουργία HTTP «αιτημάτων» (requests) στην PHP και η βιβλιοθήκη APC για την χρήση μνήμης cache στις περιπτώσεις που θέλουμε η υπηρεσία μας να ανταποκρίνεται πιο γρήγορα. Τέλος, στην εικονική μηχανή που είναι εγκατεστημένη η υπηρεσία μας κάναμε εισαγωγή των βιβλιοθηκών του Slim framework[24] για την δημιουργία του REST API και την βιβλιοθήκη του Slim-extras[25] για την εισαγωγή Basic Authentication στο REST API.

3.4.1 Παραμετροποίηση KeyRock Identity Management GE

Στην περίπτωση του KeyRock Identity Management GE δεν χρειάστηκε η παραμετροποίηση κάποιας εικονική μηχανής, καθώς το FIWARE μας προσφέρει ένα δημόσιο στιγμιότυπο της υπηρεσίας στην διεύθυνση https://account.lab.fiware.org/. Στην υλοποίηση μας χρησιμοποιήσαμε μόνο μια μέθοδο που παρέχει το REST API για την ταυτοποίηση των εγγεγραμμένων χρηστών. Αυτή η μέθοδος είναι η POST /api/v1/tokens.json με παραμέτρους το email και τον κωδικό του εγγεγραμμένου χρήστη στο FIWARE. Δηλαδή το αίτημα που στέλνουμε στον διακομιστή (server) του KeyRock Identity Management είναι της μορφής:

```
POST /api/v1/tokens.json?email=XXX@emailservice.com&password=YYY
```

Η απάντηση που λαμβάνουμε σε περίπτωση επιτυχούς ταυτοποίησης είναι της μορφής:

```
response: status=200, {“token”:”cQ9Aqz5ezhU3z6BdH1Ks”}
```

Έχοντας λάβει το session token ως απάντηση ξέρουμε ότι ο χρήστης έχει ταυτοποιηθεί μέσω του KeyRock IDM GE.

[24] http://www.slimframework.com/
[25] https://github.com/codeguy/Slim-Extras

3.4.2 Παραμετροποίηση Publish/Subscribe Context Broker GE

Η υποδομή Intellicloud μας παρέχει εγκατεστημένη σε μορφή ανενεργού στιγμιότυπου (snapshot) την υπηρεσία Context Broker GE προκειμένου να την χρησιμοποιήσουμε μετατρέποντάς την σε εικόνα (image) και δίνοντάς της μια εξωτερική διεύθυνση (floating IP 147.27.50.148). Ολοκληρώνοντας όλα τα παραπάνω η υπηρεσία ήταν ενεργή και έτοιμη να δεχτεί τις κλήσεις από την λογική της εφαρμογής.

Ο Context Broker GE όπως και ο KeyRock IDM GE δεν χρειάστηκε κάποια επιπλέον παραμετροποίηση, ώστε να είναι έτοιμος να δεχθεί δεδομένα από την εφαρμογή. Οποιαδήποτε παραμετροποίηση πραγματοποιείται στον Context Broker γίνεται μέσα από την λογική συστήματος με κλήσεις API όταν καλεστούν μέθοδοι που πραγματοποιούν συνδρομή, συνδέουν αισθητήρες κ.α.

3.4.3 Παραμετροποίηση JSON Storage GE

Η υπηρεσία JSON Storage GE παρέχεται από την υποδομή Νέφους Intellicloud σε μορφή ανενεργούς εικόνας (snapshot) όπως και ο Orion Context Broker. Η υπηρεσία είναι έτοιμη προς χρήση δημιουργώντας μια εικόνα (image) και δίνοντάς της μια εξωτερική διεύθυνση (floating IP 147.27.50.157:3000).

Στην περίπτωση του JSON Storage GE είναι αναγκαίο να γίνει μια αρχική παραμετροποίηση πριν ξεκινήσει η λειτουργία του I.I.M SE, ώστε ο SE να είναι πλήρως λειτουργικός στην αποθήκευση των δεδομένων. Όπως αναφέρθηκε σε προηγούμενη υποενότητα, ο JSON Storage GE δίνει την δυνατότητα δημιουργίας χρηστών οι οποίοι μπορούν να διατηρούν τις δικές τους βάσεις δεδομένων, συλλογές και εγγραφές. Αρχικά, δημιουργήσαμε έναν χρήστη με τ' όνομα "147.27.50.119", οποίος κρατάει όλα εκείνα τα δεδομένα που συνδέονται με το στιγμιότυπο στην IP διεύθυνση 147.27.50.119. Δηλαδή, οι πληροφορίες των χρηστών, των συνδεδεμένων αισθητήρων και των συνδρομών αποθηκεύονται σε βάσεις δεδομένων και συλλογές του χρήστη "147.27.50.119". Στον παρακάτω πίνακα παρουσιάζονται αναλυτικά οι βάσεις δεδομένων και οι αντίστοιχες συλλογές που χρειάστηκε να δημιουργηθούν.

Διεύθυνση	Αποτέλεσμα
147.27.50.157:3000/users/147.27.50.119	Δημιουργία χρήστη με όνομα "147.27.50.119"

147.27.50.157:3000/users/147.27.50.119/dbs/Users	Δημιουργία βάσης με όνομα “Users”,για την αποθήκευση πληροφορίας των χρηστών.
147.27.50.157:3000/users/147.27.50.119/dbs/Users/collections/users	Δημιουργία συλλογής με όνομα “users”,για την αποθήκευση των χρηστών του στιγμιοτύπου 147.27.50.119
147.27.50.157:3000/users/147.27.50.119/dbs/Users/collections/subscriptions	Δημιουργία συλλογής με όνομα “subscriptions”, για την αποθήκευση των συνδρομών των χρηστών του στιγμιότυπου 147.27.50.119
147.27.50.157:3000/users/147.27.50.119/dbs/Permissions	Δημιουργία βάσης με όνομα “Permissions”,για την αποθήκευση πληροφορίας σχετικά με τις άδειες χρηστών πάνω σε αισθητήρες.
147.27.50.157:3000/users/147.27.50.119/dbs/SubscriptionLog	Δημιουργία βάσης με όνομα “SubscriptionLog”,για την αποθήκευση πληροφορίας σχετικά με το ιστορικό των συνδρομών των χρηστών.
147.27.50.157:3000/users/147.27.50.119/dbs/ConnectedSensors	Δημιουργία βάσης με όνομα “ConnectedSensors”,για την αποθήκευση πληροφορίας σχετικά με τους συνδεδεμένους αισθητήρες.
147.27.50.157:3000/users/147.27.50.119/dbs/Log	Δημιουργία βάσης με όνομα “Log”,για την αποθήκευση πληροφορίας σχετικά με το ιστορικό των συνδεδεμένων αισθητήρων.

Πίνακας 1. Αρχική παραμετροποίηση του JSON storage με χρήστη το στιγμιότυπο 147.27.50.119

Επιπλέον, στην παραμετροποίηση του JSON Storage GE προστέθηκε ένας ακόμη χρήστης ο “public_user”,ο οποίος περιλαμβάνει μία βάση δεδομένων την “PredefinedSensors” και μια συλλογή “predefinedSensors”. Στην συλλογή αυτή βρίσκονται όλα τα σχήματα των αισθητήρων που έχουν προστεθεί από οποιονδήποτε «διαχειριστή», από οποιοδήποτε «στιγμιότυπο». Στην ουσία, υπάρχει μια κοινή συλλογή αισθητήρων στην οποία μπορούν οι διαχειριστές να προσθέτουν καινούριους, να ενημερώνουν ήδη υπάρχοντες, να ανακτούν κάποιον και να διαγράφουν κάποιον.

Διεύθυνση	**Αποτέλεσμα**
147.27.50.157:3000/users/public_user	Δημιουργία χρήστη με όνομα “public_user”
147.27.50.157:3000/users/public_user/dbs/PredefinedSensors	Δημιουργία βάσης με όνομα “PredefinedSensors”,για την αποθήκευση πληροφορίας των σχημάτων των αισθητήρων.

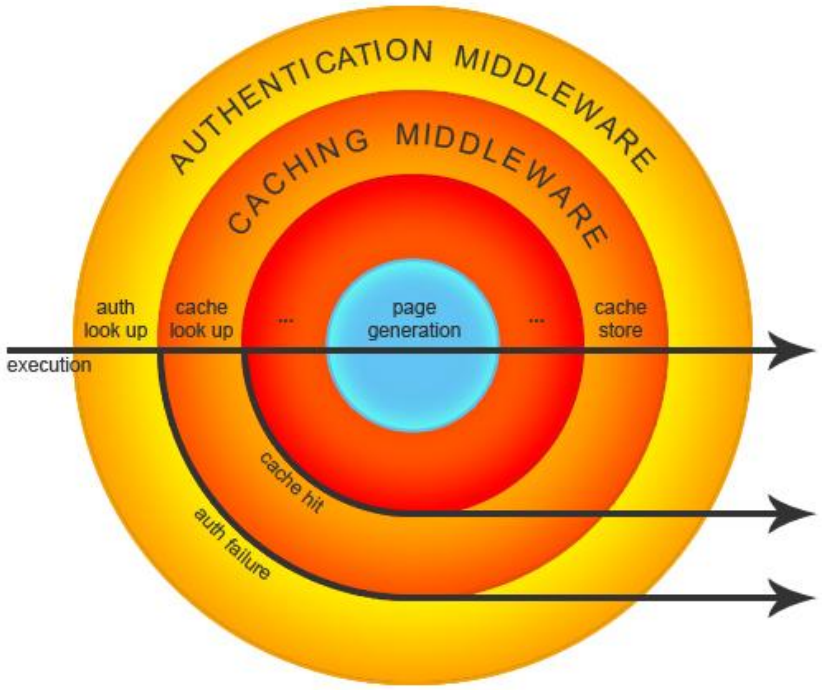
147.27.50.157:3000/users/public_user/dbs/PredefinedSensors/collections/predefinedSensors	Δημιουργία συλλογής με όνομα “predefinedSensors”,για την αποθήκευση πληροφορίας των σχημάτων των αισθητήρων.

Πίνακας 2. Αρχική παραμετροποίηση του JSON storage με χρήστη public_user

Σε περίπτωση έναρξης καινούριου στιγμιότυπου του SE σε μια διαφορετική διεύθυνση IP, τότε δίνεται η δυνατότητα στον «διαχειριστή» να εκτελέσει το αρχείο PHP **dbs_initialization.php**. Σ’ αυτό το αρχείο γίνεται η απαραίτητη παραμετροποίηση του JSON Storage GE με χρήστη αυτή τη φορά την καινούρια IP διεύθυνση του στιγμιότυπου, ώστε να λειτουργεί σύμφωνα με τις προδιαγραφές του το σύστημα.

