Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εργαστήριο Προγραμματισμού και Τεχνολογίας Ευφυών Συστημάτων

Κατανεμημένη Αρχιτεκτονική Υπηρεσιών Νέφους για το Σημασιολογικό Δίκτυο των Πραγμάτων

(Federated Service Oriented Architecture for the Semantic Internet of Things in the Cloud)

Διπλωματική Εργασία

Ραστσίνσκαγια Καρίνα

Εξεταστική επιτροπή:

Πετράκης Ευριπίδης, Καθηγητής (Επιβλέπων)

Σαμολαδάς Βασίλης, Αναπληρωτής Καθηγητής

Δεληγιαννάκης Αντώνιος, Αναπληρωτής Καθηγητής

Χανιά, Ιανουάριος 2021

Περίληψη

Στη παρούσα εργασία παρουσιάζεται μια Κατανεμημένη Αρχιτεκτονικη Σημασιολογικού Ιστού των Πραγμάτων στο Υπολογιστικό Νέφος, (Federated Service Oriented Architecture for the Semantic Internet of Things in the Cloud), η οποία απαρτίζεται από ανεξάρτητα και ισοδύναμα συστήματα Semantic Web of Things (κόμβους) που αλληλεπιδρούν μεταξύ τους. Το Fi-Swot εφαρμόζει τεχνολογίες Σημασιολογικού Ιστου του Πραγμάτων με σκοπό την καλύτερη κατανόηση των ιδιοτήτων των συσκευών, των υπηρεσιών που προσφέρουν και την διασύνδεση μεταξύ τους για την δημιουργια εφαρμογών. Κάθε κόμβος του Fi-SWoT έχει σχεδιαστεί ακολουθώντας πρότυπα Σημασιολογικού Ιστου (Semantic Web) και Μοντέλου του Ιστού των Πραγμάτων (Web Thing Model), ενώ ταυτόχρονα υιοθετεί μια υπηρεσιοκεντρική αρχιτεκτονική (SOA) και υλοποιείται ως σύνθεση των RESTful μικρο-υπηρεσιών στο υπολογιστικό νέφος (Cloud). Η πρόσβαση σε έναν κόμβο επιτρέπεται μόνο σε εξουσιοδοτημένους χρήστες του ίδιου ή άλλων κόμβων με βάση τους ρόλους των χρηστών και τις πολιτικές πρόσβασης. Το Fi-SWoT ακολουθεί ένα μοντέλο αρχιτεκτονικής 3 επιπέδων όπου κάθε επίπεδο εξυπηρετεί διαφορετική λειτουργικότητα για διαφορετικούς τύπους χρηστών. Οι κύριοι τύποι χρηστών είναι οι Διαχειριστές Συστήματος (System Administrators), οι Ιδιοκτήτες Υποδομής (Infrastructure Owners) και οι Πελάτες (Customers). Οι διαχειριστές έχουν το δικαίωμα να ελέγχουν και να τροποποιούν το ατομικό τους σύστημα SWoT καθώς και τους χρήστες που είναι συνδεδεμένοι στο σύστημά τους. Οι ιδιοκτήτες υποδομής έχουν το δικαίωμα να διαχειρίζονται τις δικές τους συσκευές και να δημιουργούν συνδρομές σε συσκευές που υπάρχουν είτε στο τοπικό τους κόμβο είτε σε απομακρυσμένο, με σκοπό να τις χρησιμοποιήσουν στις εφαρμογές που θα κατασκευάσουν. Οι πελάτες έχουν την άδεια να δημιουργούν συνδρομές σε εφαρμογές που υπάρχουν είτε στο τοπικό τους κόμβο είτε σε απομακρυσμένο, προκειμένου να έχουν πρόσβαση σε αυτές. Το προαναφερθέν μοντέλο συμβάλει στην επέκταση του συστήματος επιτρέποντας την εισαγωγή διαφόρων τύπων συσκευών και τη δημιουργία ποικίλων εφαρμογών. Η απόδοση της παραπάνω αρχιτεκτονικής ελέγχεται προσθέτοντας μεγάλο αριθμό εικονικών οντοτήτων (αισθητήρων, σπιτιών κλπ) και παράγοντας μεγάλα ποσά δεδομένων (μετρήσεων). Τα αποτελέσματα της πειραματικής διαδικασίας δείχνουν ότι το σύστημα μπορεί να ανταποκριθεί σε πραγματικούς χρόνους υπό συνθήκες μεγάλου φόρτου εργασίας.

Abstract

This paper presents a Federated Service Oriented Architecture for the Semantic Internet of Things in the Cloud, consisting of independent and equivalent Semantic Web of Things (nodes) systems that interact with each other. Fi-Swot applies Semantic Web technologies of things in order to better understand the properties of the devices, the services they offer and the interconnection between them to create applications. Each Fi-SWoT node has been designed following Semantic Web and Web Thing Model standards, while at the same time adopting a service oriented architecture (SOA) and implementing a synthesis of RESTful micro-services in the cloud computing (Cloud). Access to a node is only allowed to authorized users of the same or other nodes based on user roles and access policies. The Fi-SWoT follows a three-level architectural model where each level serves different functionality for different types of users. The main types of users are System Administrators, Infrastructure Owners and Customers. Administrators have the right to control and modify their individual SWoT system as well as users logged in to their system. Infrastructure owners have the right to manage their own devices and create subscriptions to devices that exist either at their local node or remotely, in order to use them in the applications they will build. Customers are allowed to create subscriptions to applications that exist either on their local or remote node in order to access them. The above model contributes to the expansion of the system by allowing the introduction of different types of devices and the creation of various applications. The performance of the above architecture is controlled by adding a large number of virtual entities (sensors, houses, etc.) and generating large amounts of data (measurements). The results of the experimental process show that the system can respond in real time under conditions of high workload.

Ευχαριστίες (Acknowledgements)

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κύριο Ευριπίδη Πετράκη, για την πολύτιμη βοήθειά του, την στήριξη και την καθοδήγηση που μου προσέφερε με τις συμβουλές του. Επίσης, θα ήθελα να ευχαριστήσω όλα τα μέλη του εργαστηρίου για την άριστη επικοινωνία και συνεργασία που είχαμε. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για όλη την ψυχική και υλική υποστήριξη κατά τη διάρκεια όλης της διαδικασίας των σπουδών μου, καθώς τη φίλη και συνάδελφό μου Ευφροσύνη Ανέστη, που με βοήθησε να τελειώσω έγκαιρα τις σπουδές μου, και με επιτυχία συνεργαστήκαμε στις εργαστηριακές εργασίες μας.

Περιεχόμενα

Κεφάλαιο 1 **8**
1.1 Σκοπός της εργασίας 8
1.1.1 Εξέλιξη του Διαδικτύου των Πραγμάτων (IoT) 8
1.2 Προτεινόμενη Λύση 10
1.3 Μεθοδολογία και Συνεισφορά της Εργασίας 11
1.4 Δομή της εργασίας 15
Κεφάλαιο 2 **16**
2.1 Σχετικές εργασίες 16
2.2 Διαδίκτυο των πραγμάτων (IoT) 17
2.3 Υπολογιστικό Νέφος (Cloud) 17
2.4 Web Thing Model 18
2.5 Semantic Web Of Things 19
2.6 Σχετικές Τεχνολογίες 19
2.6.1 Υπηρεσιοκεντρική Αρχιτεκτονική (SOA) 19
2.6.2 RESTful Υπηρεσίες Ιστού 19
2.6.3 Ταυτοποίηση και Εξουσιοδότηση Χρηστών 20
2.6.3.1 Keycloak 20
2.6.3.2 OAuth2.0 και OpenID Connect Authentication 21
2.6.3.3 Authorization PDP – AuthZForce 21
2.6.4 Υπηρεσίες στο Fiware 22
2.6.4.1 Publish/Subscribe Context Broker - Orion-LD Context Broker 22
2.6.4.2 Authorization PDP – AuthZForce 22
2.6.4.3 PEP Proxy – Wilma 23
2.6.4.4 Cygnus-LD 23
2.6.4.5 Backend Device Manager - IDAS GE 23
2.6.5 Υπηρεσία Mashup - Node RED 24
2.6.6 Μοντέλο πληροφορίας NGSI-LD 24
2.6.7 Βάσεις Δεδομένων 26
2.6.7.1 Βάση Δεδομένων MongoDB 26
2.6.7.2 Βάση Δεδομένων PostgreSQL 26
2.6.7.3 Apache Cassandra 27
2.6.8 Οντολογίες (SSN kai SOSA) 27
2.6.9 RDF - RDFS 28
2.6.10 OWL 28
2.6.11 SPARQL 29
2.6.12 Virtuoso 29
2.6.13 Apache Jena 29

2.6.14 Docker 30

Κεφάλαιο 3 31

3.1 Σενάριο Χρήσης 31

3.2 Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams) 35

3.3 Διαγράμματα Ακολουθίας / Δραστηριοτήτων 40

3.4 Υπηρεσίες Συστήματος ενός κόμβου 55

3.4.1 Υπηρεσία Διακομιστή μεσολάβησης του Ιστού των Πραγμάτων (Web of Things Proxy) 55

3.4.2 Διαχείριση Δεδομένων 56

Fiware Cygnus - LD 59

3.4.3 Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Publish - Subscribe 60

Service) 60

3.4.4 Υπηρεσία Οντολογίας (Ontology Service) 63

3.4.5 Υπηρεσία Σύνθεσης Εφαρμογών (Mashup Service) 65

3.4.6 Directory Service 69

3.4.7 Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών (User Authentication - Authorization) 69

3.4.8 Υπηρεσία Λήψης Απόφασης Εξουσιοδότησης (Authorization Policy Decision Point) 73

3.4.9 Διακομιστής Μεσολάβησης Επιβολής Πολιτικής (Policy Enforcement Point Proxy) 74

3.4.10 Δρομολόγησης και Ελέγχου - Σύστημα Διεπαφής Χρηστών (Application Logic - Web Application) 76

3.5 Διάγραμμα Αρχιτεκτονικής Ενός Κόμβου 79

3.6 Κατανεμημένο Σύστημα Fi-SWoT 80

3.6.1 Αρχιτεκτονικό Διάγραμμα Κατανεμημένου Συστήματος (Abstract Diagram) 81

3.6.2 Υπηρεσίες στο Κατανεμημένο Σύστημα 82

Κεφάλαιο 4 87

4.1 Περιβάλλον Υλοποίησης 87

4.2 Πειράματα 88

4.2.1 Πειραματικό περιβάλλον 89

4.2.2 Πειραματικά αποτελέσματα 89

Κεφάλαιο 5 108

5.1 Συμπεράσματα 108

5.2 Μελλοντικές Επεκτάσεις 110

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός της εργασίας

1.1.1 Εξέλιξη του Διαδικτύου των Πραγμάτων (IoT)

Το «Διαδίκτυο των Πραγμάτων» (Internet of Things) είναι ένα δίκτυο από συσκευές που είναι συνδεδεμένες μέσω του Διαδικτύου (Internet), οι οποίες μπορούν να συλλέξουν και να ανταλλάξουν δεδομένα. Το Διαδίκτυο των Πραγμάτων δημιουργήθηκε για να παρέχει βελτιωμένη και συνεχή συλλογή πληροφορίας, εξοικονόμηση χρόνου και χρήματος για τον άνθρωπο καθώς ελαχιστοποιεί την παρέμβασή του, ενώ ταυτόχρονα προσφέρει απίστευτη ευκολία στην παρακολούθηση διαφόρων τομέων της καθημερινότητας μας όπως είναι η υγεία, ένα σπίτι κλπ.

Παρόλα αυτά όσο ο αριθμός των συσκευών στο Διαδίκτυο αυξάνεται, τόσο μεγαλώνει και η ετερογένεια των δεδομένων και συσκευών. Η ετερογένεια αυτή καθιστά αδύνατη την ομοιόμορφη κατανόηση όλων των διαφορετικών οντοτήτων. Το τελευταίο δημιουργεί την ανάγκη όλες οι συσκευές του δικτύου να μπορούν να αλληλεπιδρούν ομοιόμορφα μεταξύ τους και να εντάσσονται σε οποιαδήποτε εφαρμογή, ανεξαρτήτως κατασκευαστή και πρωτοκόλλων δικτύωσης. Αυτό επιτυγχάνεται με τον «Ιστό των Πραγμάτων» (Web of Things), το οποίο στοχεύει να εντάξει όλες τις οντότητες ενός IoT συστήματος στον Ιστό, αξιοποιώντας τα ως υποδομή και καθιστώντας τα προσβάσιμα μέσω διεπαφών.

Ενώ ο Ιστός των Πραγμάτων (Web of Things) επιλύει αρκετά προβλήματα, υπάρχει δυνατότητα βελτίωσης όσον αφορά την διαλειτουργικότητα, καθώς και τη δυνατότητα των μηχανών να κατανοούν σωστά πληροφορίες από περιγραφές αντικειμένων. Ο Σημασιολογικός Ιστός των Πραγμάτων (Semantic Web of Things) αποτελεί μία επέκταση του Διαδικτύου των Πραγμάτων (IoT) και του Ιστού των Πραγμάτων (WoT), προσφέροντας επιπλέον λειτουργικότητα μέσω της αξιοποίησης των εργαλείων του Σημασιολογικού Ιστού. Ο Σημασιολογικός Ιστός έχει σαν σκοπό να εξελίξει το Διαδίκτυο των Πραγμάτων καθιστώντας τις μηχανές ικανές να κατανοούν τις ιδιότητες και τα δεδομένα των συσκευών του Διαδικτύου των

Πραγμάτων, καθώς και να τα διαχειρίζονται με έναν τρόπο που αξιοποιεί κατάλληλα τις γνώσεις των εκάστοτε συσκευών. Η μετάβαση από το κλασικό μοντέλο αναπαράστασης της πληροφορίας ενός συστήματος Διαδικτύου Πραγμάτων σε ένα σημασιολογικό μοντέλο αναπαράστασης της πληροφορίας πραγματοποιείται μέσω των εργαλείων του Σημασιολογικού Ιστού, τα οποία δίνουν ουσιαστικό περιεχόμενο στα δεδομένα με στόχο να βοηθήσουν στη καλύτερη συλλογή και επεξεργασία τους. Με αυτόν τον τρόπο, το πρόβλημα της ασάφειας των συσκευών και των δεδομένων τους βρίσκει μια λύση.

1.1.2 Ο ορισμός του προβλήματος

Είναι γνωστό ότι υπάρχουν πολλές προσεγγίσεις για το σχεδιασμό συστημάτων Semantic Web of Things (SWoT) με στόχο την αντιμετώπιση ορισμένων προκλήσεων που βασίζονται σε απαιτήσεις για συγκεκριμένες εφαρμογές ενός τομέα και συγκεκριμένες περιπτώσεις χρήσης. Αντίστοιχες προσεγγίσεις αποτελούν οι διπλωματικές εργασίες [1], [2] οι οποίες υποστηρίζουν λειτουργικότητα Σημασιολογικού Ιστού σε περιβάλλον Υπολογιστικού Νέφους. Το σύστημα ονομάστηκε **iSWoT** και σχεδιάστηκε για να επεκτείνει τη λειτουργικότητα υπαρχόντων αρχιτεκτονικών, ενσωματώνοντας την σημασιολογία όλων των οντοτήτων που ανήκουν σε αυτές, καθιστώντας το σύστημα ικανό να “κατανοήσει” τις συσκευες και τα δεδομένα τους. Αποτελεί δηλαδή μία αρχιτεκτονική για την αυτοματοποιημένη σύνδεση συσκευών στον Ιστό των Πραγμάτων (Web Of Things), και την επεξεργασία των δεδομένων σε ένα σημασιολογικό πλαίσιο. Ενσωματώνει τεχνολογίες Διαδικτύου των Πραγμάτων, Ιστού των Πραγμάτων και εφαρμόζει τεχνολογίες Σημασιολογικού Ιστού επιτρέποντας την εύκολη παραγωγή χρήσιμων εφαρμογών, προσιτών και εμπλουτισμένων με σημασιολογική πληροφορία. Έτσι, αυτές οι εφαρμογές μπορούν να κατανοήσουν τα δεδομένα που διαχειρίζονται και να παράγουν ένα συλλογιστικά ορθό αποτέλεσμα φιλικό και χρήσιμο προς τον άνθρωπο. Πρόκειται για ένα σύστημα το οποίο υποστηρίζει τη λειτουργικότητα που προβλέπει το Web Thing Model[1] της W3C, καθώς επίσης και την επέκταση του συστήματος στο Σημασιολογικό Ιστό, δημιουργώντας υπηρεσίες που προσθέτουν σημασιολογία σε όλες τις συσκευές του συστήματος.

Ωστόσο η ανάγκη για την ασφάλεια των δεδομένων που βρίσκονται στο διαδίκτυο ολοένα και αυξάνεται. Μηχανισμοί ταυτοποίησης και εξουσιοδότησης χρηστών, καθώς και έλεγχοι πρόσβασης σε πόρους των συστημάτων Σημασιολογικού Ιστού των Πραγμάτων κρίνονται απαραίτητοι. Η προστασία των σημασιολογικών πληροφοριών και των συσκευών, καθώς και η σωστή διαχείριση τους καθίσταται αναγκαία. Επιπλέον, οι ενέργειες πάνω σε συσκευές και εφαρμογές θα πρέπει να επιτρέπονται μόνο σε εξουσιοδοτημένους χρήστες του συστήματος καθώς και να διαφοροποιούνται για κάθε κατηγορία χρηστών. Η μη εξουσιοδοτημένη χρήση συσκευών μπορεί να οδηγήσει σε μη επιθυμητά αποτελέσματα όπως δημοσίευση των ιδιοτήτων τους σε ξένους κόμβους, μη ελεγχόμενη διαχείριση (π.χ. ένας χρήστης μπορεί να έχει συνδρομή σε μια συσκευή ενώ ταυτόχρονα κάποιος άλλος να την διαγράψει), ανεξέλεγκτη χρήση από “τρίτους”, από άτομα δηλαδή που

[1] https://www.w3.org/Submission/wot-model/

δεν ανήκουν στο τοπικό κόμβο κλπ. Επομένως, οι μηχανισμοί ασφάλειας δεν πρέπει να λείπουν από κανένα σύστημα. Επίσης, πολλαπλά συστήματα Σημασιολογικού Ιστού Πραγμάτων (κόμβοι SWoT) θα πρέπει να αλληλεπιδρούν μεταξύ τους θέτωντας όρους και διασφαλίζοντας ότι η επικοινωνια τους δεν θα οδηγήσει σε δημοσιευση των πληροφοριών τους.

1.2 Προτεινόμενη Λύση

Μια προσέγγιση για την επίλυση των προαναφερθέντων ζητημάτων παρουσιάζεται στη παρούσα διπλωματική εργασία, σχεδιάζοντας το **Fi-SWoT** σύστημα, το οποίο στοχεύει στη δημιουργία κατανεμημένης αρχιτεκτονικής για το Σημασιολογικό Δίκτυο των Πραγμάτων (SWoT) στο υπολογιστικό νέφος (Cloud). Συγκεκριμένα, το σύστημα αποτελείται από εγγεγραμμένους, ισοδύναμους και ανεξάρτητους οργανισμούς (ή συστήματα) Διαδικτύου των πραγμάτων, καθένα από τα οποία έχει σχεδιαστεί ακολουθώντας πρότυπα Σημασιολογικού Ιστου (Semantic Web) και Μοντέλου του Ιστού των Πραγμάτων (Web Thing Model). Το Fi-SWoT είναι σε θέση να υποστηρίξει μεγάλο αριθμό οργανισμών (κόμβων) υπό την προϋπόθεση ότι πρώτα έχουν εγγραφεί στο σύστημα. Με αυτόν τον τρόπο εξασφαλίζεται η γνώση για το πόσοι και ποιοι οργανισμοί SWoT βρίσκονται στο σύστημα μας, επιτρέποντας σε εγγεγραμμένους χρήστες να τους αναζητήσουν και να ανακτήσουν πληροφορίες τους.

Μεμονωμενα, κάθε κόμβος του Fi-SWoT διατηρεί την πληροφορία του υπό την κατοχή του διαχειριστή του και δεν επιτρέπει να ρέει ανεξέλεγκτα στο δίκτυο. Η πρόσβαση σε πληροφορίες όπως είναι οι ιδιότητες συσκευών, οι εφαρμογές κλπ, επιτρέπεται μόνο σε έγκυρους-εξουσιοδοτημένους χρήστες που ανήκουν είτε σε τοπικό είτε σε απομακρυσμένο κόμβο. Επίσης, βασική ιδέα του Fi-SWoT, την οποία ακολουθούν όλοι οι κόμβοι, είναι να αποτρέπει την είσοδο σε πόρους (υπηρεσίες) αν το αίτημα δεν περάσει πρώτα από ένα διακομιστή μεσολάβησης (pep proxy) και δεν ελεγχθούν οι άδειες του αιτούντα. Έτσι, επιτυγχάνεται η προστασία των πόρων και ο έλεγχος ενεργειών των χρηστών στο σύστημα.

Επιπλέον, η δυνατότητα αξιοποίησης των δεδομένων κάποιας συσκευής παρέχεται με την μορφή συνδρομής στους ενδιαφερόμενους χρήστες. Η συνδρομή σε μια συσκευή δίνει την δυνατότητα σε ένα προγραμματιστή εφαρμογών να την χρησιμοποιήσει για να κατασκευάσει μια εκτελέσιμη εφαρμογή. Επομένως, η χρήση συσκευών περιορίζεται θέτοντας ως απαραίτητη προϋπόθεση την εγγραφή των χρηστών σε αυτές. Οι χρήστες μπορούν να δημιουργήσουν συνδρομές σε συσκευες που ανήκουν είτε στο τοπικό τους κόμβο είτε σε απομακρυσμένο εφόσον τους έχουν χορηγηθεί οι καταλληλες άδειες.

Τέλος, το Fi-SWoT διαθέτει γραφικές διεπαφές για τις διαθέσιμες λειτουργίες καθιστώντας το ένα ευκολόχρηστο περιβάλλον, κατάλληλο να υποστηρίξει μεγάλο αριθμό χρηστών, οντοτήτων και μετρήσεων.

1.3 Μεθοδολογία και Συνεισφορά της Εργασίας

Η παρούσα εργασία βασίζεται στις διπλωματικές εργασίες iZen ([3]) και iSWoT ([1] [2]). Η πρώτη αφορά την δημιουργία ενός κατανεμημένου συστήματος, που αποτελείται από ανεξάρτητα και ισοδύναμα συστήματα IoT (κόμβοι) σε ένα δίκτυο, τα οποία αλληλεπιδρούν μεταξύ τους. Το iZen έχει σχεδιαστεί για να προσφέρει υπηρεσίες ασφάλειας, επεκτασιμότητας και υψηλής διαθεσιμότητας σε οργανισμούς που είναι πρόθυμοι να μοιράζονται πληροφορίες με άλλους οργανισμούς για έναν κοινό στόχο (π.χ. τοπικοί μετεωρολογικοί σταθμοί ενώνονται μαζί για τη δημιουργία προβλέψεων καιρού για ολόκληρη τη χώρα). Η δεύτερη εργασία υλοποιήθηκε για να σχεδιάσει ένα σύστημα που ενσωματώνει τις υπηρεσίες του Web of Things Model και του Σημασιολογικού Ιστού, προσφέροντας την δυνατότητα δημιουργίας εφαρμογών μέσω της σύνθεσης υπάρχουσας γνώσης για το σύστημα. Έτσι, το iSWoT αποτελεί μια νέα υπηρεσιακή αρχιτεκτονική για το Διαδίκτυο των πραγματων που παρέχει την λειτουργικότητα για σημασιολογική αναπαράσταση συσκευών και υπηρεσιών, σύνθεση νέων υπηρεσιών χρησιμοποιωντας υπάρχουσα γνώση για συσκευές και υπηρεσίες που παρέχουν.

Η παρούσα αρχιτεκτονική συνδυάζει των έργο των δύο προηγούμενων εργασιών και δημιουργεί μια νέα κατανεμημένη υπηρεσιοκεντρική αρχιτεκτονική για το σημασιολογικο δικτυο των πραγματων στο Υπολογιστικό Νέφος (Cloud), συνδέοντας πολλαπλά συστήματα. Τα κύρια χαρακτηριστικά που εφαρμόζονται στο πλαίσιο της τρέχουσας αρχιτεκτονικής συνοψίζονται παρακάτω :

- Βασικό χαρακτηριστικό της πλατφόρμας που αναπτύχθηκε είναι η υπηρεσιοκεντρική αρχιτεκτονική (Service Oriented Architecture). Κάθε σύστημα Cloud λειτουργεί ως αυτόνομη υπηρεσία η οποία επικοινωνεί με άλλες μέσω διεπαφών RESTful. Η υπηρεσιοκεντρική αρχιτεκτονική (βλ. 2.6.1) διευκολύνει και απλοποιεί την επέκταση του συστήματος σαν διαδικασία, καθώς κάθε ξεχωριστή υπηρεσία μπορεί να αναβαθμιστεί ή να αντικατασταθεί εξ ολοκλήρου λαμβάνοντας υπόψη μόνο την κοινή διεπαφή REST που θα πρέπει να εκθέτει, χωρίς να επηρεαστεί η λειτουργία της υπόλοιπης αρχιτεκτονικής. Ομοίως, η προσθήκη νέας λειτουργικότητας σε υπηρεσία μπορεί να γίνει εύκολα.

- Κάθε κόμβος του Fi-SWoT σχεδιάζεται ως ένα αυτόνομο σύστημα Σημασιολογικού Ιστου των Πραγμάτων πού ταυτόχρονα μπορεί να αποτελεί μέρος ενός δικτύου ομότιμων κόμβων με παρόμοια λειτουργικοτητα. Αυτο αυξάνει τις δυνατότητες δημιουργίας εφαρμογών, μεγαλύτερης κλίμακας, από συσκευές και υπηρεσίες ενός ή περισσότερων κόμβων εκμεταλλευόμενοι με τον καλύτερο τρόπο την ιδιότητα των συσκευών να αποτελούν η κάθε μια ανεξάρτητη οντότητα προσβάσιμη από κάθε μέλος του δικτύου.

- Η σχεδίαση της αρχιτεκτονικής ακολουθεί την τακτική “secure by design” (Ασφάλεια από το σχεδιασμό), εξασφαλίζοντας την προστασία των υπηρεσιών του συστήματος στο υπολογιστικό νέφος. Με αυτόν τον τρόπο οι διεπαφές REST των υπηρεσιών είναι προσβάσιμες μόνο από το ίδιο το σύστημα και τους εξουσιοδοτημένους χρήστες.

- Το σύστημα προσφέρει μηχανισμό ταυτοποίησης, προστατεύοντας τα πιστοποιητικά των χρηστών και βελτιώνοντας την ασφάλεια του συστήματος.

- Τα δεδομένα αναπαρίστανται, ανταλλάσσονται και ρέουν στο σύστημα σε μορφή NGSI-LD (βλ. 2.6.6). Επομένως, η επεξεργασία τους είναι ανεξάρτητη από το είδος συσκευής και το πρωτόκολλο επικοινωνίας.

- Η αρχιτεκτονική βασίζεται στο μοντέλο Publish Subscribe, το οποίο αποτελεί μια υποδομή ανταλλαγής μηνυμάτων που στηρίζεται σε συνδρομές. Με αυτό το μοντέλο, αφού συμβεί ένα συμβάν, αποστέλλεται σε συνδρομητές που πρέπει να ενημερωθούν. Συγκεκριμένα, πρόκειται για την υπηρεσία Publish - Subscribe Context Broker του FIWARE (βλ. Ενότητα 2.6.4.1) που υλοποιεί την λειτουργικότητα συνδρομών σε συσκευές-αισθητήρες που παράγουν δεδομένα.

- Η παρούσα εργασία υποστηρίζει τη λειτουργικότητα που προβλέπει το Web Thing Model της W3C και συνδυάζει τις γνωστές τεχνολογίες του Υπολογιστικού Νέφους με τις νέες τεχνολογίες που προτείνονται από το Σημασιολογικό Ιστό - για παράδειγμα, τα Διασυνδεδεμένα Δεδομένα (Linked Data), το πρότυπο OWL, τις οντολογίες, τη γλώσσα ερωτήσεων SPARQL, κ.ά. Ο Σημασιολογικός Ιστός δίνει στα δεδομένα μια σαφώς προσδιορισμένη σημασία και εμπλουτίζει τις περιγραφές που αφορούν τις συσκευές /αισθητήρες.

- Το σύστημα προσφέρει την εύκολη παραγωγή χρήσιμων εφαρμογών, προσιτών και εμπλουτισμένων με σημασιολογική πληροφορία, μέσω της σύνθεσης υπάρχουσας γνώσης για το σύστημα. Η σύνθεση πληροφορίας από διαφορετικές πηγές δεδομένων υλοποιείται ως Mashup [4]. Με την βοήθεια του Σημασιολογικού Ιστού, οι εφαρμογές μπορούν να κατανοήσουν τα δεδομένα που διαχειρίζονται και να παράγουν ένα συλλογιστικά ορθό αποτέλεσμα φιλικό και χρήσιμο προς τον άνθρωπο. Το παραπάνω αποτελεί έργο προηγούμενης διπλωματικής εργασίας ([2]).

- Ένα βασικό εργαλείο που χρησιμοποιείται για την υλοποίηση της αρχιτεκτονικής Fi-SWoT είναι η βάση δεδομένων Apache Cassandra, που είναι υπεύθυνη για την διαχείριση δημόσιας πληροφορίας και ενεργοποιεί την λειτουργία αναζήτησης των κόμβων που είναι συνδεδεμένοι στο Fi-SWoT, για τους χρήστες. Επιπλέον, η Cassandra εξασφαλίζει ασφάλεια και υψηλή

διαθεσιμότητα δεδομένων στους εγγεγραμμένους οργανισμούς SWoT (κόμβους).

- Το σύστημα υποστηρίζει ερωτήσεις σε όλο το δίκτυο κόμβων, δηλαδή ερωτήματα μπορούν να γίνουν είτε σε τοπικό είτε σε απομακρυσμένο κόμβο.

- Η αρχιτεκτονική που σχεδιάστηκε χαρακτηρίζεται επεκτάσιμη. Νέοι κόμβοι μπορούν να εισάγονται ή να διαγράφονται χωρίς να επηρεάζεται η λειτουργία του υπόλοιπου συστήματος.

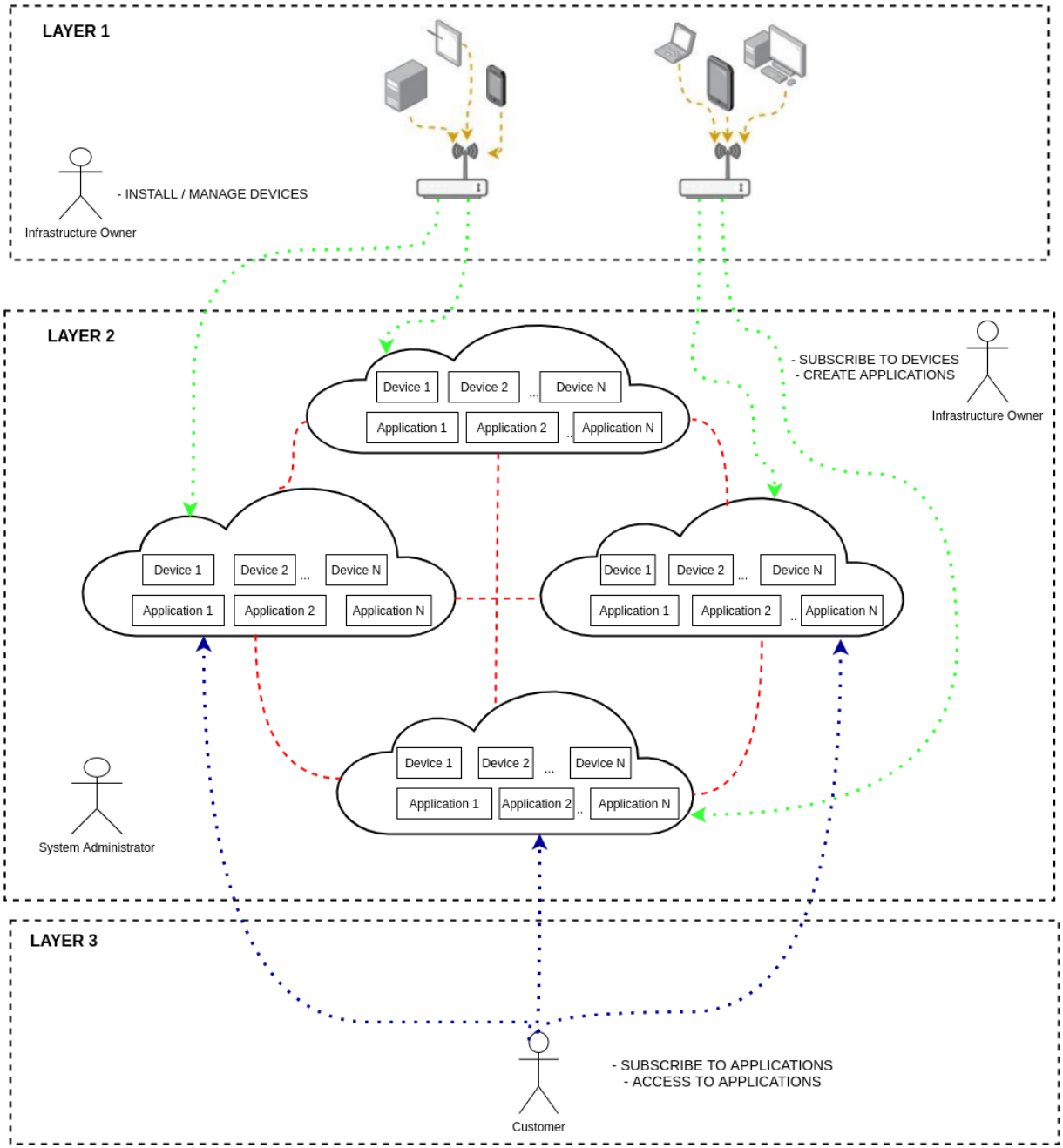
Το Fi-SWoT ακολουθεί ένα μοντέλο αρχιτεκτονικής 3 επιπέδων (3-tier architecture) όπου κάθε επίπεδο εξυπηρετεί διαφορετική λειτουργικότητα για διαφορετικούς τύπους χρηστών. Ειδικότερα, το σύστημα που προτείνουμε στηρίζεται σε επιχειρηματικό μοντέλο όπου και οι τρεις κατηγορίες χρηστών που εκμεταλλεύονται την υποδομή, μπορούν να συμβάλλουν στην επέκταση και μέγιστη αξιοποίηση του συστήματος. Το μοντέλο αρχιτεκτονικής 3 επιπέδων που υιοθετεί το σύστημα, αναπαρίσταται στην εικόνα 1.1 και στη συνέχεια γίνεται μία σύντομη ανάλυση για κάθε επίπεδο της αρχιτεκτονικής ξεχωριστά.

Το επίπεδο ένα (Layer 1) είναι το επίπεδο της υποδομής και σε αυτό το επίπεδο δραστηριοποιείται η πρώτη ομάδα χρηστών (Infrastructure Owner). Στο συγκεκριμένο επίπεδο οι ιδιοκτήτες της υποδομής έχουν το δικαίωμα να εγγράφουν νέες συσκευές στο σύστημα, τις οποίες αργότερα μπορούν να διαχειριστούν. Σκοπός των χρηστών είναι να προσφέρουν συσκευές που παρέχουν δεδομένα (μετρήσεις) και θα μπορούν να τις αξιοποιήσουν για την δημιουργία εφαρμογών.

Στο επίπεδο δύο (Layer 2) δραστηριοποιούνται δύο ομάδες χρηστών α) οι διαχειριστές του συστήματος και β) οι προγραμματιστές εφαρμογών. Σκοπός των διαχειριστών (administrators) είναι ο έλεγχος (monitoring) της πλατφόρμας σε όλα τα επίπεδα, και η διαχείριση των χρηστών (δημιουργία, ανάκτηση, διαγραφή χρηστών). Σκοπός της δεύτερης κατηγορίας χρηστών (application developers) είναι να δημιουργούν εφαρμογές χρησιμοποιώντας συσκευές της υποδομής αφού αποκτήσουν δικαιώματα χρήσης σε αυτές. Το τελευταίο επιτυγχάνεται δημιουργώντας συνδρομή στις συσκευές που θέλουν να εκμεταλλευτούν.

Στο επίπεδο τρία (Layer 3) δραστηριοποιούνται οι τελικοί χρήστες (end-users) ή αλλιώς πελάτες της υποδομής (customers). Οι τελευταίοι έχουν δικαίωμα να εγγράφονται βάσει συνδρομής και να χρησιμοποιούν την πληροφορία που παρέχουν οι εφαρμογές στις οποίες αποκτούν συνδρομή.

Βασισμένοι στην παραπανω αρχιτεκτονική, εκτιμήσαμε την αποδοση το συστήματος. Δημιουργήσαμε αιτήματα που αφορούν ενέργειες χρηστών πάνω σε συσκευές και εφαρμογές σε τοπικό, καθώς και σε απομακρυσμένο κόμβο. Με αυτό το τρόπο καταλήξαμε στο συμπέρασμα ότι το σύστημα μπορεί να ανταποκριθεί σε πραγματικό χρόνο κάτω από συνθήκες μεγάλου φόρτου εργασίας. Η ανάλυση και τα αποτελεσματα της πειραματικής διαδικασίας αποτυπώνονται λεπτομερώς στο Κεφάλαιο 4.



Εικόνα 1.1: Μοντέλο αρχιτεκτονικής 3 επιπέδων (3-tier architecture) του συστήματος.