3.5 Υλοποίηση Ταυτοποίησης Χρήστη

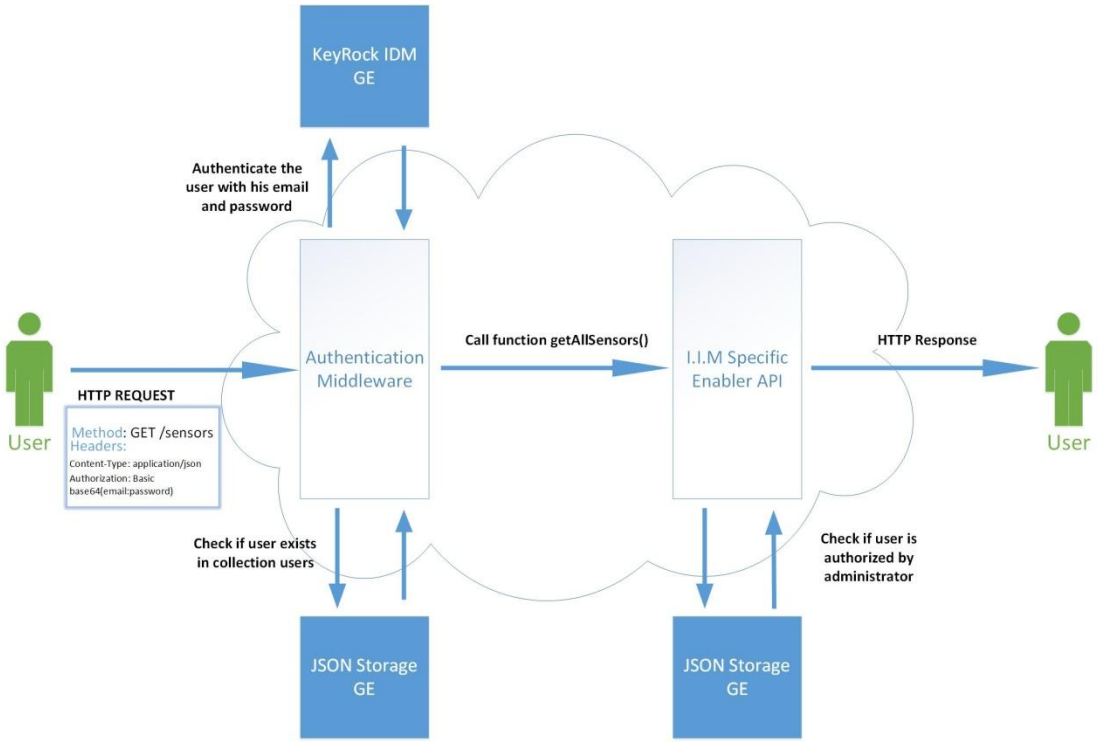
Πριν αναφερθούμε στην Υλοποίηση της Λογικής συστήματος που είναι το κυριότερο «γρανάζι» του συστήματος μας, ένα τμήμα που εκτελείται σε κάθε κλήση μεθόδου του API που υποστηρίζει το σύστημα μας είναι η Ταυτοποίηση Χρήστη μέσω ενός Middleware που μας προσφέρει το Slim framework. Το Middleware αυτό ονομάζεται HttpBasicAuth.php και μας επιτρέπει την ταυτοποίηση του χρήστη σε κάθε κλήση κάποιας μεθόδου του API. Κάθε φορά που γίνεται κλήση μιας μεθόδου του API χρειάζεται ο χρήστης να μας παρέχει το username και το password που έχει ως εγγεγραμμένος χρήστης του FIWARE. Το username και password κωδικοποιούνται στην μορφή base64(username:password), ώστε να τοποθετηθούν ως επικεφαλίδα (header) στο αίτημα (request) του χρήστη.



Σχήμα 15. Αρχιτεκτονική Middleware

Παραπάνω διακρίνεται η γενική αρχιτεκτονική της εκτέλεσης ενός αιτήματος με χρήση δύο Middleware. Το πρώτο χρησιμοποιείται για ταυτοποίηση και το δεύτερο για caching. Στην δικιά μας υπηρεσία χρησιμοποιούμε ένα Middleware για την ταυτοποίηση των χρηστών.

Ο κώδικας του Middleware που χρησιμοποιήσαμε αποτελείται κυρίως από 3 συναρτήσεις ***call***, ***authenticate*** και ***saveUser.*** Αρχικά καλείται η συνάρτηση call στην οποία καθορίζουμε ότι αν αιτηθεί η διεύθυνση 147.27.50.119/api/ τότε δεν θα ζητείται ταυτοποίηση του χρήστη. Έπειτα καλείται η συνάρτηση authenticate μέσω της οποία γίνεται η κλήση στην μέθοδο του KeyRock Identity Management GE https://account.lab.fiware.org/api/v1/tokens.json?email=123@gmail&password=123 με email και password τα στοιχεία του χρήστη. Σύμφωνα με το API του KeyRock η κλήση αυτής της μεθόδου έχει ως αποτέλεσμα την ταυτοποίηση του χρήστη και δίνει ως απάντηση στον χρήστη ένα session token και έναν κωδικό κατάστασης 200 OK. Αφού ταυτοποιηθεί ο χρήστης ελέγχουμε αν έχει ήδη αποθηκευθεί στην συλλογή users της βάσης Users του JSON Storage GE. Αν είναι ήδη αποθηκευμένος τότε τελειώνει η εκτέλεση του Middleware και η εκτέλεση του προγράμματος μετατίθεται στην εκτέλεση του API. Αν ο χρήστης δεν είναι αποθηκευμένος στην συλλογή users τότε γίνεται πρώτα ένας έλεγχος για να καθοριστεί αν θα έχει δικαιώματα «διαχειριστή» ή όχι και στην συνέχεια αποθηκεύεται στην συλλογή. Στο σημείο αυτό τελειώνει η εκτέλεση του Middleware και καλείται η συνάρτηση call για την εκτέλεση του κυρίως API.



Σχήμα 16. Χρήση Authentication Middleware για την ταυτοποίηση του χρήστη

3.6 Υλοποίηση Λογικής Συστήματος

3.6.1 Ανάλυση REST API

Η υλοποίηση της Λογικής Συστήματος του SE και το τμήμα αυτό περιλαμβάνει ένα εύχρηστο και ευρύ Restful API, το οποίο μέσα από κλήσεις μεθόδων επιτελεί συγκεκριμένες λειτουργίες. Για την καλύτερη κατανόηση του προσφερόμενου API, οι λειτουργίες του θα διαιρεθούν σε πέντε κατηγορίες όπως κάναμε και στην ανάλυση απαιτήσεων. Αυτές οι κατηγορίες που προσδιορίζουν μεθόδους του API είναι ***α) Διαχείριση Αισθητήρων***, ***β) Διαχείριση Χρηστών***, ***γ) Διαχείριση Συνδρομών***, ***δ) Διαχείριση Αδειών***, ***ε) Διαχείριση Δεδομένων Αισθητήρων.*** Στην στήλη Χρήστης αναγράφεται ποιος από τους χρήστες έχει το δικαίωμα κλήσης της μεθόδου. Όταν αναγράφεται Καταναλωτής, σημαίνει ότι και ο «διαχειριστής» και ο «καταναλωτής» έχουν δικαίωμα κλήσης. Αν αναγράφεται Διαχειριστής, τότε μόνον αυτός έχει το δικαίωμα κλήσης. Σε περίπτωση που ο «καταναλωτής» δεν έχει το δικαίωμα κλήσης τότε το σύστημα «απαντά» με σφάλμα τύπου NotAllowedException.

α) Διαχείριση Αισθητήρων

Μέθοδος	Διεύθυνση (147.27.50.119/api)	Δεδομένα (JSON format)	Αποτέλεσμα	Χρήστης
GET	/sensors	-	Ανάκτηση όλων των σχημάτων των αισθητήρων.	Καταναλωτής
GET	/sensors/{sensorId}	-	Ανάκτηση του σχήματος του αισθητήρα με αναγνωριστικό sensorId.	Καταναλωτής
POST	/sensors	{ "name": "Atmo", "attributes":[{"name":"temperature","type":"celsius"},{"name":"pressure","type":"bar"}] }	Δημιουργία ενός νέου σχήματος αισθητήρα με όνομα και χαρακτηριστικά που καθορίστηκαν από τον «διαχειριστή».	Διαχειριστής
PUT	/sensors/{sensorId}	{ "name": "Atmo2", "attributes":[{"name":"temperature","type":"kelvin"},{"name":"humidity","type":"percentage"}] }	Ανανέωση σχήματος ενός ήδη υπάρχοντος αισθητήρα με νέο όνομα ή χαρακτηριστικά.	Διαχειριστής
DELETE	/sensors/{sensorId}	-	Διαγραφή σχήματος ενός ήδη υπάρχοντος αισθητήρα.	Διαχειριστής
GET	/connected	-	Ανάκτηση όλων των συνδεδεμένων αισθητήρων.	Καταναλωτής
GET	/connected/{sensorId}	-	Ανάκτηση του συνδεδεμένου αισθητήρα με αναγνωριστικό sensorId.	Καταναλωτής
DELETE	/connected/{sensorId}	-	Διαγραφή/Αποσύνδεση του αισθητήρα με αναγνωριστικό sensorId.	Διαχειριστής

GET	/log/{sensorId}	-	Ανάκτηση του ιστορικού των δεδομένων που έχει στείλει ο αισθητήρας με αναγνωριστικό sensorId.	Διαχειριστής

Πίνακας 3. Μέθοδοι του REST API της κατηγορίας Διαχείριση Αισθητήρων

β) Διαχείριση Χρηστών

Μέθοδος	**Διεύθυνση (147.27.50.119/api)**	**Δεδομένα (JSON format)**	**Αποτέλεσμα**	**Χρήστης**
GET	/users	-	Ανάκτηση όλων των χρηστών.	Διαχειριστής
GET	/user/{userId}	-	Ανάκτηση του χρήστη με αναγνωριστικό userId.	Διαχειριστής
DELETE	/user/{userId}	-	Διαγραφή του χρήστη με αναγνωριστικό userId.	Διαχειριστής
POST	/user/admin	{ "username": “123@emailservice.com” }	Εξουσιοδότηση του «καταναλωτή» με το συγκεκριμένο username με δικαιώματα «διαχειριστή» και αντίστοιχα ο «διαχειριστής» μετά την κλήση έχει δικαιώματα «καταναλωτή».	Διαχειριστής
GET	/users/authorized	-	Ανάκτηση όλων των ταυτοποιημένων από τον «διαχειριστή» «καταναλωτών».	Διαχειριστής
GET	/users/unauthorized	-	Ανάκτηση όλων των μη ταυτοποιημένων από τον «διαχειριστή» «καταναλωτών».	Διαχειριστής
POST	/user/authorize	{ "username": “123@emailservice.com” }	Ταυτοποίηση του «καταναλωτή» με το συγκεκριμένο username από τον «διαχειριστή».	Διαχειριστής