1.4 Δομή της εργασίας

➢ Στο Κεφάλαιο 2 γίνεται μια σύντομη περιγραφή σχετικών εργασιών καθώς και τεχνολογιών που χρησιμοποιήθηκαν στην παρούσα εργασία.
➢ Στο Κεφάλαιο 3 περιγράφεται η αρχιτεκτονική του συστήματος και καταγράφονται οι υπηρεσίες που το απαρτίζουν.
➢ Στο Κεφάλαιο 4 αναλύεται το περιβάλλον υλοποίησης και εξετάζεται η απόδοση του συστήματος, η οποία μελετήθηκε με τη χρήση συνθετικών - αλλά ρεαλιστικών - δεδομένων που αφορούν αισθητήρες και άλλες συσκευές (τα δεδομένα αυτά δημιουργήθηκαν με σκοπό να μελετήσουμε την απόδοση του συστήματος σε συνθήκες μεγάλου όγκου δεδομένων και υπολογιστικού φορτίου).
➢ Στο Κεφάλαιο 5 παραθέτουμε κάποια συμπεράσματα και προτάσεις για μελλοντική διερεύνηση.

Κεφάλαιο 2

Υπόβαθρο και Υπάρχουσες Τεχνολογίες

2.1 Σχετικές εργασίες

Είναι γεγονός το ότι έχουν δημοσιευθεί ενδιαφέρουσες έρευνες που καλύπτουν σημαντικές πτυχές του σχεδιασμού και της υλοποίησης συστημάτων που αξιοποιούν τεχνολογίες του Σημασιολογικού Ιστού, παραδείγματος χάριν τα άρθρα [5] και [6]. Αντίστοιχες προσεγγίσεις για το Διαδίκτυο των Πραγμάτων και τον Σημασιολογικό Ιστό αποτελούν τα συστήματα iZen ([3]) και iSWoT ([1], [2]). Στη πρώτη περίπτωση, το iZen σχεδιάστηκε για να επιτρέπει σε συστήματα IoT (αποκαλούμενοι «κόμβοι») να συνδέονται και να επικοινωνούν μεταξύ τους με ασφάλεια, διατηρώντας παράλληλα τον πλήρη έλεγχο των δεδομένων, υπηρεσιών και συσκευών τους. Από την άλλη πλευρά, το iSWoT εστιάζει στην χρήση των τεχνολογιών του Σημασιολογικών Τεχνολογιών για την αξιοποίηση της πληροφορίας που αφορά τους αισθητήρες, τις μετρήσεις τους και τις ενέργειές τους, χρησιμοποιώντας οντολογίες και συνθέτοντας εφαρμογές που θα εξυπηρετούν τον έλεγχο της λειτουργίας ενός οικο-συστήματος 2 επιπέδων: επίπεδο πόλης και επίπεδο σπιτιού.

Ωστόσο, αναζητείται μια ολοκληρωμένη και αποδοτική κατανεμημένη αρχιτεκτονική, η οποία θα αποτελείται από ισοδύναμους και ανεξάρτητους κόμβους, όπου κάθε ένας θα βασίζεται στην ιδέα του Σημασιολογικού Ιστού και θα συνδυάζει τις κλασικές τεχνολογίες του Υπολογιστικού Νέφους με τις νέες τεχνολογίες του Σημασιολογικού Ιστού. Ο σχεδιασμός ενός κατανεμημένου συστήματος που θα περιλαμβάνει όλα τα παραπάνω, θα προσφέρει επεκτασιμότητα, ασφάλεια, θα επιτρέπει τη συλλογή και την ανάλυση δεδομένων σε πραγματικό χρόνο, και θα εξασφαλίζει την ασφαλή επικοινωνία μεταξύ των κόμβων SWoT αποτελεί ενδιαφέρουσα πρόκληση.

Στη συνέχεια, θα αναφερθούμε σε σχετικές τεχνολογίες που μπορούν να αποτελέσουν σημαντικά εργαλεία μιας κατανεμημένης αρχιτεκτονικής υπηρεσιών Υπολογιστικού Νέφους για το Σημασιολογικού Δίκτυο των Πραγμάτων και αξιοποιούνται κατάλληλα στην παρούσα εργασία.

2.2 Διαδίκτυο των πραγμάτων (IoT)

Το «Διαδίκτυο των Πραγμάτων» (Internet of Things) είναι ένα δίκτυο από συσκευές που είναι συνδεδεμένες μέσω του Διαδικτύου (Internet), οι οποίες μπορούν να συλλέγουν, να διαμοιράζονται και να χρησιμοποιούν δεδομένα προκειμένου να παρέχουν υπηρεσίες σε τομείς της κοινωνίας. Το Διαδίκτυο των Πραγμάτων προσφέρει βελτιωμένη και συνεχή συλλογή δεδομένων, εξοικονομεί χρόνο και χρήμα για τον άνθρωπο καθώς ελαχιστοποιεί την παρέμβασή του, διευκολύνει την εκτέλεση εργασιών και καθιστά δυνατή τη παρακολούθηση πεδίων της καθημερινής ζωής όπως είναι ο καιρός, η υγεία, ένα αυτοκίνητο, μια πόλη (smart cities.)

Η τεχνολογία του Διαδικτύου των πραγμάτων (IoT) έχει διεισδύσει στη καθημερινότητα μας και η χρήση της αποτελεί αναπόσπαστο κομμάτι της. Χαρακτηριστικά παραδείγματα της εφαρμογής του Διαδικτύου των Πραγμάτων είναι οι «φορετές» συσκευές (wearables devices), οι οποίες προσφέρουν ατελείωτες δυνατότητες στο πεδίο της απομακρυσμένης παρακολούθησης της υγείας ασθενών (smart healthcare), της παρακολούθησης οικοσυστημάτων όπως σύστημα αισθητήρων σε σπίτι το οποίο ελέγχει και την αυτόματη λειτουργία εγκατεστημένων συσκευών σύμφωνα με τα επίπεδα θερμοκρασίας ή υγρασίας (smart home - home automation), της επίβλεψης γεωργικών εκτάσεων και πολλές άλλες εφαρμογές. Τα παραπάνω δείχνουν ότι η δημιουργία εφαρμογών για το Διαδίκτυο των Πραγματων κρίνεται απαραίτητη σε τομείς της καθημερινότητας και έχει ως στόχο την εξυπηρέτηση, την διευκόλυνση,την ικανοποίηση των χρηστών.

2.3 Υπολογιστικό Νέφος (Cloud)

Το Υπολογιστικό Νέφος και το Διαδίκτυο των Πραγμάτων είναι άμεσα συνδεδεμένα, δημιουργώντας νέες δυνατότητες στο πεδίο της συλλογής και ανάλυσης δεδομένων. Βασικά πλεονεκτήματα του Υπολογιστικού Νέφους είναι η απλότητα, η επεκτασιμότητα και η προσιτή τιμή του, καθώς δεν απαιτεί προκαταβολική επένδυση, ενώ έχει και χαμηλό κόστος λειτουργίας. Τα παραπάνω καθιστούν το Υπολογιστικό Νέφος ιδανικό περιβάλλον ανάπτυξης εφαρμογών για το Διαδίκτυο των Πραγμάτων.

Το Υπολογιστικό Νέφος εξασφαλίζει την διάθεση υπολογιστικών πόρων μέσω διαδικτύου (π.χ. εξυπηρετητες, εφαρμογές κλπ). Ο τελικός χρήστης μπορεί να έχει απομακρυσμένη πρόσβαση σε υπηρεσίες, οι οποίες τον εξυπηρετούν και παρέχουν διευκολύνσεις. Επίσης, το Νέφος επιτρέπει την ταυτόχρονη πρόσβαση σε υπηρεσία από πολλούς χρήστες, δηλαδή την κατανάλωση ίδιου πόρου σε ευρεία κλίμακα.

Η τεχνολογία του Cloud προβλέπει και διαθέτει πόρους κατά παραγγελία (on-demand), συνεπώς οι χρήστες μπορούν αξιοποιήσουν τις υπηρεσίες του Νέφους βασιζόμενοι στις πραγματικές τους ανάγκες και να χρεωθούν σύμφωνα με την χρήση τους. Επίσης, βασική ιδιότητα αποτελεί η επεκτασιμότητα μιας υποδομής Νέφους, η οποία εξασφαλίζει την εξυπηρέτηση των απαιτήσεων και διατηρεί την απόδοση παρά

του συνεχώς αυξανόμενου πλήθους χρηστών και εφαρμογών. Επιπλέον χαρακτηριστικά του Υπολογιστικού Νέφους είναι το μειωμένο κόστος, η απεριόριστη αποθήκευση δεδομένων, η δημιουργία αντιγράφων ασφαλείας, η εύκολη πρόσβαση στις πληροφορίες κλπ.

2.4 Web Thing Model

Το Web Thing Model αποτελεί ένα μοντέλο που ορίζεται από την Κοινοπραξία του Παγκόσμιου Ιστού (World Wide Web Consortium ή W3C) σε συνδυασμό με μια διεπαφή προγραμματισμού εφαρμογών διαδικτύου (Web API) για «Πράγματα» (Things), με σκοπό αυτά να ακολουθούνται από οποιονδήποτε επιθυμεί να δημιουργήσει ένα προϊόν, μια συσκευή, μια υπηρεσία, ή μια εφαρμογή για το Web of Things. Το μοντέλο που προτείνεται από την W3C είναι μια προσπάθεια για να γίνει η αλληλεπίδραση μεταξύ των Πραγμάτων στο Διαδίκτυο των Πραγμάτων προσβάσιμη από τα πρότυπα του Ιστού (Web standards). Με αυτόν τον τρόπο, θα διευκολυνθεί η υλοποίηση των εφαρμογών του Ιστού που χρησιμοποιούν ή ανακτούν δεδομένα από αντικείμενα του πραγματικού κόσμου.

Με τον όρο “Πράγμα” (Web Thing ή απλά Thing) αναφερόμαστε στην εικονική αναπαράσταση ενός φυσικού αντικειμένου. Αυτή η αναπαράσταση αποτελεί τη βάση των πόρων (resources) του Web Things Model. Σύμφωνα με τις προδιαγραφές και τις απαιτήσεις του μοντέλου της W3C, προκειμένου ένα Πράγμα να εκθέσει τα βασικά χαρακτηριστικά του, πρέπει να υποστηρίζει το μορφότυπο JSON ως προκαθορισμένο τρόπο αναπαράστασης.

Με τον όρο **Model** (Μοντέλο ενός Πράγματος) αναφερόμαστε σε μια αναπαράσταση που βασίζεται στο μορφότυπο JSON-LD και η οποία περιγράφει τόσο τις ιδιότητες (properties) όσο και τις ενέργειες (actions) που χαρακτηρίζουν ένα Πράγμα. Αυτή η αναπαράσταση περιέχει υπερσυνδέσμους (hyperlinks) που συνδυάζονται κατάλληλα για τη δημιουργία μιας σημασιολογικής περιγραφής της αντίστοιχης οντότητας. Επομένως, το μοντέλο ενός Πράγματος εκμεταλλεύεται τις τεχνολογίες του Σημασιολογικού Ιστού, με σκοπό την αποσαφήνιση των ιδιοτήτων που έχει το εκάστοτε Πράγμα (π.χ. τι μετράει, πού βρίσκεται, κλπ.).

Οι λειτουργίες - υπηρεσίες που προτείνονται από το Web Thing Model αφορούν κυρίως την ανάκτηση, την προσθήκη, την ενημέρωση - ή ακόμα και τη διαγραφή - δεδομένων σχετικών με τα Πράγματα (για παράδειγμα, την ανάκτηση ή την ενημέρωση μιας περιγραφής JSON ή της περιγραφής JSON-LD ενός Πράγματος).

2.5 Semantic Web Of Things

Το Semantic Web Of Things (SWoT) προτείνει το σχεδιασμό διαλειτουργικών IoT υπηρεσιών στον Ιστό, χρησιμοποιώντας τεχνολογίες Σημασιολογικού Ιστού. Το SWoT είναι μια σημασιολογική επέκταση του Web of Things (WoT) που επιτρέπει σε εφαρμογές να μοιράζονται περιεχόμενα, υπηρεσίες πέραν των ορίων τους και να αναπτύσσουν νέες εφαρμογές ως σύνθεση υπαρχόντων. Μέσω των τεχνολογιών του Σημασιολογικού Ιστού των Πραγμάτων οι υπάρχουσες εφαρμογές και οντότητες του Διαδικτύου των Πραγμάτων αποκτούν σημασία,ενώ αρχίζουν να γίνονται σε ένα βαθμό κατανοητές και σε επίπεδο μηχανής. Αυτό επιτυγχάνεται χάρη σε μεθόδους και τεχνολογίες όπως αυτή των Διασυνδεδεμένων Δεδομένων **(Linked - Data)**[2] .

2.6 Σχετικές Τεχνολογίες

2.6.1 Υπηρεσιοκεντρική Αρχιτεκτονική (SOA)

Η Υπηρεσιοκεντρική Αρχιτεκτονική[3] (Service Oriented Architecture ή SOA) είναι ένα μοντέλο αρχιτεκτονικής βασισμένο στην υπηρεσία, όπου υπηρεσία είναι μια σαφώς καθορισμένη και αυτοτελής λειτουργικότητα. Η βασική ιδέα της αρχιτεκτονικής στηρίζεται στην ανεξαρτησία μεταξύ των υπηρεσιών που επικοινωνούν για την επίτευξη μιας δραστηριότητας. Μια υπηρεσία δεν χρειάζεται να γνωρίζει τις τεχνικές λεπτομέρειες μιας άλλης υπηρεσίας με την οποία αλληλοεπιδρά, παρά μόνο το τελικό σημείο διεπαφής της μέσω του οποίου πραγματοποιείται η μετάδοση των απαραίτητων δεδομένων.

Στην Υπηρεσιοκεντρική Αρχιτεκτονική, μια εφαρμογή δημιουργείται συνθέτοντας μικρές, αυτοτελείς, λειτουργικές μονάδες (υπηρεσίες). Το τελευταίο δίνει το πλεονέκτημα της επαναχρησιμοποίησης των μονάδων σε πολλαπλές εφαρμογές ανεξάρτητα από τις αλληλεπιδράσεις τους με άλλες μονάδες. Συμβάλει στην εύκολη συντήρηση εφαρμογών εφοσον μια υπηρεσία είναι μια ανεξάρτητη οντότητα, μπορεί εύκολα να ενημερωθεί ή να συντηρηθεί χωρίς να χρειάζεται να ληφθούν υπόψη άλλες υπηρεσίες της εφαρμογής. Επιπλέον, η Υπηρεσιοκεντρική Αρχιτεκτονική προσφέρει βελτιωμένη δυνατότητα επεκτασιμότητας,διαθεσιμότητα και ευελιξία για συνεργασία, καθώς επιτρέπει την επικοινωνία διαφορετικών μονάδων ανεξαρτήτως από την πλατφόρμας και την τεχνολογίας.

2.6.2 RESTful Υπηρεσίες Ιστού

Το αρχιτεκτονικό πρότυπο Representational State Transfer[4] (REST), αξιοποιώντας τις πολύ σημαντικές ιδιότητές του (απόδοση, επεκτασιμότητα και τροποποιησιμότητα), προσφέρει στις Υπηρεσίες Ιστού τη δυνατότητα να λειτουργούν βέλτιστα στον παγκόσμιο ιστό. Με βάση το πρότυπο REST, τα δεδομένα

[2] https://en.wikipedia.org/wiki/Linked_data
[3] https://en.wikipedia.org/wiki/Service-oriented_architecture
[4] ttps://en.wikipedia.org/wiki/Representational_state_transfer

και η λειτουργικότητα θεωρούνται πόροι (resources) και η είσοδος σε αυτούς πραγματοποιείται από μια αυστηρά καθορισμένη κοινή διεπαφή που στηρίζεται στη χρήση των πρότυπων μεθόδων HTTP. Συγκεκριμένα, κάθε πόρος είναι προσβάσιμος μέσω ενός Uniform Resource Identifier (URI), δηλαδή από έναν υπερσύνδεσμο του διαδικτύου. Επίσης, κάθε πόρος απαρτίζεται από μια ταυτότητα και έναν τύπο δεδομένων, ενώ υποστηρίζει και ένα σύνολο ενεργειών. Η ταυτότητα του πόρου είναι το προαναφερθέν URI και οι ενέργειες υλοποιούνται από τις τυπικές μεθόδους HTTP: την ανάκτηση / ανάγνωση (GET), την δημιουργία (POST), την ενημέρωση (PUT), και την διαγραφή (DELETE). Με τη λειτουργία GET επιτυγχάνεται η ανάκτηση της τρέχουσας κατάστασης ενός πόρου (με χρήση κάποιου είδους αναπαράστασης), με τη λειτουργία POST δημιουργείται ένας νέος πόρος, ενώ με τη λειτουργία PUT ενημερώνεται η τρέχουσα κατάσταση του πόρου. Τέλος, με τη λειτουργία DELETE διαγράφεται ένας συγκεκριμένος πόρος. Για την ένδειξη επιτυχίας ή αποτυχίας της λειτουργίας χρησιμοποιείται ο κωδικός κατάστασης HTTP, όπου κάθε κωδικός έχει την δική του ερμηνεία.

2.6.3 Ταυτοποίηση και Εξουσιοδότηση Χρηστών

Με τον όρο ταυτοποίηση αναφερόμαστε στην επιβεβαίωση ενός έγκυρου χρήστη και την επιτυχή του σύνδεση στο σύστημα. Από την άλλη πλευρά, η εξουσιοδότηση σχετίζεται με τον έλεγχο αδειών που έχει ένας χρήστης και την λήψη αποφάσεων έγκρισης ή απόρριψης αιτημάτων που πραγματοποιούνται από τον ίδιο σε πόρους του συστήματος. Στη συνέχεια παρατίθονται τα εργαλεία που χρησιμοποιούνται για την σύνθεση της υπηρεσίας ταυτοποίησης και εξουσιοδότησης χρηστών στην παρούσα αρχιτεκτονικη.

2.6.3.1 Keycloak

Το Keycloak[5] είναι μια υπηρεσία ταυτοποίησης και εξουσιοδότησης χρηστών. Είναι υπεύθυνο για την εγγραφή των χρηστών, την πιστοποίηση τους, την αποθήκευση και διαχείριση των στοιχείων τους. Επιπλέον, υποστηρίζει λειτουργίες που σχετίζονται με την ασφάλεια του συστήματος, για παράδειγμα την προστασία εφαρμογών και υπηρεσιών, την ανάθεση αδειών σε χρήστες για πρόσβαση σε πόρους, καθώς και την αντιστοίχιση χρηστών με υπάρχοντες ρόλους υιοθετώντας με αυτό τον τρόπο τα δικαιώματα του εκάστοτε ρόλου (Role Mapping). Το Keycloak βασίζεται σε τυπικά πρωτόκολλα επικοινωνίας και παρέχει υποστήριξη για OpenID Connect, OAuth2.0. Επίσης, μπορεί να πιστοποιήσει τους εγγεγραμμένους χρήστες χρησιμοποιώντας μηχανισμό OpenID Connect, ο οποίος περιγράφεται στη συνεχεια του κεφαλαίου.

[5] https://www.keycloak.org/documentation

2.6.3.2 OAuth2.0 και OpenID Connect Authentication

Μέσω του πρωτοκόλλου OAuth2.0[6], παρέχεται στους πελάτες μια “ασφαλής - εξουσιοδοτημένη πρόσβαση” σε πόρους διακομιστών, καθώς ορίζεται μια διαδικασία για τους κατόχους πόρων που επιτρέπουν την πρόσβαση τρίτων (δηλαδή πελατών) στους πόρους του διακομιστή της υπηρεσίας τους, χωρίς όμως να χρειάζεται να μοιραστούν τα διαπιστευτήριά τους. Ο μηχανισμός OAuth2.0 έχει σχεδιαστεί έτσι ώστε να λειτουργεί με το πρωτόκολλο HTTP (Hypertext Transfer Protocol) και να συμβάλει στην έκδοση ειδικών αναγνωριστικών πρόσβασης, που ονομάζονται Oauth2 tokens, σε τρίτους πελάτες. Η παραπάνω έκδοση πραγματοποιείται μέσω μιας υπηρεσίας εξουσιοδότησης (π.χ. Keycloak IdM), αφού πρώτα δοθεί η έγκριση του ιδιοκτήτη των πόρων. Επομένως, η πρόσβαση στους προστατευόμενους πόρους (που φιλοξενούνται από το διακομιστή πόρων) πραγματοποιείται με τη χρήση του διακριτικού OAuth2 token.

Το OpenID Connect[7] αποτελεί ένα απλό επίπεδο ταυτότητας πάνω από το πρωτόκολλο OAuth 2.0 βελτιώνοντας τον έλεγχο ταυτότητας ενός χρήστη. Ενώ το OAuth 2.0 αφορά την πρόσβαση σε πόρους και την κοινή χρήση, το OIDC αφορά τον έλεγχο ταυτότητας χρήστη. Επιτρέπει σε τρίτους πελάτες να επαληθεύουν την ταυτότητα του τελικού χρήστη και να λαμβάνουν βασικές πληροφορίες για το προφίλ του. Το OpenID Connect προσθέτει δύο αξιοσημείωτες δομές ταυτότητας στο μοντέλο έκδοσης διακριτικών του OAuth :

- Ένα διακριτικό ταυτότητας (OAuth2 Token), το οποίο περιέχει τα διαπιστευτήρια ασφαλείας για μια περίοδο σύνδεσης και προσδιορίζει τον χρήστη μαζί με τα δικαιώματα του.
- Ένα τυποποιημένο χαρακτηριστικό ταυτότητας API (Id Token), με το οποίο ένας πελάτης μπορεί να ανακτήσει τα επιθυμητά χαρακτηριστικά ταυτότητας για έναν δεδομένο χρήστη.

2.6.3.3 Authorization PDP – AuthZForce

Το AuthZForce, αποτελεί ένα μηχανισμό απόφασης, για την έγκριση ή απόρριψη αιτημάτων πρόσβασης από χρήστες ή υπηρεσίες, σε πόρους του συστήματος. Το AuthZForce εξακριβώνει με βάση τις εφαρμοστέες πολιτικές πρόσβασης του συστήματος, εάν κάποιος χρήστης ή υπηρεσία έχει το δικαίωμα πρόσβασης σε ένα συγκεκριμένο πόρο του συστήματος. Η υπηρεσία παρέχει ένα REST API μέσω του οποίου δέχεται αιτήματα πρόσβασης -χρηστών ή υπηρεσιών- και απαντάει στην έξοδο της, θετικά εάν το αίτημα πρόσβασης είναι εξουσιοδοτημένο ή αρνητικά διαφορετικά.

[6] https://oauth.net/2/
[7] https://openid.net/specs/openid-connect-core-1_0.html#toc

2.6.4 Υπηρεσίες στο Fiware

Το FIWARE προσφέρει υπηρεσίες γενικού σκοπού (GEs), οι οποίες περιέχουν απλές διεπαφές προγραμματισμού εφαρμογών (REST APIs). Χάρη στις υπηρεσίες αυτές, το FIWARE επιτρέπει την ανάπτυξη και τη διανομή εφαρμογών υπηρεσιοκεντρικής αρχιτεκτονικής στο Υπολογιστικό Νέφος. Ο κατάλογος υπηρεσιών του FIWARE περιλαμβάνει μια πλούσια συλλογή από γενικούς ενεργοποιητές (GE) που προσφέρουν στους προγραμματιστές τη δυνατότητα εύκολης και γρήγορης υλοποίησης σύνθετων λειτουργιών, με αποτέλεσμα να διευκολύνεται η διαδικασία ανάπτυξης εφαρμογών. Οι υπηρεσίες του FIWARE είναι όλες ανοικτού κώδικα και διατίθενται στο κοινό δωρεάν. Οι υπηρεσίες του καταλόγου του οικοσυστήματος FIWARE που χρησιμοποιήθηκαν στην παρούσα εργασία είναι οι παρακάτω:

2.6.4.1 Publish/Subscribe Context Broker - Orion-LD Context Broker

Ο Orion-LD Context Broker[8] αποτελεί επέκταση του Orion Context Broker υποστηρίζοντας το μοντέλο αναπαράστασης - ανταλλαγής δεδομένων NGSI-LD (βλ. 2.6.6), το οποίο και τον καθιστά συμβατό με την τεχνολογία των Διασυνδεδεμένων Δεδομένων (Linked Data). Συνιστά την υλοποίηση μιας Υπηρεσίας Διαχείρισης Συμβάντων και Συνδρομών και παρέχει για τη διαχείρισή τους RESTful διεπαφές NGSI-LD. Χάρη σε αυτές τις διεπαφές, οι χρήστες έχουν τη δυνατότητα να εκτελέσουν διάφορες λειτουργίες, όπως:

- Εγγραφή οντοτήτων για παρακολούθηση των συμβάντων αυτών.
- Ενημέρωση των πληροφοριών των καταχωρημένων οντοτήτων.
- Ειδοποίηση όταν πραγματοποιούνται αλλαγές στις πληροφορίες των οντοτήτων.
- Ανάκτηση πληροφοριών των εγγεγραμμένων οντοτήτων.
- Διαγραφή οντοτήτων.

2.6.4.2 Authorization PDP – AuthZForce

Όπως αναφέρθηκε και στην παράγραφο 2.6.3.3, το AuthZForce είναι υπεύθυνο για την λήψη αποφάσεων (Έγκριση ή Απόρριψη) εξετάζοντας πρώτα τις πολιτικές πρόσβασης που έχουν καθοριστεί για τον αιτούντα.

Το σύνολο αδειών πρόσβασης συντάσσεται σε γλώσσα XML και δημιουργεί ένα αρχείο το οποίο ονομάζεται **XACML**[9]. Το τελευταίο σχεδιάστηκε για να εκφράζει πολιτικές ασφαλείας και δικαιώματα πρόσβασης, χρήστη ή υπηρεσίας, σε πληροφορίες και πόρους της υποδομής.

[8] https://github.com/FIWARE/context.Orion-LD
[9] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

2.6.4.3 PEP Proxy – Wilma

Το PEP Proxy-Wilma έχει τον ρόλο του μεσολαβητή (REST) αιτημάτων μεταξύ υπηρεσιών του συστήματος ή μεταξύ χρηστών και υπηρεσιών του συστήματος. Αναλυτικότερα, το PEP-Proxy δέχεται από μία υπηρεσία ή χρήστη, ένα αίτημα πρόσβασης σε πόρο του συστήματος. Το PEP-Proxy θα «ρωτήσει» την υπηρεσία AuthZForce (μέσω της REST διεπαφής του AuthZForce) εάν το αίτημα πρόσβασης στον συγκεκριμένο πόρο εγκρίνεται ή απορρίπτεται. Εφόσον το αίτημα εγκριθεί, το PEP-Proxy θα το προωθήσει στην υπηρεσία -του συστήματος- η οποία διαθέτει το συγκεκριμένο πόρο και θα επιστρέψει την απάντηση της στον αιτούντα. Διαφορετικά το αίτημα αγνοείται. Με αυτόν τον τρόπο, μόνο οι εγγεγραμμένοι-εξουσιοδοτημένοι χρήστες και οι εξουσιοδοτημένες υπηρεσίες θα μπορούν να έχουν πρόσβαση σε πόρους του συστήματος.

2.6.4.4 Cygnus-LD

Το Cygnus-LD[10] συμβάλλει στη σύνδεση μεταξύ Orion-LD Context Broker (που αποτελεί πηγή δεδομένων NGSI-LD) και PostgreSQL (βλ. 2.6.7.2), η οποία αποτελεί την προκαθορισμένη βάση δεδομένων του. Συγκεκριμένα, το Cygnus-LD είναι απολύτως συμβατό με τις ροές δεδομένων NGSI-LD και κατάλληλο να αναγνωρίζει την πληροφορία σε μορφή NGSI-LD, να τη διαχειρίζεται και να την αποθηκεύει ορθά στη PostgreSQL. Επιπλέον, διασφαλίζει ότι μια αποτυχημένη μεταφορά πληροφοριών στη βάση δεδομένων, μπορεί να επαναληφθεί χωρίς να χαθούν τα δεδομένα.

2.6.4.5 Backend Device Manager - IDAS GE

Η υπηρεσία IDAS είναι μια υλοποίηση της υπηρεσίας Διαχείρισης Συσκευών Backend του FIWARE και πραγματοποιεί τη διασύνδεση των αισθητήρων - συσκευών με το Υπολογιστικό Νέφος. Αυτή η υπηρεσία, χρησιμοποιώντας εξειδικευμένους IoT Πράκτορες (Agents), μπορεί να λαμβάνει δεδομένα και να μεταφράζει ειδικά πρωτόκολλα IoT (HTTP, MQTT, Coap, LoRaWAN0.1) στο πρωτόκολλο NGSI-LD, που είναι ένα από τα πρότυπα αναπαράστασης - ανταλλαγής δεδομένων του FIWARE. Κάθε IoT Πράκτορας είναι ένα “συστατικό” της εν λόγω υπηρεσίας και εξειδικεύεται σε συγκεκριμένο πρωτόκολλο IoT, το οποίο μεταφράζει σε NGSI-LD. Παράλληλα, η υπηρεσία IDAS διαθέτει το δικό της μηχανισμό προστασίας, καθώς λαμβάνει δεδομένα μόνο από τις εγγεγραμμένες φυσικές συσκευές οι οποίες έχουν προστεθεί από κάποιο χρήστη στον πίνακα συσκευών της υπηρεσίας. Έτσι, απορρίπτονται αιτήματα από κάθε μη εγγεγραμμένη συσκευή.

[10] https://fiware-cygnus.readthedocs.io/en/stable/cygnus-ngsi-ld/quick_start_guide/index.html

2.6.5 Υπηρεσία Mashup - Node RED

Πρόκειται για μία υπηρεσία Mashup η οποία παρέχει την δυνατότητα δημιουργίας εκτελέσιμων εφαρμογών ως σύνθεση κόμβων που ανταλλάσσουν δεδομένα σε προκαθορισμένες συνδέσεις. Ένας κόμβος για το Node-RED[11] είναι μια ανεξάρτητη και αυτοτελής μονάδα με σαφώς καθορισμένη λειτουργικότητα. Διαθέτει ένα runtime περιβάλλον[12] σε Node.js μέσω του οποίου εκτελούνται οι εφαρμογές. Ουσιαστικά, κάθε εφαρμογή της υπηρεσίας αποτελείται από μια ροή κόμβων (flow), εκ των οποίων κάθε ένας έχει την δική του αρμοδιότητα και όλοι μαζί συντελούν στη δημιουργία της εφαρμογής. Η εφαρμογή αποθηκεύεται στην βάση αποθήκευσης εφαρμογών της υπηρεσίας Node-RED, όπου μετά είναι έτοιμη προς εκτέλεση. Η υπηρεσία προσφέρει μια RESTful διεπαφή με μεθόδους για την δημιουργία/επεξεργασία και διαγραφή εκτελέσιμων εφαρμογών.

2.6.6 Μοντέλο πληροφορίας NGSI-LD

Το μοντέλο πληροφοριών FIWARE NGSI-2 εξελίχθηκε σε NGSI-LD[13] με σκοπό να υποστηρίζει καλύτερα Διασυνδεδεμένα Δεδομένα (σχέσεις οντοτήτων), καθώς και να χρησιμοποιείται σε συνδυασμό με σημασιολογικά εργαλεία (π.χ. αποθήκευση της πληροφορίας σε τριπλέτες, συμβατό με την γλώσσα ερωτήσεων SPARQL κλπ.). Το νέο αυτό μοντέλο αντιπροσωπεύει την πληροφορία χρησιμοποιώντας ορθή JSON-LD αναπαράσταση, μια μορφή σειριοποίησης με βάση το JSON για τα συνδεδεμένα δεδομένα. Ωστόσο, ενώ υπάρχει συμβατότητα μεταξύ NGSI-LD και JSON-LD δεν μπορούμε να ισχυριστούμε ότι οι δύο μορφές αναπαράστασης επιτυγχάνουν απόλυτη ταύτιση. Ακολουθεί μια περιγραφή των βασικών δομών της NGSI-LD πληροφορίας.

Entity: Οι οντότητες αποτελούν το κέντρο βάρους στο μοντέλο πληροφοριών FIWARE NGSI-LD. Μια οντότητα αντιπροσωπεύει ένα πράγμα, δηλαδή οποιοδήποτε φυσικό ή λογικό αντικείμενο (π.χ. έναν αισθητήρα, ένα άτομο, ένα δωμάτιο κ.λπ.). Κάθε οντότητα έχει ένα αναγνωριστικό οντότητας (id) και ένα τύπο (type) πχ “Sensor”. Μία οντότητα μπορεί να έχει από ένα έως n χαρακτηριστικά (Attributes).

Attributes: Τα χαρακτηριστικά αποτελούν ιδιότητες των οντοτήτων. Κάθε χαρακτηριστικό θα πρέπει να αποτελείται από τον τύπο του (Property ή Relationship) και από την τιμή του. Ο τύπος του καθορίζει σε τι τύπο κόμβου JSON-LD ανήκει. Συγκεκριμένα, αν πρόκειται για χαρακτηριστικό τύπου Property αναφερόμαστε σε DataType Property και η περιγραφή του ενσωματώνει το attribute “value” που υποδηλώνει τη τιμή που έχει. Αντιθέτως, όταν έχουμε χαρακτηριστικό τύπου Relationship αναφερόμαστε σε Object Property και η περιγραφή του ενσωματώνει το attribute “object”, το οποίο δείχνει (με URL) στην οντότητα με την οποία σχετίζεται η εκάστοτε οντότητα που το περιέχει.

[11] https://nodered.org
[12] https://techterms.com/definition/rte
[13] https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_howto/index.html

Context: Πρόκειται για ένα πεδίο το οποίο παρέχει τους υπερσυνδέσμους που είναι απαραίτητοι σε ένα JSON-LD μοντέλο και δείχνουν στην οντολογία η οποία περιγράφει την εκάστοτε οντότητα, καθώς και τις οντότητες με τις οποίες σχετίζεται.

Συνεπώς, μια οντότητα αποτελείται συνήθως από όλα τα παραπάνω στοιχεία και ακολουθεί τους κανόνες σύνταξης του προτύπου NGSI-LD. Επίσης, στο μοντέλο πληροφοριών NGSI-LD δεν υπάρχουν μεταδεδομένα χαρακτηριστικού (metadata), αλλά μόνο Ιδιότητες Ιδιοτήτων (DataType Property) ή Ιδιότητες Σχέσεων (Object Property). Ένα απλό παράδειγμα οντότητας φαίνεται παρακάτω:

```
{
  "id": "urn:ngsi-ld:Observation:ChaniaTemperatureObservation",
  "type" : "Observation",
  "context": {
      "type": "Property",
      "value": "http://example.org/data/ChaniaTemperatureObservation"
  },
  "madeObservation": {
    "type": "Relationship",
    "value": {
      "type": "Relationship",
      "object": "http://example.org/data/ChaniaTemperatureObservation",
      "context": "http://www.w3.org/ns/sosa/madeObservation"
    }
  },
  "madeBySensor": {
    "type": "Property",
    "value": {
      "type": "Relationship",
      "object": "http://example.org/data/ChaniaTemperatureSensor",
      "context": "http://www.w3.org/ns/sosa/madeBySensor"
    }
  },
  "hasSimpleResult": {
    "type": "Property",
    "value": 25
  },
  "resultTime": {
    "type": "Property",
    "value": "2020-28-09t13:55:00Z"
  },
  "@context": [
    "https://schema.lab.fiware.org/ld/context",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ]
}
```

Εικόνα 2.1 : Παράδειγμα μετρήσιμης οντότητας σε NGSI-LD αναπαράσταση

Στην εικόνα 2.1 φαίνεται η NGSI-LD αναπαράσταση μιας μετρήσιμης οντότητας (Observation). Η παραπάνω σχεδιαστική λογική ακολουθήθηκε για όλες τις οντότητες του συστήματος.

2.6.7 Βάσεις Δεδομένων

2.6.7.1 Βάση Δεδομένων MongoDB

Το MongoDB[14] είναι ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) ανοιχτού κώδικα. Χρησιμοποιεί ένα μοντέλο βάσης δεδομένων που προορίζεται για έγγραφα και υποστηρίζει διάφορες μορφές δεδομένων όπως το μορφότυπο JSON. Συνιστά μια από τις πολυάριθμες μη σχεσιακές τεχνολογίες βάσεων δεδομένων που δημιουργήθηκαν με το banner “NoSQL” προκειμένου να χρησιμοποιηθούν σε μεγάλες εφαρμογές δεδομένων και άλλες εργασίες επεξεργασίας δεδομένων που δεν ταιριάζουν καλά σε ένα άκαμπτο σχεσιακό μοντέλο. Σε αντίθεση με τις σχεσιακές βάσεις δεδομένων που χρησιμοποιούν πίνακες και σειρές, η αρχιτεκτονική του MongoDB εμπεριέχει συλλογές και έγγραφα.

2.6.7.2 Βάση Δεδομένων PostgreSQL

Η **PostgreSQL** είναι μια σχεσιακή βάση δεδομένων ανοικτού κώδικα με πολλές δυνατότητες. Παρά την σπάνια χρήση σχεσιακών βάσεων δεδομένων σε μεγάλα συστήματα, φαίνεται ότι PostgreSQL δεν έχει το χαρακτήρα της κλασικής σχεσιακής βάσης που είχε παλαιότερα, καθώς έχει αρχίσει να συγκρίνεται με **NoSQL βάσεις** και σε πολλές περιπτώσεις να "κερδίζει τη μάχη" στη σύγκριση με αυτές. Συγκεκριμένα πρόκειται για μια SQL βάση δεδομένων, που έχει ορισμένες στρατηγικές για το χειρισμό της ευρετηρίασης (indexing), την αύξηση ταυτοχρονισμού (concurrency) και την εφαρμογή βελτιστοποιήσεων και βελτιώσεων απόδοσης, όπως προηγμένη ευρετηρίαση, κατανομή πίνακα και άλλους μηχανισμούς.

‘Ερευνες ([7]) επιβεβαιώνουν ότι η PostgreSQL είναι ταχύτερη από την MongoDB σε περίπτωση που τα δεδομένα διαχειρίζονται ως πίνακες δηλαδή σε μορφή key-value pairs. Επιπλέον, αναφέρουν ότι το μέγεθος των δεδομένων που αποθηκεύονται στη βάση είναι πολυ μικροτερο στη PostgreSQL έχοντας κέρδος στους διαθέσιμους πόρους από άποψη χωρητικότητας. Σε αυτό το σημείο αξίζει να τονίσουμε ότι στη παρούσα εργασία οι πληροφορίες που μας ενδιαφέρουν, δηλαδή οι ιστορικές τιμές των οντοτήτων και η ημερομηνία λήψης της εκάστοτε μέτρησης, κατανέμονται στις στήλες των πινάκων της PostgreSQL, οι οποίες και δημιουργούνται αυτόματα χωρίς ανθρώπινη παρέμβαση. Επίσης, η επιλογή της PostgreSQL έγινε κατά σύμβαση εφόσον αποτελεί την προκαθορισμένη βάση δεδομένων του Cygnus-LD (βλ. 2.6.4.4).

[14] https://www.mongodb.com/

Τέλος, συμπαιρένουμε ότι πρόκειται για μια βάση δεδομένων η οποία μπορεί να συγκριθεί με μη-σχεσιακές βάσεις ([8]) και να ανταποκριθεί σε εφαρμογές με μεγάλα δεδομένα. Παρόλα αυτά δεν μπορούμε να πούμε ότι κρίνεται κατάλληλη για όλα τα συστήματα. Η επιλογή της εξαρτάται από το πεδίο χρήσης.

2.6.7.3 Apache Cassandra

Η Cassandra[15] είναι μια μη-σχεσιακή (NoSQL), κατανεμημένη βάση δεδομένων που εκτελείται σε σύμπλεγμα διακομιστών, σχεδιασμένη για διαχείριση μεγάλων ποσοτήτων δεδομένων. Σε αντίθεση με άλλες βάσεις δεδομένων, στη Cassandra όλοι οι κόμβοι του συμπλέγματος είναι ισοδύναμοι και επικοινωνούν ο ένας τον άλλον, ενώ σε περίπτωση αποτυχίας ενός κόμβου του συμπλέγματος, άλλοι κόμβοι αναλαμβάνουν για να ολοκληρώσουν την εργασία. Ένα επιπλέον πλεονέκτημα αυτής της βάσης δεδομένων είναι η δυνατότητα προσθήκης (ή κατάργησης) ενός διακομιστή στο (ή από) το σύμπλεγμα ανά πάσα στιγμή χωρίς να απαιτείται χρόνος διακοπής.

2.6.8 Οντολογίες (SSN kai SOSA)

Οι οντολογίες[16] αποτελούν οργανωμένες συλλογές πληροφοριών, που περιλαμβάνουν αναπαραστάσεις, επίσημες ονομασίες και ορισμούς κατηγοριών, ιδιοτήτων και σχέσεων μεταξύ των εννοιών, των δεδομένων και των οντοτήτων που απαρτίζουν ένα ή παραπάνω εννοιολογικά πεδία. Ορίζουν δηλαδή κοινό λεξιλόγιο και διαμοιρασμένη γνώση. Οι οντολογίες αξιοποιούνται από γνωστές τεχνολογίες του Σημασιολογικού Ιστού όπως η γλώσσα ερωτήσεων SPARQL, καθώς επίσης χρησιμοποιούνται στην Τεχνητή Νοημοσύνη, στη βιοπληροφορική, και σε άλλες επιστήμες ως μια μορφή αναπαράστασης γνώσης για τον κόσμο.

Παράδειγμα αποτελούν οι οντολογίες Semantic Sensor Network[17] (SSN) και SOSA (Sensor, Observation, Sample, and Actuator). Η SSN είναι μια οντολογία για την περιγραφή αισθητήρων, των μετρήσεών τους (observations), των ενεργειών τους, των ιδιοτήτων τους και των ενεργοποιητών τους. Η SSN ενσωματώνει στον πυρήνα της την αυτοδύναμη οντολογία SOSA (Sensor, Observation, Sample, and Actuator), την οποία χρησιμοποιεί για τις βασικές κλάσεις και ιδιότητές της. Το βασικό πλεονέκτημα της οντολογίας SSN σε σχέση με την SOSA είναι ότι έχει προσθέσει εκφραστικότητα στην SOSA, λόγω των παραπάνω κλάσεων και ιδιοτήτων που περιλαμβάνει, κι έτσι μπορεί να περιγράψει ένα μεγαλύτερο πλήθος οντοτήτων και με μεγαλύτερη ακρίβεια. Γι' αυτό το λόγο, έχει χρησιμοποιηθεί στις περισσότερες υλοποιήσεις που αφορούν το Διαδίκτυο των Πραγμάτων (IoT projects).

[15] http://cassandra.apache.org/
[16] https://en.wikipedia.org/wiki/Ontology_(information_science)
[17] https://www.w3.org/TR/vocab-ssn/

2.6.9 RDF - RDFS

Resource Description Framework[18] (RDF) ονομάζεται μια οικογένεια προδιαγραφών (specifications) που αναπτύχθηκε από την Κοινοπραξία του Παγκόσμιου Ιστού και χρησιμοποιείται ως μια γενική μέθοδος για την εννοιολογική περιγραφή ή για τη μοντελοποίηση των πληροφοριών που πραγματοποιείται στους πόρους του Διαδικτύου. Ακολουθεί μια ποικιλία συντακτικών σημειογραφιών (syntax notations) και μορφών σειριοποίησης δεδομένων. Το πρότυπο RDF βασίζεται στην ιδέα της δημιουργίας δηλώσεων (statements) που αφορούν πόρους - συγκεκριμένα πόρους του Διαδικτύου - με χρήση εκφράσεων της μορφής υποκείμενο - κατηγόρημα - αντικείμενο (subject - predicate - object) , που είναι γνωστές ως τριπλέτες (triples). Το υποκείμενο υποδηλώνει τον πόρο, το κατηγόρημα υποδηλώνει χαρακτηριστικά ή πτυχές του πόρου, και εκφράζει μια σχέση μεταξύ του υποκειμένου και του αντικειμένου.

Resource Description Framework Schema[19] (RDFS) ονομάζεται ένα σύνολο κλάσεων με συγκεκριμένες ιδιότητες, οι οποίες χρησιμοποιούν το επεκτάσιμο RDF μοντέλο δεδομένων αναπαράστασης γνώσης, που παρέχει βασικά στοιχεία για την περιγραφή των οντολογιών, οι οποίες προορίζονται για τη δόμηση πόρων RDF. Οι πόροι αυτοί μπορούν να αποθηκευτούν σε μια δομή αποθήκευσης τριπλετών (triplestore ή RDF store), από όπου μπορούν να ανακτηθούν με τη γλώσσα ερωτήσεων SPARQL.

2.6.10 OWL

Web Ontology Language[20] (OWL) ονομάζεται μια οικογένεια γλωσσών αναπαράστασης γνώσης που χρησιμοποιούνται στο χώρο του Σημασιολογικού Ιστού για τη συγγραφή οντολογιών. Οι γλώσσες OWL χαρακτηρίζονται από επίσημη σημασιολογία. Χτίζονται με βάση το XML πρότυπο για αντικείμενα, που ονομάζεται RDF (Resource Description Framework) και αναπτύχθηκε από την Κοινοπραξία του Παγκόσμιου Ιστού (World Wide Web Consortium). Συγκεκριμένα, η γλώσσα OWL επιτυγχάνει την επέκταση της ικανότητας των XML, RDF και RDF Schema (RDFS) να αναπαριστούν οντολογίες του παγκόσμιου ιστού. Η OWL και το πρότυπο RDF έχουν προσελκύσει σε μεγάλο βαθμό το ενδιαφέρον του ακαδημαϊκού χώρου, του χώρου της Ιατρικής καθώς και αυτού της διαφήμισης.

[18] https://en.wikipedia.org/wiki/Class_diagram
[19] https://en.wikipedia.org/wiki/RDF_Schema
[20] https://en.wikipedia.org/wiki/Web_Ontology_Language

2.6.11 SPARQL

Η SPARQL είναι μια γλώσσα ερωτημάτων για δεδομένα RDF (Resource Description Framework), δηλαδή μια σημασιολογική γλώσσα ερωτήσεων για βάσεις δεδομένων, η οποία είναι κατάλληλη για την ανάκτηση και το χειρισμό δεδομένων που έχουν αποθηκευτεί σε μορφή RDF. Αυτό σημαίνει ότι είναι δυνατή η επεξεργασία δεδομένων τα οποία βρίσκονται αποθηκευμένα σε βάσεις δεδομένων ως σύνολα τριπλετών με τη μορφή υποκείμενο - κατηγόρημα - αντικείμενο (subject - predicate - object). Συγκεκριμένα, χρησιμοποιώντας την γλώσσα SPARQL, νέα δεδομένα μπορούν να προστεθούν σε γράφους, καθώς και να ανακτηθούν, να ενημερωθούν ή να διαγραφούν υπάρχοντα δεδομένα.

2.6.12 Virtuoso

Ο Virtuoso (Openlink) Universal Server είναι ένα ενδιάμεσο λογισμικό (middleware) που συνδυάζει τη λειτουργικότητα ενός παραδοσιακού Σχεσιακού συστήματος βάσεων δεδομένων (RDBMS) με τη λειτουργικότητα μιας Αντικειμενο-σχεσιακής βάσης δεδομένων (ORDBMS), μιας εικονικής βάσης δεδομένων, του προτύπου RDF, του προτύπου XML, ενός εξυπηρετητή διαδικτυακής εφαρμογής (web application server) και ενός εξυπηρετητή αρχείων (file server) σε ένα ενιαίο σύστημα. Αντί για να έχουμε διαφορετικούς εξυπηρετητές, απ’ τους οποίους ο καθένας θα είναι αφιερωμένος στο να πραγματοποιεί μια λειτουργικότητα από τις προαναφερθείσες, ο Virtuoso είναι ένας “universal server” (καθολικός εξυπηρετητής). Αυτό σημαίνει ότι επιτρέπει μια ενιαία πολυνηματική (multithreaded) διαδικασία εξυπηρετητή που εφαρμόζει πολλαπλά πρωτόκολλα.

Χάρη στο τερματικό της Virtuoso (Virtuoso SPARQL Endpoint), μπορούμε να εκτελέσουμε SPARQL ερωτήματα για οποιοδήποτε γράφο από αυτούς που βρίσκονται στην Virtuoso. Με τον όρο γράφο αναφερόμαστε σε σημασιολογικό γράφο, ο οποίος αποτελεί ένα δίκτυο κόμβων, που αναπαριστά σημασιολογικές σχέσεις μεταξύ εννοιών και απαρτίζεται από ένα σύνολο τριάδων RDF. Με άλλα λόγια, ο Virtuoso χρησιμοποιείται για την αποθήκευση σημασιολογικής πληροφορίας, η οποία φιλοξενείται στο γράφο σε μορφή τριπλετών.

2.6.13 Apache Jena

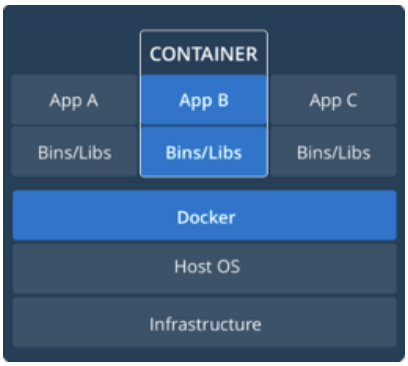
Το Apache Jena Semantic Web Framework αποτελεί ένα πλαίσιο εφαρμογών Σημασιολογικού Ιστού βασισμένο σε γλώσσα Java. Παρέχει ένα προγραμματιστικό περιβάλλον για επεξεργασία μεταδεδομένων RDF, RDFS, OWL, SPARQL και ένα μηχανισμό αιτίασης - συλλογισμού (reasoner), δηλαδή μια μηχανή εξαγωγής λογικών συμπερασμάτων βασισμένη σε κανόνες. Αξιοποιεί την έννοια του Μοντέλου, δηλαδή

μιας διεπαφής η οποία ορίζει τον τρόπο αποθήκευσης γράφων RDF καθώς και τις μεθόδους που μπορούν να κληθούν πάνω στο Μοντέλο. Οι μέθοδοι αφορούν λειτουργίες όπως η προσθήκη τριπλετών (triples) δεδομένων, η προσθήκη πόρων και ιδιοτήτων σε πόρους, η ανάκτηση τριάδων, πόρων και ιδιοτήτων, η εκτύπωση μεταδεδομένων RDF στην έξοδο (π.χ. σε αρχείο RDF/XML), κλπ.

2.6.14 Docker

Το Docker[21] είναι μια πλατφόρμα που προσφέρει Εικονοποίηση (Virtualization) σε επίπεδο Λειτουργικού Συστήματος. Συσκευάζει εφαρμογές και όλες τις εξαρτήσεις τους με τη μορφή ενός κιβωτίου (container), σε ένα απομονωμένο περιβάλλον από το κανονικό σύστημα.

Όπως δείχνει η εικόνα 2.2, κάθε εφαρμογή αναπτύσσεται και εκτελείται σε ξεχωριστό δοχείο (container), το οποίο έχει τις δικές του εξαρτήσεις και βιβλιοθήκες, διασφαλίζει την ανεξαρτησία του από άλλες εφαρμογές, ενώ παράλληλα παρέχει στους προγραμματιστές την απαραίτητη ασφάλεια για την κατασκευή εφαρμογών που δεν θα αλληλεπιδρούν μεταξύ τους.



Εικόνα 2.2 - Docker

[21]https://www.docker.com/

Κεφάλαιο 3

Αρχιτεκτονική Συστήματος

3.1 Σενάριο Χρήσης

Η αρχιτεκτονική του συστήματος αποτελείται από τρία επίπεδα όπου κάθε ένα από αυτά εξυπηρετεί διαφορετική λειτουργικότητα για διαφορετικούς τύπους χρηστών. Κύριου τύποι χρηστών είναι στο σύστημα είναι οι διαχειριστές (Administrators), οι ιδιοκτήτες της υποδομής (Infrastructure Owners) και οι τελικοί χρήστες (Customers). Η διαφοροποίηση της λειτουργικότητας του συστήματος ανά τύπο χρηστών συμβάλει στην απλοποίηση της περιγραφή του συστήματος σε υπο-ενότητες ή υπο-συστήματα και στη διευκόλυνση επέκτασης του και στα τρία επίπεδα με την προσθήκη μεγαλύτερου αριθμού αισθητήρων διάφορων τύπων, περισσότερων εφαρμογών με διαφορετικό σκοπό που μπορούν να χρησιμοποιούν κοινούς ή διαφορετικούς αισθητήρες.

Στην παρούσα εργασία κάθε ομάδα χρηστών παρέχει συγκεκριμένες λειτουργίες και οι χρήστες εκτελούν τις ενέργειες που τους επιτρέπει η κατηγορία στην οποία ανήκουν, μέσω γραφικών διεπαφών. Ακολουθεί μια εκτενής περιγραφή των λειτουργιών που αφορούν κάθε μια από τις παραπάνω κατηγορίες χρηστών.

Διαχειριστής του συστήματος (Administrator)

1. **Διαχείριση εικονικών οντοτήτων/μοντέλων πραγμάτων.** Ο διαχειριστής έχει το δικαίωμα να διαχειρίζεται όλους τους τύπους εικονικών οντοτήτων που έχουν οριστεί στο σύστημα. Συγκεκριμένα, μια εικονική οντότητα αναλαμβάνει να περιγράψει όλα τα μοντέλα αισθητήρων, μετρήσεων, εφαρμογών, σπιτιών κλπ. Επομένως ο διαχειριστής μπορεί να δημιουργεί, να ενημερώνει, να ανακτά και να διαγράφει οποιαδήποτε εικονική οντότητα, η οποία αντιπροσωπεύει ένα Μοντέλο του Ιστού των Πραγμάτων. Έχει δηλαδή την άδεια να εφαρμόζει τις τυπικές HTTP μεθόδους (Create, Read, Update, Delete) σε οντότητες. Κάθε μια από τις παραπάνω ενέργειες απασχολεί διαφορετική υπηρεσία του συστήματος και αναλύεται σε ακόλουθη ενότητα (βλ. 3.3). Οι λειτουργίες που αναφέρθηκαν είναι διαθέσιμες μέσω γραφικής διεπαφής αφού ο διαχειριστής συνδεθεί στο τοπικό του κόμβο και λάβει χώρα η υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών.
2. **Αναζήτηση συσκευών – Δημιουργία/Κατάργηση συνδρομής σε συσκευές.** Ο διαχειριστής μπορεί να αναζητήσει συσκευές και να επιλέξει εκείνες οι οποίες πληρούν τις προδιαγραφές που επιθυμεί. Οι προδιαγραφές

αυτές μπορεί να αφορούν τα χαρακτηριστικά τους, την πόλη στην οποία βρίσκεται η θέση τους, καθώς και τον τύπο συσκευής (μοντέλο). Ο διαχειριστής έχει την δυνατότητα να επιλέξει συσκευές που προκύπτουν από την αναζήτηση και να δημιουργήσει νέα συνδρομή σε αυτές ή να καταργήσει υπάρχουσα συνδρομή.

3. **Προβολή συνδρομών σε συσκευές.** Ο διαχειριστής μπορεί να ανακτήσει, μέσω γραφικής διεπαφής, μια λίστα με συσκευές που έχει εγγραφεί με την μορφή συνδρομής.
4. **Δημιουργία νέας εφαρμογής.** Ο διαχειριστής μπορεί να δημιουργήσει μια εφαρμογή αφού είναι συνδρομητής σε ένα υποσύνολο συσκευών και μπορεί να χρησιμοποιεί τα δεδομένα τους. Τα δεδομένα αφορούν ιστορικές αλλά και τρέχουσες μετρήσεις, οι οποίες βρίσκονται αποθηκευμένες στην βάση δεδομένων ιστορικού του συστήματος. Η εφαρμογή μπορεί να είναι είτε σημασιολογικής είτε ιστορικής σημασίας. Πιο συγκεκριμένα, μπορεί να περιλαμβάνει τη σημασιολογική έννοια των τρεχουσών τιμών κάποιων χαρακτηριστικών για μια πόλη καθώς και τις συμβουλές της οντολογίας για επιλεγμένους τομείς της καθημερινότητας ή γραφικές απεικονίσεις «μίξεων» (Mashup’s) από δεδομένα συσκευών (π.χ. Διάγραμμα γραμμής με την ελάχιστη τιμή της θερμοκρασίας ανά ώρα για 24 ώρες σε ένα σπίτι). Μία εφαρμογή έχει ένα “τελικό σημείο” (URI). και η πρόσβαση σε αυτή γίνεται μέσω του URI, από ένα πρόγραμμα περιήγησης ιστού.
5. **Επεξεργασία εφαρμογής.** Ο διαχειριστής μπορεί ανά πάσα στιγμή να τροποποιήσει τις εφαρμογές του, προσθέτοντας ή αφαιρώντας mashups αισθητήρων και τομείς της καθημερινότητας.
6. **Προβολή εφαρμογών.** Ο διαχειριστής έχει την δυνατότητα μέσω γραφικής διεπαφής να βλέπει ποιες εφαρμογές έχουν δημιουργηθεί συνολικά στο σύστημα καθώς και μια περιγραφή για την κάθε εφαρμογή (όνομα, γεωγραφική τοποθεσία, σκοπό, τι υπολογίζει, τομείς της καθημερινότητας).
7. **Διαγραφή εφαρμογής.** Ο διαχειριστής μπορεί να ανακτήσει την λίστα με όλες τις εφαρμογές που έχει δημιουργήσει και να καταργήσει όποια από αυτές θέλει, αφού πρώτα γίνει έλεγχος ότι κανένας άλλος χρήστης δεν είναι συνδρομητής σε αυτήν.
8. **Διαχείριση Χρηστών.** Ο διαχειριστής μπορεί να δημιουργήσει ένα νέο χρήστη στο σύστημα εισάγοντας τα απαραίτητα στοιχεία (username, password, email) και να τον κατατάξει σε μία από τις διαθέσιμες κατηγορίες χρηστών (ιδιοκτήτης υποδομής, τελικός χρήστης). Επίσης, έχει το δικαίωμα να ανακτήσει όλους τους χρήστες του συστήματος με τα στοιχεία τους και να επεξεργαστεί τις πληροφορίες που συνθέτουν το προφίλ ενός χρήστη. Τέλος, μπορεί να διαγράψει έναν χρήστη αφού ελεγχθεί ότι δεν έχει την κατοχή του συσκευές, εφαρμογές και δεν είναι συνδρομητής σε συσκευές.
9. **Εποπτεία συστήματος.** Ο διαχειριστής έχει την δυνατότητα μέσω γραφικής διεπαφής, να παρακολουθεί ανά πάσα στιγμή ποιοι χρήστες υπάρχουν στο σύστημα, πόσες συσκευές υπάρχουν στο σύστημα, πόσες εφαρμογές έχουν δημιουργηθεί στο σύστημα. Επίσης, μπορεί να δει πόσοι αυτόνομοι κόμβοι

Σημασιολογικού Ιστού έχουν εγγραφεί στο δίκτυο και τις πληροφορίες που μοιράζονται δημόσια. Επιπλέον, μπορεί να διαχειριστεί αιτήματα εγγραφής από χρήστες που ανήκουν σε απομακρυσμένους κόμβους του Fi-SWoT. Πιο συγκεκριμένα, αιτήματα συνδρομής στο κόμβο από “ξένους” χρήστες, δρομολογούνται στο διαχειριστή, ο οποίος θα πρέπει να αποφασίσει εάν θα το αποδεχτεί και θα επιτρέψει την είσοδο του εκάστοτε χρήστη στα δεδομένα του κόμβου ή όχι.

Ιδιοκτήτες Υποδομής (Infrastructure Owners)

1. **Εγγραφή στο σύστημα.** Ο χρήστης για να εγγραφεί σε ένα κόμβο του δικτύου, συμπληρώνει τα στοιχεία του, όπως όνομα, email, κωδικό πρόσβασης. Επιπλέον, επιλέγει την κατηγορία χρηστών στην οποία θέλει να ανήκει, σε αυτήν την περίπτωση ιδιοκτήτης υποδομής. Όταν συμπληρώσει τις απαραίτητες πληροφορίες, ο διαχειριστής αυτού του κόμβου τον εγγράφει σύστημα και του αναθέτει τον αντίστοιχο ρόλο μαζί με τα αντίστοιχα δικαιώματά.
2. **Διαχείριση συσκευών.** Ο χρήστης έχει το δικαίωμα να εκτελεί ενέργειες σε συσκευές του συστήματος όπως δημιουργία (Create), ανάκτηση (Get), ενημέρωση (Update), διαγραφή (Delete). Συγκεκριμένα, για να εισάγει μια νέα συσκευή θα πρέπει πρώτα να επιβεβαιωθεί ότι υπάρχει το αντίστοιχο μοντέλο (Web Thing Model) για την εκάστοτε συσκευή και το όνομα της να είναι μοναδικό, ενώ για την διαγραφή μιας υπάρχουσας συσκευής θα πρέπει πρώτα να γίνει έλεγχος ότι δεν χρησιμοποιείται από κάποιον χρήστη. Οι λειτουργίες ανάκτησης και ενημέρωσης συσκευής αφορούν την αναπαράσταση των ιδιοτήτων της και την αλλαγή αυτών. Κάθε μια από τις παραπάνω ενέργειες απασχολεί διαφορετική υπηρεσία του συστήματος και αναλύεται σε ακόλουθη ενότητα (βλ. 3.3). Οι λειτουργίες που αναφέρθηκαν είναι διαθέσιμες μέσω γραφικής διεπαφής αφού ο χρήστης συνδεθεί στο τοπικό του κόμβο και λάβει χώρα η υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών.
3. **Αναζήτηση συσκευών – Δημιουργία/Κατάργηση συνδρομής σε συσκευές.** Ο χρήστης μπορεί να αναζητήσει συσκευές και να επιλέξει εκείνες οι οποίες πληρούν τις προδιαγραφές που επιθυμεί. Οι προδιαγραφές αυτές μπορεί να αφορούν τα χαρακτηριστικά τους, την πόλη στην οποία βρίσκεται η θέση τους, καθώς και τον τύπο συσκευής (μοντέλο). Ο χρήστης έχει την δυνατότητα να επιλέξει συσκευές που προκύπτουν από την αναζήτηση και να δημιουργήσει νέα συνδρομή σε αυτές ή να καταργήσει υπάρχουσα συνδρομή.
4. **Προβολή συνδρομών σε συσκευές.** Ο χρήστης μπορεί να ανακτήσει, μέσω γραφικής διεπαφής, μια λίστα με συσκευές που έχει εγγραφεί με την μορφή συνδρομής.
5. **Δημιουργία νέας εφαρμογής.** Ο χρήστης μπορεί να δημιουργήσει μια εφαρμογή αφού είναι συνδρομητής σε ένα υποσύνολο συσκευών και μπορεί να χρησιμοποιεί τα δεδομένα τους. Τα δεδομένα αφορούν ιστορικές αλλά και

τρέχουσες μετρήσεις, οι οποίες βρίσκονται αποθηκευμένες στην βάση δεδομένων ιστορικού του συστήματος. Η εφαρμογή μπορεί να είναι είτε σημασιολογικής είτε ιστορικής σημασίας. Πιο συγκεκριμένα, μπορεί να περιλαμβάνει τη σημασιολογική έννοια των τρεχουσών τιμών κάποιων χαρακτηριστικών για μια πόλη καθώς και τις συμβουλές της οντολογίας για επιλεγμένους τομείς ή γραφικές απεικονίσεις «μίξεων» (Mashup's) από δεδομένα συσκευών (π.χ. Διάγραμμα γραμμής με την ελάχιστη τιμή της θερμοκρασίας ανά ώρα για 24 ώρες σε ένα σπίτι). Μία εφαρμογή έχει ένα "τελικό σημείο" (URI). και η πρόσβαση σε αυτή γίνεται μέσω του URI, από ένα πρόγραμμα περιήγησης ιστού.

6. **Επεξεργασία εφαρμογής.** Ο χρήστης μπορεί ανά πάσα στιγμή να τροποποιήσει τις εφαρμογές του, προσθέτοντας ή αφαιρώντας mashups αισθητήρων και τομείς της καθημερινότητας.
7. **Προβολή εφαρμογών.** Ο χρήστης έχει την δυνατότητα μέσω γραφικής διεπαφής να βλέπει ποιες εφαρμογές έχουν δημιουργηθεί συνολικά στο σύστημα καθώς και μια περιγραφή για την κάθε εφαρμογή (όνομα, γεωγραφική τοποθεσία, σκοπό, τι υπολογίζει, τομείς της καθημερινότητας).
8. **Διαγραφή εφαρμογής.** Ο χρήστης μπορεί να ανακτήσει την λίστα με όλες τις εφαρμογές που έχει δημιουργήσει και να καταργήσει όποια από αυτές θέλει, αφού πρώτα γίνει έλεγχος ότι κανένας άλλος χρήστης δεν είναι συνδρομητής σε αυτήν.
9. **Αναζήτηση διαθέσιμων κόμβων - Δημιουργία συνδρομής.** Μέσω γραφικής διεπαφής, ο χρήστης μπορεί να δει όλους τους διαθέσιμους κόμβους Σημασιολογικού Ιστού που είναι καταχωρημένοι στο Fi-SWoT και να υποβάλλει αίτηση συνδρομής σε όποιον επιθυμεί με στόχο να μπορεί να χρησιμοποιεί τις συσκευές του εκάστοτε απομακρυσμένου κόμβου, καθώς και να δημιουργεί εφαρμογές σε αυτόν. Πριν στείλει αίτημα, μπορεί να ανακαλύψει τις δημόσιες πληροφορίες των κόμβων και να επιλέξει αυτόν που τον ενδιαφέρει. Όταν ο χρήστης ζητήσει εγγραφή σε έναν ξένο κόμβο, το αίτημα δρομολογειται στον αντίστοιχο διαχειριστή ο οποίος αποφασίζει εάν θα αποδεχτεί τον χρήστη ή όχι. Αν ο διαχειριστής αποδεχτεί το αίτημα, θα εκχωρήσει στον αντίστοιχο χρήστη τους κατάλληλους ρόλους με τα αντίστοιχα δικαιώματά τους, προκειμένου να έχει πρόσβαση σε πόρους του συστήματος του.

Τελικοί Χρήστες (Customers)

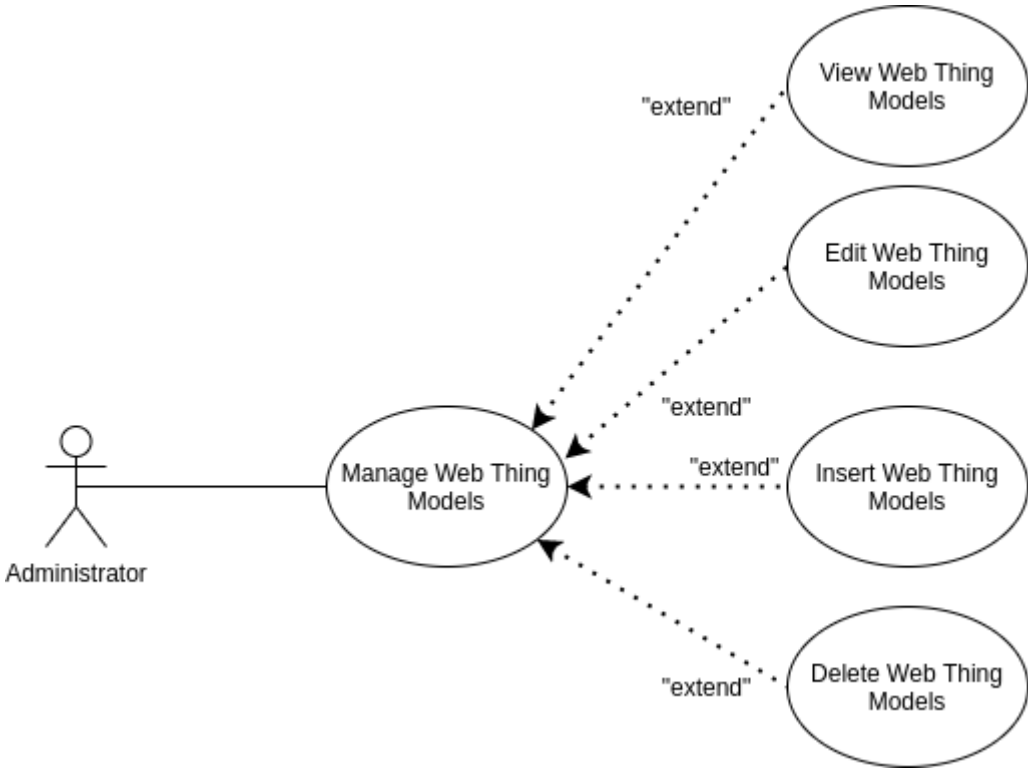
1. **Εγγραφή στο σύστημα.** Ο χρήστης για να εγγραφεί σε ένα κόμβο του δικτύου , συμπληρώνει τα στοιχεία του, όπως όνομα, email, κωδικό πρόσβασης. Επιπλέον, επιλέγει την κατηγορία χρηστών στην οποία θέλει να ανήκει, σε αυτήν την περίπτωση τελικός χρήστης/πελάτης. Όταν συμπληρώσει τις απαραίτητες πληροφορίες, ο διαχειριστής αυτού του κόμβου τον εγγράφει σύστημα και του αναθέτει τον αντίστοιχο ρόλο μαζί με τα αντίστοιχα δικαιώματά.

2. **Αναζήτηση διαθέσιμων εφαρμογών – Δημιουργία/Κατάργηση συνδρομής σε εφαρμογές.** Ο τελικός χρήστης μπορεί να αναζητήσει τις διαθέσιμες εφαρμογές του συστήματος και να δει μια σύντομη περιγραφή για την κάθε μια. Με αυτό τον τρόπο, ο τελικός χρήστης μπορεί να δημιουργήσει συνδρομή σε μία ή περισσότερες εφαρμογές που τον ενδιαφέρουν, προκειμένου να έχει το δικαίωμα πρόσβασης σε αυτές. (Διαφορετικά δεν μπορεί να έχει πρόσβαση σε κάποια εφαρμογή). Αντίστοιχα, με τον ίδιο τρόπο μπορεί να καταργήσει μία υπάρχουσα συνδρομή σε κάποια εφαρμογή.
3. **Πρόσβαση σε εφαρμογές.** Εφόσον ο τελικός χρήστης διατηρεί συνδρομή σε μία εφαρμογή, μπορεί να έχει πρόσβαση σε αυτήν μεταβαίνοντας στο τελικό σημείο επαφής (URI) της εφαρμογής από το πρόγραμμα περιήγησης που χρησιμοποιεί.
4. **Προβολή συνδρομών σε εφαρμογές.** Ο τελικός χρήστης μπορεί να ανακαλύψει μέσω γραφικής διεπαφής σε ποιες εφαρμογές διατηρεί συνδρομή καθώς και πληροφορίες σχετικά με την κάθε εφαρμογή.
5. **Αναζήτηση διαθέσιμων κόμβων - Δημιουργία συνδρομής.** Μέσω γραφικής διεπαφής, ο τελικός χρήστης μπορεί να δει όλους τους διαθέσιμους κόμβους Σημασιολογικού Ιστού που είναι καταχωρημένοι στο Fi-SWoT και να υποβάλλει αίτηση συνδρομής σε όποιον επιθυμεί με στόχο να μπορεί να χρησιμοποιεί τις εφαρμογές του εκάστοτε απομακρυσμένου κόμβου. Πριν στείλει αίτημα, μπορεί να ανακαλύψει τις δημόσιες πληροφορίες των κόμβων και να επιλέξει αυτόν που τον ενδιαφέρει. Όταν ο χρήστης ζητήσει εγγραφή σε έναν ξένο κόμβο, το αίτημα δρομολογειται στον αντίστοιχο διαχειριστή ο οποίος αποφασίζει εάν θα αποδεχτεί τον χρήστη ή όχι. Αν ο διαχειριστής αποδεχτεί το αίτημα, θα εκχωρήσει στον αντίστοιχο χρήστη τους κατάλληλους ρόλους με τα αντίστοιχα δικαιώματά τους, προκειμένου να έχει πρόσβαση σε πόρους του συστήματος του.

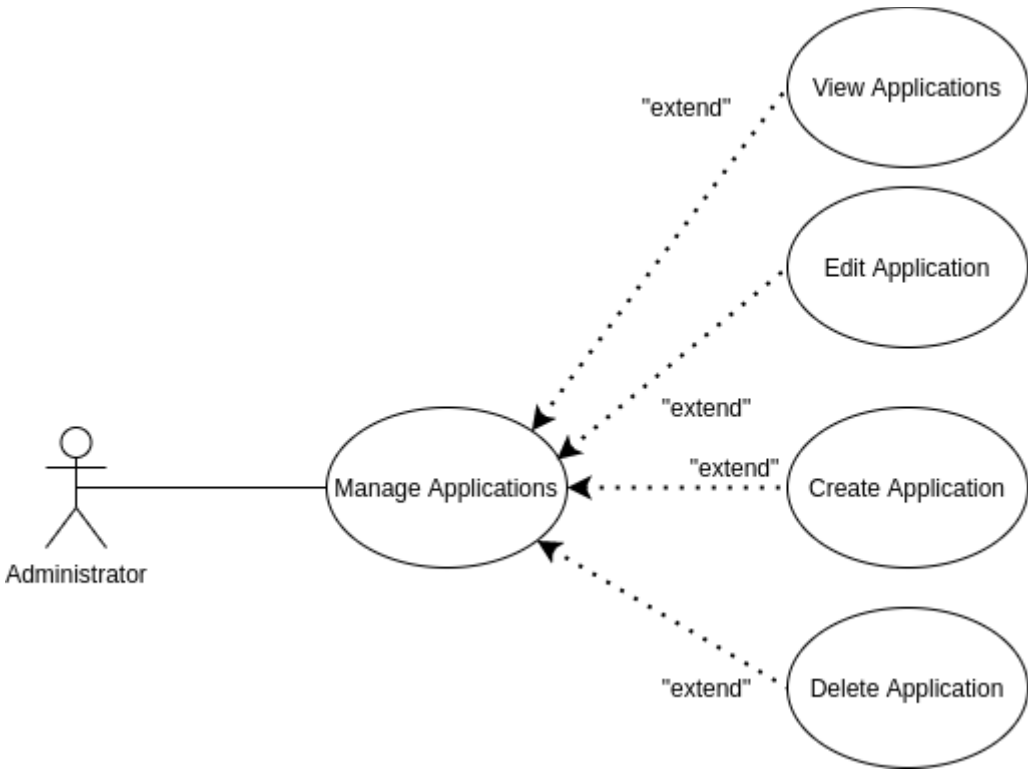
3.2 Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams)

Έχοντας περιγράψει στην Ενότητα 3.1 τις λειτουργικές χρηστών ανά κατηγορία, για την καλύτερη αποτύπωσή τους ομαδοποιούνται και παρουσιάζονται παρακάτω σε διαγράμματα περιπτώσεων χρήσης της UML (Use Case Diagrams).

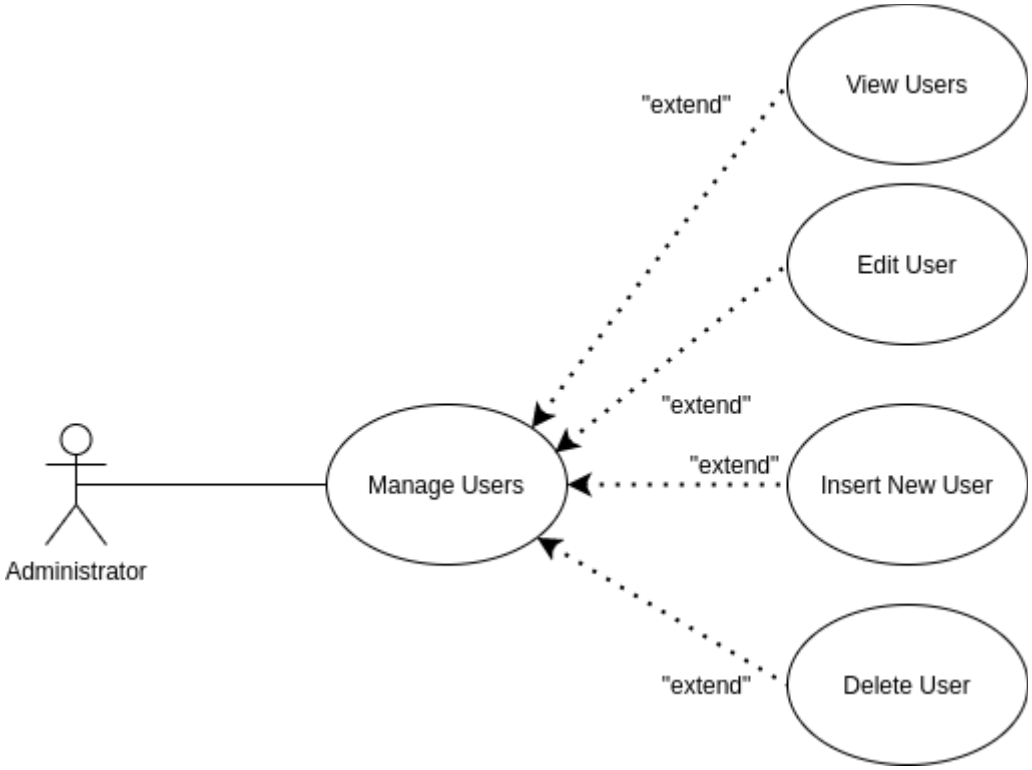
Για τον διαχειριστή του κάθε κόμβου, οι λειτουργίες που περιγράφηκαν στην ενότητα 3.1, αποτυπώνονται στα διαγράμματα των εικόνων 3.1.1, 3.1.2, 3.1.3 και 3.1.4, προκειμένου να είναι πιο ευδιάκριτα.