Πίνακας 4. Μέθοδοι του REST API της κατηγορίας Διαχείριση Χρηστών

γ) Διαχείριση Συνδρομών

Μέθοδος	**Διεύθυνση (147.27.50.119/api)**	**Δεδομένα (JSON format)**	**Αποτέλεσμα**	**Χρήστης**
GET	/subscriptions	-	Ανάκτηση όλων των συνδρομών των χρηστών.	Διαχειριστής
GET	/subscriptions/{subscriptionId}	-	Ανάκτηση της συνδρομής χρήστη με αναγνωριστικό subscriptionId.	Διαχειριστής
GET	/subscription/log	-	Ανάκτηση του ιστορικού των δεδομένων όλων των αισθητήρων που είχε συνδρομή ο ταυτοποιημένος χρήστης.	Καταναλωτής
DELETE	/subscriptions/{subscriptionId	-	Διαγραφή της συνδρομής χρήστη με αναγνωριστικό subscriptionId.	Διαχειριστής
GET	/subscription	-	Ανάκτηση της συνδρομής του ταυτοποιημένου χρήστη.	Καταναλωτής
POST	/subscription	{ "subAttributes":[{"name":"a1","sensorid":"test1"},{"name":"a2","sensorid":"test1"}], "subType":"ONCHANGE", "condAttributes":[{"name":"a1","sensorid":"test1"}] }	Δημιουργία μιας καινούριας συνδρομής του ταυτοποιημένου χρήστη.	Καταναλωτής
PUT	/subscription	{ "subAttributes":[{"name":"x-axis","sensorid":"Kinect"},{"name":"y-axis","sensorid":"Kinec	Ανανέωση της υπάρχουσας συνδρομής του ταυτοποιημένου χρήστη.	Καταναλωτής

		t"},{"name":"a1","sensorid":"test1"}], "subType":"ONTIMEINTERVAL", "interval":5 }		
DELETE	/subscription	-	Διαγραφή της συνδρομής του ταυτοποιημένου χρήστη.	Καταναλωτής

Πίνακας 5. Μέθοδοι του REST API της κατηγορίας Διαχείριση Συνδρομών

δ) Διαχείριση Αδειών

Μέθοδος	**Διεύθυνση (147.27.50.119/api)**	**Δεδομένα (JSON format)**	**Αποτέλεσμα**	**Χρήστης**
GET	/sensors/{sensorId}/permission	-	Ανάκτηση της άδειας για τον αισθητήρα με αναγνωριστικό sensorId.	Καταναλωτής

POST	/sensors/{sensorId}/permission	{
		"users":[“123@emailservice.com”]
		}
			Δημιουργία της άδειας για τον αισθητήρα με αναγνωριστικό sensorId.	Διαχειριστής

PUT	/sensors/{sensorId}/permission	{
		"users":[“dem2@emailservice.com”]
		}
			Ανανέωση της άδειας για τον αισθητήρα με αναγνωριστικό sensorId.	Διαχειριστής

DELETE	/sensors/{sensorId}/permission	-	Διαγραφή της άδειας για τον αισθητήρα με αναγνωριστικό sensorId.	Διαχειριστής

Πίνακας 6. Μέθοδοι του REST API της κατηγορίας Διαχείριση Αδειών

ε) Διαχείριση Δεδομένων Αισθητήρων

Μέθοδος	Διεύθυνση (147.27.50.119/api)	Δεδομένα (JSON format)	Αποτέλεσμα	Χρήστης
GET	/contextNotifications	-	Ανάκτηση των δεδομένων από τους αισθητήρες που έχει κάνει συνδρομή ο χρήστης.	Καταναλωτής
POST	/event	{"id": "Atmo", "attributes":[{"name":"temperature","type":"Celsius","value":"23"},{"name":"humidity","type":"percentage","value":"70"}] }	Δημιουργία γεγονότος και προώθηση των δεδομένων του αισθητήρα στην υπηρεσία.	Διαχειριστής

Πίνακας 7. Μέθοδοι του REST API της κατηγορίας Διαχείριση Δεδομένων Αισθητήρων

3.6.2 Υλοποίηση μεθόδων

Ο διαχωρισμός που έγινε στην προηγούμενη υποενότητα σχετικά με τις λειτουργίες του REST API, θα μας βοηθήσει στην ανάλυση των βημάτων που ακολουθήθηκαν για την υλοποίηση της κάθε μεθόδου ξεχωριστά.

α) Διαχείριση Αισθητήρων

- **GET /sensors**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Πραγματοποιείται κλήση GET στην διεύθυνση http://147.27.50.157:3000/users/public_user/dbs/PredefinedSensors/collections/predefinedSensors/records του JSON Storage GE, για την ανάκτηση όλων των διαθέσιμων σχημάτων των αισθητήρων. Σε περίπτωση που η κλήση αποτύχει για οποιονδήποτε λόγο το σύστημα «απαντά» με σφάλμα τύπου CurlException.
  - Στο τέλος εμφανίζει το αποτέλεσμα της κλήσης στον χρήστη.
- **GET /sensors/{sensorId}**

- Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
- Έλεγχος αν υπάρχει ο αισθητήρας με μοναδικό αναγνωριστικό το sensorId στη διεύθυνση http://147.27.50.157:3000/users/public_user/dbs/PredefinedSensors/collections/predefinedSensors/finds. Αν υπάρχει εμφανίζει το αποτέλεσμα της κλήσης στον χρήστη, αλλιώς το σύστημα «απαντά» με σφάλμα τύπου NotFoundException.

- **POST /sensors**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος των δεδομένων που λαμβάνει ο SE μέσω του POST, ώστε να είναι σε μορφή JSON και επιπλέον να είναι στην μορφή που έχουμε καθορίσει. Τα δεδομένα στην συγκεκριμένη μέθοδο πρέπει να είναι της μορφής:

    {
    "name":"Atmo",
    "attributes":[{"name":"temperature","type":"celsius"},{"name":"pressure","type":"bar"}]
    }

    Δίνουμε δηλαδή το όνομα του αισθητήρα και τα χαρακτηριστικά από τα οποία παίρνουμε μετρήσεις. Επίσης ελέγχεται αν τα ονόματα των χαρακτηριστικών είναι μοναδικά, ώστε να αποφευχθεί κάποιο λάθος σε πιθανή συνδρομή ενός τέτοιου χαρακτηριστικού. Σε περίπτωση που τα δεδομένα δεν είναι σε σχήμα JSON το σύστημα «απαντά» με σφάλμα τύπου JsonException και αν δεν είναι στην παραπάνω μορφή ή τα ονόματα των χαρακτηριστικών δεν είναι μοναδικά «απαντά» με σφάλμα τύπου DataFormatException.

  - Από το σύστημα δίνεται αυτόματα ένα μοναδικό αναγνωριστικό (id) για τον συγκεκριμένο αισθητήρα, ώστε να μπορούμε να έχουμε πολλούς αισθητήρες του ίδιου τύπου π.χ Atmo αλλά θα έχουν διαφορετικό αναγνωριστικό π.χ Atmo1
  - Στη συνέχεια γίνεται κλήση στη διεύθυνση http://147.27.50.157:3000/users/public_user/dbs/PredefinedSensors/collections/predefinedSensors/records με μέθοδο POST του JSON Storage GE και δεδομένα αυτά που μας είχε στείλει ο χρήστης, ώστε να προστεθεί το σχήμα του αισθητήρα στην συλλογή predefinedSensors. Σε περίπτωση οποιουδήποτε σφάλματος στην κλήση της μεθόδου του JSON Storage GE το σύστημα «απαντά» με σφάλμα τύπου CurlException. Σε περίπτωση επιτυχίας το σύστημα απαντά με μήνυμα { "response":"The sensor added successfully" }.

- **PUT /sensors/{sensorId}**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».

- Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId, υπάρχει στην συλλογή predefinedSensors. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει προχωράμε στο επόμενο βήμα.
- Αν ο αισθητήρας με αναγνωριστικό sensorId είναι συνδεδεμένος, τότε το σύστημα «απαντά» με σφάλμα τύπου NotAllowedException. Σε περίπτωση που δεν είναι συνδεδεμένος τότε ελέγχει τα δεδομένα που έχει λάβει από τον χρήστη. Αν δεν είναι σε μορφή JSON ή δεν είναι στην μορφή που έχουμε καθορίσει τότε το σύστημα «απαντά» με JsonException και DataFormatException αντίστοιχα. Τα δεδομένα του χρήστη πρέπει να είναι της μορφής:

  {
  "name":"Atmo",
  "attributes":[{"name":"temperature","type":"celsius"},{"name":"pressure","type":"bar"}]
  }
- Αν τα δεδομένα είναι σύμφωνα με τις προδιαγραφές τότε πραγματοποιείται ανανέωση του σχήματος του αισθητήρα στην συλλογή predefinedSensors του JSON storage GE και το σύστημα «απαντά» { "response":"The sensor updated successfully" }.

- **DELETE /sensors/{sensorId}**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId, υπάρχει στην συλλογή predefinedSensors. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει προχωράμε στο επόμενο βήμα.
  - Αν ο αισθητήρας με αναγνωριστικό sensorId είναι συνδεδεμένος, τότε το σύστημα διαγράφει τον συνδεδεμένο αισθητήρα από την συλλογή connectedSensors και έπειτα τον διαγράφει και από την συλλογή predefinedSensors. Σε περίπτωση που διαγραφούν χωρίς σφάλμα το σύστημα «απαντά» { "response":"Sensor has been deleted successfully" }, αλλιώς «απαντά» με σφάλμα τύπου CurlException.

- **GET /connected/{sensorId}**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId, υπάρχει στην συλλογή connectedSensors. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει τότε εμφανίζεται ο συγκεκριμένος αισθητήρας στον χρήστη.

- **GET /connected**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Πραγματοποιείται κλήση στη συλλογή connectedSensors και το αποτέλεσμα της κλήσης εμφανίζεται στον χρήστη. Σε περίπτωση σφάλματος κάτα την κλήση το σύστημα «απαντά» με σφάλμα τύπου CurlException.

- **DELETE /connected/{sensorId}**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId, υπάρχει στην συλλογή connectedSensors. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει προχωράμε στο επόμενο βήμα.
  - Ύστερα πραγματοποιείται κλήση στον JSON Storage GE για την διαγραφή του από την συλλογή connectedSensors. Σε περίπτωση σφάλματος της κλήσης το σύστημα «απαντά» με σφάλμα τύπου CurlException.
  - Στη συνέχεια, διαγράφεται η οντότητα του συγκεκριμένου αισθητήρα από τον Context Broker GE με κλήση στην διεύθυνση http://147.27.50.148:1026/NGSI10/contextEntities/{id} με id τον αισθητήρα που θέλουμε να διαγράψουμε. Αν η διαγραφή έγινε με επιτυχία τότε το σύστημα «απαντά» με {"response":"The connected sensor deleted successfully"}. Σε περίπτωση σφάλματος στην κλήση του Context Broker GE το σύστημα «απαντά» με σφάλμα τύπου CurlException.

- **GET /sensors/{sensorId}/log**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId, υπάρχει στην συλλογή connectedSensors. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει προχωράμε στο επόμενο βήμα.
  - Στη συνέχεια πραγματοποιείται κλήση στον JSON Storage GE στην βάση Log και συλλογή το id του αισθητήρα, ώστε να ανακτηθούν τα δεδομένα του που είναι αποθηκευμένα εκεί. Στη συνέχεια τα δεδομένα εμφανίζονται στον χρήστη. Σε περίπτωση σφάλματος κατά την κλήση το σύστημα «απαντά» με σφάλμα τύπου Curl Exception.

β) Διαχείριση Χρηστών

- **GET /users**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».

- Πραγματοποιείται κλήση στον JSON Storage GE στη συλλογή users της βάσης Users, ώστε να εμφανίσει το σύστημα ως αποτέλεσμα στον χρήστη όλους τους χρήστες του συγκεκριμένου στιγμιότυπου. Σε περίπτωση σφάλματος κατά την κλήση το σύστημα «απαντά» με σφάλμα τύπου Curl Exception.

- **GET /user/{userId}**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο χρήστης με μοναδικό αναγνωριστικό userId, υπάρχει στην συλλογή users. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει τότε τον εμφανίζει στον χρήστη ως «απάντηση».