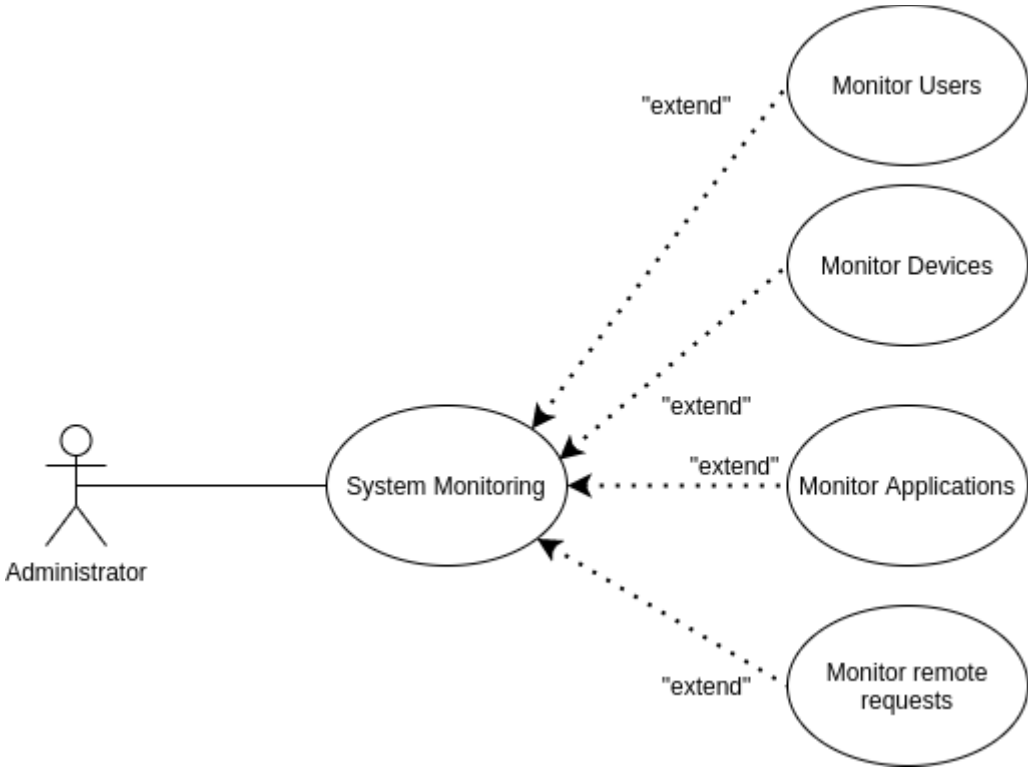
Εικόνα 3.1.1*: Διάγραμμα περιπτώσεων χρήσης (Use case Diagram) για τον διαχειριστή του συστήματος.*



Εικόνα 3.1.2*: Διάγραμμα περιπτώσεων χρήσης (Use case Diagram) για τον διαχειριστή του συστήματος.*

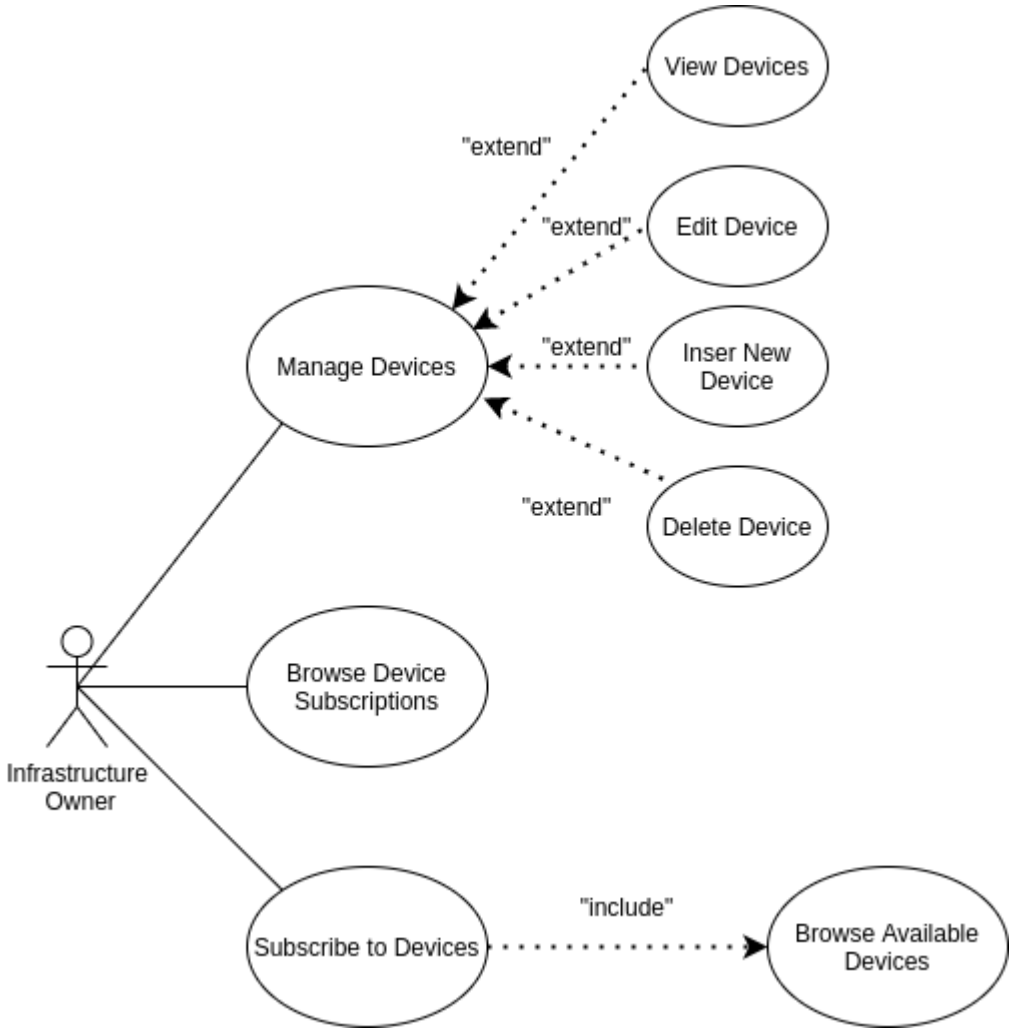


Εικόνα 3.1.3*: Διάγραμμα περιπτώσεων χρήσης (Use case Diagram) για τον διαχειριστή του συστήματος.*

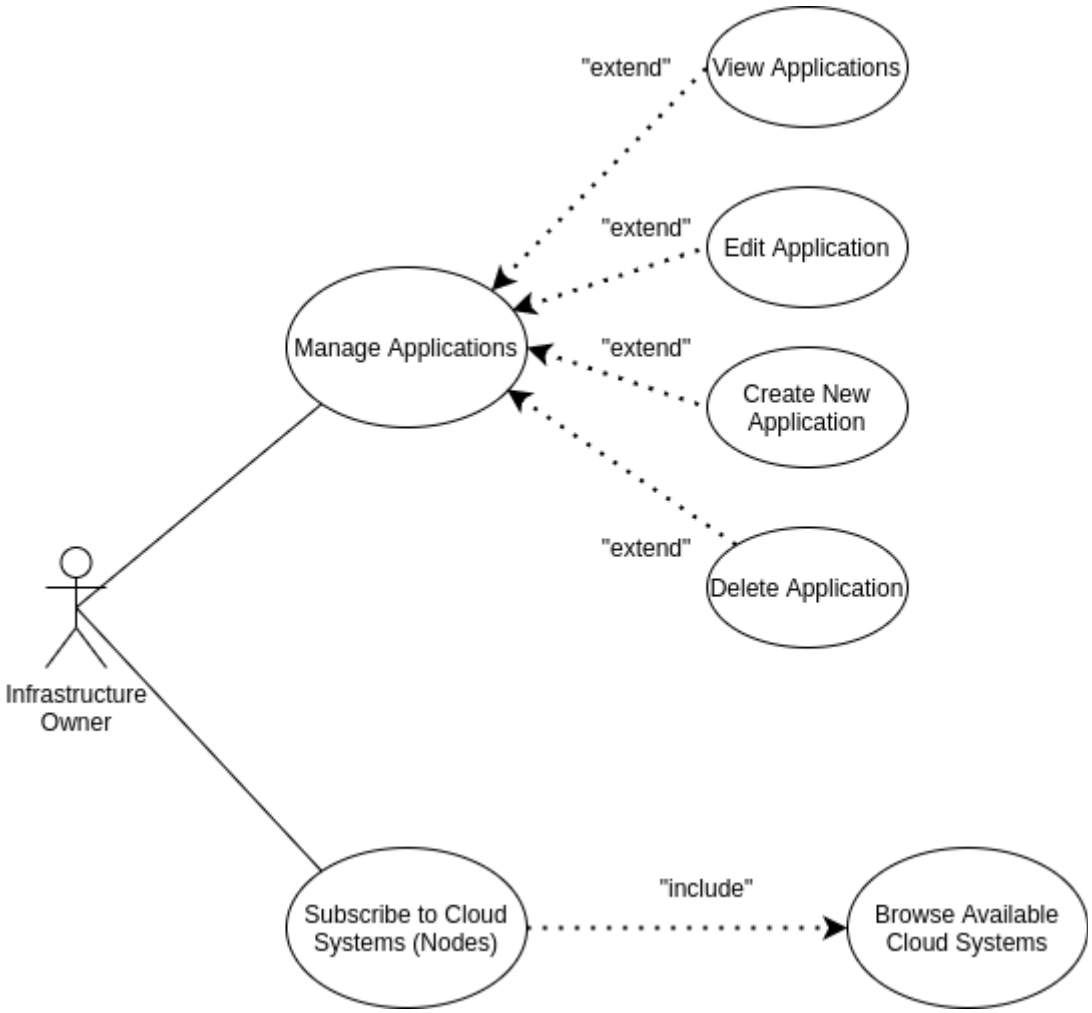


Εικόνα 3.1.4*: Διάγραμμα περιπτώσεων χρήσης (Use case Diagram) για τον διαχειριστή του συστήματος.*

Αντίστοιχα, για τον ιδιοκτήτη υποδομής, οι λειτουργίες που περιγράφηκαν στην ενότητα 3.1, αποτυπώνονται στα διαγράμματα των εικόνων 3.2.1 και 3.2.2, προκειμένου να είναι πιο ευδιάκριτα.

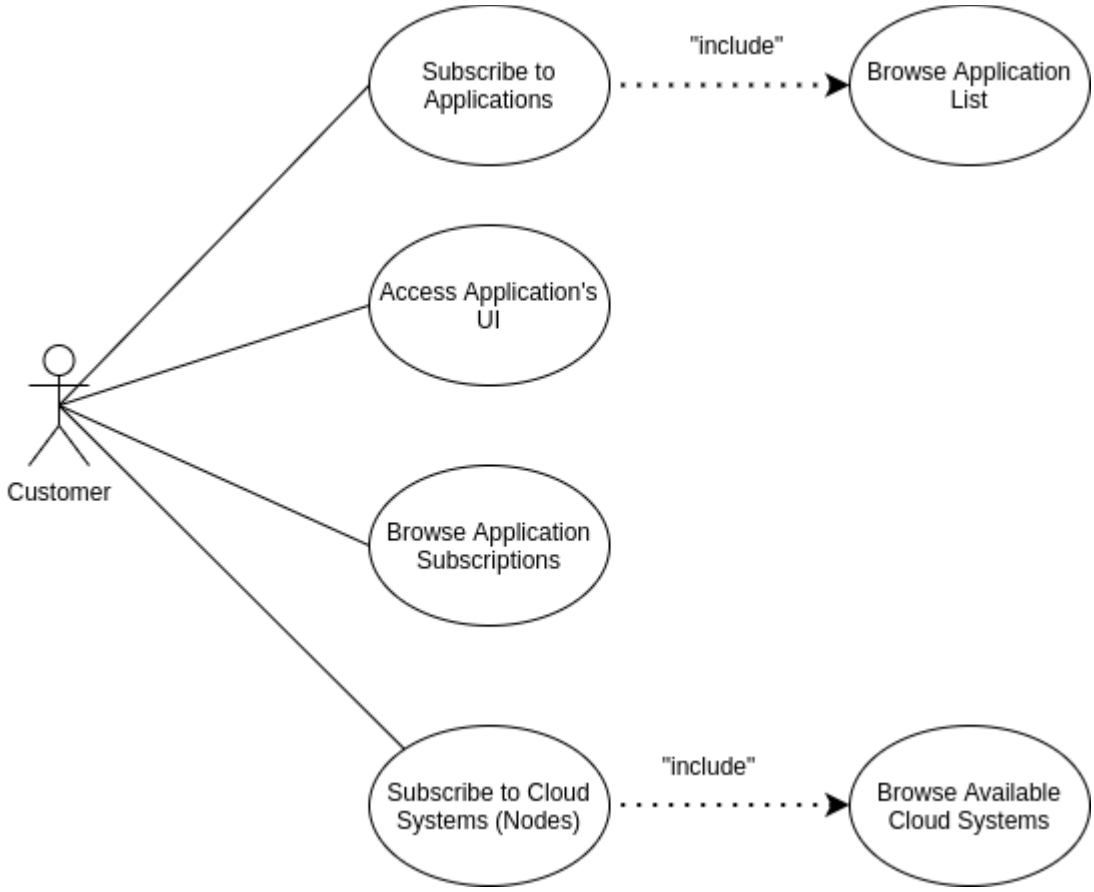


Εικόνα 3.2.1*: Διάγραμμα περιπτώσεων χρήσης (Use case Diagram) για τον ιδιοκτήτη υποδομής.*



Εικόνα 3.2.2*: Διάγραμμα περιπτώσεων χρήσης (Use case Diagram) για τον ιδιοκτήτη υποδομής.*

Τέλος, για τον τελικό χρήστη, οι λειτουργίες που περιγράφηκαν στην ενότητα 3.1, αποτυπώνονται στο διάγραμμα της εικόνας 3.3.



***Εικόνα 3.3**: Διάγραμμα περιπτώσεων χρήσης (Use case Diagram) για τον τελικό χρήστη του συστήματος.*

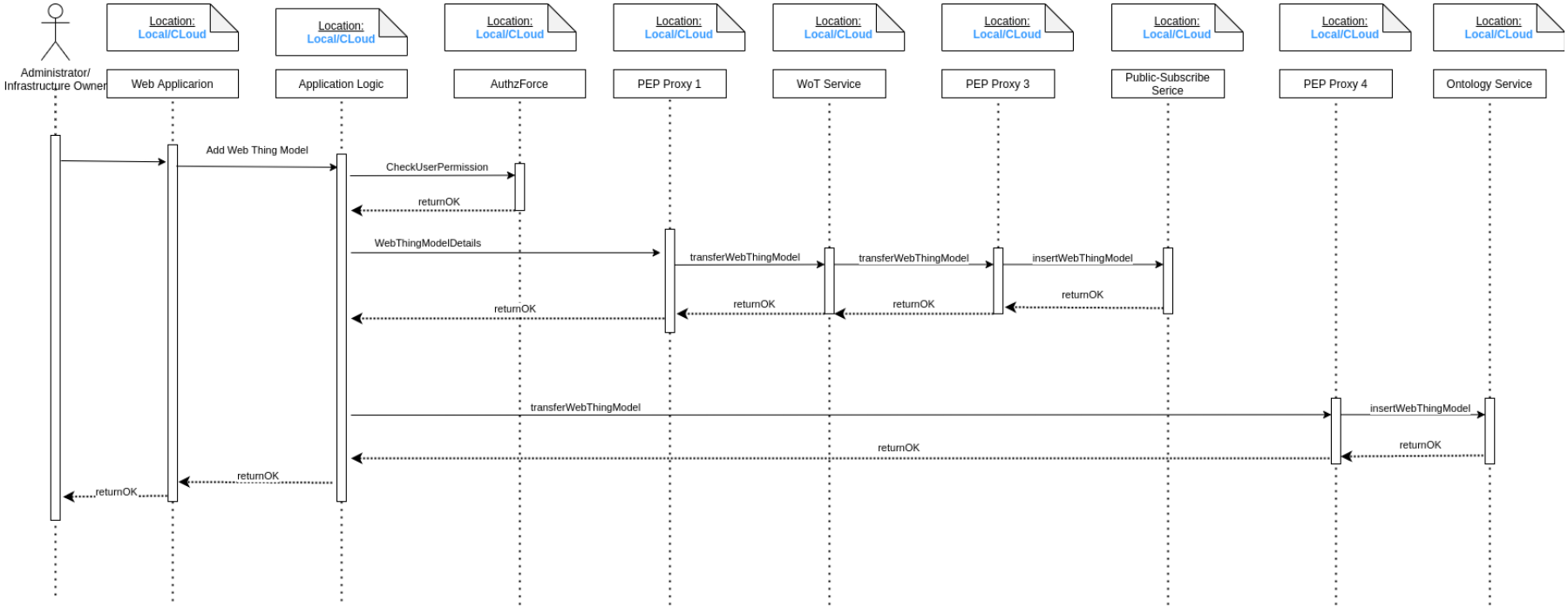
3.3 Διαγράμματα Ακολουθίας / Δραστηριοτήτων

Αυτή η ενότητα δείχνει τα διαγράμματα ακολουθίας / δραστηριοτήτων για κάθε τύπο χρήστη. Τα διαγράμματα ακολουθίας στοχεύουν στην παρουσίαση των πιο σημαντικών λειτουργιών του συστήματος για αυτές τις κατηγορίες χρηστών. Τα ακόλουθα διαγράμματα αφορούν λειτουργίες που εκτελούν οι χρήστες **τοπικά**, δηλαδή στο κόμβο στον οποίο είναι συνδεδεμένοι. Οι ενέργειες των χρηστών σε απομακρυσμένους κόμβους του Fi-SWoT αναλύονται και παρουσιάζονται σε επόμενη ενότητα (βλ. 3.6.3).

Όλα τα αιτήματα που υποβάλλουν οι χρήστες μέσω των γραφικών διεπαφών, πρέπει να περάσουν από ένα κοινό στάδιο. Αυτό το στάδιο περιλαμβάνει την ταυτοποίηση του χρήστη από τον τοπικό Keycloak. Μόλις ο χρήστης αναγνωριστεί από την τοπική υπηρεσία, λαμβάνει χώρα το επόμενο στάδιο, δηλαδή η εξουσιοδότηση του χρήστη. Επιπλέον, όλα τα αιτήματα δρομολογούνται στις υπηρεσίες τοπικού κόμβου χρησιμοποιώντας πρωτόκολλο **HTTP**.

- **Προσθήκη μοντέλου πράγματος (Add a Web Thing Model)**

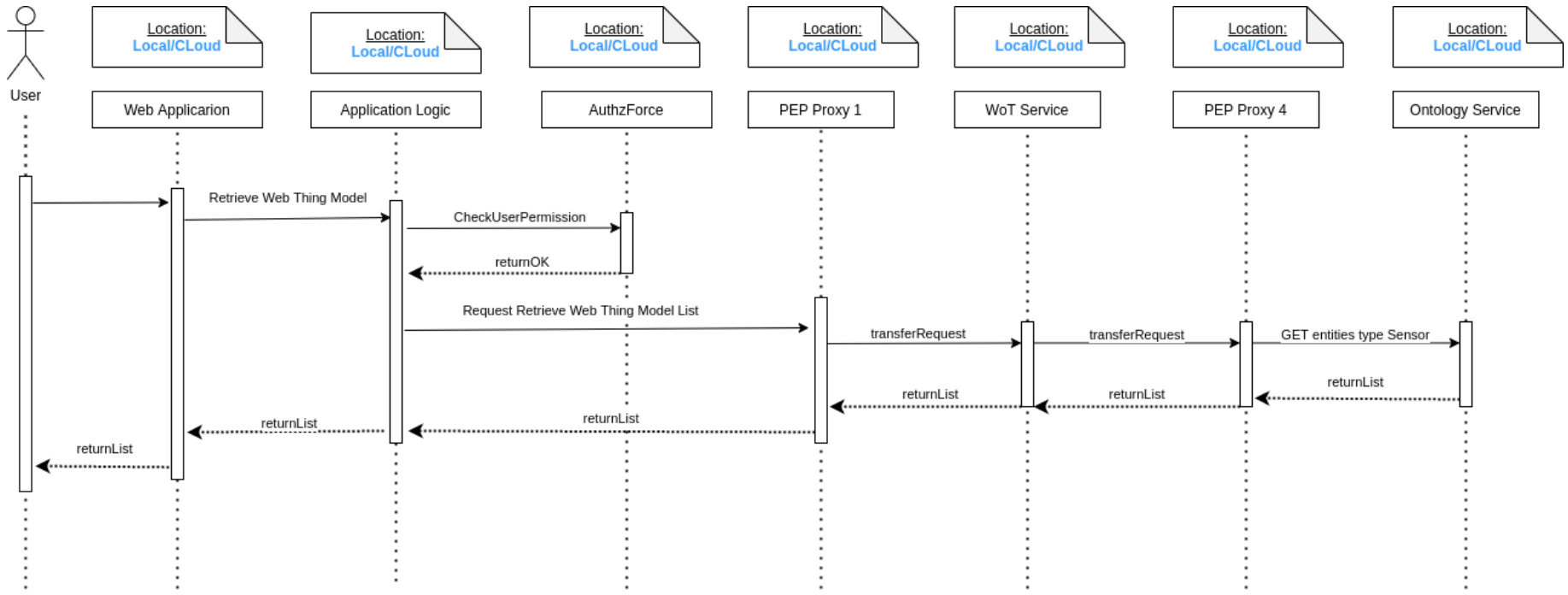
Ο διαχειριστής ή ο ιδιοκτήτης της υποδομής είναι υπεύθυνος για την προσθήκη ενός νέου μοντέλου της μορφής NGSI-LD στο σύστημα, δηλαδή μιας νέας εικονικής οντότητας (ενός αισθητήρας, μιας μέτρησης κλπ). Ο χρήστης αφού συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να εκτελέσει την λειτουργία Add a Web Thing Model στο τοπικό του κόμβο. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, ο χρήστης ανακατευθύνεται σε νέα σελίδα, όπου μπορεί να εισάγει την NGSI-LD αναπαράσταση της οντότητας που θέλει να προσθέσει. Αφού ολοκληρώσει την περιγραφή του νέου μοντέλου πράγματος, επιλέγει την εισαγωγή της νέας πληροφορίας στο σύστημα και το αίτημα κατευθύνεται στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας, δηλαδή της υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key (κάθε PEP Proxy-Wilma έχει τον δικό του μυστικό κωδικό Master Key ο οποίος καθορίζεται κατά την αρχικοποίηση του εκάστοτε Διακομιστή Μεσολάβησης). Εφόσον ο κωδικός Master Key είναι σωστός, το Pep Proxy προωθεί την επικοινωνία στην υπηρεσία για την οποία μεσολαβεί. Έπειτα η υπηρεσία WoT, προωθεί την πληροφορία του νέου μοντέλου στις κατάλληλες υπηρεσίες για να αποθηκευτεί σωστά στο σύστημα. Αυτές είναι η υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Publish - Subscribe Service) και η υπηρεσία της Οντολογίας. Αξίζει να σημειωθεί ότι και οι δύο αποτελούν προστατευόμενες υπηρεσίες, και η πρόσβαση σε αυτές γίνεται με την χορήγηση του σωστού κωδικού Master Key στον εκάστοτε διακομιστή μεσολάβησης. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.4.



***Εικόνα 3.4**: Διάγραμμα ακολουθίας για την προσθήκη νέου μοντέλου πράγματος.*

- **Ανάκτηση μοντέλων πραγμάτων (Retrieve a list of Web Things Models)**

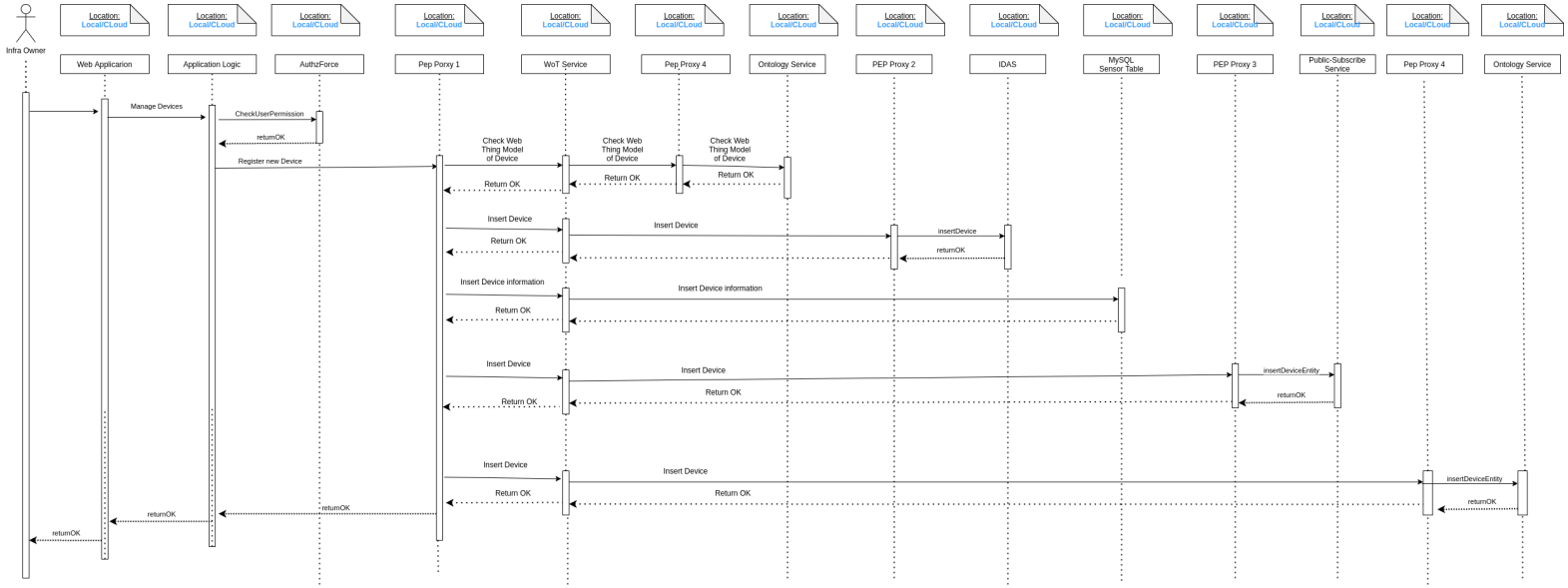
Ένας χρήστης μπορεί να ανακτήσει μία λίστα με τα μοντέλα πραγμάτων που είναι αποθηκευμένα στο σύστημα και τον ενδιαφέρουν, και στη συνέχεια να επιλέξει όποιο επιθυμεί για να δει την περιγραφή του. Ο χρήστης αφού συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να εκτελέσει την λειτουργία ανάκτηση του μοντέλου ενός πράγματος στο τοπικό του κόμβο. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, το αίτημα κατευθύνεται στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας, δηλαδή της υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key (κάθε PEP Proxy-Wilma έχει τον δικό του μυστικό κωδικό Master Key ο οποίος καθορίζεται κατά την αρχικοποίηση του εκάστοτε Διακομιστή Μεσολάβησης). Εφόσον ο κωδικός Master Key είναι σωστός, το Pep Proxy προωθεί την επικοινωνία στην υπηρεσία για την οποία μεσολαβεί. Έπειτα η υπηρεσία WoT, η οποία είναι υπεύθυνη για να απαντάει ερωτήματα σημασιολογικού ιστού, προωθεί το ερώτημα στο διακομιστή μεσολάβησης της υπηρεσίας της οντολογίας στην οποία είναι αποθηκευμένα όλα τα μοντέλα πραγμάτων. Με αυτό τον τρόπο επιστρέφονται όλα τα μοντέλα πραγμάτων και ο χρήστης μπορεί να επιλέξει ένα από αυτά και να ανακτήσει την NGSI-LD αναπαράσταση του. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.5.



***Εικόνα 3 5**: Διάγραμμα ακολουθίας για την ανάκτηση μοντέλων πραγμάτων.*

- **Εγγραφή μιας νέας συσκευής (Register a new device)**

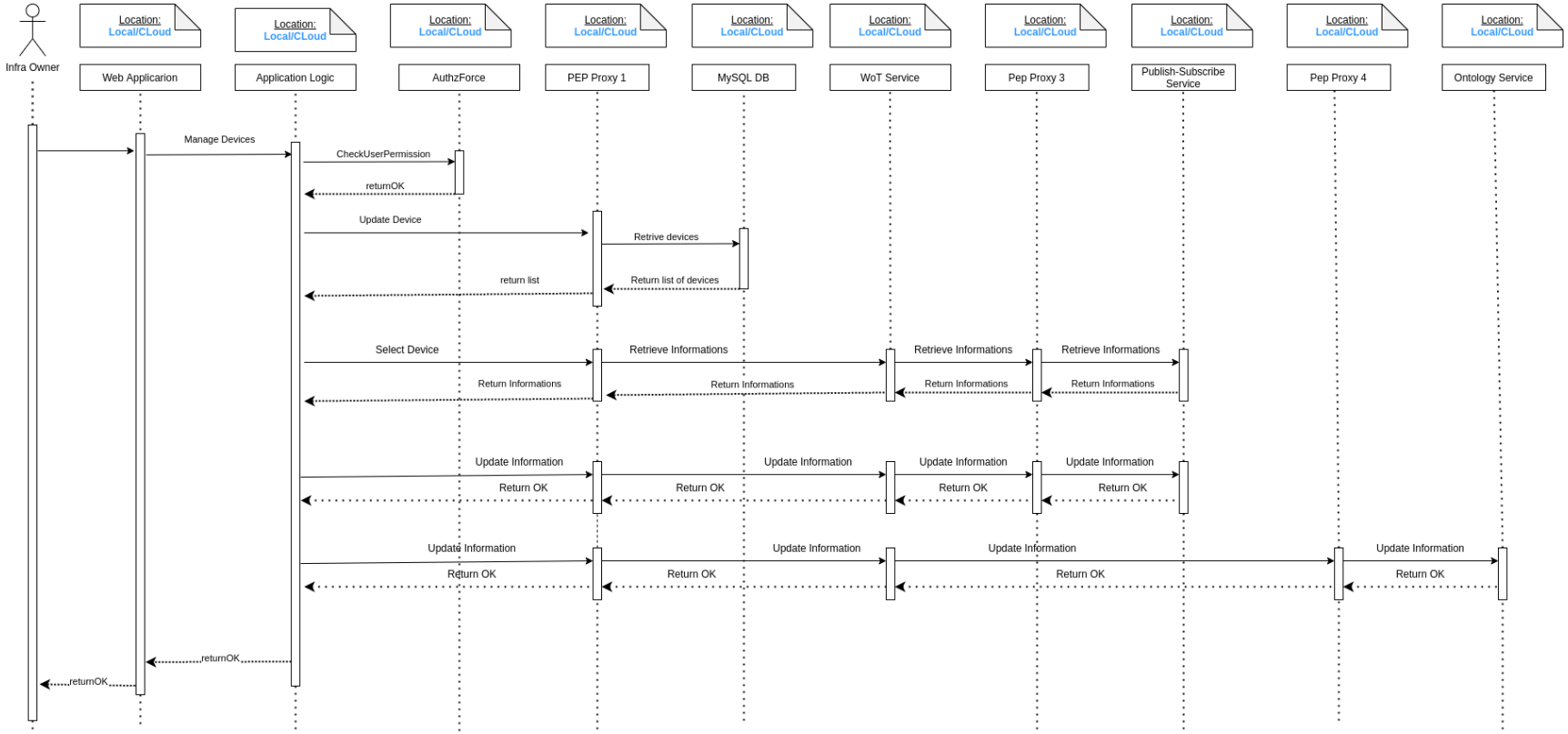
Ο διαχειριστής ή ό ιδιοκτήτης της υποδομής μπορεί να εισάγει μια νέα συσκευή στο κόμβο που είναι συνδεδεμένος με στόχο να λαμβάνει μετρήσεις για ένα χαρακτηριστικό (π.χ. θερμοκρασία, υγρασία) του “έξω κόσμου”. Ο χρήστης αφού συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να εκτελέσει την λειτουργία εγγραφή νέας συσκευής στο τοπικό του κόμβο. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, ο διαχειριστής ανακατευθύνεται σε νέα σελίδα, όπου μπορεί να συμπληρώσει τα στοιχεία της συσκευής. Προτού όμως γίνει αυτό ο χρήστης ερωτάται από το σύστημα για το όνομα του Πράγματος που συνιστά την εικονική οντότητα της συγκεκριμένης συσκευής. Έτσι, γίνεται άμεσα έλεγχος για το αν η οντολογία του συστήματος “γνωρίζει” αυτό το Πράγμα, δηλαδή για το αν το μοντέλο του Πράγματος έχει καταχωρηθεί στο σύστημα και συμπεριλαμβάνεται στις γνώσεις της οντολογίας. Το παραπάνω ερώτημα στέλνεται στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key (κάθε PEP Proxy-Wilma έχει τον δικό του μυστικό κωδικό Master Key). Εφόσον ο κωδικός Master Key είναι σωστός, το Pep Proxy προωθεί την επικοινωνία στην υπηρεσία που “προστατεύει”, η οποία με την σειρά της δρομολογεί το ερώτημα στο διακομιστή μεσολάβησης της υπηρεσίας της οντολογίας με τον αντίστοιχο μυστικό κωδικό. Αφού το ερώτημα φτάσει στην υπηρεσία της οντολογίας, εκτελείται και επιστρέφεται στο χρήστη το αποτέλεσμα. Αν ο χρήστης λάβει θετική απάντηση, μπορεί να προχωρήσει στην εγγραφή της συσκευής. Αφού εισάγει τα απαραίτητα στοιχεία, μέσω της υπηρεσίας Web Thing Model στέλνεται ένα αίτημα POST στο διακομιστή μεσολάβησης της υπηρεσίας Διαχείρισης Συσκευών - Backend IDAS και στη συνέχεια η πληροφορία πηγαίνει στην ίδια την υπηρεσία για να επιτευχθεί η διασύνδεση της συσκευής με το σύστημα. Έπειτα, σε περίπτωση ορθής αποθήκευσης, ο IDAS επιστρέφει θετική απάντηση και η υπηρεσία Web Thing Model αναλαμβάνει η πληροφορία της εκάστοτε συσκευής να αποθηκευτεί και στις υπόλοιπες απαραίτητες υπηρεσίες. Συγκεκριμένα προωθεί την πληροφορία στις υπηρεσίες Διαχείρισης Συμβάντων και Συνδρομών (Publish - Subscribe Service) και Οντολογίας. Αξίζει να σημειωθεί ότι και οι δύο αποτελούν προστατευόμενες υπηρεσίες, και η πρόσβαση σε αυτές γίνεται με την χορήγηση του σωστού κωδικού Master Key στον εκάστοτε διακομιστή μεσολάβησης. Επιπλέον, η υπηρεσία Web Thing Model δρομολογεί στη σχεσιακή βάση δεδομένων πληροφορίες όπως όνομα νέας συσκευής και όνομα δημιουργού. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.6.