- **POST /user/admin**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν τα δεδομένα που στέλνει ο χρήστης με την κλήση της μεθόδου POST είναι σε μορφή JSON και σύμφωνα με τις προδιαγραφές του συστήματος. Τα δεδομένα του χρήστη πρέπει να είναι της μορφής:
    {
    "username":"123@gmail.com"
    }
    Όπου το χαρακτηριστικό username προσδιορίζει τον χρήστη ο οποίος θα λάβει τα δικαιώματα «διαχειριστή».
  - Έλεγχος αν το username που δίνει ο χρήστης υπάρχει στην συλλογή users μέσω μιας κλήσης στο JSON Storage GE. Αν δεν υπάρχει χρήστης με username αυτό που μας έδωσε ο χρήστης το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Σε περίπτωση που βρεθεί χρήστης με αυτό το username τότε ελέγχεται αν είναι «διαχειριστής» ή ταυτοποιημένος από τον «διαχειριστή» «καταναλωτής». Αν είναι «διαχειριστής» το σύστημα «απαντά» με σφάλμα τύπου InvalidTypeException. Εν συνεχεία, με διαδοχικές κλήσεις στην συλλογή users του JSON Storage GE, αλλάζω τα δικαιώματα του «διαχειριστή» με τον «καταναλωτή» με το αντίστοιχο username και εμφανίζεται στον χρήστη το μήνυμα {"response":"The change of administrator completed successfully"}.

- **GET /users/unauthorized**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Πραγματοποιείται κλήση στον JSON Storage GE στη συλλογή users της βάσης Users, ώστε να εμφανίσει το σύστημα ως αποτέλεσμα στον χρήστη όλους τους χρήστες που έχουν το χαρακτηριστικό authorized την τιμή

false. Σε περίπτωση σφάλματος κατά την κλήση το σύστημα «απαντά» με σφάλμα τύπου Curl Exception.

- **GET /users/authorized**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Πραγματοποιείται κλήση στον JSON Storage GE στη συλλογή users της βάσης Users, ώστε να εμφανίσει το σύστημα ως αποτέλεσμα στον χρήστη όλους τους χρήστες που έχουν το χαρακτηριστικό authorized την τιμή true. Σε περίπτωση σφάλματος κατά την κλήση το σύστημα «απαντά» με σφάλμα τύπου Curl Exception.

- **POST /user/authorize**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν τα δεδομένα που στέλνει ο χρήστης με την κλήση της μεθόδου POST είναι σε μορφή JSON και σύμφωνα με τις προδιαγραφές του συστήματος. Τα δεδομένα του χρήστη πρέπει να είναι της μορφής:
    {
    "username":"123@gmail.com"
    }
    Όπου το χαρακτηριστικό username προσδιορίζει τον «καταναλωτή» ο οποίος θα ταυτοποιηθεί από τον «διαχειριστή».
  - Έλεγχος αν το username που δίνει ο χρήστης υπάρχει στην συλλογή users μέσω μιας κλήσης στο JSON Storage GE. Αν δεν υπάρχει χρήστης με username αυτό που μας έδωσε ο χρήστης το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Σε περίπτωση που βρεθεί χρήστης με αυτό το email τότε ελέγχεται αν είναι «διαχειριστής» ή ταυτοποιημένος από τον «διαχειριστή» «καταναλωτής». Αν είναι είτε «διαχειριστής» είτε ταυτοποιημένος «καταναλωτής», το σύστημα «απαντά» με σφάλμα τύπου NotAllowedException. Εν συνεχεία, με διαδοχικές κλήσεις στην συλλογή users του JSON Storage GE, ταυτοποιείται ο «καταναλωτής» με το αντίστοιχο username και εμφανίζεται στον χρήστη το μήνυμα {"response":"User has been authorized"}.

- **DELETE /user/{userId}**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο χρήστης με μοναδικό αναγνωριστικό userId, υπάρχει στην συλλογή users. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει τότε ελέγχεται αν είναι «καταναλωτής» ή «διαχειριστής». Αν είναι «διαχειριστής», εμφανίζεται σφάλμα τύπου NotAllowedException. Στην περίπτωση που είναι «καταναλωτής», τότε με δύο κλήσεις στον JSON Storage GE διαγράφεται ο χρήστης από την συλλογή users και έπειτα διαγράφεται το ιστορικό των

δεδομένων από τις συνδρομές που έχει κάνει ο χρήστης από τη βάση SubscriptionLog. Τέλος, εμφανίζεται στον χρήστη το μήνυμα { "response":"The user has been deleted successfully" }.

γ) Διαχείριση Συνδρομών

- **GET /subscriptions**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Πραγματοποιείται κλήση στον JSON Storage GE στη συλλογή subscriptions της βάσης Users, ώστε να εμφανίσει το σύστημα ως αποτέλεσμα στον χρήστη όλες τις συνδρομές που έχουν γίνει. Σε περίπτωση σφάλματος κατά την κλήση το σύστημα «απαντά» με σφάλμα τύπου Curl Exception.

- **GET /subscriptions/{subscriptionId}**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν η συνδρομή με μοναδικό αναγνωριστικό subscriptionId, υπάρχει στην συλλογή subscriptions του JSON Storage GE. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν υπάρχει τότε εμφανίζεται ως απάντηση στον χρήστη. Σε περίπτωση σφάλματος κατά την κλήση στον JSON Storage GE το σύστημα «απαντά» με σφάλμα τύπου Curl Exception.

- **GET /subscription/log**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Μέσω της συνάρτησης getUserInfo(), παίρνουμε τα στοιχεία του χρήστη όπως το μοναδικό του αναγνωριστικό userId. Στη συνέχεια πραγματοποιείται κλήση στον JSON Storage GE στη συλλογή του χρήστη με αναγνωριστικό το userId της βάσης SubscriptionLog. Τέλος, εμφανίζονται στον χρήστη το ιστορικό όλων των δεδομένων των αισθητήρων που έχει ή είχε στο παρελθόν συνδρομή.

- **DELETE /subscriptions/{subscriptionId}**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν η συνδρομή με μοναδικό αναγνωριστικό subscriptionId, υπάρχει στην συλλογή subscriptions του JSON Storage GE. Αν δεν υπάρχει τότε το σύστημα «απαντά» με σφάλμα τύπου

NotFoundException. Αν υπάρχει τότε συνεχίζεται η εκτέλεση στο επόμενο βήμα.

- Πραγματοποιείται κλήση στον JSON Storage GE με σκοπό την διαγραφή της συνδρομής από την συλλογή subscriptions. Στη συνέχεια, πραγματοποιείται κλήση στην διεύθυνση http://147.27.50.148:1026/NGSI10/unsubscribeContext του Context Broker GE ώστε να διαγραφεί η συνδρομή του χρήστη και να μην στέλνονται πλέον ενημερώσεις των δεδομένων για την συγκεκριμένη συνδρομή.

- **GET /subscription**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Αρχικά ανακτάμε τα στοιχεία του ταυτοποιημένου χρήστη με την βοήθεια της μνήμης cache και ελέγχουμε αν ο χρήστης έχει ήδη συνδρομή στο σύστημα. Σε περίπτωση που δεν έχει το σύστημα «απαντά» με σφάλμα τύπου NotFoundException. Αν όμως ο χρήστης έχει συνδρομή τότε πραγματοποιείται κλήση στον JSON Storage GE στη συλλογή subscriptions ώστε να ανακτήσουμε και να εμφανίσουμε τα στοιχεία συνδρομής του συγκεκριμένου χρήστη. Σε περίπτωση σφάλματος κατά την κλήση στον JSON Storage GE το σύστημα «απαντά» με σφάλμα τύπου Curl Exception.

- **POST /subscription**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Έλεγχος αν τα δεδομένα που στέλνει ο χρήστης με την κλήση της μεθόδου POST είναι σε μορφή JSON και σύμφωνα με τις προδιαγραφές του συστήματος. Τα δεδομένα του χρήστη όταν θέλει να κάνει συνδρομή είναι δύο ειδών, αναλόγως με τον τύπο της συνδρομής που θέλει να κάνει. Αν θέλει να δέχεται δεδομένα από αισθητήρες όταν αλλάζει η τιμή κάποιου/κάποιων χαρακτηριστικού/κών αυτών των αισθητήρων, τότε το χαρακτηριστικό (property) subType των δεδομένων που στέλνει ο χρήστης πρέπει να είναι ONCHANGE. Το χαρακτηριστικό (attribute), του αισθητήρα που θέλουμε να αλλάζει καθορίζεται στο condAttributes. Στο condAttributes μπορούν να μπουν παραπάνω από ένα χαρακτηριστικά και όταν αλλάζει ένα από αυτά τότε ενημερώνεται ο χρήστης. Για subType=ONCHANGE, η μορφή των δεδομένων του χρήστη πρέπει να είναι:

    {
    "subAttributes":[{"name":"a1","sensorid":"test1"},{"name":"a2","sensorid":"test1"}],
    "subType":"ONCHANGE",
    "condAttributes":[{"name":"a1","sensorid":"test1"}]

}

Αντιθέτως, αν ο χρήστης επιθυμεί να ενημερώνεται ανά ένα χρονικό διάστημα που καθορίζει αυτός τότε πρέπει να καθορίσει το χαρακτηριστικό (property) subType με ONTIMEINTERVAL. Σ' αυτή την περίπτωση αρκεί να καθορίσει ένα ακόμα χαρακτηριστικό (property) interval. Το interval είναι το χρονικό διάστημα σε δευτερόλεπτα μετά των πέρας των οποίων ο χρήστης δέχεται ενημερώσεις από τον Context Broker GE για τα δεδομένα των αισθητήρων που έχει κάνει συνδρομή. Για subType=ONTIMEINTERVAL, η μορφή των δεδομένων του χρήστη πρέπει να είναι:

{
"subAttributes":[{"name":"a1","sensorid":"test1"},{"name":"a2","sensorid":"test1"}],
"subType":"ONTIMEINTERVAL",
"interval":"3"
}

- Στη συνέχεια, αφού βεβαιωθούμε ότι τα δεδομένα που μας παρέχει ο χρήστης είναι σύμφωνα με τις προδιαγραφές που έχουμε θέσει, γίνεται έλεγχος όσων αφορά τις άδειες που έχει καθορίσει ο «διαχειριστής». Αν για παράδειγμα κάποιος αισθητήρας που θέλει να κάνει συνδρομή ο χρήστης δεν έχει καθοριστεί να έχει άδεια συνδρομής ο συγκεκριμένος χρήστης τότε το σύστημα «απαντά» με σφάλμα τύπου SubscribeException. Αν μπορεί να πραγματοποιήσει συνδρομή τότε συνεχίζεται η εκτέλεση στο επόμενο βήμα.
- Στο επόμενο βήμα διαμορφώνονται κατάλληλα τα δεδομένα του χρήστη ώστε να γίνει η κλήση στον Context Broker GE για την αίτηση συνδρομής στην διεύθυνση http://147.27.50.148:1026/NGSI10/subscribeContext.
- Αφού πραγματοποιηθεί με επιτυχία η κλήση στον Context Broker, έπειτα πραγματοποιείται κλήση στον JSON Storage GE στην συλλογή subscriptions, ώστε να αποθηκευθεί η συνδρομή που μόλις πραγματοποιήθηκε.