***Εικόνα 3.6**: Διάγραμμα ακολουθίας για την εισαγωγή νέας συσκευή.*

- **Ενημέρωση ενός μοντέλου συσκευής (Update the model of a device)**

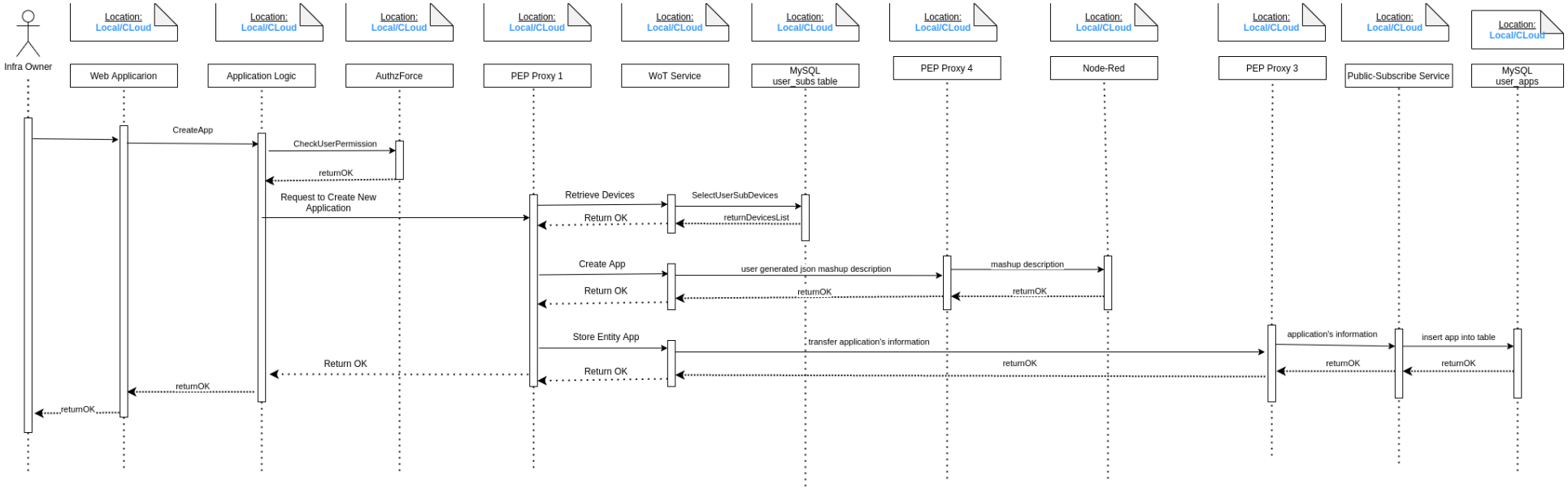
Ο διαχειριστής ή ο ιδιοκτήτης της υποδομής έχει το δικαίωμα να τροποποιήσει στοιχεία των συσκευών που βρίσκονται στη κατοχή του. Πιο συγκεκριμένα, αφού ο χρήστης συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να εκτελέσει την λειτουργία ενημέρωση συσκευής στο τοπικό του κόμβο, η οποία αντιστοιχίζεται στην υπηρεσία Update the model of a Thing που προτείνει το Web Thing Model της W3C. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, ο διαχειριστής ανακατευθύνεται σε νέα σελίδα, όπου μπορεί να ανακτήσει όλες τις συσκευές που του ανήκουν. Το αίτημα ανάκτησης όλων των συσκευών του χρήστη, στέλνεται πρώτα στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key και έπειτα η υπηρεσία προωθεί το αίτημα στη βάση δεδομένων. Στο παράθυρο περιήγησης του χρήστη, επιστρέφεται μια λίστα με όλες τις συσκευές, που μπορεί να δει την αναπαράσταση τους σε NGSI-LD και να επιλέξει ποιά από αυτές θέλει να τροποποιήσει. Ο χρήστης επιλέγει πρώτα να δει την αναπαράσταση μιας συσκευής σε μορφή NGSI-LD και η υπηρεσία Web Thing Model προωθεί το αίτημα στο διακομιστή μεσολάβησης της υπηρεσίας Διαχείρισης Συμβάντων και Συνδρομών, ο οποίος προωθεί την επικοινωνία στην υπηρεσία που βρίσκεται αποθηκευμένη η ζητούμενη πληροφορία. Με αυτό τον τρόπο, επιστρέφεται στο χρήστη μια εκτενής περιγραφή της συσκευής. Στη συνέχεια, ο χρήστης εισάγει τις νέες τιμές και η υπηρεσία Web Thing Model αναλαμβάνει να ενημερωθούν οι κατάλληλες υπηρεσίες, όπου δηλαδή βρίσκονται αποθηκευμένες οι πληροφορίες της εκάστοτε συσκευής. Επομένως, αιτήματα τύπου UPDATE στέλνονται στις υπηρεσίες Διαχείρισης Συμβάντων και Συνδρομών (Orion-LD Context Broker) και Οντολογίας, αφού το επιτρέψει ο εκάστοτε διακομιστής μεσολάβησης (κωδικός Master Key). Έτσι, η περιγραφή του μοντέλου της συσκευής ενημερώνεται και αποθηκεύεται εκ νέου στο σύστημα. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.7.



***Εικόνα 3.7**: Διάγραμμα ακολουθίας για την ενημέρωση υπάρχουσας συσκευής.*

- **Δημιουργία νέας εφαρμογής (Create new application)**

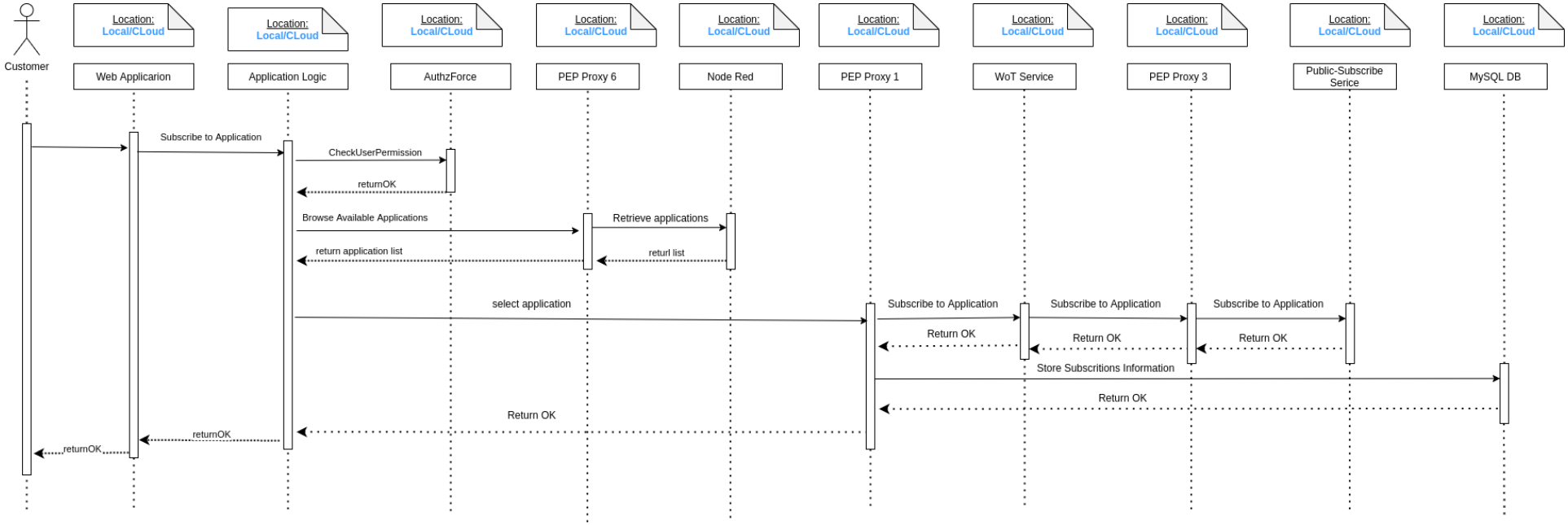
Ο διαχειριστής ή ο ιδιοκτήτης της υποδομής μπορεί να δημιουργήσει νέα εφαρμογή στο κόμβο που είναι συνδεδεμένος. Ο χρήστης αφού συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να δημιουργήσει μια νέα εφαρμογή. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, το αίτημα κατευθύνεται πρώτα στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας, δηλαδή της υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key (κάθε PEP Proxy-Wilma έχει τον δικό του μυστικό κωδικό Master Key ο οποίος καθορίζεται κατά την αρχικοποίηση του εκάστοτε Διακομιστή Μεσολάβησης). Εφόσον ο κωδικός Master Key είναι σωστός, το Pep Proxy προωθεί την επικοινωνία στην υπηρεσία για την οποία μεσολαβεί. Έπειτα η υπηρεσία WoT, είναι υπεύθυνη να ρωτήσει την σχεσιακή βάση δεδομένων και να επιστρέψει μόνο συσκευές που περιλαμβάνονται στη συνδρομή του χρήστη και μπορεί να χρησιμοποιήσει στη κατασκευή εφαρμογών. Στη συνέχεια, ο χρήστης επιλέγει τα mashup και την λειτουργία που θα έχει η εφαρμογή μέσω γραφικής διεπαφής (όνομα, σκοπός, πεδίο εφαρμογής, ιστορική ή σημασιολογική εφαρμογή κλπ). Η πληροφορία αυτή αποστέλλεται μετά, μέσω ενός αιτήματός, στο διακομιστή μεσολάβησης της υπηρεσίας Node-RED και δημιουργείται η ροή (flow) που αντιστοιχεί στην εκάστοτε εφαρμογή. Η ροή που δημιουργήθηκε αποθηκεύεται στην βάση αποθήκευσης των ροών του Node-RED. Ταυτόχρονα, αποστέλλεται και άλλο ένα αίτημα στο διακομιστή μεσολάβησης της υπηρεσίας Δημοσίευσης - Συνδρομής (Orion-LD Context Broker) ώστε να δημιουργηθεί η οντότητα της εφαρμογής στο σύστημα και να δημοσιοποιηθεί. Πλέον, η εφαρμογή αποκτά ένα μοναδικό αναγνωριστικό και μπορεί να ανακτηθεί μέσω του συνδέσμου (URL) που οδηγεί σε αυτή. Τέλος, στη σχεσιακή βάση δεδομένων αποθηκεύονται οι πληροφορίες όνομα εφαρμογής και όνομα κατόχου. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.8.



***Εικόνα 3.8**: Διάγραμμα ακολουθίας για την δημιουργία νέας εφαρμογής.*

- **Συνδρομή σε εφαρμογή (Subscribe to application)**

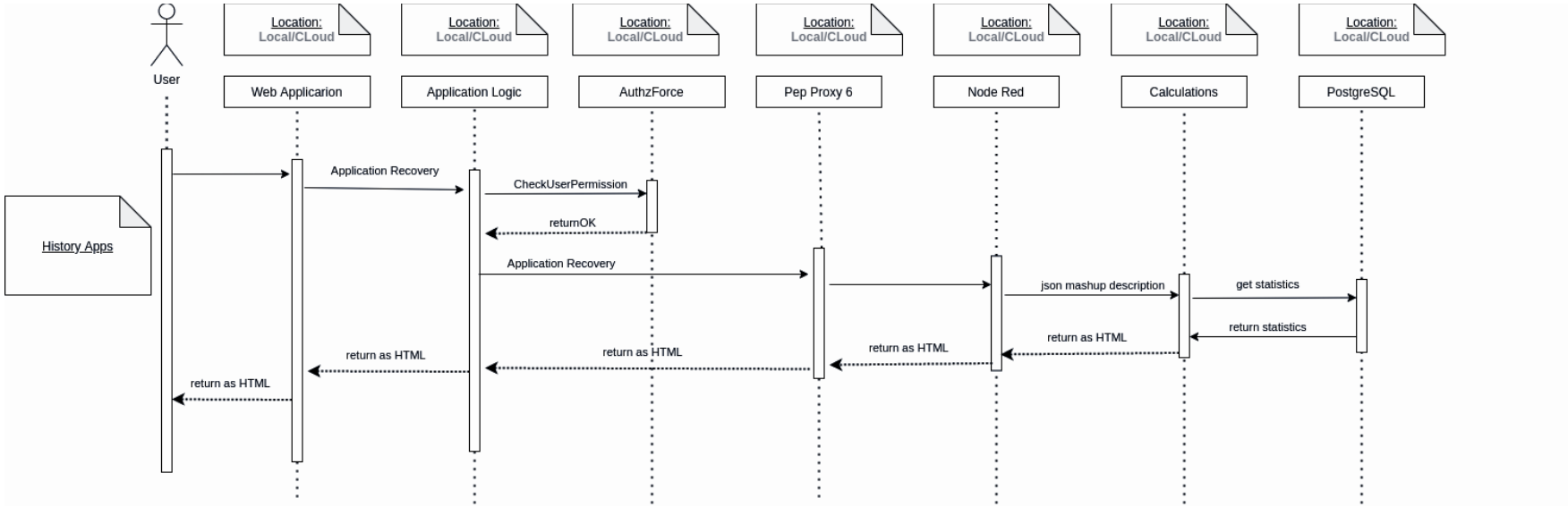
Ο τελικός χρήστης μπορεί να δημιουργήσει συνδρομή σε πόρους του συστήματος όπως είναι οι εφαρμογές. Ο τελικός χρήστης αφού συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να δημιουργήσει συνδρομή σε εφαρμογή με σκοπό να αποκτήσει πρόσβαση σε αυτήν. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, το αίτημα κατευθύνεται στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας, δηλαδή της υπηρεσίας Node-RED, περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key (κάθε PEP Proxy-Wilma έχει τον δικό του μυστικό κωδικό Master Key ο οποίος καθορίζεται κατά την αρχικοποίηση του εκάστοτε Διακομιστή Μεσολάβησης). Εφόσον ο κωδικός Master Key είναι σωστός, το Pep Proxy προωθεί την επικοινωνία στην υπηρεσία Node-RED, η οποία θα επιστρέψει στον χρήστη όλες τις υπάρχουσες εφαρμογές και αυτός θα κληθεί να επιλέξει εκείνη που καλύπτει τις απαιτήσεις του. Εφόσον ο χρήστης διαλέξει μια, στέλνεται ένα αίτημα συνδρομής στο διακομιστή μεσολάβησης της υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (WoT Service), ο οποίος προωθεί την επικοινωνία στην υπηρεσία για την οποία μεσολαβεί. Η τελευταία αναλαμβάνει να δρομολογήσει το αίτημα στον αντίστοιχο διακομιστή μεσολάβησης της υπηρεσίας Δημοσίευσης - Συνδρομής (Orion-LD Context Broker), η οποία είναι υπεύθυνη για την δημιουργία συνδρομών σε οντότητες τύπου εφαρμογών. Τέλος, στη σχεσιακή βάση δεδομένων αποθηκεύονται οι πληροφορίες όνομα εφαρμογής και όνομα συνδρομητή. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.9.



Εικόνα 3.9: *Διάγραμμα ακολουθίας για την δημιουργία συνδρομής σε εφαρμογή.*

- **Ανάκτηση ιστορικής εφαρμογής**

Ο τελικός χρήστης μπορεί να ανακτήσει και να χρησιμοποιήσει εφαρμογές στις οποίες έχει συνδρομή. Ο τελικός χρήστης αφού συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να ανατρέξει σε μια υπάρχουσα εφαρμογή, στην οποία έχει δικαίωμα πρόσβασης. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, το αίτημα κατευθύνεται στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας, δηλαδή της υπηρεσίας Node-RED, περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key (κάθε PEP Proxy-Wilma έχει τον δικό του μυστικό κωδικό Master Key ο οποίος καθορίζεται κατά την αρχικοποίηση του εκάστοτε Διακομιστή Μεσολάβησης). Εφόσον ο κωδικός Master Key είναι σωστός, το Pep Proxy προωθεί την επικοινωνία στην υπηρεσία για την οποία μεσολαβεί. Σε αυτήν την περίπτωση στέλνεται ένα αίτημα στην υπηρεσία Node-RED και συγκεκριμένα στην ροή (flow) που αντιστοιχεί στην εφαρμογή. Στη συνέχεια στέλνεται ένα POST αίτημα στην υπηρεσία Calculations με τα δεδομένα που ζητάει ο χρήστης, η οποία αναλύει την περιγραφή mashup της εφαρμογής, αποφασίζει τι είδους εφαρμογή είναι και ποιες υπηρεσίες θα χρησιμοποιηθούν στη συνέχεια για την εκτέλεση της εφαρμογής. Η τρέχουσα δραστηριότητα αφορά εφαρμογή ιστορικού περιεχομενου. Επομένως, στέλνονται τα κατάλληλα ερωτήματα στην βάση δεδομένων PostgreSQL και μόλις επιστραφούν τα αποτελέσματα στην υπηρεσία Calculations, παίρνουν την HTML μορφή που θα απεικονιστεί στο παράθυρο περιήγησης του χρήστη. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.10.



Εικόνα 3.10: *Διάγραμμα ακολουθίας για την ανάκτηση ιστορικής εφαρμογής.*

3.4 Υπηρεσίες Συστήματος ενός κόμβου

Ακολουθεί μια εκτενής περιγραφή των επιμέρους υπηρεσιών του συστήματος, καθώς και της αλληλεπίδρασης που υπάρχει μεταξύ τους προκειμένου να πραγματοποιηθούν οι ζητούμενες λειτουργίες.

3.4.1 Υπηρεσία Διακομιστή μεσολάβησης του Ιστού των Πραγμάτων (Web of Things Proxy)

Η υπηρεσία Διακομιστή μεσολάβησης του Ιστού των Πραγμάτων αποτελείται από δύο υπηρεσίες: την υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και την Υπηρεσία Διαχείρισης Συσκευών Backend IDAS. Ο Διακομιστής μεσολάβησης αλληλεπιδρά με τις εξωτερικές οντότητες του Ιστού των Πραγμάτων (τα ερεθίσματα από τον “έξω κόσμο”), δηλαδή με τους χρήστες (χάρη στην πρώτη υπηρεσία) αλλά και με τις συσκευές - αισθητήρες (μέσω της δεύτερης). Με αυτόν τον τρόπο, μεσολαβεί ανάμεσα στο Υπολογιστικό Νέφος και τον έξω κόσμο, επιτρέποντας την είσοδο δεδομένων (μετρήσεων) από τον έξω κόσμο, καθώς και έξοδο προς αυτόν (π.χ. όταν δίνεται εντολή προς μια συσκευή).

Αναλυτικά, η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων είναι υπεύθυνη για την υλοποίηση όλων των λειτουργιών που προβλέπει το Web Thing Model. Γι’ αυτό το λόγο, επικοινωνεί άμεσα με την Διαδικτυακή Εφαρμογή (Web Application) του συστήματος, δηλαδή μια γραφική διεπαφή για τους χρήστες, με σκοπό να συμβάλει στην εξυπηρέτηση όλων των λειτουργιών που επιλέγει ο χρήστης μέσω αυτής. Παράλληλα, συνεργάζεται τόσο με την Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών του συστήματος όσο και με την Υπηρεσία Οντολογίας, με τις οποίες έχει αμφίδρομη σχέση, καθώς μπορεί να λαμβάνει και να στέλνει δεδομένα προς αυτές. Για παράδειγμα, μπορεί να ανακτά πληροφορίες (σχετικές με τις συσκευές) από την οντολογία ή από τη βάση δεδομένων της Υπηρεσίας Διαχείρισης Συμβάντων και Συνδρομών (Orion-LD Context Broker) και να τις εκτυπώνει στο χρήστη μέσω της Διαδικτυακής Εφαρμογής. Και αντίστροφα, μπορεί να προσθέτει / ενημερώνει / διαγράφει πληροφορίες στις προαναφερθείσες αποθηκευτικές δομές, σύμφωνα με τις επιλογές του χρήστη στη Διαδικτυακή Εφαρμογή. Επιπλέον, η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων επικοινωνεί και με την Υπηρεσία Διαχείρισης Συσκευών Backend IDAS, είτε στην περίπτωση που ο χρήστης επιθυμεί να εγγράψει μια συσκευή στη λίστα των συσκευών της υπηρεσίας IDAS (ούτως ώστε το σύστημα να λαμβάνει στη συνέχεια δεδομένα από αυτήν τη συσκευή) είτε στην περίπτωση που απλά θέλει να ελέγξει αν μια συσκευή βρίσκεται στην παραπάνω λίστα συσκευών.

Όπως προαναφέρθηκε, η Υπηρεσία Διαχείρισης Συσκευών Backend IDAS του οικοσυστήματος FIWARE, αναλαμβάνει να εξυπηρετήσει την επικοινωνία των συσκευών με το Υπολογιστικό Νέφος, εφόσον αυτές έχουν συνδεθεί στην υπηρεσία και μεταδίδουν πληροφορίες. Η παραπάνω υπηρεσία δέχεται τις μετρήσεις που στέλνει κάθε συσκευή, ανεξάρτητα από το πρωτόκολλο κάθε συσκευής, και μεταφέρει

την πληροφορία στην Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών αποκλειστικά σε μορφή NGSI-LD. Επομένως, ο ρόλος της υπηρεσίας IDAS είναι πολύ σημαντικός, διότι επιτυγχάνει προσαρμογή πρωτοκόλλου (protocol adaptation) και καταφέρνει να καταστήσει την επικοινωνία ανεξάρτητη από το πρωτόκολλο μετάδοσης της συσκευής. Έτσι, η πληροφορία από συσκευές διαφορετικού είδους και πρωτοκόλλων αποθηκεύεται στο σύστημα με την ίδια μορφή και η διαχείρισή της γίνεται ακριβώς με τον ίδιο τρόπο. Σε αυτήν την πληροφορία, συμπεριλαμβάνονται μεταδεδομένα, όπως η τοποθεσία της συσκευής, η ιδιότητα που υφίσταται μέτρηση, το πρωτόκολλο της συσκευής, κλπ. Συμπεραίνουμε, δηλαδή, ότι αυτή η υπηρεσία είναι το σημείο εισόδου της πληροφορίας των συσκευών στο Υπολογιστικό Νέφος.

3.4.2 Διαχείριση Δεδομένων

Το σύστημα μας διαχειρίζεται μεγάλη ποικιλία πληροφοριών. Για αυτό τον λόγο, για την καλύτερη επεξεργασία και διαχείριση, κάθε τύπος πληροφορίας αντιστοιχίζεται σε διαφορετική δομή αποθήκευσης.

Συγκεκριμένα, η μη-σχεσιακή βάση δεδομένων Orion-LD MongoDB χρησιμοποιείται από την υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Orion-LD Context Broker) για την αποθήκευση της πληροφορίας όλων των οντοτήτων που δημιουργούνται στο σύστημα, καθώς και των συνδρομών που δημιουργούνται ως προς τις οντότητες. Η MongoDB είναι κατάλληλη για την διαχείριση μεγάλου όγκου δεδομένων και ταυτόχρονα πολύ γρήγορη. Επίσης, είναι συμβατή με δεδομένα της μορφής NGSI-LD, που αποτελεί το μοντέλο πληροφορίας και την αναπαράσταση όλων των εικονικών οντοτήτων που διαχειρίζεται ο Orion-LD Context Broker. Επιπλέον, η MongoDB χρησιμοποιείται για την αποθήκευση μόνο των τρεχουσών τιμών συσκευών - αισθητήρων, που είναι συνδεδεμένες στο σύστημα.

Οι ιστορικές μετρήσεις των συσκευών αποθηκεύονται στη σχεσιακή βάση δεδομένων PostgreSQL. Αυτή η βάση αποτελεί κομμάτι της ιστορικής υπηρεσίας του συστήματος και αναλύεται στη συνέχεια. Ωστόσο, μπορεί να εξυπηρετήσει μεγάλο όγκο δεδομένων μετρήσεων (χρονικής σειράς) και η ανάκτηση ιστορικών τιμών πραγματοποιείται μέσα από ερωτήματα SQL που εκτελούνται στη συγκεκριμένη βάση.

Η σχεσιακή βάση δεδομένων του συστήματος (Relational DB) χρησιμοποιείται για την αποθήκευση πληροφοριών σχετικών με τους χρήστες. Αυτή η βάση χρησιμοποιείται από την υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών (βλ. παράγραφο 3.7) και αποθηκεύει στοιχεία κάθε εγγεγραμμένου χρήστη όπως όνομα,κωδικός, email, γεωγραγική θέση (κόμβος) κλπ. Λόγω της φύσης της πληροφορίας που αφορά το προφίλ ενός χρήστη, απαιτείται η χρήση σχεσιακής βάσης δεδομένων για την αποθήκευση αυτής της πληροφορίας.

Η σχεσιακή βάση δεδομένων του συστήματος (MySQL DB) χρησιμοποιείται για την αποθήκευση πληροφοριών σχετικών με τους χρηστές, τις συσκευές και τις εφαρμογές που έχουν στην κατοχή τους, καθώς και τις συνδρομές τους σε αυτές. Οι παραπάνω πληροφορίες αποθηκεύονται σε μορφή πίνακα (key-value pairs) και απαιτούνται για να ξέρουμε ποιές συσκευές ή εφαρμογές ανήκουν σε ένα χρήστη και

έχει την δυνατότητα να εκτελέσει ενέργειες ενημέρωσης / διαγραφής.

Η βάση δεδομένων “Applications Storage” χρησιμοποιείται από την υπηρεσία Σύνθεσης Εφαρμογών για να αποθηκεύονται οι εφαρμογές που κατασκευάζουν οι χρήστες.

Τέλος, στο Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) της Virtuoso περιέχεται ο γράφος RDF, στον οποίο έχει αποθηκευτεί η οντολογία του συστήματος. Το Virtuoso αποτελεί ένα ένα σύστημα αποθήκευσης τριπλετών (triple store) της οντολογίας. Η οντολογία που δημιουργήθηκε και χρησιμοποιείται από το σύστημα περιέχει (σε μορφή τριπλετών) τόσο το αρχικό μοντέλο οντολογίας όσο και τα δεδομένα αλλά και τα μεταδεδομένα που αφορούν τους αισθητήρες / συσκευές, όπως είναι οι μετρήσεις τους, τα σπίτια και οι πόλεις όπου αυτοί βρίσκονται, κλπ.

Υπηρεσία Αποθήκευσης των Ιστορικών Δεδομένων (History Service)

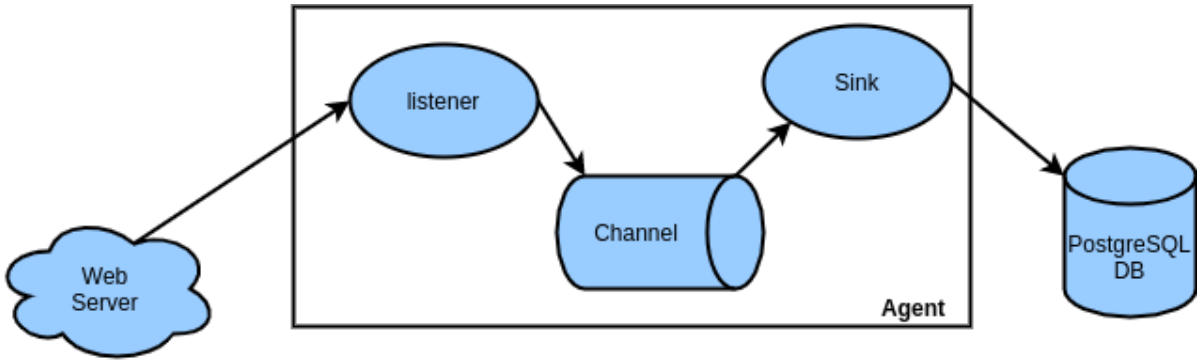
Η Υπηρεσία Αποθήκευσης των Ιστορικών Δεδομένων αναλαμβάνει να δέχεται ροές δεδομένων που προέρχονται από μετρήσεις συσκευών και συγκεκριμένα συσκευών που οι χρήστες χρησιμοποιούν , δηλαδή έχουν δημιουργήσει συνδρομή. Όπως και οι υπόλοιπες υπηρεσίες, περιλαμβάνει μικρότερες υπηρεσίες, που συνεργάζονται μεταξύ τους για να εξυπηρετήσουν τις ανάγκες της υπηρεσίας που διαχειρίζεται τα ιστορικά δεδομένα. Συγκεκριμένα, η παρούσα υπηρεσία αποτελείται από μια σχεσιακή βάση δεδομένων PostgreSQL από την υπηρεσία Cygnus-LD, που αποτελεί σύνδεσμο μεταξύ του Orion-LD Context Broker (στέλνει δεδομένα μετρήσεων σε μορφή NGSI-LD) και της βάσης δεδομένων PostgreSQL. Το Cygnus-LD δέχεται τις ροές δεδομένων NGSI-LD και τις αποθηκεύει στην PostgreSQL DB, δηλαδή στην ιστορική βάση δεδομένων του συστήματος. Επιπλέον, έχει τη δυνατότητα να αποθηκεύει τα ιστορικά δεδομένα με χρονολογική σειρά στην ιστορική βάση δεδομένων και να τα οργανώνει σε πίνακες χωρίς ανθρώπινη παρέμβαση. Με αυτόν τον τρόπο, διατηρείται το ιστορικό των μετρήσεων των αισθητήρων. Στη συνέχεια, για τον υπολογισμό στατιστικών όπως ανάκτηση του αθροίσματος, του μέγιστου ή του ελάχιστου από ένα πλήθος τιμών, καθώς και για συγκεκριμένα χρονικά διαστήματα (π.χ. για ανάκτηση δεδομένων κατά το διάστημα ενός ή παραπάνω λεπτών, ωρών, ημερών ή ακόμα και μηνών), ορίζονται εξειδικευμένα ερωτήματα (SQL) τα οποία απευθύνονται στην ιστορική βάση δεδομένων PostgreSQL. Τα αποτελέσματα που λαμβάνονται μέσω των παραπάνω ερωτημάτων, και συγκεκριμένα τα στατιστικά, αξιοποιούνται από άλλη υπηρεσία (Mashup Service), όπως φαίνεται και στο διάγραμμα αρχιτεκτονικής του συστήματος (βλ. Εικόνα 3.23). Το Mashup Service μπορεί να εκμεταλλευτεί κατάλληλα το γεγονός ότι λαμβάνει στατιστικά για επιλεγμένες χρονικές περιόδους, ούτως ώστε να εξυπηρετήσει τις ανάγκες των εφαρμογών. Παρακάτω παρατίθενται τα SQL ερωτήματα που είναι υπεύθυνα για τον υπολογισμό στατιστικών που χρησιμοποιούν οι εφαρμογές.

SQL Ερώτημα	Περιγραφή Ερωτήματος
SELECT max(hassimpleresult::integer), date (recvtime::timestamp) as real_date, extract (hour from recvtime::timestamp) as hour from tourguide.".$ids[$x]." where date_part('day',age(CURRENT_TIMESTAMP, recvtime::timestamp)) < 1 GROUP BY hour,real_date ORDER BY real_date,hour	Ερώτημα ανάκτησης μέγιστης τιμής μέτρησης παρατήρησης που έκανε ο αισθητήρας που παρακολουθεί το συγκεκριμένο χαρακτηριστικό ανά ώρα για τις τελευταίες 24 ώρες.
SELECT min(hassimpleresult::integer), date (recvtime::timestamp) as real_date, extract (hour from recvtime::timestamp) as hour from tourguide.".$ids[$x]." where date_part('day',age(CURRENT_TIMESTAMP, recvtime::timestamp)) < 1 GROUP BY hour,real_date ORDER BY real_date,hour	Ερώτημα ανάκτησης ελάχιστης τιμής μέτρησης παρατήρησης που έκανε ο αισθητήρας που παρακολουθεί το συγκεκριμένο χαρακτηριστικό ανά ώρα για τις τελευταίες 24 ώρες.
SELECT max(hassimpleresult::integer), date (recvtime::timestamp) as real_date from tourguide.".$ids[$x]." where date_part('day',age(CURRENT_TIMESTAMP, recvtime::timestamp)) < 8 GROUP BY real_date ORDER BY real_date	Ερώτημα ανάκτησης μέγιστης τιμής μέτρησης παρατήρησης που έκανε ο αισθητήρας που παρακολουθεί το συγκεκριμένο χαρακτηριστικό ανά ημέρα για τις τελευταίες 7 μέρες.
SELECT min(hassimpleresult::integer), date (recvtime::timestamp) as real_date from tourguide.".$ids[$x]." where date_part('day',age(CURRENT_TIMESTAMP, recvtime::timestamp)) < 8 GROUP BY real_date ORDER BY real_date	Ερώτημα ανάκτησης ελάχιστης τιμής μέτρησης παρατήρησης που έκανε ο αισθητήρας που παρακολουθεί το συγκεκριμένο χαρακτηριστικό ανά ημέρα για τις τελευταίες 7 μέρες.
SELECT avg(hassimpleresult::integer), date (recvtime::timestamp) as real_date, extract (hour from recvtime::timestamp) as hour from tourguide.".$ids[$x]." where date_part('day',age(CURRENT_TIMESTAMP, recvtime::timestamp)) < 1 GROUP BY hour,real_date ORDER BY real_date,hour	Ερώτημα ανάκτησης μέσης τιμής μέτρησης παρατήρησης που έκανε ο αισθητήρας που παρακολουθεί το συγκεκριμένο χαρακτηριστικό ανά ώρα για τις τελευταίες 24 ώρες.
SELECT avg(hassimpleresult::integer), date (recvtime::timestamp) as real_date from tourguide.".$ids[$x]." where date_part('day',age(CURRENT_TIMESTAMP, recvtime::timestamp)) < 8 GROUP BY real_date ORDER BY real_date	Ερώτημα ανάκτησης μέσης τιμής μέτρησης παρατήρησης που έκανε ο αισθητήρας που παρακολουθεί το συγκεκριμένο χαρακτηριστικό ανά ημέρα για τις τελευταίες 7 μέρες.

Πίνακας 3.1 *: Αναπαράσταση SQL ερωτημάτων*

Fiware Cygnus - LD

Η υπηρεσία Cygnus-LD είναι βασισμένη στην αρχιτεκτονική “Apache Flume” και παρέχει πράκτορες (agents), οι οποίοι είναι υπεύθυνοι για τη συλλογή ροών δεδομένων NGSI-LD και για την αποθήκευσή τους σε κάποια εξωτερική βάση δεδομένων που έχει προκαθοριστεί. Ένας πράκτορας (Εικόνα 3.11) απαρτίζεται από ένα “ακροατή” (listener), που είναι υπεύθυνος για τη λήψη των δεδομένων, ένα “κανάλι” (Channel) στο οποίο ο ακροατής προωθεί τα δεδομένα και έναν “προορισμό” (Sink), ο οποίος “παραλαμβάνει” τα δεδομένα με σκοπό να τα αποθηκεύσει σε μία εξωτερική βάση δεδομένων (η οποία ενδέχεται να είναι σε ξεχωριστή εικονική μηχανή από τον Πράκτορα). Το Cygnus-LD παρέχει εξειδικευμένο πράκτορα, που έχει τη δυνατότητα να στηρίξει τη συλλογή και τη διατήρηση NGSI-LD δεδομένων στη βάση δεδομένων - αποθετηρίων PostgreSQL, η οποία αποτελεί το προκαθορισμένο προορισμό (Sink) του Cygnus-LD.



Εικόνα 3.11 *: Apache flume Agent*

Το σενάριο της παρούσας εργασίας απαιτεί την χρήση της υπηρεσίας Cygnus-LD γιατί είναι απολύτως συμβατή με την υπηρεσία Orion-LD Context Broker και με την μορφή NGSI-LD. Συγκεκριμένα, έχει την δυνατότητα να αναγνωρίζει την πληροφορία σε μορφή NGSI-LD και ξέρει πώς να τη διαχειριστεί, δηλαδή πως να την μεταφέρει από τον Orion-LD στην PostgreSQL εξασφαλίζοντας μηδενική πιθανότητα λάθους.

Αξιοποιώντας τη λειτουργία συνδρομής που παρέχει η υπηρεσία Orion-LD Context Broker, ο πράκτορας αποτελεί συνδρομητή για όλες τις οντότητες συσκευών (μετρήσεων) του Orion-LD. Με αυτόν τον τρόπο, οποιαδήποτε αλλαγή συμβαίνει στο χαρακτηριστικό (attribute) “hasSimpleResult”, το οποίο διατηρεί την μέτρηση της συσκευής, πυροδοτεί μια ενημέρωση από τον Orion-LD προς το τελικό σημείο του συνδρομητή πράκτορα. Συνεπώς, ο πράκτορας λαμβάνει όλες τις νέες μετρήσεις των αισθητήρων (όταν αυτές προκύπτουν). Αμέσως μετά, αναλαμβάνει να αποθηκεύσει τις μετρήσεις χωρίς να γίνει κάποιο εσωτερικό configuration (υπάρχει έτοιμο), απλά και μόνο με το subscription, στην ιστορική βάση δεδομένων, διατηρώντας έτσι ένα ιστορικό δεδομένων χρονικής σειράς για τις μετρήσεις όλων των αισθητήρων που έχουν συνδεθεί στο σύστημα. Το τελευταίο αποτελεί σημαντική λειτουργία που επιτυγχάνεται από τον Cygnus-LD. Στην PostgreSQL, το σχήμα (schema) αποθήκευσης των δεδομένων δημιουργείται αυτόματα από τον Cygnus-LD και τα

δεδομένα που κατανέμονται ως στήλες στο πίνακα είναι η ώρα λήψης της εκάστοτε νέας μέτρησης (recvTime), το αναγνωριστικό της οντότητας, ο τύπος της οντότητας και η νέα μέτρηση (τιμή) του χαρακτηριστικού hasSimpleResult.

Σε αυτό το σημείο, καθώς η λειτουργία ενός πράκτορα της υπηρεσίας Cygnus-LD έχει γίνει πιο κατανοητή, αξίζει να σημειωθεί ένα ακόμα θετικό που προσφέρει σε επίπεδο αρχιτεκτονικής. Όπως εξηγήθηκε στην αρχή της ενότητας ο πράκτορας αποτελείται από ένα ακροατή, ένα κανάλι και ένα “sink”. Το κανάλι λειτουργεί σαν προσωρινή αποθήκη δεδομένων που λαμβάνονται από τον ακροατή, το Sink αναλαμβάνει να «πάρει» δεδομένα που βρίσκονται προσωρινά στο κανάλι και να τα αποθηκεύσει στην εξωτερική βάση δεδομένων. Εφόσον τα δεδομένα αποθηκευτούν με επιτυχία στην εξωτερική βάση δεδομένων, διαγράφονται από το κανάλι, διαφορετικά τα δεδομένα συνεχίζουν να διατηρούνται στο κανάλι. Με αυτόν τον τρόπο, μία αποτυχημένη εγγραφή (πχ. λόγω καθυστέρησης δικτύου, λόγου προσωρινής υπερφόρτωσης της βάσης δεδομένων κλπ) μπορεί να επαναληφθεί χωρίς να χαθούν τα δεδομένα. Τέλος, το Cygnus-LD, χρησιμοποιεί κανάλι το οποίο τρέχει απευθείας στη μνήμη (MemoryChannel) και συμβάλλει στη γρήγορη μεταφορά δεδομένων που εισέρχονται, με αποτέλεσμα να υπερτερεί σε σχέση με άλλα που τρέχουν στο δίσκο.

3.4.3 Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Publish - Subscribe Service)

Στόχος της Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών είναι να επιτρέπει την αλληλεπίδραση διαφόρων οντοτήτων και να διαχειρίζεται δεδομένα πλαισίου. Η υπηρεσία απαρτίζεται από τον Orion-LD Context Broker και από μια μη-σχεσιακή βάση δεδομένων Orion-LD MongoDB.

Η επιλογή του Orion-LD Context Broker έγινε για τις ανάγκες του συστήματος να υποστηρίζεται η μορφοποίηση Διασυνδεδεμένων Δεδομένων ώστε να εκφραστούν σωστά μέσω της οντολογίας οι οντότητες του συστήματος και να δημιουργηθεί μια σύνδεση μεταξύ τους, διαμορφώνοντας έτσι έναν Ιστό από Διασυνδεδεμένα Δεδομένα. Το μοντέλο πληροφορίας NGSI-LD αποτελεί ένα JSON-LD shcema και υποστηρίζει τα παραπάνω, σε αντίθεση με ένα μοντέλο πληροφορίας NGSI2 (μοντέλο αναπαράστασης δεδομένων για Orion Context Broker). Ο Orion-LD παρέχει για την διαχείριση δεδομένων RESTful διεπαφές NGSI-LD, ενώ διαθέτει τις ίδιες λειτουργίες με τον κλασικό Orion CB, όπως :

- Εγγραφή οντοτήτων για παρακολούθηση των συμβάντων αυτών.
- Ενημέρωση των πληροφοριών των καταχωρημένων οντοτήτων.
- Ειδοποίηση όταν πραγματοποιούνται αλλαγές στις πληροφορίες των οντοτήτων.
- Ανάκτηση πληροφοριών των εγγεγραμμένων οντοτήτων.
- Διαγραφή οντοτήτων.

Η Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών δρα ως μεσολαβητής για τις παρακάτω οντότητες:

- “Αισθητήρων” (Sensors)
- “Μετρήσεων” (Observations)
- “Σπιτιών” (Homes)
- “Εφαρμογών” (Applications)
- “Εκτελέσεων ενέργειας” (Actuations)
- “Μετρούμενων ιδιοτήτων” (Observable properties)
- “Χωρικών οντοτήτων” (Spatial things)

Οι οντότητες που ανήκουν στις παραπάνω κατηγορίες αποθηκεύονται σε μορφή NGSI-LD (βλ. 2.6.6) στη βάση δεδομένων Orion-LD MongoDB (βλ. “Διαχείριση Δεδομένων”), που είναι συμβεβλημένη με την υπηρεσία Orion-LD Context Broker του FIWARE. Αυτό έπρεπε να γίνει με στόχο η πληροφορία που παρέχει η οντολογία για τις οντότητες του συστήματος να απεικονίζεται και στην υπηρεσία Publish-Subscribe, έτσι ώστε οι οντότητες του συστήματος που δημοσιεύονται από την υπηρεσία αυτή να ανήκουν και να περιγράφονται ταυτόχρονα από την οντολογία SSN.

Η υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών επικοινωνεί άμεσα με τις υπηρεσίες Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων, Οντολογίας και Διαχείρισης Ιστορικών Δεδομένων.

Στην πρώτη περίπτωση, όπως έχει ήδη σημειωθεί, για οποιαδήποτε ενέργεια στις οντότητες το αίτημα δρομολογείται πρώτα στην υπηρεσία Web Thing Model, η οποία με την σειρά της το προωθεί στην κατάλληλη υπηρεσία. Επομένως, λειτουργίες ανάκτησης / εισαγωγής / ενημέρωσης / διαγραφής οντοτήτων και συνδρομής δεν εκτελούνται άμεσα στην υπηρεσία Publish - Subscribe αλλά έμμεσα μέσω της υπηρεσίας Web Thing Model, γεγονός που δικαιολογεί και την σύνδεση των δύο υπηρεσιών.

Η υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών συνεργάζεται με την Οντολογία όταν υπάρχουν συνδρομές προς οντότητες του συστήματος. Συγκεκριμένα, οι πληροφορίες των οντοτήτων θα πρέπει να είναι ίδιες τόσο στην αποθηκευτική δομή του Orion-LD όσο και σε αυτή της οντολογίας. Επομένως, στην περίπτωση που υπάρξει κάποια αλλαγή στις τιμές των χαρακτηριστικών της οντότητας και εφόσον υπάρχει συνδρομή, πυροδοτείται μια ενημέρωση, από την υπηρεσία Publish-Subscribe, με περιεχόμενο αυτήν την αλλαγή και στην υπηρεσία της Οντολογίας, η οποία αποθηκεύει την νέα τιμή στο γράφο της.

Τέλος, η υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών επικοινωνεί με την υπηρεσία Διαχείρισης Ιστορικών Δεδομένων χάρη στην λειτουργία συνδρομής που προσφέρει ο Orion-LD. Η σύνδεση των δύο υπηρεσιών απαιτείται για την ροή και την αποθήκευση ιστορικών δεδομένων συσκευών. Έτσι η επικοινωνία μεταξύ Publish-Subscribe Service και History Service πραγματοποιείται μόνο όταν υπάρχουν νέες μετρήσεις.

Orion-LD Context Broker - Δημιουργία συνδρομής σε οντότητες

Ο Orion-LD Context Broker έχει - μεταξύ άλλων - μια πολύ σημαντική λειτουργία: μπορεί να δημιουργεί συνδρομή[22] σε συγκεκριμένες (ή συγκεκριμένου τύπου) οντότητες, γεγονός που του δίνει τη δυνατότητα να πυροδοτεί ειδοποιήσεις για οποιαδήποτε αλλαγή συμβεί στα χαρακτηριστικά (attributes) κάποιας οντότητας. Η εκάστοτε ενημέρωση θα αποσταλεί από την υπηρεσία Orion-LD Context Broker σε ένα URI που έχει προκαθοριστεί (από τον συνδρομητή), μέσω ενός REST αιτήματος της μεθόδου POST. Το αίτημα αυτό περιέχει στο σώμα του την πληροφορία της αλλαγής (των τιμών των χαρακτηριστικών), στη μορφή που ορίζει το πρότυπο NGSI-LD.

Η δυνατότητα αυτή αξιοποιείται τόσο από την Υπηρεσία Αποθήκευσης των Ιστορικών Δεδομένων όσο και από την Υπηρεσία Οντολογίας (με ξεχωριστή συνδρομή από κάθε υπηρεσία). Αμφότερες οι υπηρεσίες, δρώντας ως συνδρομητές, δέχονται ειδοποιήσεις σχετικά με τις αλλαγές στις μετρήσεις των αισθητήρων. Η πρώτη υπηρεσία αποθηκεύει τις αλλαγές αυτές στην ιστορική βάση δεδομένων του συστήματος, ενώ η δεύτερη τις αποθηκεύει στην οντολογία του συστήματος (κατά τη διαδικασία αυτή πραγματοποιείται και η διαδικασία του reasoning ώστε να ελεγχθούν οι νέες τιμές π.χ. σχετικά με τα όρια στα οποία κυμαίνονται). Έτσι, με κάθε αλλαγή στις τιμές των μετρήσεων, η ιστορική βάση προσθέτει τις νέες τιμές, διατηρώντας το ιστορικό μετρήσεων χρονικής σειράς, ενώ η οντολογία ενημερώνει τις υπάρχουσες τιμές που έχουν αποθηκευτεί για κάθε μέτρηση (δηλαδή κρατάει μόνο τις τρέχουσες τιμές, όπως συμβαίνει και στον Orion-LD Context Broker).

Για να επιτευχθεί η καθεμία από τις παραπάνω συνδρομές, απαιτείται η αποστολή ενός αιτήματος στην υπηρεσία Orion-LD Context Broker, το οποίο έχει ως αποτέλεσμα τη δημιουργία συνδρομής προς τις οντότητες των μετρήσεων των αισθητήρων. Για παράδειγμα, το αίτημα για να δημιουργηθεί η συνδρομή της Υπηρεσίας Αποθήκευσης των Ιστορικών Δεδομένων προς τις οντότητες των μετρήσεων είναι αυτό που φαίνεται στην εικόνα 3.12:

[22] https://documenter.getpostman.com/view/513743/SzfDvjN5?version=latest

```
{
  "description": "Device sub",
  "type": "Subscription",
  "entities": [
    {
      "id": "urn:ngsi-ld:Observation:[device_name]",
      "type": "Observation"
    }
  ],
  "watchedAttributes":["hasSimpleResult"],
  "notification": {
    "attributes": [
      "hasSimpleResult"
    ],
    "format": "keyValues",
    "endpoint": {
      "uri": "http://172.16.1.15:5050/notify",
      "accept": "application/ld+json"
    }
  },
  "@context": "https://fiware.github.io/data-models/context.jsonld"
}
```

***Εικόνα 3.12**: Στιγμιότυπο της συνδρομής της Υπηρεσίας Αποθήκευσης των Ιστορικών Δεδομένων προς τις οντότητες των μετρήσεων των αισθητήρων (σε μορφή NGSI-LD).*

Η παραπάνω αναπαράσταση NGSI-LD αποτελείται από τα ακόλουθα πεδία:

- **description:** Μια σύντομη περιγραφή της συνδρομής.
- **type:** Λειτουργία συνδρομής.
- **entities:** Περιέχει τις οντότητες στις οποίες δημιουργείται η συνδρομή και τον τύπο τους.
- **watchedAttributes:** Υποδηλώνει ποια χαρακτηριστικά της οντότητας παρακολουθούνται.
- **notification:** Υποδηλώνει τη συνθήκη αλλαγής της οντότητας και για ποιο ή για ποια χαρακτηριστικά μας ενδιαφέρει να λαμβάνουμε ενημερώσεις σχετικά με τις αλλαγές στα δεδομένα τους. Στην παρούσα περίπτωση, ο τύπος “Observation” εκπροσωπεί πάντα μια συγκεκριμένου είδους μέτρηση και αυτό που μας ενδιαφέρει είναι το attribute “hasSimpleResult” που κρατάει την τιμή της μέτρησης.
- **uri:** Υποδηλώνει το σύνδεσμο στον οποίο θα γίνει η αποστολή των ειδοποιήσεων.
- **@context:** Εισάγεται σε οντότητες για να παρέχει πλήρως αναγνωρισμένα ονόματα (URI) που σχετίζονται με όρους.

3.4.4 Υπηρεσία Οντολογίας (Ontology Service)

Η Υπηρεσία Οντολογίας είναι υπεύθυνη για την αλληλεπίδραση της οντολογίας του συστήματος με τις υπόλοιπες υπηρεσίες της πλατφόρμας. Το έργο της οντολογίας είναι πολύ σημαντικό, αφού προσδίδει εκφραστικότητα στις συσκευές

που είναι συνδεδεμένες στο διαδίκτυο προσφέροντας με αυτό τον τρόπο στις μηχανές την δυνατότητα να κατανοούν την σημασία τους και να αντιλαμβάνονται σωστά τις μετρήσεις που παίρνουν. Η υπηρεσία αποτελείται από το Σύστημα Διαχείρισης Βάσεων Δεδομένων της Virtuoso, στο οποίο φιλοξενείται ο γράφος της οντολογίας, καθώς και την διεπαφή προγραμματισμού εφαρμογών Jena API, η οποία μεσολαβεί σε κάθε επικοινωνία της οντολογίας με το υπόλοιπο σύστημα.

Η διεπαφή Jena API είναι το εργαλείο που επιτρέπει την ανάκτηση, την προσθήκη, την ενημέρωση αλλά και τη διαγραφή δεδομένων από την οντολογία, καθώς και την αξιοποίηση κάποιου μηχανισμού αιτίασης. Στη παρούσα αρχιτεκτονική χρησιμοποιείται ο μηχανισμός συλλογισμού **Pellet Reasoner**[23] ο οποίος χρησιμοποιείται σε συνδυασμό με τις βιβλιοθήκες του Apache Jena και αξιοποιείται για την εύρεση ασυνεπειών αλλά και για την εξαγωγή νέων πληροφοριών από τις πληροφορίες που αντιπροσωπεύονται σε κάποια οντολογία. Μια πιο αναλυτική περιγραφή του μηχανισμού αιτίασης βρίσκεται στη διπλωματική εργασία iSWoT ([2]).

Για τις ανάγκες περιγραφής των συσκευών / αισθητήρων επιλέχθηκε η οντολογία SSN, η οποία παρέχει μεγαλύτερη εκφραστικότητα στην περιγραφή των οντοτήτων. Αυτό συμβαίνει επειδή η οντολογία SSN περιέχει περισσότερες κλάσεις και ιδιότητες, επομένως είναι ικανή να περιγράψει περισσότερες έννοιες και οντότητες (και με μεγαλύτερη ακρίβεια).

Η υπηρεσία της Οντολογίας αλληλεπιδρά άμεσα με την υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) στις περιπτώσεις ανάκτησης / δημιουργίας / διαγραφής / ενημέρωσης οποιασδήποτε εικονικής οντότητας του συστήματος. Αυτό συμβαίνει γιατί όπως έχει ήδη αναφερθεί, όλες οι εγγεγραμμένες οντότητες ενός κόμβου είναι αποθηκευμένες στο γράφο της οντολογίας, επομένως θα πρέπει να ενημερώνεται για οποιαδήποτε τροποποίηση. Επιπλέον, η υπηρεσία της Οντολογίας αλληλεπιδρά και με την υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Publish - Subscribe Service), στη περίπτωση που μια συσκευή λάβει νέα μέτρηση και μέσω της λειτουργίας συνδρομής πυροδοτείται η αλλαγή της από τον Orion-LD. Η αλλαγή αυτή θα πρέπει να σημειωθεί τόσο στη βάση δεδομένων του Orion-LD όσο και στη βάση δεδομένων της οντολογίας.

Τέλος, η υπηρεσία της Οντολογίας σχετίζεται με την υπηρεσία Σύνθεσης Εφαρμογών (Mashup Service). Η δημιουργία σημασιολογικών εφαρμογών απαιτεί την συνεργασία των δύο υπηρεσιών. Η ιδέα κατασκευής εφαρμογών σημασιολογικού τύπου, βασίζεται στη λήψη των τελευταίων (τρεχουσών) μετρήσεων των αισθητήρων που χρησιμοποιούνται στην εφαρμογή και στην ερμηνεία αυτών μέσω του reasoner Pellet. Η περιγραφή και η υλοποίηση για την ερμηνεία των παραπάνω μετρήσεων και για την εξαγωγή συμπερασμάτων αποτελεί έργο προηγούμενης διπλωματικής εργασίας ([2]).

[23] https://www.w3.org/2001/sw/wiki/Pellet

3.4.5 Υπηρεσία Σύνθεσης Εφαρμογών (Mashup Service)

Η υπηρεσία Mashup καθιστά δυνατή τη δημιουργία εφαρμογών, που είναι υποκείμενες στο Σημασιολογικό Ιστό, από προγραμματιστές του συστήματος. Ο προγραμματιστής μέσω γραφικής διεπαφής μπορεί να επιλέξει το είδος της εφαρμογής, δηλαδή αν η εφαρμογή αφορά ένα σπίτι ή μία πόλη από αυτές που υποστηρίζει το σύστημα. Στη συνέχεια επιλέγει για ποιο σπίτι ή πόλη συγκεκριμένα επιθυμεί να δει πληροφορίες, δηλαδή, να δημιουργηθεί μια εφαρμογή για αυτό. Εν τέλει, επιλέγει αν η εφαρμογή θα είναι τύπου ιστορικών δεδομένων ή σημασιολογικού τύπου τρεχουσών τιμών.

Προκειμένου να παραχθεί η εκτελέσιμη εφαρμογή, η λειτουργικότητα της όπως την διαμόρφωσε ο προγραμματιστής, περιγράφεται σε μία πρότυπη δομή JSON. Στην εικόνα 3.13 φαίνεται μια δομή JSON για εφαρμογή πόλης ενώ στην εικόνα 3.14 εφαρμογή σπιτιού.

```
{
    "appname": "AthensApp",
    "appscope": "weather",
    "info": [
        {
            "attributes": [
                "temperautre",
                "humidity"
            ],
            "city": "Athens",
            "domains": [
                "WaterSport",
                "Weather",
                "Garment"
            ],
            "func": "LIVE"
        }
    ]
}
```

Εικόνα 3.13 *: Αναπαράσταση των mashup μίας εφαρμογής πόλης σε JSON.*

```
{
    "appname": "Appartment100App",
    "appscope": "home",
    "info": [
        {
            "attributes": [
                "luminocity"
            ],
            "city": "Appartment100",
            "domains": [],
            "func": "MAX_24hr"
        }
    ]
}
```

Εικόνα 3.14 *: Αναπαράσταση των mashup μίας εφαρμογής σπιτιού σε JSON.*

Τα παραπάνω χαρακτηριστικά που φαίνονται στις εικόνες, συνθέτουν την συνολική πληροφορία μιας εκτελέσιμης εφαρμογής. Αναλυτικά, τα χαρακτηριστικά έχουν την εξής σημασία :

- **appname** : Το όνομα της εφαρμογής.
- **appscope** : Ο τύπος της εφαρμογής, δηλαδή αν αφορά ένα σπίτι ή μια πόλη (weather για πόλη, home για σπίτι).
- **info** : Πίνακας ο οποίος περιέχει το σύνολο των mashup της εφαρμογής.
- **attributes** : Πρόκειται για το υποσύνολο των χαρακτηριστικών μέτρησης αισθητήρων που ο χρήστης μπορεί να χρησιμοποιήσει (μέσω συνδρομής) και για τα οποία ενδιαφέρεται να λάβει πληροφορία. Τα χαρακτηριστικά αυτά περιλαμβάνονται στο mashup, το οποίο θα απεικονίζει μία σύνθεση από δεδομένα τους.
- **city** : Αφορά το όνομα (id) του σπιτιού ή της πόλης για το οποίο μας ενδιαφέρει να λάβουμε πληροφορίες από τους αισθητήρες που έχουμε επιλέξει και μπορούμε να χρησιμοποιήσουμε.
- **domains** : Στην περίπτωση που η εφαρμογή αφορά ταυτόχρονα πόλη και τρέχουσες τιμές, ο χρήστης μπορεί να ρωτήσει για έναν τομέα της καθημερινότητας και να του απεικονιστεί συμβουλή, η οποία παράγεται ανάλογα με την τιμή που έχουν οι αισθητήρες που μετρούν τα χαρακτηριστικά που επέλεξε.
- **func** : Πρόκειται για τον «τύπο» του mashup, μία κωδική λέξη (συμβολοσειρά) η οποία ορίζει τι κατάσταση (ή συμβάν) απεικονίζει το mashup με την σύνθεση των δεδομένων. Κάθε διαφορετικός «τύπος» mashup που υποστηρίζει το σύστημα αντιστοιχείται με μία -μοναδική- κωδική λέξη όπως βλέπουμε παρακάτω:
    - **MAX_24hr** : Γράφημα γραμμής που απεικονίζει τη μέγιστη τιμή του χαρακτηριστικού ανά ώρα, για τις τελευταίες 24 ώρες.
    - **MIN_24hr** : Γράφημα γραμμής που απεικονίζει την ελάχιστη τιμή του χαρακτηριστικού ανά ώρα, για τις τελευταίες 24 ώρες.
    - **AVERAGE_24hr** : Γράφημα γραμμής που απεικονίζει τη μέση τιμή του χαρακτηριστικού ανά ώρα, για τις τελευταίες 24 ώρες.
    - **MAX_7D** : Γράφημα γραμμής που απεικονίζει τη μέγιστη τιμή του χαρακτηριστικού ανά μέρα, για τις τελευταίες 7 μέρες.
    - **MIN_7D** : Γράφημα γραμμής που απεικονίζει την ελάχιστη τιμή του χαρακτηριστικού ανά μέρα.
    - **AVERAGE_7D** : Γράφημα γραμμής που απεικονίζει τη μέση τιμή του χαρακτηριστικού ανά μέρα, για τις τελευταίες 7 μέρες.
    - **LIVE** : Έχουμε κάνει την επιλογή αυτή η κωδική λέξη να δηλώνει ότι η εφαρμογή είναι σημασιολογικού περιεχομένου, ενώ όλες οι προηγούμενες αποτελούν ιστορικές εφαρμογές. Με αυτό τον τρόπο, αναπαριστάται η σημασιολογική ερμηνεία της οντολογίας πάνω στις τρέχουσες τιμές των χαρακτηριστικών (attributes) της πόλης που επιλέχθηκε. Επίσης, αναπαριστώνται συμβουλές ανάλογα με τις επιλογές που έγιναν στο πεδίο **domains**. Μια εκτενέστερη περιγραφή

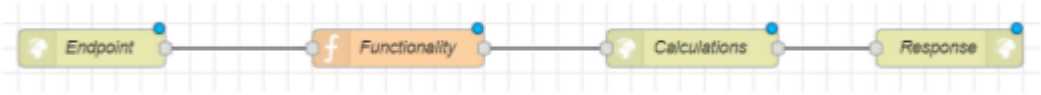
για την συγκεκριμένη κωδική λέξη βρίσκεται σε προηγούμενη διπλωματική εργασία ([2]).

Κάθε διαφορετική κωδική λέξη από τις παραπάνω, αντιστοιχίζεται σε έναν συγκεκριμένο αλγόριθμο που υλοποιεί το mashup της αντίστοιχης κωδικής λέξης.

Εντός της υπηρεσίας Mashup του συστήματος, υλοποιείται η σύνθεση των επιμέρους υπηρεσιών Node-RED και Calculations, οι οποίες συνεργάζονται για την παραγωγή του τελικού αποτελέσματος, δηλαδή την δημιουργία εκτελέσιμης εφαρμογής. Στη συνεχεια ακολουθει μια αναλυτική περιγραφή των δυο προαναφερθέντων υπηρεσιών.

Node-RED

Η υπηρεσία Node-RED έχει τον πρωταρχικό ρόλο για την δημιουργία εκτελέσιμων εφαρμογών από προγραμματιστές, παρέχοντας παράλληλα ένα runtime περιβάλλον που επιτρέπει την εκτέλεση τους, από τελικούς χρήστες του συστήματος. Εντός της υπηρεσίας Node-RED, η εφαρμογή υφίσταται σαν σύνθεση τεσσάρων κόμβων (nodes) που ανταλλάσσουν δεδομένα σε προκαθορισμένες συνδέσεις (εικόνα 3.15).



Εικόνα 3.15 *: Πρότυπο ροής (flow) σύνθεσης εκτελέσιμης εφαρμογής.*

Οι κόμβοι εκτελούνται με προκαθορισμένη σειρά ροής (flow) και η σύνθεση τους σαν σύνολο αποτελεί την εκτελέσιμη εφαρμογή. Ουσιαστικά, κάθε εφαρμογή υφισταται αρχικά σαν μια ροή (flow) η οποία αποθηκεύεται στην βάση αποθήκευσης εφαρμογών της υπηρεσίας Node-RED, όπου μετά είναι έτοιμη προς εκτέλεση. Κάθε κόμβος έχει μία συγκεκριμένη λειτουργικότητα. Παρακάτω περιγράφεται συνοπτικά ο ρόλος κάθε κόμβου της ροής.

1. Ο πρώτος κόμβος “Endpoint”, αποτελεί ένα ακροατή (Listener) ο οποίος δέχεται HTTP αιτήματα και αποτελεί το σημείο επαφής (endpoint) της εφαρμογής. Με την προσπέλαση του τελικού σημείου της εφαρμογής η εκτέλεση της ροής προχωράει στο δεύτερο κόμβο.
2. Ο δεύτερος κόμβος “Functionality” περιέχει αποθηκευμένη την πληροφορία JSON που αναπαριστά τα mashup της εφαρμογής (π.χ. εικόνα 3.13 και 3.14). Σκοπός αυτού του κόμβου είναι να μεταδώσει την πληροφορία JSON στον επόμενο κόμβο της ροής “Calculations”.
3. Ο Τρίτος κόμβος “Calculations” δέχεται σαν είσοδο την πληροφορία JSON από τον κόμβο “Functionality”. Σε αυτό το στάδιο, εκτελούνται οι αλγόριθμοι οι οποίοι υλοποιούν τα mashup που ορίζει η πληροφορία JSON που ελήφθη στην είσοδο και συγκεκριμένα της τιμής που περιέχεται στο πεδίο **“func”**.
4. Η υπηρεσία που είναι υπεύθυνη για την λειτουργία του συγκεκριμένου κόμβο αναλύεται παρακάτω. Όταν οι υπολογισμοί ολοκληρωθούν, παράγεται στην

έξοδο του κόμβου ο προσαρμοσμένος κώδικας HTML/Javascript που περιέχει την γραφική αναπαράσταση των mashup της εφαρμογής. Ο κώδικας αυτός προωθείται στον επόμενο κόμβο της ροής “Response”.

5. Ο τελικός κόμβος “Response” προωθεί τον κώδικα -HTML/Javascript- που έλαβε στην είσοδο, στο πρόγραμμα περιήγησης του τελικού χρήστη, ο οποίος λαμβάνει την πληροφορία της εφαρμογής που προσπέλασε.

Επομένως, μέσω ενός REST αιτήματος προς το Node-RED, μπορούμε είτε να δημιουργήσουμε μία εκτελέσιμη εφαρμογή περιγράφοντας στο σώμα του αιτήματος, τους κόμβους που θα συνθέτουν την εφαρμογή είτε να την ανακτήσουμε περιλαμβάνοντας το τελικό σημείο επαφής της (Endpoint).

Calculations

Πρόκειται για ένα διακομιστή Apache ο οποίος περιέχει όλους τους αλγορίθμους (σε γλώσσα PHP) που υλοποιούν τα διαφορετικά mashup δεδομένων που υποστηρίζει το σύστημα. Είναι προγραμματισμένος ώστε να εκτελεί την παρακάτω διαδικασία:

1. Δέχεται HTTP αιτήματα της μεθόδου POST, με σώμα -αιτήματος- την πρότυπη δομή JSON η οποία περιγράφει τα mashup που περιλαμβάνει μία εφαρμογή. Τα αιτήματα αυτά έρχονται από την υπηρεσία Node-RED και συγκεκριμένα από τον τρίτο κόμβο με όνομα Calculations, ο οποίος αναλύθηκε προηγουμένως (υποενότητα Node-RED).
2. Απαντάει στην έξοδο του, τον προσαρμοσμένο κώδικα HTML/Javascript ο οποίος σχεδιάζει την γραφική αναπαράσταση των mashup που περιγράφονται στην δομή JSON που έλαβε στην είσοδο. Η έξοδος του είναι πάλι ο τρίτος κόμβος Calculations του Node-RED ο οποίος στη συνέχεια προωθεί την πληροφορία στον τελευταίο κόμβο Response.

Με την άφιξη ενός αιτήματος γίνεται ανάλυση της δομής JSON. Για κάθε διαφορετικό mashup που ζητείται θα καλεστεί ο κατάλληλος αλγόριθμος για τον υπολογισμό του με βάση τα πεδία “appscope” και “func” που περιέχει. Ο αλγόριθμος θα λάβει ως ορίσματα εισόδου:

1. Την λίστα των μετρήσιμων χαρακτηριστικών “attributes” (πχ. Temperature, humidity) που περιλαμβάνονται στο mashup,
2. Το σπίτι ή πόλη στο οποίο ανήκουν τα παραπάνω χαρακτηριστικά “city” (πχ. Athens, Room145) που θα απεικονίζει το mashup (βλ. παρακάτω).
3. Τομείς της καθημερινότητας που ο χρήστης ενδιαφέρεται να λάβει απάντηση (**domains**) .

Ουσιαστικά η υπηρεσία Calculations αναλύει την περιγραφή mashup της εφαρμογής, αποφασίζει τι είδους εφαρμογή είναι και ποιες υπηρεσίες θα χρησιμοποιηθούν στη συνέχεια για την εκτέλεση της εφαρμογής. Πιο συγκεκριμένα, αν πρόκειται για ιστορική εφαρμογή εκτελούνται τα κατάλληλα SQL ερωτήματα στην ιστορική βάση δεδομένων PostgreSQL (βλ. Πίνακα 3.1), ενώ αν πρόκειται για σημασιολογική

εφαρμογή (τιμή πεδίου func:LIVE) στέλνονται τα κατάλληλα αιτήματα στην υπηρεσία οντολογίας. Στην δεύτερη περίπτωση, η υπηρεσία της οντολογίας εκτελεί τα ακόλουθα βήματα :

1. Η υπηρεσία Jena API ζητάει από την υπηρεσία Virtuoso τον γράφο της οντολογίας.
2. Μόλις πάρει τον γράφο, για κάθε χαρακτηριστικό και τομέα της καθημερινότητας (αν πρόκειται για πόλη) που βρίσκονται στα πεδία περιγραφής της εφαρμογής, εκτελείται συλλογισμός μέσω του Pellet reasoner και με SPARQL ερωτήματα επιστρέφονται αποτελέσματα με σημασιολογική έννοια και όχι νούμερα .

Εφόσον η εκτέλεση των αλγορίθμων ολοκληρωθεί, ο διακομιστής Calculations επιστρέφει στην έξοδο του ένα προσαρμοσμένο κώδικα HTML/Javascript. Ο κώδικας αυτός είναι προσαρμοσμένος ώστε να σχεδιάζει γραφικά τα αποτελέσματα των αλγορίθμων σε διαγράμματα (όπου έχει ζητηθεί). Για την γραφική απεικόνιση ενός διαγράμματος γίνεται χρήση της βιβλιοθήκης **Google Charts**[24].

Επομένως,κατά την δημιουργία μιας εφαρμογής από ένα προγραμματιστή, ανεξαρτήτως του τύπου της, παράγεται η σύνθεση κόμβων που υλοποιεί την εκτελέσιμη εφαρμογή και αποθηκεύεται στην βάση δεδομένων (MongoDB) της υπηρεσίας Mashup σαν μια ροή (flow) του Node-RED. Στη συνέχεια η εφαρμογή είναι διαθέσιμη στους τελικούς χρήστες του συστήματος. Η υπηρεσία Calculation αναλαμβάνει την διαφοροποίηση των τύπων των εφαρμογών, αναλύοντας την περιγραφή JSON της κάθε εφαρμογής και δρομολογώντας τα κατάλληλα αιτήματα στις υπηρεσίες (οντολογίας ή ιστορική) που πρέπει.

3.4.6 Directory Service

Η συγκεκριμένη υπηρεσία είναι υπεύθυνη για την ανακάλυψη των κόμβων, που είναι εγγεγραμμένοι στο σύστημα του Fi-SWoT. Ουσιαστικά δίνει την δυνατότητα σε χρήστες τοπικών κόμβων να αναζητήσουν απομακρυσμένους κόμβους του δικτύου και να δημιουργήσουν συνδρομή. Επίσης, μέσω του Directory Service ένας χρήστης μπορεί να εκτελέσει ενέργειες σε ξένο κόμβο, εφόσον έχει εγγραφεί και ανήκει στους έγκυρους χρήστες του. Μια πιο αναλυτική περιγραφή της υπηρεσίας ακολουθεί στην παράγραφο 3.6.2.

3.4.7 Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών (User Authentication - Authorization)

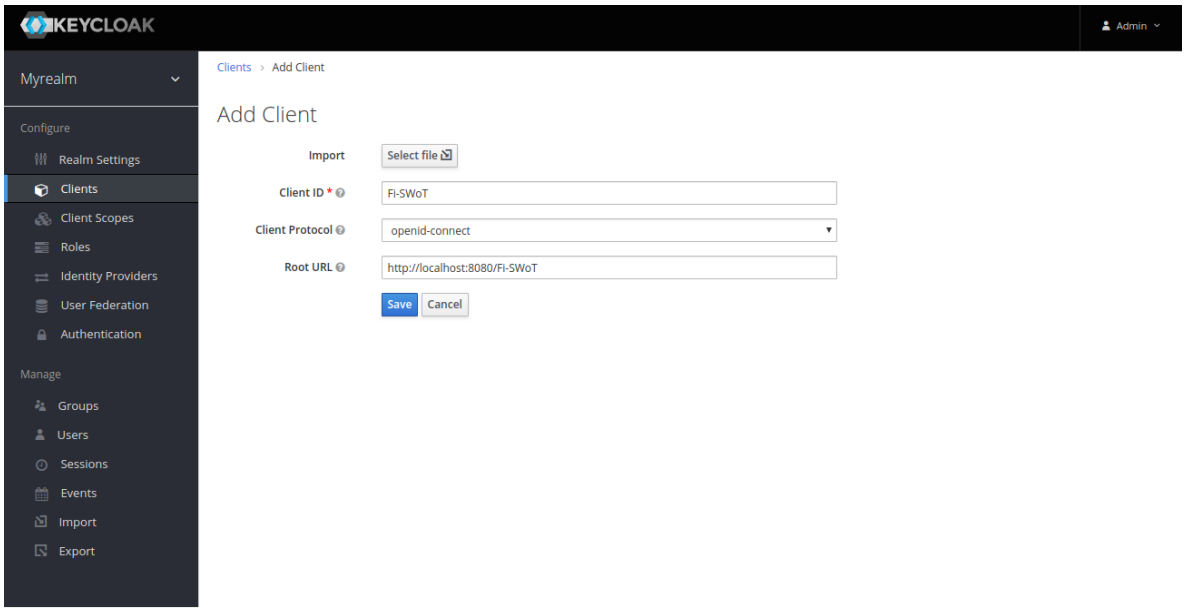
Η υπηρεσία αυτή συνιστά αφετηρία για το σύστημα, καθώς αναλαμβάνει την εγγραφή και τη σύνδεση των χρηστών. Κατά την εγγραφή του χρήστη, ορίζονται τα χαρακτηριστικά που συνθέτουν το προσωπικό του προφίλ όπως όνομα, email, κωδικό πρόσβασης, ρόλος του χρήστη. Η υπηρεσία παρέχει τις εξής λειτουργίες :

[24] https://developers.google.com/chart/

1. Την εγγραφή του συστήματος από κάποιο διαχειριστή (administrator) στην υπηρεσία Keycloak.
2. Την εγγραφή ενός νέου χρήστη στο σύστημα ούτως ώστε να έχει πρόσβαση σε αυτό.
3. Τη διαδικασία ταυτοποίησης ενός διαχειριστή ή χρήστη μέσω της εν λόγω υπηρεσίας.

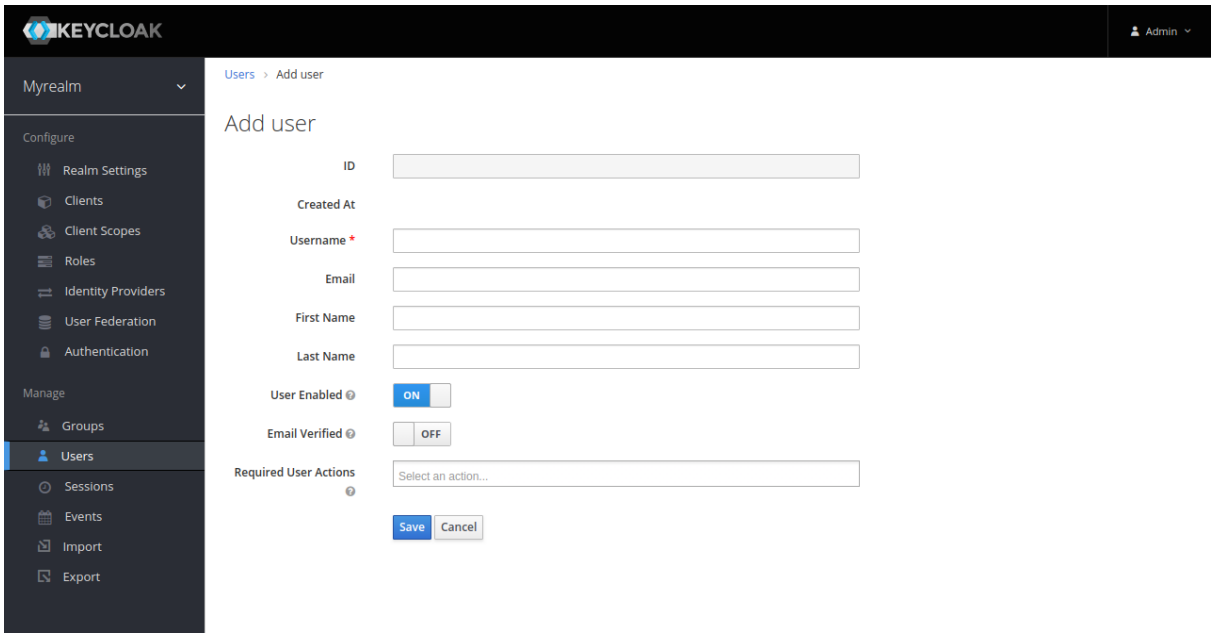
Ακολουθεί μια αναλυτική περιγραφή των παραπάνω λειτουργιών:

1. Αρχικά, πραγματοποιείται η διαδικασία εγγραφής του συστήματος στην υπηρεσία Keycloak IDM από το διαχειριστή. Συγκεκριμένα, ο πίνακας ελέγχου της υπηρεσίας Keycloak παρέχει ένα γραφικό περιβάλλον (Εικόνα 3.16), στο οποίο μπορεί εύκολα να επιτευχθεί η εγγραφή, με αποτέλεσμα την αυτόματη παραγωγή δύο μοναδικών αναγνωριστικών που σχετίζονται με την ταυτοποίηση (client_id και client_secret). Τα δύο αυτά αναγνωριστικά συνιστούν για την υπηρεσία Keycloak την ταυτότητα του συστήματος. Επομένως, κάθε αίτημα σύνδεσης στο σύστημα πρέπει οπωσδήποτε να περιλαμβάνει στο σώμα αιτήματος (request body) του τα παραπάνω αναγνωριστικά.



Εικόνα 3.16 *: Στιγμιότυπο από το γραφικό περιβάλλον όπου γίνεται η εγγραφή του συστήματος.*

2. Στη συνέχεια, κάποιος χρήστης που επιθυμεί να αποκτήσει πρόσβαση στο σύστημα, θα πρέπει πρωτίστως να δημιουργήσει ένα νέο λογαριασμό στην υπηρεσία Keyrock IdM. Το γραφικό περιβάλλον, όπου γίνεται η εγγραφή (sign up) ενός νέου χρήστη, από το διαχειριστή του συστήματος, παρέχεται από τον πίνακα ελέγχου της υπηρεσίας Keycloak και φαίνεται στην εικόνα 3.17.