- **DELETE /subscription**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Μέσω της συνάρτησης getUserInfo(), παίρνουμε τα στοιχεία του χρήστη όπως το μοναδικό αναγνωριστικό συνδρομής subId. Σε περίπτωση που δεν έχει συνδρομή (subId=NULL) το σύστημα «απαντά» με σφάλμα τύπου NotFoundException Αν υπάρχει τότε συνεχίζεται η εκτέλεση στο επόμενο βήμα.
  - Πραγματοποιείται κλήση στον JSON Storage GE με σκοπό την διαγραφή της συνδρομής από την συλλογή subscriptions. Στη συνέχεια, πραγματοποιείται κλήση στην διεύθυνση http://147.27.50.148:1026/NGSI10/unsubscribeContext του Context Broker GE ώστε να διαγραφεί η συνδρομή του χρήστη και να μην

στέλνονται πλέον ενημερώσεις των δεδομένων για την συγκεκριμένη συνδρομή.

- **PUT /subscription**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
  - Μέσω της συνάρτησης getUserInfo(), παίρνουμε τα στοιχεία του χρήστη όπως το μοναδικό αναγνωριστικό συνδρομής subId. Σε περίπτωση που δεν έχει συνδρομή (subId=NULL) το σύστημα «απαντά» με σφάλμα τύπου NotFoundException Αν υπάρχει τότε συνεχίζεται η εκτέλεση στο επόμενο βήμα.
  - Έλεγχος αν τα δεδομένα που στέλνει ο χρήστης είναι σε μορφή JSON και σύμφωνα με τις προδιαγραφές του συστήματος. Τα δεδομένα του χρήστη όταν θέλει να ανανεώσει την συνδρομή του είναι δύο ειδών, αναλόγως με τον τύπο της συνδρομής που θέλει να κάνει. Αν θέλει να δέχεται δεδομένα από αισθητήρες όταν αλλάζει η τιμή κάποιου/κάποιων χαρακτηριστικού/κών αυτών των αισθητήρων, τότε το χαρακτηριστικό (property) subType των δεδομένων που στέλνει ο χρήστης πρέπει να είναι ONCHANGE. Το χαρακτηριστικό (attribute), του αισθητήρα που θέλουμε να αλλάζει καθορίζεται στο condAttributes. Στο condAttributes μπορούν να μπουν παραπάνω από ένα χαρακτηριστικά και όταν αλλάζει ένα από αυτά τότε ενημερώνεται ο χρήστης. Για subType=ONCHANGE, η μορφή των δεδομένων του χρήστη πρέπει να είναι:

    {
    "subType":"ONCHANGE",
    "condAttributes":[{"name":"a1","sensorid":"test1"}]
    }

    Αντιθέτως, αν ο χρήστης επιθυμεί να ενημερώνεται ανά ένα χρονικό διάστημα που καθορίζει αυτός τότε πρέπει να καθορίσει το χαρακτηριστικό (property) subType με ONTIMEINTERVAL. Σ' αυτή την περίπτωση αρκεί να καθορίσει ένα ακόμα χαρακτηριστικό (property) interval. Το interval είναι το χρονικό διάστημα σε δευτερόλεπτα μετά των πέρας των οποίων ο χρήστης δέχεται ενημερώσεις από τον Context Broker GE για τα δεδομένα των αισθητήρων που έχει κάνει συνδρομή. Για subType=ONTIMEINTERVAL, η μορφή των δεδομένων του χρήστη πρέπει να είναι:

    {
    "subType":"ONTIMEINTERVAL",
    "interval":"3"
    }

    Τα δεδομένα των αισθητήρων στα οποία κάνει συνδρομή δεν μπορούν να ανανεωθούν μέσω της κλήσης αυτής της μεθόδου καθώς, ο Context Broker GE δεν μας προσφέρει αυτή τη δυνατότητα. Μπορείς μόνο να ανανεώσεις τον τύπο της συνδρομής.

- Στο επόμενο βήμα διαμορφώνονται κατάλληλα τα δεδομένα του χρήστη ώστε να γίνει η κλήση στον Context Broker GE για την αίτηση ανανέωσης της ήδη υπάρχουσας συνδρομής στην διεύθυνση http://147.27.50.148:1026/NGSI10/subscribeContext.
- Αφού πραγματοποιηθεί με επιτυχία η κλήση στον Context Broker, έπειτα πραγματοποιείται κλήση στον JSON Storage GE στην συλλογή subscriptions, ώστε να ανανεωθεί η ήδη υπάρχουσα συνδρομή.

δ) Διαχείριση Αδειών

- **POST /sensors/{sensorId}/permissions**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId υπάρχει στην συλλογή predefinedSensors του JSON Storage GE. Αν δεν υπάρχει εμφανίζεται στον χρήστη σφάλμα τύπου NotFoundException. Στη συνέχεια ελέγχεται αν ο συγκεκριμένος αισθητήρας έχει ήδη άδεια για κάποιους χρήστες. Αν υπάρχει αποθηκευμένη άδεια τότε εμφανίζεται στον χρήστη PermissionException γιατί η άδεια κάθε αισθητήρα μπορεί να είναι μία εγγραφή στον JSON Storage GE. Σε αντίθετη περίπτωση συνεχίζεται η εκτέλεση στο επόμενο βήμα.
  - Έλεγχος των δεδομένων που δίνει ο χρήστης μέσω της μεθόδου, ώστε να είναι σε μορφή JSON και να είναι σύμφωνα με τις προδιαγραφές. Τα δεδομένα του χρήστη πρέπει να είναι της μορφής:

    {
    "users":["123@gmail.com","kdmortal@gmail.com"]
    }

    Στο χαρακτηριστικό (property) του JSON users δίνονται όλα τα usernames των χρηστών, στους οποίους επιτρέπει ο «διαχειριστής» να κάνουν συνδρομή για τα δεδομένα του αισθητήρα με αναγνωριστικό sensorId.
  - Έλεγχος αν τα usernames των χρηστών που δόθηκαν είναι όντως χρήστες της εφαρμογής. Σε περίπτωση που δεν είναι το σύστημα «απαντά» με σφάλμα τύπου NotFoundException.
  - Στη συνέχεια, αποθηκεύεται η άδεια του συγκεκριμένου αισθητήρα στην βάση Permissions και συλλογή το αναγνωριστικό του αισθητήρα sensorId του JSON Storage GE.

- **GET /sensors/{sensorId}/permissions**
  - Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».

- Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId υπάρχει στην συλλογή predefinedSensors του JSON Storage GE. Αν δεν υπάρχει εμφανίζεται στον χρήστη σφάλμα τύπου NotFoundException. Στη συνέχεια ελέγχεται αν ο συγκεκριμένος αισθητήρας έχει ήδη άδεια για κάποιους χρήστες. Αν υπάρχει αποθηκευμένη άδεια τότε εμφανίζονται στον χρήστη όλοι οι χρήστες, οι οποίοι μπορούν να κάνουν συνδρομή στον συγκεκριμένο αισθητήρα.

- **DELETE /sensors/{sensorId}/permissions**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId υπάρχει στην συλλογή predefinedSensors του JSON Storage GE. Αν δεν υπάρχει εμφανίζεται στον χρήστη σφάλμα τύπου NotFoundException. Στη συνέχεια ελέγχεται αν ο συγκεκριμένος αισθητήρας έχει ήδη άδεια για κάποιους χρήστες. Αν υπάρχει αποθηκευμένη άδεια τότε με κλήση στον JSON Storage GE διαγράφεται η συλλογή για τον αισθητήρα sensorId.

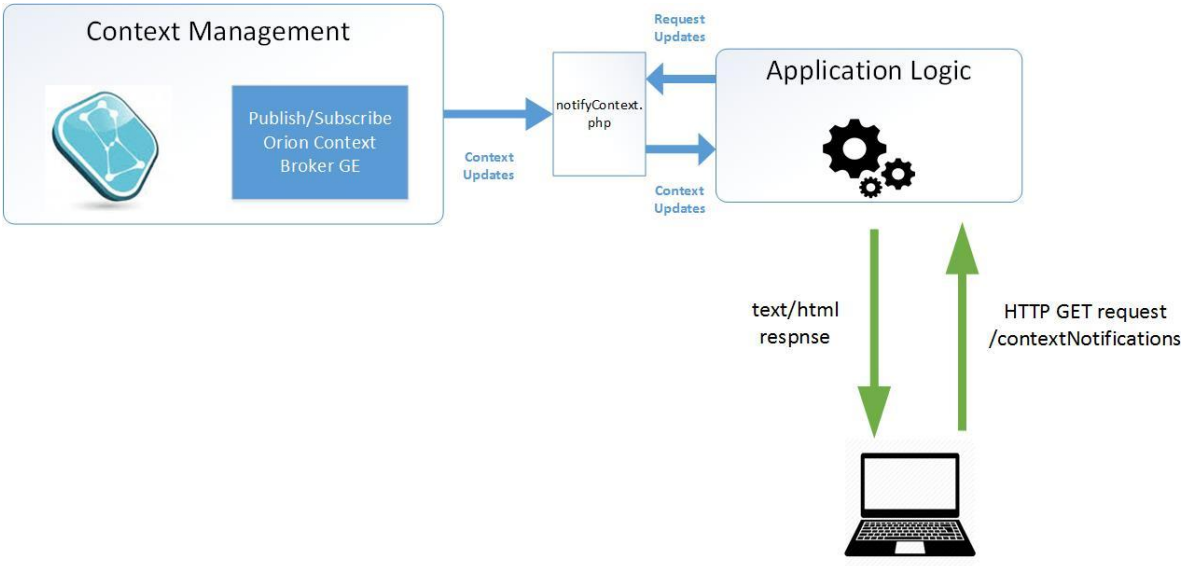
- **PUT /sensors/{sensorId}/permissions**
  - Έλεγχος αν ο χρήστης είναι ο «διαχειριστής».
  - Έλεγχος αν ο αισθητήρας με μοναδικό αναγνωριστικό sensorId υπάρχει στην συλλογή predefinedSensors του JSON Storage GE. Αν δεν υπάρχει εμφανίζεται στον χρήστη σφάλμα τύπου NotFoundException. Στη συνέχεια ελέγχεται αν ο συγκεκριμένος αισθητήρας έχει ήδη άδεια για κάποιους χρήστες. Αν δεν υπάρχει αποθηκευμένη άδεια τότε εμφανίζεται στον χρήστη PermissionException. Σε αντίθετη περίπτωση συνεχίζεται η εκτέλεση στο επόμενο βήμα.
  - Έλεγχος των δεδομένων που δίνει ο χρήστης μέσω της μεθόδου, ώστε να είναι σε μορφή JSON και να είναι σύμφωνα με τις προδιαγραφές. Τα δεδομένα του χρήστη πρέπει να είναι της μορφής:

    {
    "users":["123@gmail.com","kdmortal@gmail.com"]
    }

    Στο χαρακτηριστικό (property) του JSON users δίνονται όλα τα usernames των χρηστών, στους οποίους επιτρέπει ο «διαχειριστής» να κάνουν συνδρομή για τα δεδομένα του αισθητήρα με αναγνωριστικό sensorId.
  - Έλεγχος αν τα usernames των χρηστών που δόθηκαν είναι όντως χρήστες της εφαρμογής. Σε περίπτωση που δεν είναι το σύστημα «απαντά» με σφάλμα τύπου NotFoundException.
  - Στη συνέχεια, ανανεώνεται η ήδη υπάρχουσα άδεια του συγκεκριμένου αισθητήρα στην βάση Permissions και συλλογή το αναγνωριστικό του αισθητήρα sensorId του JSON Storage GE.