Εικόνα 3.17 *: Στιγμιότυπο από το γραφικό περιβάλλον όπου γίνεται η εγγραφή ενός χρήστη.*

3. Τέλος, οποιοσδήποτε χρήστης επιθυμεί να εισέλθει στο σύστημα, πρέπει να περάσει επιτυχώς από τη διαδικασία ταυτοποίησης. Αυτή πραγματοποιείται κατά το αίτημα εισόδου του χρήστη στο σύστημα, με το οποίο δίνει τα απαραίτητα στοιχεία του λογαριασμού του (e-mail και password). Ειδικότερα, η Λογική Εφαρμογής (Application Logic) πραγματοποιεί ένα αίτημα REST στην υπηρεσία Keycloak IdM. Το σώμα του αιτήματος (request body) περιέχει τα δύο αναγνωριστικά (client_id,client_secret) και τα στοιχεία εισόδου του χρήστη, προκειμένου να γίνει η ταυτοποίηση. Μόλις αυτή ολοκληρωθεί επιτυχώς, επιστρέφεται ένα OAuth2 token από την υπηρεσία Keycloak IdM στη Λογική Εφαρμογής και η είσοδος του χρήστη πραγματοποιείται με επιτυχία. Αμέσως μετά, δημιουργείται στον εξυπηρετητή του χρήστη μια συνεδρία (session) που αποθηκεύει το OAuth2 token του χρήστη κι έτσι ο χρήστης μπορεί να παραμείνει και να περιηγηθεί σε κάθε σελίδα της Διαδικτυακής Εφαρμογής του Συστήματος για όλο το διάστημα κατά το οποίο η συνεδρία παραμένει ενεργή (μέχρι δηλαδή ο χρήστης να αποσυνδεθεί). Στην παρούσα εργασία, η λειτουργία αυτή υλοποιήθηκε με τη χρήση της μεταβλητής $_SESSION [OAuth2_token] (στη γλώσσα προγραμματισμού php), η οποία αποθήκευσε την τιμή του token.

Στον πίνακα που ακολουθεί (REST table) γίνεται μια περιγραφή βασικών HTTP μεθόδων της υπηρεσίας Keycloak IdM που χρησιμοποιήθηκαν στη παρούσα εργασία:

Μέθοδος	URL Μεθόδου	Header	Σώμα αιτήματος	Περιγραφή μεθόδου
POST	/ auth/realms/myrealm/protocol/openid-connect/token	-	{ grant_type=password &username=”USERNAME” &password=”PASSWORD” &scope=openid &client_id=”client_id” &client_secret=”client_secret” }	Δίνεται ένα έγκυρο username, password και επιστρέφεται ένα OAuth2 token + Id Token.
POST	/ auth/realms/myrealm/protocol/openid-connect/token	-	{ grant_type=password &username=”adminusername” &password=”adminpassword” &scope=openid &client_id=”client_id” &client_secret=”client_secret” }	Δίνονται στοιχεία διαχειριστή. Επιστρέφεται το αναγνωριστικό πρόσβασης : X_subj_token .
POST	/auth/realms/myrealm/ protocol/openid-connect/userinfo	-	{ 'access_token=”user_access_token” }	Δίνεται αναγνωριστικό πρόσβασης ενός χρήστη και επιστρέφονται πληροφορίες για αυτόν.
GET	/auth/admin/realms/ myrealm/clients/service_name/roles/role_name	Authorization: Bearer: X_subj_token	-	Η μέθοδος ελέγχει αν υπάρχει ένας συγκεκριμένος ρόλος στην υπηρεσία.
POST	/auth/admin/realms /myrealm/clients /service_name/roles	Authorization: Bearer: X_subj_token	{ “name”: “role_name” }	Εισαγωγή νέου ρόλου στην υπηρεσία.
POST	/auth/admin/realms/ myrealm/users/user_id /role-mappings/clients /service_name	Authorization: Bearer: X_subj_token	{ "name":"role_name", "id":"'.role_id”, "clientRole": true, "composite": false, “containerId”:”service_name” }	Ανάθεση ρόλου σε χρήστη της υπηρεσίας (role mapping).
DELETE	/auth/admin/realms/ myrealm/users/user_id	Authorization: Bearer: X_subj_token	-	Διαγραφή ενός χρήστη από την υπηρεσία.

***Πίνακας 3.2**: HTTP μέθοδοι της υπηρεσίας Keycloak IDM.*

3.4.8 Υπηρεσία Λήψης Απόφασης Εξουσιοδότησης (Authorization Policy Decision Point)

Η υπηρεσία αυτή ευθύνεται για την λήψη αποφάσεων έγκρισης ή απόρριψης αιτημάτων πρόσβασης που γίνονται από χρήστες προς υπηρεσίες του συστήματος, με κριτήριο τις εφαρμοστέες πολιτικές πρόσβασης που συνιστούν οι ρόλοι εντός της υπηρεσίας Ταυτοποίησης και Εξουσιοδότησης Χρηστών.

Οι αποφάσεις έγκρισης η απόρριψης αιτημάτων πρόσβασης, προκύπτουν από τους κανόνες ελέγχου πρόσβασης. Ένας κανόνας ελέγχου πρόσβασης σχετίζεται άρρηκτα με ένα ρόλο εντός της υπηρεσίας Ταυτοποίησης και Εξουσιοδότησης Χρηστών καθώς περιγράφει τις προϋποθέσεις που πρέπει να ακολουθεί το αίτημα πρόσβασης οποιουδήποτε χρήστη με αυτό το ρόλο, προκειμένου εγκριθεί.

Οι κανόνες ελέγχου πρόσβασης ακολουθούν το πρότυπο XACML και δημιουργούνται κατά την εγγραφή ενός νέου χρήστη στο σύστημα. Ανάλογα την κατηγορία στην οποία επιλέγει ο χρήστης να ανήκει (Infrastructure Owner ή Customer), δημιουργείται και το αντίστοιχο XACML αρχείο που ενσωματώνει άδειες πρόσβασης σε πόρους. Με αυτόν τον τρόπο, τα αιτήματα των χρηστών για πρόσβαση σε υπηρεσίες του συστήματος, προωθούνται πρώτα για αξιολόγηση στην υπηρεσία Λήψης Απόφασης Εξουσιοδότησης.

Ακολουθεί ένας πίνακας με τις περιγραφές των HTTP μεθόδων της υπηρεσίας AuthZForce που χρησιμοποιήθηκαν στη παρούσα εργασία:

Μέθοδος	URL Μεθόδου	Σώμα αιτήματος	Περιγραφή μεθόδου
POST	/pap/policies	αρχείο xml	Δημιουργία νέου XACML με πολιτικές πρόσβασης
POST	/pdp	αρχείο xml με το ζητούμενο πόρο	Έλεγχος πρόσβασης σε πόρο του συστήματος
DELETE	/ pap/policies/rolename."/1	-	Διαγραφή XACML αρχείου που σχετίζεται με ένα ρόλο

Πίνακας 3.3: *HTTP μέθοδοι της υπηρεσίας AuthZForce.*

3.4.9 Διακομιστής Μεσολάβησης Επιβολής Πολιτικής (Policy Enforcement Point Proxy)

Ένας διακομιστής μεσολάβησης είναι ένας διακομιστής που ενεργεί ως ενδιάμεσος για τις αιτήσεις από χρήστες που αναζητούν πόρους από άλλους διακομιστές. Ένας χρήστης συνδέεται με τον διακομιστή μεσολάβησης, ζητώντας κάποια υπηρεσία, όπως ένα αρχείο, μια σύνδεση, μια ιστοσελίδα ή άλλο πόρο που διατίθεται από διαφορετικό διακομιστή. Ο διακομιστής μεσολάβησης αναλαμβάνει να προωθήσει το αίτημα στον διακομιστή που διαθέτει την συγκεκριμένη υπηρεσία και αφού λάβει την απάντηση του ερωτήματος την προωθεί στον αιτούντα. Οι υπηρεσίες της αρχιτεκτονικής που διαθέτουν πόρους οι οποίοι δεν είναι θεμιτό να είναι προσβάσιμοι από μη εξουσιοδοτημένες υπηρεσίες ή χρήστες, δεν προσφέρουν δημόσια την REST διεπαφή τους. Επομένως, κάθε υπηρεσία προκειμένου να μπορεί να εξυπηρετεί εξωτερικά (Σε σχέση με την εικονική μηχανή στην οποία «τρέχει») αιτήματα, διαθέτει ένα τοπικό διακομιστή μεσολάβησης ο οποίος αναλαμβάνει να δέχεται στο δημόσιο τελικό του σημείο τα αιτήματα που προορίζονται για εκείνη και της τα προωθεί.

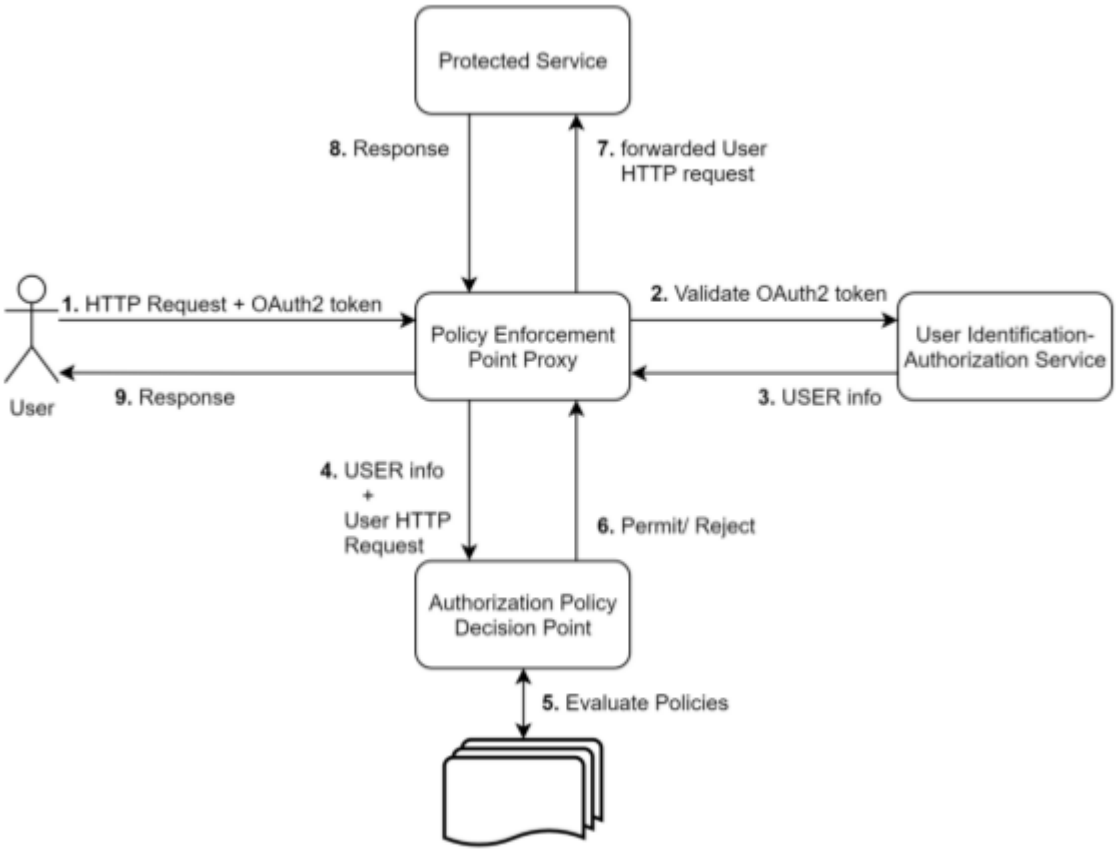
Ο Διακομιστής Μεσολάβησης Επιβολής Πολιτικής, είναι ένας διακομιστής μεσολάβησης που απαιτεί υποχρεωτικά στην κεφαλίδα των HTTP αιτημάτων που λαμβάνει, ένα από τα δύο παρακάτω διακριτικά, διαφορετικά τα αιτήματα αγνοούνται:

- OAuth2 token: Ένα έγκυρο OAuth2 token αντιστοιχεί σε ένα χρήστη και έχει δημιουργηθεί από την υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών κατά την είσοδο του στο σύστημα.
- Master Key: Πρόκειται για ένα μυστικό κωδικό ο οποίος καθορίζεται κατά την αρχικοποίηση του Διακομιστή Μεσολάβησης Επιβολής Πολιτικής. Κάθε διαφορετικός Διακομιστής Μεσολάβησης Επιβολής Πολιτικής της αρχιτεκτονικής έχει το δικό του -μοναδικό- κωδικό Master key.

Στην περίπτωση όπου το αίτημα που δέχεται, φέρει στην κεφαλίδα του το μυστικό κωδικό “Master Key”, εφόσον ο κωδικός Master Key είναι σωστός, ο Διακομιστής PEP το προωθεί στην υπηρεσία για την οποία μεσολαβεί και επιστρέφει την απάντηση της στον αιτούντα (Εικόνα 3.18).

***Εικόνα 3.18**: Λειτουργία Διακομιστή PEP με την χρήση του μυστικού κωδικού Master Key.*

Στην επόμενη περίπτωση αιτημάτων με χρήση OAuth2 token, η διαδικασία απεικονίζεται στην Εικόνα 3.19 και παρακάτω αναλύονται οι ενέργειες που εκτελούνται με τη σειρά που αυτές λαμβάνουν χώρα.



***Εικόνα 3.19**: Λειτουργία Διακομιστή PEP με την χρήση του OAuth2 token.*

1. Ο Διακομιστής PEP δέχεται ένα HTTP αίτημα που περιλαμβάνει ένα OAuth2 token.
2. Ελέγχει αν το διακριτικό που φέρει στην κεφαλίδα του το αίτημα που δέχτηκε, αποτελεί ένα έγκυρο OAuth2 token . Ο έλεγχος αυτός συμβαίνει μέσω REST επικοινωνίας με την υπηρεσία ταυτοποίησης και εξουσιοδότησης χρηστών.
3. Εφόσον η εγκυρότητα του OAuth2 token επιβεβαιωθεί, η υπηρεσία Ταυτοποίησης και Εξουσιοδότησης, επιστρέφει στον Διακομιστή PEP, τις πληροφορίες που σχετίζονται με την ταυτότητα και τους ρόλους του χρήστη που αφορά το OAuth2 token. Σε περίπτωση που το OAuth2 token δεν είναι έγκυρο η διαδικασία σταματάει εδώ.
4. Σειρά έχει να αξιολογηθεί αν το αίτημα το οποίο έκανε ο χρήστης στο βήμα 1), εγκρίνεται βάσει των ρόλων που διαθέτει. Ο Διακομιστής PEP μέσω REST επικοινωνίας, προωθεί τις πληροφορίες που έλαβε στο βήμα 3) και το αίτημα που έλαβε στο βήμα 1) στην υπηρεσία Λήψης Απόφασης Εξουσιοδότησης προκειμένου η υπηρεσία να αποφασίσει αν το αίτημα εγκρίνεται ή όχι.
5. Η υπηρεσία αξιολογεί αν το αίτημα του χρήστη εγκρίνεται ή απορρίπτεται.
6. Ο Διακομιστής PEP ενημερώνεται για το αποτέλεσμα της αξιολόγησης.

7. Εφόσον το αίτημα εγκρίνεται, ο Διακομιστής PEP προωθεί το αίτημα στην προστατευόμενη υπηρεσία για την οποία μεσολαβεί.
8. Η υπηρεσία επιστρέφει στον Διακομιστή PEP την απάντηση του αιτήματος.
9. Ο Διακομιστής PEP προωθεί την απάντηση της υπηρεσίας στον χρήστη.

Στην παρούσα αρχιτεκτονική, χρησιμοποιούνται εφτά διαφορετικοί διακομιστές PEP με το κάθε ένα να «προστατεύει» αντίστοιχα τις υπηρεσίες (1) Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων, (2) Διαχείρισης Συσκευών Backend IDAS, (3) Οντολογίας, (4) Διαχείρισης Συμβάντων και Συνδρομών, (5) Διαχείρισης Ιστορικών Δεδομένων, (6) Υπηρεσία Mashup, (7) Directory Service.

Οι διακομιστές PEP[25] 1 και 7 (βλ. Διάγραμμα Αρχιτεκτονικής Ενός Κόμβου 3.23) λειτουργούν με βάση την διαδικασία που αποτυπώνεται στο διάγραμμα της εικόνας 3.19. Στις υπόλοιπες λειτουργίες, όπου η επικοινωνία συμβαίνει ανάμεσα σε υπηρεσίες του συστήματος (και όχι ανάμεσα σε μία υπηρεσία του συστήματος και αίτημα που δέχεται από χρήστη), δεν τίθεται σκοπιμότητα ταυτοποίησης ή διαμόρφωσης εξουσιοδότησης διαφορετικών βαθμίδων. Επομένως, σε αυτό το σενάριο μία υπηρεσία του συστήματος, όταν κάνει αίτημα σε μία άλλη υπηρεσία του συστήματος, περιλαμβάνει στην κεφαλίδα του αιτήματος της τον κατάλληλο κωδικό Master key του υπεύθυνου διακομιστή PEP. Οι διακομιστές PEP σε αυτή την περίπτωση λειτουργούν σύμφωνα με την διαδικασία που αποτυπώθηκε στο διάγραμμα της εικόνα 3.18.

Με αυτό τον τρόπο εξασφαλίζεται από την υποδομή του συστήματος ότι μία υπηρεσία που λαμβάνει την παραπάνω «ασφάλεια» δεν είναι διαθέσιμη σε αιτήματα εκτός του συστήματος.

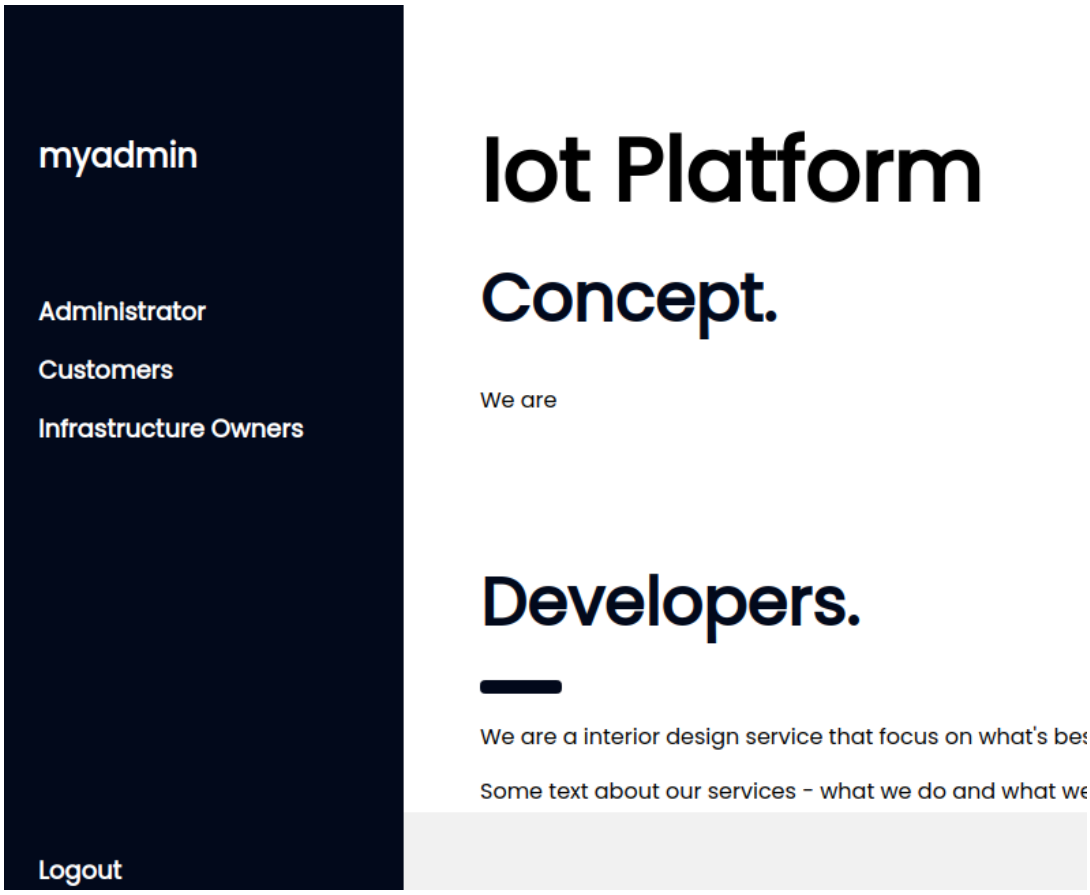
3.4.10 Δρομολόγησης και Ελέγχου - Σύστημα Διεπαφής Χρηστών (Application Logic - Web Application)

Η Λογική Εφαρμογής αποτελεί κομβικό σημείο του συστήματος καθώς περιλαμβάνει τον κώδικα για την ενορχήστρωση των επιμέρους υπηρεσιών, ώστε το σύστημα να επιτυγχάνει την λειτουργικότητα που διατυπώνεται στις προδιαγραφές που έχουν οριστεί. Το σύστημα διεπαφής χρηστών (Web Application) θεωρείται τμήμα της λογικής εφαρμογής καθώς περιλαμβάνει τον απαραίτητο κώδικα για την υλοποίηση των γραφικών διεπαφών (για όλους τους διαφορετικούς τύπους χρηστών) του συστήματος. Τα αιτήματα των χρηστών του συστήματος δημιουργούνται από το σύστημα διεπαφής χρηστών και δρομολογούνται στην λογική εφαρμογής για την κατάλληλη προώθηση τους. Στη συνέχεια διατυπώνονται δύο σχετικά παραδείγματα της λειτουργίας τους.

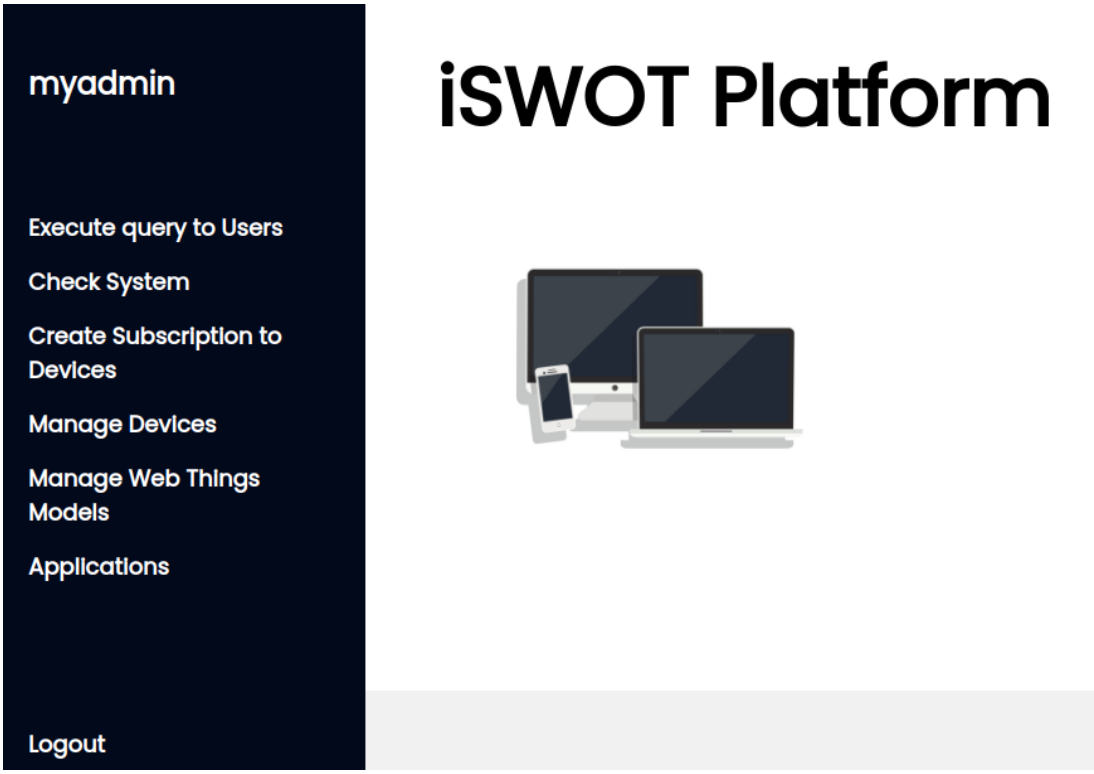
[25] https://www.keycloak.org/docs/6.0/authorization_services/#_enforcer_overview

- Ένας χρήστης μέσω του συστήματος διεπαφής χρηστών δημιουργεί αίτημα εισόδου στο σύστημα (“Log in” με την χρήση Username και Password). Το αίτημα προωθείται στην υπηρεσία «Λογική εφαρμογής», η οποία με την σειρά της το δρομολογεί στην υπηρεσία ταυτοποίησης και εξουσιοδότησης του συστήματος (Keycloak). Εφόσον τα στοιχεία του επαληθευτούν, ο χρήστης θα εισέλθει στο σύστημα. Κατόπιν, η υπηρεσία «Λογική εφαρμογής» δημιουργεί μία συνεδρία σύνδεσης (session) για το χρήστη, η οποία περιλαμβάνει τα στοιχεία εισόδου του και το OAuth2 token που του αντιστοιχεί.
- Ένας τελικός χρήστης μέσω του συστήματος διεπαφής χρηστών, ζητάει πρόσβαση σε μία εφαρμογή της συνδρομής του. Το HTTP αίτημα του, δρομολογείται στην υπηρεσία «Λογική Εφαρμογής». Η «Λογική εφαρμογής» δρομολογεί το αίτημα στην υπηρεσία Mashup (Υπηρεσία που εκτελεί τις εφαρμογές του συστήματος, βλ. 3.4.5). Η «λογική εφαρμογής» επιστρέφει το αποτέλεσμα στο χρήστη και εκείνος βλέπει στην οθόνη του την πληροφορία της εφαρμογής.

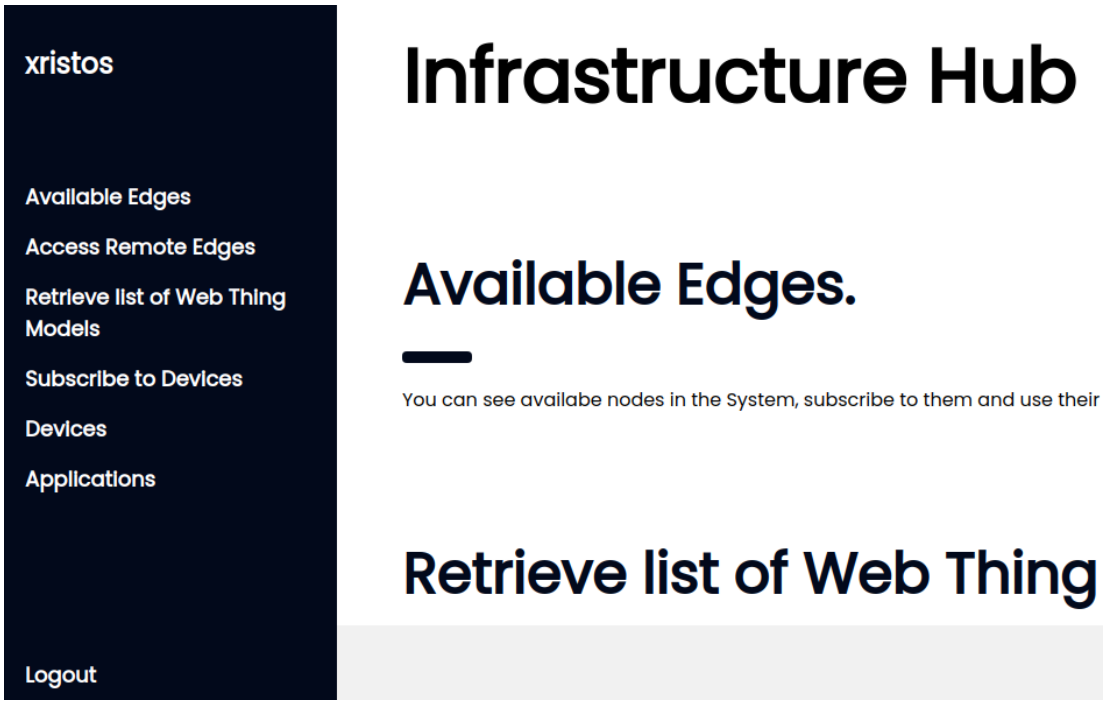
Με την ίδια λογική, όλα τα αιτήματα που προκύπτουν από τις γραφικές διεπαφές του συστήματος διεπαφής χρηστών δρομολογούνται μέσω της υπηρεσίας «λογική εφαρμογής» στις κατάλληλες υπηρεσίες. Ακολουθούν ορισμένα στιγμιότυπα γραφικών διεπαφών του συστήματος για τους διαφορετικούς τύπους χρηστών.



***Εικόνα 3.20**: Στιγμιότυπο επιλογής κατηγορίας χρήστη.*



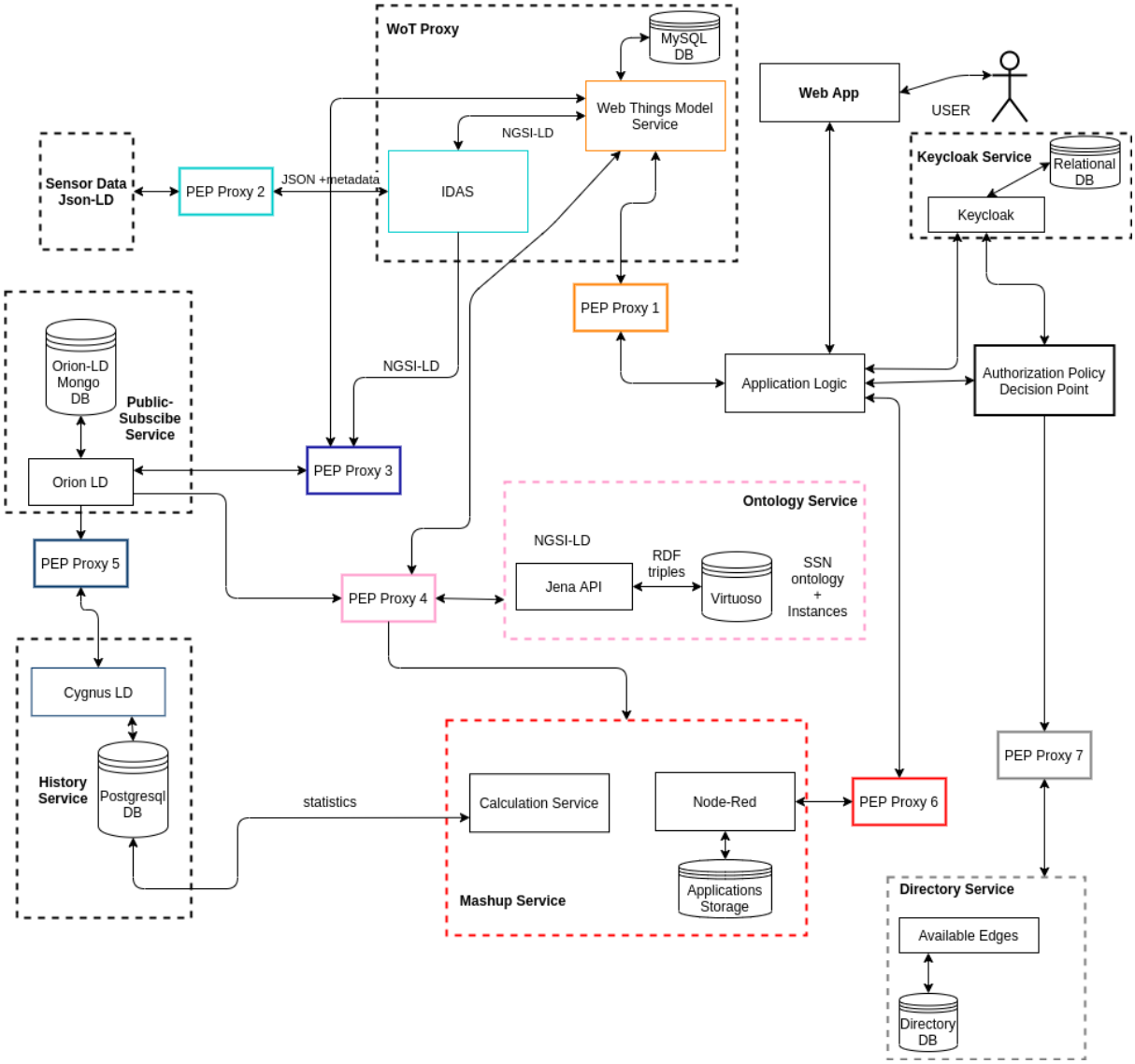
***Εικόνα 3.21**: Στιγμιότυπο διαθέσιμων λειτουργιών για τον διαχειριστή του συστήματος.*



***Εικόνα 3.22**: Στιγμιότυπο διαθέσιμων λειτουργιών για χρήστη τύπου Infrastructure Owner.*

3.5 Διάγραμμα Αρχιτεκτονικής Ενός Κόμβου

Ακολουθεί το συνολικό διάγραμμα αρχιτεκτονικής ενός κόμβου του συστήματος Fi-SWoT. Στο σχήμα φαίνονται όλες οι υπηρεσίες που αναλύθηκαν σε προηγούμενη ενότητα και πως αλληλεπιδρούν μεταξύ τους, καθιστώντας την παρούσα αρχιτεκτονική μέρος του Σημασιολογικού Ιστού των Πραγμάτων.



Εικόνα 3.23 : *Διάγραμμα Αρχιτεκτονικής Συστήματος Ενός Κόμβου.*

3.6 Κατανεμημένο Σύστημα Fi-SWoT

Το Fi-SWoT αποτελεί ένα κατανεμημένο σύστημα που περιλαμβάνει ανεξάρτητα και ισοδύναμα σύστημα Σημασιολογικού Ιστού (κόμβοι), τα οποία αλληλεπιδρούν μεταξύ τους. Κάθε κόμβος περιλαμβάνει όλες τις υπηρεσίες που περιγράφηκαν παραπάνω και αντιστοιχίζεται στο διάγραμμα της εικόνας 3.23. Στο Fi-SWoT δεν υπάρχει “κύριος” κόμβος, αλλά όπως αναφέρθηκε όλοι οι κόμβοι θεωρούνται ίσοι και έχουν τις ίδιες δυνατότητες. Ως κατανεμημένο σύστημα έχει τα ακόλουθα πλεονεκτήματα :

- Επιτυγχάνει την εύκολη ανταλλαγή δεδομένων μεταξύ κόμβων, χάρη στην επικοινωνία τους.
- Υποστηρίζει δυνατότητα κλιμάκωσης, επιτρέποντας την εύκολη εισαγωγή νέου κόμβου στο σύστημα.
- Η αποτυχία ενός κόμβου δεν συνεπάγεται την αποτυχία ολόκληρου του κατανεμημένου συστήματος.
- Δυνατότητα πολλαπλής κοινής χρήσης πόρων από διαφορετικούς κόμβους.

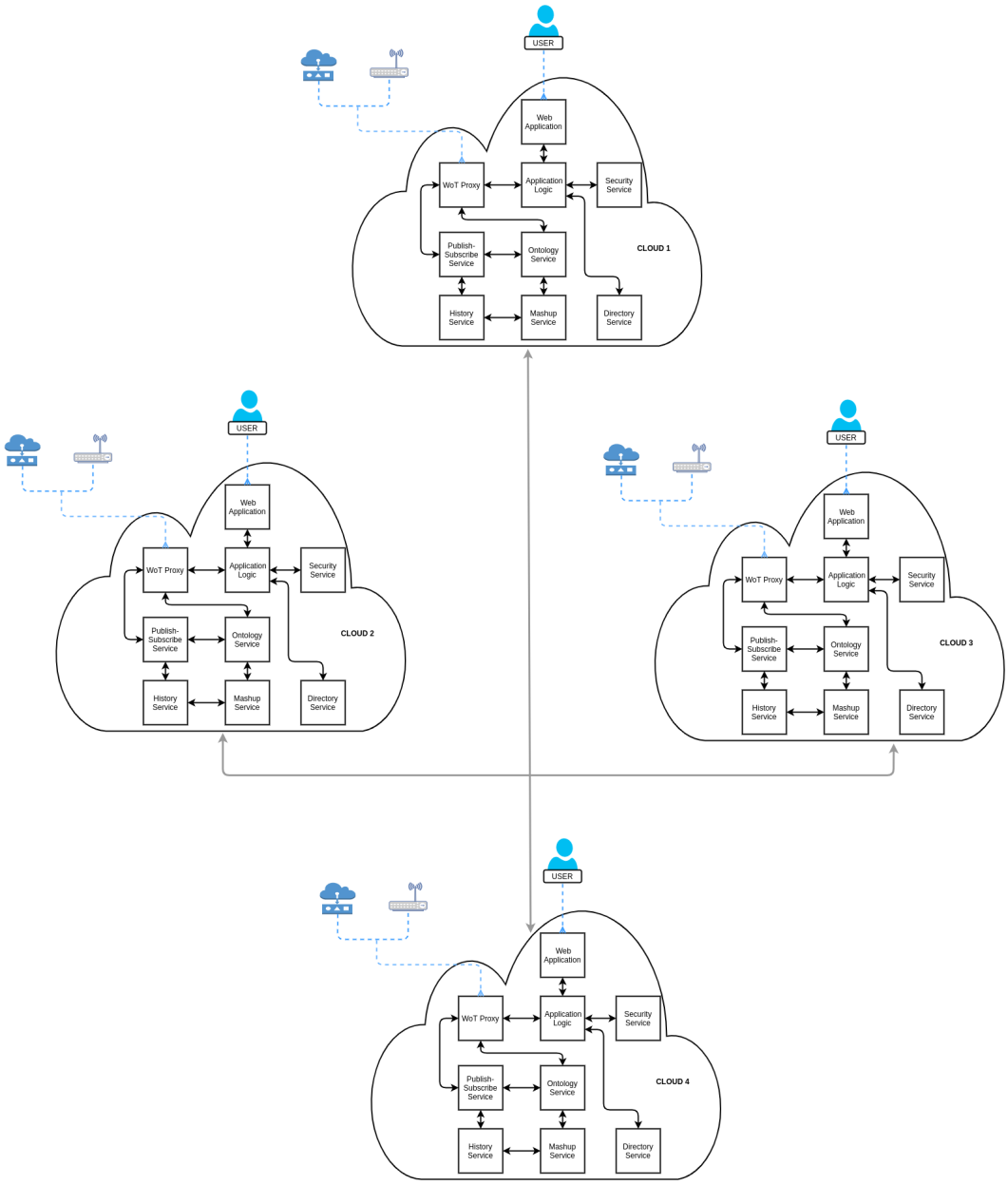
Το Fi-SWoT σχεδιάστηκε με στόχο να επιτρέπεται η ανταλλαγή πληροφοριών μεταξύ των κόμβων, με ασφάλεια και παράλληλα διατηρώντας τον πλήρη έλεγχο των δεδομένων, πόρων τους. Οι κόμβοι μοιράζονται πληροφορίες και υπηρεσίες κατάστασης, αλλά χωρίς να επηρεάζει ο ένας τον άλλον. Για παράδειγμα αν ένας κόμβος αποτύχει (“πέσει”), οι υπόλοιποι θα εξακολουθούν να λειτουργούν χωρίς να επηρεαστούν από το παραπάνω συμβάν. Επιπλέον, κάθε σύστημα Σημασιολογικού Ιστού (κόμβος) είναι υπεύθυνο για την προστασία (π.χ. μη εξουσιοδοτημένη πρόσβαση) των δικών του συσκευών και υπηρεσιών. Ο κάτοχος ή ο διαχειριστής του είναι ο υπεύθυνος για την παροχή πρόσβασης σε χρήστες ενός ξένου κόμβου.