ε) Διαχείριση Δεδομένων Αισθητήρων

- **GET /contextNotifications**
  Μέσω αυτής της μεθόδου ο χρήστης ενημερώνεται σε πραγματικό χρόνο για τα δεδομένα των αισθητήρων στα οποία έχει κάνει συνδρομή. Για την λειτουργία της συνάρτησης χρησιμοποιείται το αρχείο **notifyContext.php**, στο οποίο γίνεται η σύνδεση των ενημερώσεων του Context Broker, με τα δεδομένα που εμφανίζονται στον χρήστη. Για την υλοποίηση της notifyContext.php, έγινε χρήση των Server Side Events[26] ώστε ασύγχρονα να στέλνονται δεδομένα στον χρήστη όταν γίνει ενημέρωση από τον Context Broker. Η λειτουργία αυτής της μεθόδου μπορεί να κατανοηθεί καλύτερα από το παρακάτω σχήμα:



Σχήμα 17. Λειτουργία μεθόδου GET /contextNotifications

- Έλεγχος αν ο χρήστης είναι ταυτοποιημένος από τον «διαχειριστή».
- Έλεγχος αν ο χρήστης έχει ήδη κάποια συνδρομή. Σε περίπτωση που δεν έχει εμφανίζεται σφάλμα τύπου NotFoundException.
- Το σύστημα «απαντά» στον χρήστη με κώδικα HTML ο οποίος συνδέεται με το αρχείο notifyContext, ώστε όταν δεχθεί ενημέρωση από τον Context Broker GE να εμφανισθούν τα δεδομένα του αισθητήρα σε πραγματικό χρόνο στον χρήστη. Τα δεδομένα που εμφανίζονται στον χρήστη ως απάντηση του συστήματος είναι της μορφής:
  {"subscriptionId":"553e93a498702c804cc318ce","originator":"localhost","contextResponses":[{"contextElement":{

[26] http://www.w3schools.com/html/html5_serversentevents.asp

"type":"Sensor","isPattern":"false","id":"Atmo","attributes":[{"name":"temperature_Atmo","type":"celsius","value":"24"},{"name":"pressure_Atmo","type":"bar","value":"70"}]},"statusCode":{"code":"200","reasonPhrase":"OK"}}]}

σε περίπτωση που ο χρήστης έχει κάνει συνδρομή στα χαρακτηριστικά temperature και pressure του περιβαλλοντικού αισθητήρα Atmo. Το JSON που παρουσιάζεται ως απάντηση στον χρήστη είναι η ενημέρωση που λαμβάνουμε από τον Context Broker GE σύμφωνα με την συνδρομή που κάναμε.

- **POST /event**

  Μέσω αυτής της μεθόδου ο «διαχειριστής» έχει τη δυνατότητα να στείλει δεδομένα αισθητήρων στο σύστημα. Για την πλήρη λειτουργία αυτής της μεθόδου γίνεται χρήση δύο συναρτήσεων της **checkEvent**, η οποία ελέγχει αν τα δεδομένα που στέλνει ο «διαχειριστής» είναι JSON,είναι σύμφωνα με τις προδιαγραφές και επιπλέον αν συμβαδίζουν με το σχήμα του αισθητήρα. Η δεύτερη συνάρτηση είναι η dataReciever, η οποία κάνει τις απαραίτητες κλήσεις σε Context Broker για την δημιουργία ή ανανέωση της οντότητας αλλά και στον JSON Storage GE για την αποθήκευση του αισθητήρα στη συλλογή connectedSensors, αποθήκευση των δεδομένων του αισθητήρα κ.α. Αναλυτικά η υλοποίηση της μεθόδου περιγράφεται στα παρακάτω βήματα:

  - Έλεγχος αν ο χρήστης είναι «διαχειριστής».
  - Έλεγχος των δεδομένων που στέλνει ο χρήστης μέσω της συνάρτησης checkEvent, ώστε να είναι σε μορφή JSON, σύμφωνα με το σχήμα του αισθητήρα και σύμφωνα με τις προδιαγραφές που έχουμε θέσει. Τα δεδομένα του χρήστη πρέπει να είναι της μορφής:

    {
    "id": "Atmo",
    "attributes":[{"name":"temperature","type":"celsius","value":"20"},{"name":"humidity","type":"percentage","value":"75"}]
    }

    Όπου id είναι το μοναδικό αναγνωριστικό του αισθητήρα και στο χαρακτηριστικό attributes περιλαμβάνονται τα δεδομένα του αισθητήρα με τις πιο πρόσφατες τιμές.
  - Στη συνέχεια πραγματοποιείται έλεγχος αν ο αισθητήρας με το id που δίνει ο χρήστης είναι αποθηκευμένος στην συλλογή predefinedSensors του JSON Storage GE. Σε περίπτωση που δεν βρεθεί σχήμα αισθητήρα με αναγνωριστικό το id που στέλνει ο χρήστης, τότε του εμφανίζεται σφάλμα τύπου NotFoundException. Αλλιώς, η εκτέλεση συνεχίζεται στο επόμενο βήμα.
  - Έπειτα ελέγχουμε αν ο αισθητήρας είναι ήδη συνδεδεμένος. Αν δεν είναι τότε πραγματοποιείται κλήση στον Context Broker ώστε να δημιουργήσει

την οντότητα και στην συνέχεια καλείται δύο φορές η υπηρεσία JSON Storage GE, αρχικά για την αποθήκευση του αισθητήρα στη συλλογή με τους συνδεδεμένους και έπειτα για την δημιουργία της συλλογής με το αναγνωριστικό (sensorId) του αισθητήρα όπου θα αποθηκευτούν τα δεδομένα του στη βάση δεδομένων Log. Στην περίπτωση που είναι ήδη συνδεδεμένος, απλά ανανεώνουμε την αποθηκευμένη οντότητα πραγματοποιώντας κλήση στον Context Broker στην διεύθυνση http://147.27.50.148:1026/NGSI10/updateContext.

- Στη συνέχεια, είτε ο αισθητήρας είναι ήδη συνδεδεμένος είτε όχι αποθηκεύουμε τα δεδομένα του στη συλλογή που δημιουργήσαμε με το αναγνωριστικό (sensorId) του στη βάση δεδομένων Log της υπηρεσίας JSON Storage GE.

3.6.3 Αιτήματα (Requests) και Απαντήσεις (Responses)

Στην κλήση κάθε μεθόδου του API, όταν δημιουργούμε το αίτημα που θα σταλεί στον SE, είναι υποχρεωτικό να καθορίσουμε δύο επικεφαλίδες (headers). Η μια είναι αναγκαία για την ταυτοποίηση του χρήστη όπως είχαμε αναφέρει και στην σχετική υποενότητα 3.7. Η χρήση του API είναι περιορισμένη μόνο για χρήστες που είναι εγγεγραμμένοι στο FIWARE, γι' αυτό το λόγο σε κάθε αίτημα προς τον SE είναι απαραίτητο να γίνεται βασική HTTP ταυτοποίηση (basic HTTP authentication) με το username και τον κωδικό του χρήστη. Επιπλέον απαραίτητη είναι και η εισαγωγή της επικεφαλίδας Content-type στην τιμή application/json καθώς ο SE στέλνει και δέχεται δεδομένα σε μορφή JSON.

Επικεφαλίδα	**Τιμή**
Content-type	application/json
Authorization	Basic {base64(username:password)}

Πίνακας 8. Απαραίτητες επικεφαλίδες στην κλήση μεθόδων του API

Αντίστοιχος πίνακας με τις «απαντήσεις» (responses) που δέχεται ο χρήστης όσων αφορά τα σφάλματα που παράγουν οι «εξαιρέσεις» και οι κωδικοί κατάστασης που τις συνοδεύουν.

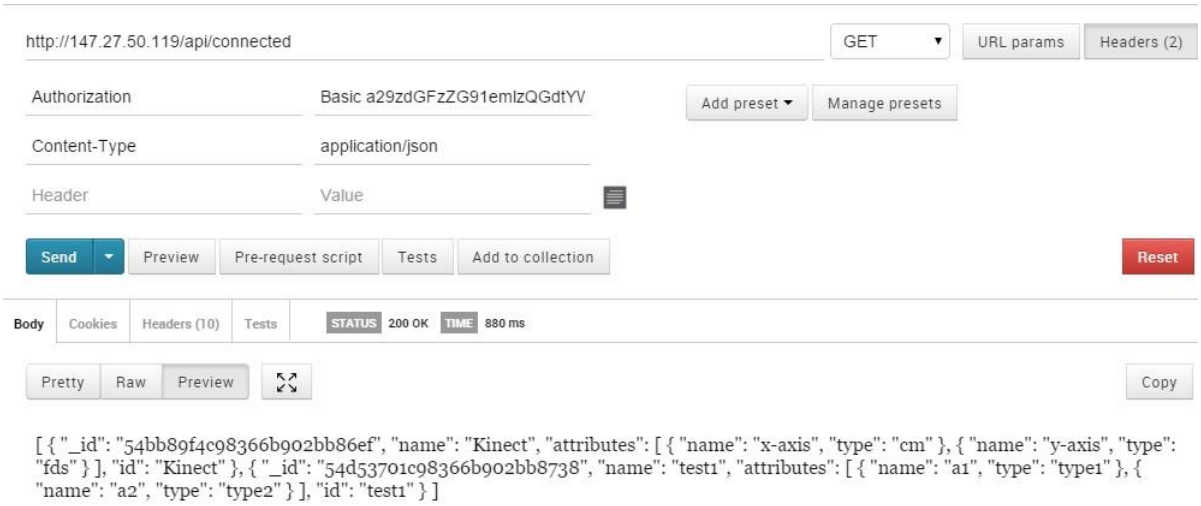
Εξαίρεση (Exception)	Κωδικός Κατάστασης (Status Code)
CurlException	500
NotFoundException	404
SubscriptionException	400
ContextBrokerException	500
JsonException	400
DataFormatException	400
NotAllowedException	403
PermissionException	400
InvalidTypeException	400

Πίνακας 9. Εξαιρέσεις και Κωδικοί Κατάστασης ως απαντήσεις σφάλματος στον χρήστη

3.6.4 Παραδείγματα

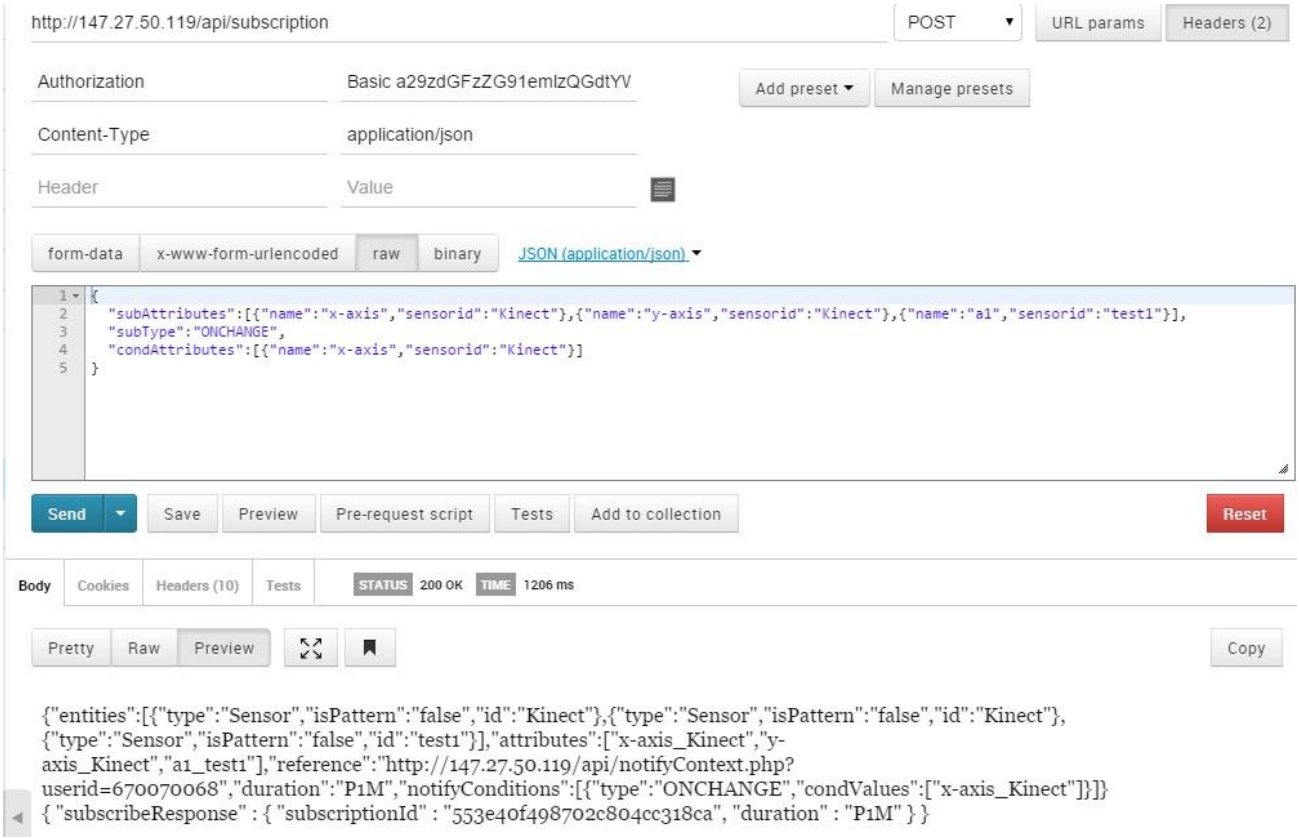
Παρακάτω γίνεται χρήση κάποιων μεθόδων του API για την καλύτερη κατανόηση της λειτουργίας τους.

Αρχικά, θέλουμε να ανακτήσουμε όλους τους συνδεδεμένους στο σύστημα αισθητήρες. Αυτό θα γίνει με χρήση της μεθόδου GET /connected του API. Στην παρακάτω εικόνα εμφανίζεται το αίτημα που γίνεται στον SE, αλλά και η απάντηση που παίρνει ο χρήστης από τον SE.



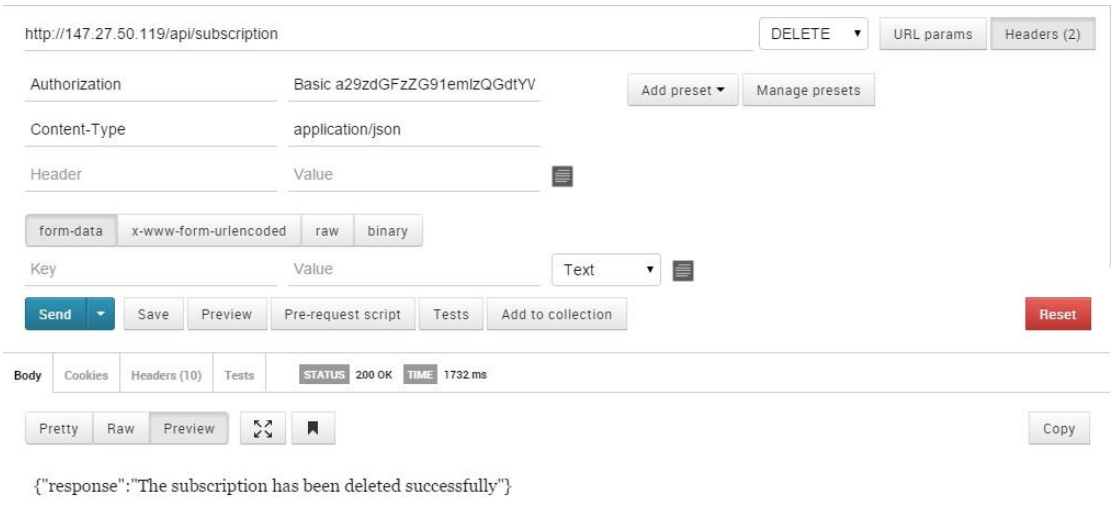
Σχήμα 18. Αίτημα και απάντηση του SE στην κλήση της μεθόδου GET /connected

Έπειτα θέλουμε να κάνει συνδρομή ο χρήστης σε όλα τα δεδομένα του αισθητήρα Kinect και στο χαρακτηριστικό (attribute) a1 του αισθητήρα test1. Στο συγκεκριμένο παράδειγμα επέλεξε ο χρήστης να ενημερώνεται για τα χαρακτηριστικά x-axis,y-axis του αισθητήρα Kinect και το χαρακτηριστικό a1 του αισθητήρα test1. Όταν αλλάξει η τιμή του χαρακτηριστικού x-axis του αισθητήρα Kinect, τότε ο Context Broker θα ενημερώσει τα δεδομένα που επέλεξε να κάνει συνδρομή ο χρήστης.



Σχήμα 19. Αίτημα και απάντηση του SE στην κλήση της μεθόδου POST /subscription

Στη συνέχεια, διαγράφουμε την συνδρομή του χρήστη.



Σχήμα 20. Αίτημα και απάντηση του SE στην κλήση της μεθόδου DELETE /subscription

3.6.5 Απαραίτητα εργαλεία

Για την ολοκλήρωση της υλοποίησης της διπλωματικής εργασίας έγινε εκτενής χρήση κάποιων εργαλείων:

- Filezilla[27], για την σύνδεση στην εικονική μηχανή μέσω SSH (Secure Shell)[28] και την μεταφορά αρχείων σε αυτή μέσω sftp.
- Postman[29], για την δημιουργία των απαραίτητων requests και κλήση των αντίστοιχων μεθόδων του API. Επίσης χρησιμοποιήθηκε για την αποσφαλμάτωση των μεθόδων της υπηρεσίας.
- Slim Framework, με τις βιβλιοθήκες PHP που μας προσέφερε βοήθησαν αρκετά στην ελαχιστοποίηση της προσπάθειας από μέρους μας στην δημιουργία του REST API και της ταυτοποίησης του χρήστη με την παροχή ειδικού Middleware.

3.7 Πείραμα και αποτελέσματα

Για την βαθύτερη κατανόηση της απόδοσης του SE σε πραγματικές συνθήκες προσομοιώσαμε την λειτουργία δύο αισθητήρων Atmo και Zephyr, ώστε να ελέγξουμε κατά πόσο μπορεί ο SE να ανταποκριθεί πλήρως λειτουργικά σε συνεχή δεδομένα από αισθητήρες. Επιλέξαμε να ελέγξουμε την απόδοση της μεθόδου /event POST, καθώς είναι η μέθοδος του API που χρησιμοποιείται πιο συχνά, επομένως επηρεάζει την απόδοση του API σε μεγαλύτερο βαθμό από οποιαδήποτε άλλη μέθοδο.

Το πείραμα που πραγματοποιήσαμε εκτελείται σε 4 φάσεις. Και στις 4 φάσεις του πειράματος περιλαμβάνονται 2 χρήστες που κάνουν συνδρομή στα δεδομένα που παρέχουν δύο αισθητήρες στο σύστημα ο Zephyr που είναι ιατρικός αισθητήρας και μετράει τον καρδιακό παλμό σε bpm (beats per minute) και τον περιβαλλοντικό αισθητήρα Atmo, ο οποίος μετράει την εξωτερική θερμοκρασία σε βαθμούς κελσίου και την υγρασία σε εκατοστιαίο ποσοστό. Για την προσομοίωση της λειτουργίας αυτών των αισθητήρων δημιουργήσαμε 2 αρχεία php τα οποία στέλνουν τα δεδομένα των αισθητήρων σε ένα παράθυρο 5 λεπτών ανά ένα χρονικό διάστημα που καθορίζει η εκάστοτε φάση του πειράματος. Το χρονικά διαστήματα ανανέωσης των δεδομένων των αισθητήρων καλύπτουν ένα μεγάλο εύρος από 15 δευτερόλεπτα στην $1^{η}$ φάση μέχρι 2 δευτερόλεπτα στην $4^{η}$ φάση, ώστε να αποτυπώνεται πιο καθαρά το

[27] https://filezilla-project.org/
[28] http://en.wikipedia.org/wiki/Secure_Shell
[29] https://www.getpostman.com/

αποτέλεσμα της κάθε φάσης του πειράματος. Αξίζει να σημειωθεί ότι, το κάτω όριο στον καθορισμό του χρονικού διαστήματος ήταν τα 2 δευτερόλεπτα, καθώς μικρότερο χρονικό διάστημα θα είχε ως αποτέλεσμα να μην εμφανίζονται τα δεδομένα στον χρήστη (λόγω της εκτέλεσης της μεθόδου POST /event με μέσο χρόνο απόκρισης ~1.6).

Παρακάτω αποτυπώνονται τα αποτελέσματα των πειραμάτων, που είναι η διαφορά του χρόνου μεταξύ της χρονικής στιγμής που γίνεται η κλήση στον SE (κλήση της POST /event) για να στείλει τα δεδομένα των αισθητήρων και την χρονική στιγμή που εμφανίζονται τα δεδομένα στον χρήστη.

Zephyr	**Atmo**	**Αποτέλεσμα Zephyr**	**Αποτέλεσμα Atmo**
15 δευτερόλεπτα	15 δευτερόλεπτα	~1.6 δευτερόλεπτα	~1.6 δευτερόλεπτα
10 δευτερόλεπτα	10 δευτερόλεπτα	~1.6 δευτερόλεπτα	~1.6 δευτερόλεπτα
5 δευτερόλεπτα	5 δευτερόλεπτα	~1.6 δευτερόλεπτα	~1.6 δευτερόλεπτα
Τυχαίο (μεταξύ 2 με 4 δευτερόλεπτα)	Τυχαίο (μεταξύ 2 με 4 δευτερόλεπτα)	~1.6 δευτερόλεπτα	~1.6 δευτερόλεπτα

Πίνακας 10. Πείραμα και αποτελέσματα

Συμπεραίνουμε, λοιπόν ότι όσο μικρό χρονικό διάστημα ανανέωσης των δεδομένων και αν θέσουμε, δεν επηρεάζεται σε μεγάλο βαθμό η διαφορά των χρόνων μεταξύ της αποστολής των δεδομένων και της εμφάνισης στον χρήστη. Αυτή η χρονική διαφορά είναι περίπου 1.6 δευτερόλεπτα. Και αυτό συμβαίνει γιατί η εκτέλεση της συνάρτησης POST /event του API λόγω των ελέγχων που κάνει στο JSON του χρήστη και λόγω των κλήσεων στις διάφορες υπηρεσίες έχει μια σταθερή χρονική καθυστέρηση γύρω στα 1.6 δευτερόλεπτα. Αυτή η καθυστέρηση δικαιολογείται λόγω των αιτημάτων που γίνονται στη μέση περίπτωση, ένα αίτημα στον Context Broker GE με μέσο χρόνο απόκρισης 700 ms και δύο αιτήματα που πραγματοποιούνται στον JSON Storage GE με μέσο χρόνο απόκρισης 400 ms. Αν συμπεριληφθούν στους χρόνους αυτούς και οι καθυστερήσεις λόγω του δικτύου τότε γίνεται αντιληπτό ότι η μέση χρονική καθυστέρηση των 1.6 δευτερολέπτων για την ειδοποίηση του συνδρομητή είναι ρεαλιστική.

Κεφάλαιο 4ο

4. Συμπεράσματα και Μελλοντική Δουλειά

4.1 Συμπεράσματα

Η υπηρεσία I.I.M που αναπτύχθηκε στα πλαίσια της διπλωματικής μου εργασίας, πέτυχε τους στόχους που είχαμε θέσει εξαρχής. Καθοριστική συμβολή στην επίτευξη των στόχων έπαιξε η τεχνολογία Υπολογιστικού Νέφους, η οποία συνδυάστηκε με την ιδέα του Διαδικτύου των Πραγμάτων. Μπορέσαμε και εκμεταλλευτήκαμε τα πλεονεκτήματα που μας προσέφεραν αυτές οι τεχνολογίες συνδυασμένες, οπότε είμαστε σε θέση να συνοψίσουμε τους αρχικούς στόχους από άποψη λειτουργικότητας, οι οποίοι και επιτεύχθηκαν.

Η λύση που προτείνουμε είναι η υπηρεσία I.I.M (Intellicloud IoT Management), η οποία διαχειρίζεται τους χρήστες και τους αισθητήρες με στόχο την άμεση ενημέρωση συνδρομητών-χρηστών, σε ενημερώσεις των δεδομένων του εκάστοτε αισθητήρα.

Οι βασικές λειτουργίες που υποστηρίζονται:

- Προσθήκη/αφαίρεση/ανανέωση αισθητήρων από τον διαχειριστή.
- Προσθήκη/αφαίρεση/ανανέωση συνδρομών χρήστη.
- Προσθήκη/αφαίρεση/ανανέωση δικαιωμάτων χρηστών στην συνδρομή αισθητήρων, από τον διαχειριστή.
- Άμεση ενημέρωση συνδρομητών σε ενημερώσεις αισθητήρων που έχουν συνδρομή.
- Ταυτοποίηση/διαγραφή χρηστών από τον διαχειριστή.
- Βάση δεδομένων με το ιστορικό των δεδομένων των αισθητήρων.

Κυριότερα πλεονεκτήματα της υπηρεσίας I.I.M:

- ***Επεκτάσιμο***, μπορούν να προστεθούν υπηρεσίες για την επέκταση των λειτουργιών του SE, όπως για παράδειγμα εργαλεία ανάλυσης δεδομένων.
- ***Εύκολο στην χρήση***, με ένα απλό και εύχρηστο REST (Representational State Transfer) API που προσφέρεται σε οποιονδήποτε θέλει να το χρησιμοποιήσει.

- ***Υποστήριξη πολλών διαφορετικών αισθητήρων και χρηστών/ελαστικότητα,*** προσαρμογή των υπολογιστικών πόρων της υποδομής νέφους σε περίπτωση απαίτησης από την εφαρμογή.
- ***Συμβατότητα,*** με υπηρεσίες που προσφέρουν περιβάλλοντα Νέφους όπως το FIWARE.

Σημαντικό κομμάτι του συστήματός αποδείχθηκε η χρήση των υπηρεσιών γενικού σκοπού. Με την εισαγωγή αυτών στην υλοποίηση του συστήματος, καταφέραμε να το κάνουμε πλήρως λειτουργικό και να πληροί τις προδιαγραφές που αρχικά είχαμε θέσει. Ακόμα, ο φόρτος εργασίας μειώθηκε, καθώς η πολυπλοκότητα του συστήματος απαλείφτηκε και ο κώδικας που ήταν απαραίτητος να αναπτύξουμε ήταν σημαντικά λιγότερος. Τέλος, η ευελιξία που μας προσέφερε η υποδομή Νέφους Intellicloud μέσω της παροχής υπολογιστικών πόρων έπαιξε σημαντικό ρόλο στην ολοκλήρωση της εργασίας καθώς ήμασταν σε θέση να δεσμεύουμε δυναμικά και on-demand υπολογιστικούς πόρους ανά πάσα ώρα και στιγμή χωρίς να επιβαρύνουμε την υποδομή Νέφους.

4.2 Περιορισμοί

Σημαντικά ήταν τα προβλήματα και οι δυσκολίες τις οποίες κληθήκαμε να ξεπεράσουμε κατά την υλοποίηση του συστήματος. Παρακάτω παρουσιάζονται συνοπτικά.

- Μεγάλος χρόνος απόκρισης στην αποστολή αιτημάτων στην υπηρεσία γενικού σκοπού Context Broker GE όσων αφορά την ανανέωση των δεδομένων της οντότητας. Ο μέσος χρόνος των 700 ms, κρίνεται αρκετά μεγάλος για εφαρμογές που υλοποιούνται με στόχο την άμεση απόκριση στα αιτήματα του χρήστη, όπως εφαρμογές πραγματικού χρόνου (real-time).

- Στην περίπτωση της συνδρομής χρήστη το αίτημα που φτάνει στον Context Broker GE δεν ελέγχεται ώστε να εντοπίσει αν τα δεδομένα που του παρείχε ο χρήστης είναι όντως έγκυρα. Για παράδειγμα, αν ένας χρήστης κάνει συνδρομή στην οντότητα Kinect στα χαρακτηριστικά test1 και test2, ο Context Broker δεν θα ελέγξει αν όντως η οντότητα Kinect περιέχει αυτά τα χαρακτηριστικά, με αποτέλεσμα να γίνεται η συνδρομή χωρίς να εμφανίζονται ενημερώσεις του αισθητήρα στον χρήστη. Αυτός ο έλεγχος θα έπρεπε να ήταν ενσωματωμένος στην λειτουργικότητα του Context Broker GE κάνοντας μια απλή σύγκριση με τη βάση όπου κρατούνται οι οντότητες.

Κάτι τέτοιο ήταν αδύνατο να πραγματοποιηθεί από εμάς, καθώς θα προσέθετε επιπλέον πολυπλοκότητα και μείωση της απόδοσης του SE.

- Λόγω των περιορισμών των υπηρεσιών γενικού σκοπού και σε συνδυασμό με τις καθυστερήσεις που εισάγονται λόγω του δικτύου, γίνεται αντιληπτό ότι το σύστημα μας δεν ανταποκρίνεται λειτουργικά σε εφαρμογές που στέλνουν πάρα πολλά δεδομένα σε πολύ μικρό χρονικό διάστημα (μικρότερο από 1.6 δευτερόλεπτα), όπως για παράδειγμα αισθητήρες κίνησης. Αυτό συμβαίνει διότι λόγω του αυξανόμενου φόρτου στις ειδοποιήσεις των συνδρομητών θα προστίθεται συνεχώς μια επιπλέον καθυστέρηση πράγμα που θα κάνει το σύστημα μας λιγότερο αποδοτικό σε τέτοιες περιπτώσεις.

- Δεν δοκιμάστηκε η χρήση της υπηρεσίας Sensor Data Collection Service για επικοινωνία με την υπηρεσία I.I.M, καθώς ακόμα βρίσκεται σε δοκιμαστικό στάδιο και δεν υποστηρίζει μεγάλο εύρος συσκευών.

4.3 Μελλοντική Δουλειά

Το σύστημα που αναπτύξαμε έχει την δυνατότητα με συγκεκριμένες επεκτάσεις, οι οποίες επισημαίνονται συνοπτικά παρακάτω να γίνει περισσότερο λειτουργικό και αποδοτικό.

- Ο χρήστης να έχει τη δυνατότητα να καθορίζει μοτίβα (patterns) σε συγκεκριμένα δεδομένα αισθητήρων στα οποία έχει συνδρομή και να μπορεί να ειδοποιείται όταν κάποιο μοτίβο ικανοποιηθεί. Για παράδειγμα αν η θερμοκρασία ενός εσωτερικού χώρου ανεβαίνει συνεχώς σε διάστημα 5 λεπτών τότε ειδοποιείται ο χρήστης.

- Έρευνα για την διασφάλιση της ασφάλειας στην διαχείριση των δεδομένων από την υπηρεσία.

- Αυτόματη αρχικοποίηση των βάσεων και των συλλογών όταν αρχικοποιείται ένα στιγμιότυπο, χωρίς να χρειάζεται η παρέμβαση του χρήστη. Όταν αρχικοποιηθεί η εικονική μηχανή για $1^{η}$ φορά να εκτελείται ένα αρχείο στο λειτουργικό σύστημα.

- Έρευνα για τη χρήση ειδικών εργαλείων για την ανάλυση και διαχείριση των τεράστιων σε όγκο δεδομένων (Big Data Analysis) που παράγονται.

- Έρευνα για την αυτόματη προσαρμογή των υπολογιστικών πόρων στις ανάγκες του συστήματος (scalability).

Βιβλιογραφία

1. Sensor Data Collector Service Documentation http://catalogue.fi-star.eu/enablers/sensor-data-collection-service
2. A view of Cloud Computing by Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia http://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext
3. The NIST Definition of Cloud by Peter Mell, Timothy Grance http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf
4. Cloud Computing, A Practical Approach by Toby Velte, Anthony Velte, Robert Elsenpeter http://dl.acm.org/citation.cfm?id=1594816
5. Cloud Computing Security Risk Assessment by Daniele Catteddu, Giles Hogben http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment
6. Addressing cloud computing security issues by Dimitrios Zissis, Dimitrios Lekkas http://www.sciencedirect.com/science/article/pii/S0167739X10002554
7. OpenStack: Toward an Open-Source Solution for Cloud Computing, by Omar SEFRAOUI, Mohammed AISSAOUI, Mohsine ELEULDJ http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.245.229&rep=rep1&type=pdf
8. Service-oriented architecture by Perrey R., Lycett M. http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=1210138&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1210138
9. REST Representational State Transfer, by Michael Jakl http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.7334&rep=rep1&type=pdf

10. The application/json Media Type for Javascript Object Notation (JSON) https://tools.ietf.org/html/rfc4627
11. Virtualization and Cloud Computing, by Yuping Xin and Yongzhao Zhan http://link.springer.com/chapter/10.1007/978-3-642-27323-0_39
12. Internet of Things, by Feng Xia, Laurence T. Yang, Lizhe Wang, Alexey Vinel http://www.homeworkmarket.com/sites/default/files/q5/04/07/danainfo.acppwiszgmk2n0u279qu76contentserver.pdf
13. http://www.theinquirer.net/inquirer/news/2402238/ibm-to-splash-out-usd3bn-on-iot-and-cloud-based-open-platform
14. KeyRock Identity Management GE Documentation http://catalogue.fiware.org/enablers/documentation-18
15. Context Broker GE Documentation http://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker/documentation