Σε αυτό το σημείο αξίζει να αναφέρουμε ότι κάθε κόμβος του Fi-SWoT περιέχει την υπηρεσία **Directory**, η οποία υλοποιείται ως κόμβος της βάσης δεδομένων Cassandra. Η Cassandra αποτελεί μια κατανεμημένη βάση δεδομένων, επιτρέποντας στους κόμβους του Fi-SWoT να “γνωρίζονται” μεταξύ τους και να μοιράζονται πληροφορίες όπως τοποθεσίες, τύπους αισθητήρων, εφαρμογών κλπ. Αυτό επιτρέπει στους χρήστες να ανακαλύπτουν τους διαθέσιμους κόμβους και να ανακτούν τις δημόσιες πληροφορίες τους χωρίς λειτουργία εξουσιοδότησης. Ωστόσο αν κάποιος χρήστης επιθυμεί να εκμεταλλευτεί περαιτέρω λειτουργίες ενός απομακρυσμένου κόμβου, θα χρειαστεί άδεια πρόσβασης από τον εκάστοτε διαχειριστή του κόμβου.

Με αυτό τον τρόπο, η αρχιτεκτονική iSWoT του [1] και [2] αναδιαμορφώνεται για να μετατραπεί σε έναν αυτόνομο κόμβο SWoT στο οικοσύστημα Fi-SWoT.

3.6.1 Αρχιτεκτονικό Διάγραμμα Κατανεμημένου Συστήματος (Abstract Diagram)

Ακολουθεί ένα συνοπτικό διάγραμμα του οικοσυστήματος Fi-SWoT με τέσσερις κόμβους. Κάθε κόμβος έχει σχεδιαστεί ως σύνθεση RESTful μικρο-υπηρεσιών στο υπολογιστικό νέφος. Οι κόμβοι είναι διαλειτουργικοί και επεκτάσιμη, δηλαδή οι υπηρεσίες μπορούν να προστεθούν ή να αφαιρεθούν.



Εικόνα 3.24 : *Αρχιτεκτονικό Διάγραμμα Fi-SWoT με 4 κόμβους.*

3.6.2 Υπηρεσίες στο Κατανεμημένο Σύστημα

Directory Service

Η υπηρεσία Directory διατηρεί πληροφορίες για την ανακάλυψη όλων των εγγεγραμμένων κόμβων του Fi-SWoT, χρησιμοποιώντας τη βάση δεδομένων **Cassandra**. Οι κόμβοι περιγράφονται από το αναγνωριστικό-όνομα, τα στοιχεία επικοινωνίας τους (IP) και στατιστικά όπως αριθμός εγγεγραμμένων χρηστών, συσκευών, εφαρμογών, ενώ κάθε ο κόμβος καθίσταται διαθέσιμος για συνδρομή σε χρήστες άλλων κόμβων. Βασική προϋπόθεση για να είναι ένας κόμβος ανιχνεύσιμος από τους χρήστες είναι η εγγραφή του στην βάση δεδομένων Cassandra που αποτελεί αναπόσπαστο κομμάτι του Directory Service. Σε αυτό το σημείο αξίζει να αναφέρουμε ορισμένα πλεονεκτήματα της Cassandra:

- είναι εξαιρετικά επεκτάσιμη και ικανή να υποστηρίξει χιλιάδες ταυτόχρονους χρήστες ή λειτουργίες ανά δευτερόλεπτο.
- δεν υπάρχει σημείο αποτυχίας σηματοδότησης (δηλαδή σε περίπτωση αποτυχίας κόμβου, δεν θα υπάρξει απώλεια δεδομένων καθώς άλλοι κόμβοι θα αναλάβουν την εξυπηρέτηση).
- νέοι κόμβοι μπορούν να αφαιρεθούν ή να προστεθούν στο Fi-SWoT χωρίς διακοπή λειτουργίας

Η υπηρεσία αποθηκεύει την περιγραφή κάθε κόμβου, καθώς και τους τύπους συσκευών, εφαρμογών που διαθέτει. Τα παραπάνω αποτελούν δημόσιες πληροφορίες που ανήκουν σε κόμβους και μπορούν να ανακαλυφθούν εύκολα από χρήστες άλλων κόμβων χάρη στη Cassandra. Ωστόσο, ενέργειες όπως συνδρομή και πρόσβαση σε αισθητήρες και υπηρεσίες, δημιουργία εφαρμογών σε ξένους κόμβους, επιτρέπεται μόνο σε εξουσιοδοτημένους χρήστες. Ο διαχειριστής του εκάστοτε συστήματος είναι ο μόνος υπεύθυνος για τη χορήγηση πρόσβασης στον κόμβο.

Ένας χρήστης που έχει πραγματοποιήσει συνδρομή σε απομακρυσμένο κόμβο, μπορεί να συνδεθεί στον εκάστοτε απομακρυσμένο κόμβο και να εκτελέσει λειτουργίες με βάση τις πολιτικές πρόσβασης. Οι ενέργειες που επιτρέπονται σε ένα ξένο κόμβο διαμορφώνονται σύμφωνα με τον τύπο χρήστη και είναι οι εξής :

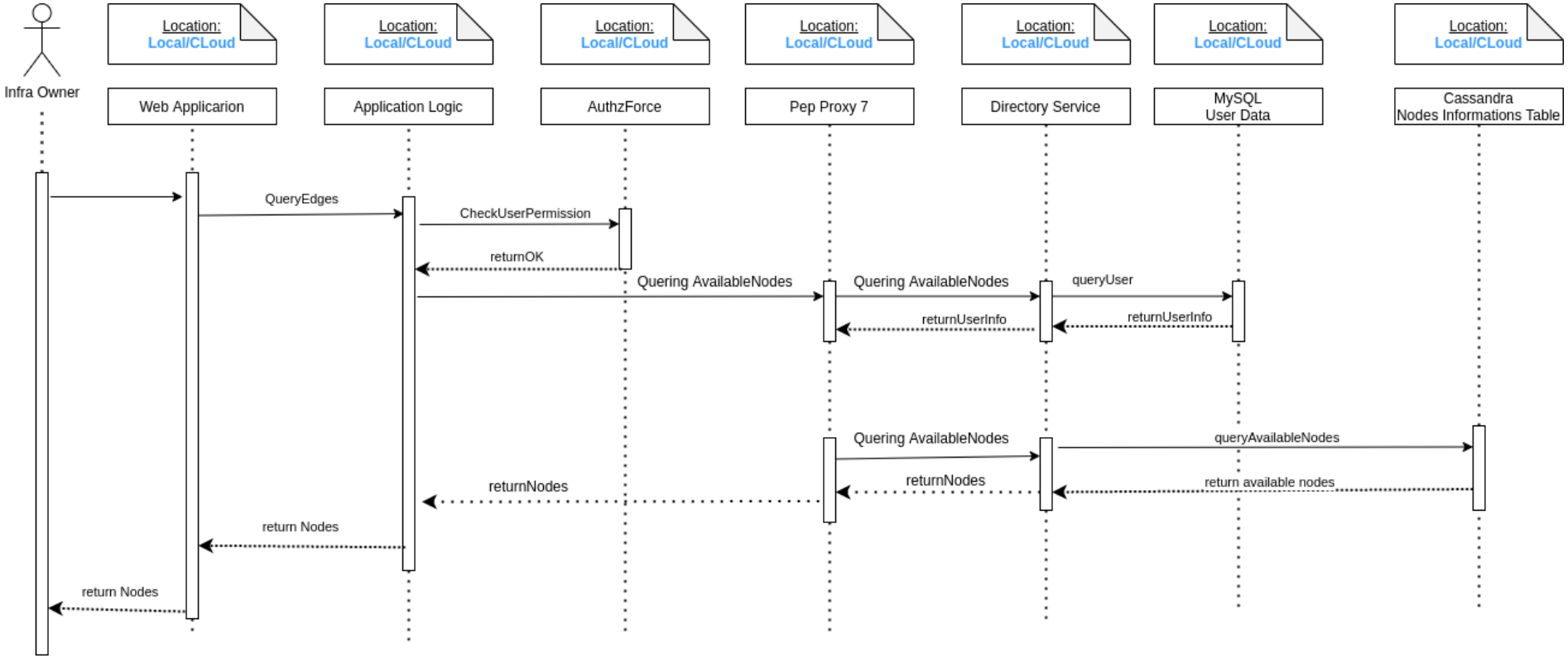
- Ιδιοκτήτης Υποδομης (Infrastructure Owner)
  - Ανάκτηση Συσκευών
  - Δημιουργία συνδρομής σε συσκευή
  - Αναζήτηση εφαρμογών
  - Δημιουργία νέας εφαρμογής
- Πελάτης / Τελικός Χρήστης (Customer)
  - Αναζήτηση Εφαρμογών
  - Δημιουργία συνδρομής σε εφαρμογή
  - Ανάκτηση / Χρήση εφαρμογής

Επίσης, υπηρεσία Directory καθιστά το Fi-SWoT επεκτάσιμο αφού χάρη σε αυτή μπορούν να εισάγονται νέοι κόμβοι στο σύστημα και να γνωστοποιούνται στους ήδη υπάρχοντες. Τέλος, η υπηρεσία είναι προσβάσιμη από όλους τους χρήστες με απώτερο σκοπό την κοινοποίηση όλων των κόμβων του Fi-SWoT.

Ακολουθεί ένα διάγραμμα ακολουθίας το οποίο περιγράφει την αναζήτηση διαθέσιμων κόμβων του συστήματος. Η συγκεκριμένη δραστηριότητα αξιοποιεί την τρέχουσα υπηρεσία και έχει ως στόχο την παροχή πληροφοριών γενικού σκοπού για ξένους κόμβους (π.χ. ποιοι είναι, πόσους χρήστες έχουν, ποιές εφαρμογές και συσκευές διαθέτουν) σε χρήστες που δεν έχουν εξουσιοδότηση πρόσβασης .

- **Αναζήτηση διαθέσιμων κόμβων στο Fi-SWoT**

Ένας ιδιοκτήτης υποδομής ή τελικός χρήστης μπορεί να ανακαλύψει όλους του εγγεγραμμένους κόμβους του Fi-SWoT και να ανακτήσει τις δημόσιες πληροφορίες τους. Ο χρήστης αφού συνδεθεί στο σύστημα, μέσω της Διαδικτυακής Εφαρμογής (Web Application) επιλέγει να δει τους διαθέσιμους κόμβους. Η διαδικτυακή εφαρμογή προωθεί το αίτημα στην τοπική εφαρμογή λογικής (Application Logic), η οποία δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες. Προτού το αίτημα φτάσει στις κατάλληλές υπηρεσίες, ελέγχεται εάν ο χρήστης έχει το δικαίωμα να εκτελέσει την επιλεγμένη ενέργεια. Ο έλεγχος αυτός πραγματοποιείται καλώντας την υπηρεσία AuthZForce, η οποία εγκρίνει ή απορρίπτει το αίτημα με βάση το ρόλο του χρήστη και τα δικαιώματα του. Εάν το AuthZForce επιστρέψει θετική απάντηση, το αίτημα κατευθύνεται στο διακομιστή μεσολάβησης (Pep Proxy) της προστατευόμενης υπηρεσίας, δηλαδή του Directory Service περιλαμβάνοντας στην κεφαλίδα του αιτήματος τον μυστικό κωδικό Master Key (κάθε PEP Proxy-Wilma έχει τον δικό του μυστικό κωδικό Master Key). Εφόσον ο κωδικός Master Key είναι σωστός, το Pep Proxy προωθεί την επικοινωνία στην υπηρεσία για την οποία μεσολαβεί. Έπειτα η υπηρεσία Directory, αναλαμβάνει να επιστρέψει στον χρήστη μια λίστα με τους διαθέσιμους κόμβους. Το τελευταίο επιτυγχάνεται μέσω της βάσης δεδομένων Cassandra, η οποία διατηρεί τις περιγραφές όλων των κόμβων. Στο παράθυρο περιήγησης του χρήστη εμφανίζονται όλοι οι διαθέσιμοι κόμβοι, καθώς και σε ποιους έχει εγγραφεί, δηλαδή έχει δικαιώματα πρόσβασης. Επίσης, σε κάθε διαθέσιμο κόμβο υπάρχουν οι επιλογές ανάκτησης πληροφοριών γενικού σκοπού, συνδρομής (Subscribe) και κατάργησης συνδρομής, σε περίπτωση που ο χρήστης δεν επιθυμεί να έχει πρόσβαση στον εκάστοτε κόμβο. Η παραπάνω περιγραφή απεικονίζεται ως διάγραμμα ακολουθίας στην εικόνα 3.24.



Εικόνα 3.24 : *Διάγραμμα ακολουθίας αναζήτησης διαθέσιμων κόμβων στο Fi-SWoT.*

Μηχανισμός OpenID Connect

Το OpenID Connect αποτελεί ένα απλό επίπεδο ταυτότητας πάνω από το πρωτόκολλο OAuth 2.0. Ενώ το OAuth 2.0 αφορά την πρόσβαση σε πόρους και την εξουσιοδότηση χρηστών, το OIDC αφορά την ταυτοποίηση του χρήστη. Επιτρέπει δηλαδή σε τρίτους πελάτες να επαληθεύουν την ταυτότητα του τελικού χρήστη και να λαμβάνουν βασικές πληροφορίες για το προφίλ του χρησιμοποιώντας REST API.

Συγκεκριμένα, το OpenID Connect αποτελεί επέκταση του OAuth 2.0. Γνωρίζει το που, πότε, πως έγινε η ταυτοποίηση ενός χρήστη, καθώς και επιτρέπει στο χρήστη να συνδεθεί σε ένα σύστημα με ένα συνδυασμό αναγνωριστικών (όνομα και κωδικός) για να έχει πρόσβαση σε πολλαπλές υπηρεσίες του εώς ότου αποσυνδεθεί (Single Sign-On). Βελτιώνει τον μηχανισμό ταυτοποίησης και εξουσιοδότησης, μειώνοντας την πιθανότητα έκθεσης πληροφοριών του χρήστη κατα την διάρκεια πρόσβασης σε έναν οργανισμό. Επιπλέον, συμβάλει στη ανάκτηση πρόσθετων πληροφοριών των χρηστών, παρέχοντας ένα μοναδικό αναγνωριστικό γνωστό ως Id Token. Το τελευταίο επιβεβαιώνει την ταυτότητα του χρήστη περιλαμβάνοντας στοιχεία όπως όνομα, ηλεκτρονική διεύθυνση, από ποιό πάροχο διατίθεται, πότε παράχθηκε, πότε λήγει. Το Id Token κωδικοποιείται ως JSON Web Token[26] (JWT) και η έκδοση του πραγματοποιείται καλώντας την υπηρεσία ταυτοποίησης (π.χ. Keycloak IDM).

Πιο αναλυτικά, ένα JWT αποτελείται από τρία μέρη τα οποία είναι τα εξής :

- **Header :** δηλώνει τον τύπο του διακριτικού, το οποίο είναι JWT και τον αλγόριθμο υπογραφής που χρησιμοποιείται.
- **Payload :** περιέχει πληροφορίες για τον οργανισμό που ο χρήστης είναι μέλος καθώς και στοιχεία του χρήστη όπως όνομα, email. Επιπλέον, παρέχει πληροφορίες που αφορούν το ίδιο το διακριτικό όπως ποιός είναι ο εκδότης του, ο χρόνος λήξης, ο χρόνος έκδοσης.
- **Signature :** χρησιμοποιείται για να επαληθεύσει ότι ο αποστολέας του JWT είναι αυτός που λέει ότι είναι και για να διασφαλίσει ότι οι πληροφορία δεν άλλαξε στην πορεία.

Οι παραπάνω πληροφορίες είναι κωδικοποιημένες σε JSON μορφή και ενώνονται με το σύμβολο “.” ανάμεσα τους. Η αποκωδικοποίηση τους οδηγεί σε ανάκτηση των πληροφοριών που αναφέρθηκαν και αποδεικνύει ότι πρόκειται για έναν έγκυρο χρήστη.

Στη παρούσα εργασία ο μηχανισμός OpenID Connect χρησιμοποιείται στη περίπτωση που ένας χρήστης θέλει να αποκτήσει πρόσβαση σε έναν απομακρυσμένο κόμβο. Ο χρήστης προκειμένου να εκτελέσει ενέργειες στον εκάστοτε κόμβο, θα πρέπει πρώτα να ενταχθεί στους πιστοποιημένους χρήστες του. Επομένως, όταν ένας χρήστης (τοπικού κόμβου) στέλνει ένα αίτημα συνδρομής σε ένα ξένο κόμβο, ο τελευταίος ζητάει επιβεβαίωση από τον τοπικό κόμβο. Αυτό πραγματοποιείται δρομολογώντας ένα αίτημα στην υπηρεσία ταυτοποίησης του τοπικού κόμβου περιλαμβάνοντας στη κεφαλίδα του αιτήματος την ενεργοποίηση του

[26] https://developer.yahoo.com/oauth2/guide/openid_connect/decode_id_token.html

μηχανισμού **OpenID Connect** και τα στοιχεία του χρήστη. Η απάντηση που επιστρέφεται περιέχει δύο αναγνωριστικά (access token και id token). Με αυτόν τον τρόπο επαληθεύεται η εγκυρότητα του χρήστη και ότι είναι ασφαλές να αποκτήσει πρόσβαση στον απομακρυσμένο κόμβο. Επιπλέον, ο ξένος κόμβος μπορεί να ανακτήσει πληροφορίες του χρήστη που παρέχονται στο Id Token. Στη συνέχεια, ο ξένος κόμβος εισάγει τον χρήστη στους πιστοποιημένους δικούς του χρήστες παρέχοντας του πρόσβαση σε πόρους του και δικαιώματα εκτέλεσης συγκεκριμένων ενεργειών. Τέλος, ο χρήστης με την σειρά του μπορεί να συνδεθεί στο ξένο κόμβο και να στέλνει αιτήματα σε αυτόν για όσο χρόνο διαρκεί το session του.

Κεφάλαιο 4

Ανάλυση απόδοσης

4.1 Περιβάλλον Υλοποίησης

Η υποδομή του συστήματος που υλοποιήθηκε στη παρούσα εργασία στεγάζεται στο περιβάλλον Google Cloud Platform[27] και απαρτίζεται από τέσσερις (4) εικονικές μηχανές. Το GCP είναι ένας πάροχος υπολογιστικών πόρων για την ανάπτυξη και τη λειτουργία εφαρμογών στον ιστό, ενώ παρέχει ένα φιλικό και εύχρηστο περιβάλλον προς τους χρήστες του. Τα τεχνικά χαρακτηριστικά κάθε εικονικής μηχανής που δημιουργήθηκε στο GCP είναι τα εξής:

CPU	2 CPU cores, x86_64@2.8GHz
MEMORY	8GB
HDD	40GB
OS	Ubuntu 16.04

Κάθε εικονική μηχανή στο GCP διαθέτει μια εσωτερική διεύθυνση (internal ip), ενώ είναι στη κρίση του χρήστη αν θα ορίσει μια στατική εξωτερική διεύθυνση (external ip) για την εκάστοτε μηχανή ώστε να μπορεί να επικοινωνεί με το εξωτερικό δίκτυο, να στέλνει και να λαμβάνει δεδομένα. Είναι ευθύνη του χρήστη αν θα επιτρέψει την πρόσβαση στο VM από οποιαδήποτε άλλη συσκευή ή όχι. Το τελευταίο επιτυγχάνεται ορίζοντας ένα εύρος διευθύνσεων που αντιστοιχίζονται σε πόρους-συσκευές, οι οποίοι θα έχουν την δυνατότητα πρόσβασης στο VM.

Για τις ανάγκες της παρούσας αρχιτεκτονικής, κάθε εικονική μηχανή (VM) του GCP αποτελείται από ένα περιβάλλον Docker και μια βάση δεδομένων, Apache Cassandra και αντιπροσωπεύει ένα κόμβο του Fi-SWoT. Σε κάθε docker πλατφόρμα υπάρχουν 23 κιβώτια (containers) στα οποία εκτελούνται οι ακόλουθες υπηρεσίες :

- Web Application
- Application Logic
- Keycloak
- AuthZForce
- Web Things Models Service
- Mashup Service

[27] https://cloud.google.com/docs

- Node Red
- Orion-LD Context Broker
- Cygnus-LD
- IDAS
- MySQL Database
- MongoDB
- PostgreSQL
- Apache Tomcat
- Ontology
- Directory Service
- Pep Proxy για κάθε προστατευόμενη υπηρεσία (7 συνολικά)

Η προσπέλαση των εικονικών μηχανών πραγματοποιείται μέσω ενός παραθύρου περιήγησης γνωρίζοντας την αντίστοιχη εξωτερική ip. Ωστόσο για την πρόσβαση στις υπηρεσίες που χρησιμοποιούνται ως γραφική διεπαφή (πχ Web Application) στο σύστημα έχει οριστεί ένα port. Με αυτό τον τρόπο, η υπηρεσία καθίσταται προβιβάσιμη μεσω του URL: “http://external_ip:service_port”.

4.2 Πειράματα

Σκοπός της πειραματικής διαδικασίας είναι να εξετάσει την απόδοση του συστήματος σε πραγματικές συνθήκες φόρτου. Προκειμένου να αποδειχθεί η απόδοση του συστήματος, χρησιμοποιήθηκε το εργαλείο Apache Bench το οποίο έχει την δυνατότητα δημιουργίας πολλών ταυτόχρονων αιτημάτων και υπολογισμού σημαντικών παραμέτρων όπως είναι ο χρόνος εξυπηρέτησης συγκεκριμένου αριθμού αιτημάτων. Το Apache Bench έχει την ιδιότητα να παράγει συνθήκες φόρτου σε κάθε υπηρεσία του συστήματος ξεχωριστά, ορίζοντας τον αριθμό των αιτημάτων που θα κληθεί η κάθε υπηρεσία να εξυπηρετήσει καθώς και τον αριθμό αυτών που θα εκτελεστούν ταυτόχρονα. Κάθε εντολή του Apache Bench θα πρέπει να περιέχει το συνολικό αριθμό των αιτημάτων, τον αριθμό ταυτοχρονισμού, την διεύθυνση που πρόκειται να “χτυπήσει” και τα δεδομένα εισόδου που θα χρησιμοποιήσει η εκάστοτε εφαρμογή της υπηρεσίας.

Επιπλέον, για να παρακολουθήσουμε τους φυσικούς πόρων των εικονικών μηχανών κατά την διάρκεια των πειραμάτων, χρησιμοποιήσαμε το εργαλείο HTOP. Το HTOP εκτελείται κατευθείαν από τη γραμμή εντολών και μας ενημερώνει σχετικά με την κατανάλωση πόρων ανά διεργασία, καθώς και για τη συνολική κατανάλωση πόρων από το σύστημα σε πραγματικό χρόνο.

Τα ακόλουθα πειράματα διακρίνονται σε δύο ενότητες. Η πρώτη αφορά αιτήματα που πραγματοποιούνται στο τοπικό κόμβο, δηλαδή η απάντηση έρχεται από τον κόμβο στον οποίο ο χρήστης είναι συνδεδεμένος, ενώ η δεύτερη ενότητα “μοιράζει” τα αιτήματα στους εγγεγραμμένους κόμβους. Συγκεκριμένα, για 1000 αιτήματα και δύο κόμβους ο κάθε κόμβος θα κληθεί να εξυπηρετήσει 500 αιτήματα, για 1000 αιτήματα και τρεις κόμβους ο κάθε κόμβος θα κληθεί να εξυπηρετήσει 333

αιτήματα και ούτω κάθεξης. Σε αυτή την περίπτωση, το ένα μέρος των αιτημάτων δρομολογείται στο τοπικό κόμβο και το υπόλοιπο σε απομακρυσμένους.

Επομένως, σκοπός είναι να δείξουμε την απόδοση του συστήματος για πολλά ταυτόχρονα αιτήματα που δρομολογούνται σε τοπικό κόμβο αλλά και σε απομακρυσμένους κόμβους, οι οποίοι έχουν πολλούς χρήστες και πολλά δεδομένα.

4.2.1 Πειραματικό περιβάλλον

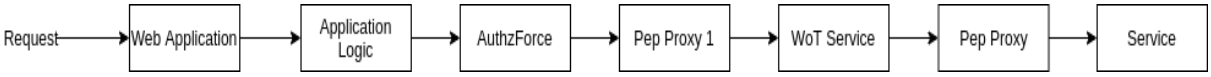
Όπως έχει ήδη αναφερθεί, η παρούσα εργασία αποτελείται από τέσσερις (πανομοιότυπες) εικονικές μηχανές, εκ των οποίων η κάθε μια αντιπροσωπεύει ένα κόμβο του Fi-SWoT. Προκειμένου να παράξουμε συνθήκες φόρτου στο σύστημα, φτιάξαμε ένα υποθετικό σενάριο όπου κάθε εικονική μηχανή (κόμβος) του συστήματος αντιπροσωπεύει μια πόλη και περιλαμβάνει εικονικές οντότητες 1050 σπιτιών, οι οποίες περιέχουν αισθητήρες. Άρα πρόκειται για μια πόλη, στην οποία έχουμε 1050 σπίτια, με 3 είδη αισθητήρων (υγρασίας, θερμοκρασίας, φωτεινότητας) και σε όλα τα σπίτια υπάρχει ένας αισθητήρας από κάθε είδος. Οι οντότητες αυτές αποθηκεύτηκαν τόσο στην οντολογία όσο και στη βάση δεδομένων του Orion-LD Context Broker, που φιλοξενεί τις πιο πρόσφατες τιμές. Επιπλέον, κάθε αισθητήρας μεταφέρει 100 μετρήσεις ανά ώρα. Συνολικά, δηλαδή παράγονται $24 \times 1000 = 24000$ μετρήσεις από κάθε αισθητήρα και $3 \times 24000 = 72000$ μετρήσεις από όλους τους αισθητήρες την ημέρα. Τα παραπάνω δεδομένα επαναλαμβάνονται και αποθηκεύονται ξεχωριστά σε κάθε κόμβο του συστήματος.

4.2.2 Πειραματικά αποτελέσματα

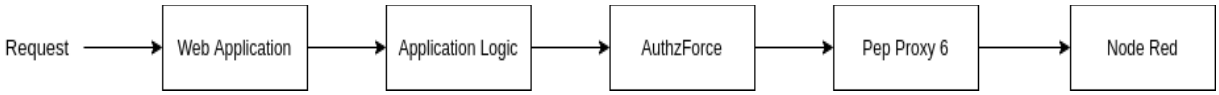
Κάθε πειραματική διαδικασία ακολουθεί το ίδιο μοτίβο, δηλαδή περιλαμβάνει 1000 αιτήματα με σκοπό να εξεταστεί η λειτουργικότητα και η απόδοση των εκάστοτε υπηρεσιών. Όλα τα πειράματα επαναλαμβάνονται για διαφορετικό αριθμό ταυτοχρονισμού, δηλαδή για διαφορετικό αριθμό ταυτόχρονων αιτημάτων. Τα αποτελέσματα των μετρήσεων αφορούν τον μέσο χρόνο εξυπηρέτησης ανά αίτημα και χωρίζονται σε ομάδες με βάση τον ταυτοχρονισμό που χρησιμοποιήθηκε για να σταλούν τα αιτήματα. Πιο αναλυτικά, τα αιτήματα στέλνονται ανά ένα, ανά εκατό, και ανά διακόσια. Με ταυτοχρονισμό ανά ένα σημαίνει ότι τα αιτήματα στέλνονται διαδοχικά το ένα μετά το άλλο, με ταυτοχρονισμό ανά εκατό σημαίνει ότι στέλνονται διαδοχικά σε ομάδες των εκατό αιτημάτων και ούτω καθεξής. Για κάθε αριθμό ταυτοχρονισμού παρατηρείται ο μέσος χρόνος εξυπηρέτησης ανά αίτημα, η κατανάλωση CPU και μνήμη RAM σε μορφή ποσοστού. Τα παραπάνω αποτελέσματα έχουν αποτυπωθεί σε πίνακες, οι οποίοι φαίνονται στη συνέχεια του Κεφαλαίου.

Κάθε πείραμα αντιπροσωπεύει ερωτήματα εγγεγραμμένων χρηστών του συστήματος και ακολουθεί μια σειρά βημάτων πριν εκτελεστεί το αίτημα. Τα κύρια σημεία πρόσβασης που δέχονται ερωτήματα είναι η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), η υπηρεσία Σύνθεσης Εφαρμογών (Mashup Service), καθώς και το Directory Service (κυρίως για

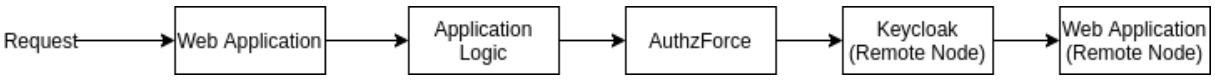
την δεύτερη ενότητα πειραμάτων). Επομένως, τα ερωτήματα εκτελούν τις εξής ακολουθίες βημάτων :



Εικόνα 4.1 : *Ακολουθία βημάτων που εκτελεί η πρώτη ενότητα ερωτημάτων.*



Εικόνα 4.2 : *Ακολουθία βημάτων που εκτελεί η πρώτη ενότητα ερωτημάτων.*



Εικόνα 4.3 : *Ακολουθία βημάτων που εκτελεί η δεύτερη ενότητα ερωτημάτων.*

Τα παραπάνω workflows είναι κοινά για όλα τα πειράματα που ακολουθούν.

Αιτήματα σε τοπικό κόμβο

Πείραμα 1

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα ανεύρεσης ιδιοτήτων συγκεκριμένου μοντέλου αισθητήρα, δηλαδή ψάχνει όλα τα χαρακτηριστικά μιας συγκεκριμένης οντότητας της Οντολογίας.
Υπηρεσίες : Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το αίτημα δρομολογείται στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και έπειτα φτάνει στην προστατευόμενη υπηρεσία. Εκεί η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων επεξεργάζεται το ερώτημα, το οποίο στη συνέχεια προωθεί στο τοπικό διακομιστή μεσολάβησης (Pep Proxy) της υπηρεσίας της Οντολογίας, ο οποίος με την σειρά του αναλαμβάνει να προωθήσει το ερώτημα στην Υπηρεσία Οντολογίας - αρχικά στο Jena API και στη συνέχεια στο γράφο της οντολογίας στην Virtuoso (όπου βρίσκονται αποθηκευμένα όλα τα μοντέλα αισθητήρων σε μορφή τριπλετών). Το ερώτημα εκτελείται από την οντολογία στην Virtuoso και η απάντηση επιστρέφει στο χρήστη ακολουθώντας την αντίθετη πορεία.
Λεπτομέρειες : Το ερώτημα πραγματοποιήθηκε πάνω σε μια συλλογή από ***68.138*** τριπλέτες, οι οποίες φιλοξενούνται στον γράφο της οντολογίας και αναπαριστούν όλα τα δεδομένα που προσθέσαμε στο σύστημα προκειμένου να εξετάσουμε την απόδοση του. Η αναζήτηση υλοποιείται σε μια βάση με μεγάλο πλήθος δεδομένων - ώστε η απόκριση να είναι ρεαλιστική - και επιτυγχάνεται χρησιμοποιώντας την γλώσσα SPARQL. Το αποτέλεσμα περιλαμβάνει τριπλέτες με κάθε μια να έχει ως υποκείμενο το ζητούμενο μοντέλο αισθητήρα.

Αίτημα REST : GET
http://34.76.235.126:8060/web_app/wot/retrieve_model/retrieve_model.php?q=ChaniaTemperatureSensor

Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στον Πίνακα 4.1.1. Παρατηρούμε ότι ο μέγιστος - με διαφορά - από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (17.662 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (9.121 ms). Το ποσοστό κατανάλωσης της RAM βρίσκεται σε χαμηλά επίπεδα, ενώ της CPU σε ψηλότερα ωστόσο δε φτάνει το 100%. Το ερώτημα, είναι γρήγορο και δεν απαιτεί μεγάλη κατανάλωση μνήμης RAM παρά τις συνθήκες φόρτου που επικρατούν στο σύστημα. Αξίζει να σημειωθεί ότι στη συγκεκριμένη περίπτωση, η απάντηση μπορεί να δοθεί και από την βάση MongoDB του Orion-LD Context Broker, αφού όλα τα μοντέλα αισθητήρων είναι αποθηκευμένα εκεί ως NGSI-LD οντότητες. Η επιλογή έγινε με βάση τον χρόνο εκτέλεσης του ερωτήματος. Παρατηρήθηκε ότι η υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών παρουσίαζε αρκετά αυξημένους χρόνους. Επομένως, αφού είδαμε πόσο “στοιχίζει” το ερώτημα σε κάθε υπηρεσία, διαλέξαμε να δρομολογείται στην πιο γρήγορη. Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης, προωθώντας το ίδιο ερώτημα στην υπηρεσία Publish - Subscribe, σημειώνονται στον Πίνακα 4.1.2.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	17.662	71	6,7
100	9.121	60	7.3
200	9.300	82.3	7.3

Πίνακας 4.1.1 : *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης ιδιοτήτων συγκεκριμένου μοντέλου αισθητήρα, στην υπηρεσία Ontology (Πείραμα 1).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	43	58	0.8
100	31.943	84	0.63
200	32.042	85	0,5

Πίνακας 4.1.2 : *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης ιδιοτήτων συγκεκριμένου μοντέλου αισθητήρα, στην υπηρεσία Publish-Subscribe(Πείραμα 1).*

Πείραμα 2

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα εύρεσης όλων των διαφορετικών μοντέλων των Πραγμάτων που έχουν αποθηκευτεί στο σύστημα και συγκεκριμένα στην Οντολογία του συστήματος (ώστε στη συνέχεια να επιλέξει ένα από αυτά και να ανακτήσει πληροφορία).
Υπηρεσίες : Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service). Το αίτημα ακολουθεί την ίδια πορεία με αυτήν στο Πείραμα 1, με την διαφορά ότι εδώ ανακτώνται όλα τα μοντέλα αισθητήρων που έχουν αποθηκευτεί στο γράφο της οντολογίας ως οντότητες.
Λεπτομέρειες : Το αίτημα που εξετάζεται στο παρόν πείραμα αφορά την εύρεση όλων των μοντέλων των αισθητήρων και γίνεται στην ίδια βάση δεδομένων με το Πείραμα 1, υπό τις ίδιες συνθήκες φόρτου. Το ερώτημα αυτό επιστρέφει μόνο τα ονόματα όλων των (διαφορετικών) μοντέλων, δηλαδή το μοναδικό αναγνωριστικό της οντότητας του κάθε μοντέλου στην οντολογία.
Αίτημα REST : GET
http://34.76.235.126:8060/web_app/wot/retrieve_model/things.php
Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στον Πίνακα 4.2.1. Παρατηρούμε ότι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (40.742 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (29.040 ms). Το ποσοστό κατανάλωσης της RAM εξακολουθεί βρίσκεται σε χαμηλά επίπεδα, ενώ της CPU σε ψηλότερα και φτάνει το 100% στη τελευταία περίπτωση. Το ερώτημα απαιτεί περισσότερο χρόνο σε σχέση με το Πείραμα 1, και αυτό είναι λογικό εφόσον πρόκειται να αναζητήσει όλες τις οντότητες τύπου αισθητήρα ενώ δεν σταματάει όταν βρει μια συγκεκριμένη. Παρόλα αυτά δεν απαιτεί μεγάλη κατανάλωση μνήμης RAM παρά τις συνθήκες φόρτου που επικρατούν στο σύστημα. Αξίζει να σημειωθεί ότι και σε αυτό το πείραμα, η απάντηση μπορεί να δοθεί από την βάση MongoDB του Orion-LD Context Broker, αφού όλα τα μοντέλα αισθητήρων είναι αποθηκευμένα εκεί ως NGSI-LD οντότητες. Αντίστοιχα, αφού εξετάσαμε πόσο “στοιχίζει” το ερώτημα σε κάθε υπηρεσία, διαλέξαμε να δρομολογείται στην πιο γρήγορη. Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης, προωθώντας το ίδιο ερώτημα στην υπηρεσία Publish - Subscribe, σημειώνονται στον Πίνακα 4.2.2.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	40.742	65	6,5
100	29.040	72	6,5
200	36.211	100	6,5

Πίνακας 4.2.1 : *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών μοντέλων των Πραγμάτων στην υπηρεσία Ontology (Πείραμα 2).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	169.089	63	0,8
100	124.14	88	0.8
200	130.047	100	0,1

Πίνακας 4.2.2 : *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών μοντέλων των Πραγμάτων στην υπηρεσία Publish-Subscribe (Πείραμα 2).*

Πείραμα 3

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα ενημέρωσης ενός μοντέλου που έχει αποθηκευτεί στο σύστημα. Ο χρήστης με αυτήν την ενημέρωση αλλάζει μόνο ένα attribute ενός συγκεκριμένου μοντέλου που έχει καταχωρηθεί στο σύστημα.
Υπηρεσίες : Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το αίτημα δρομολογείται στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και έπειτα φτάνει στην προστατευόμενη υπηρεσία. Η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων επεξεργάζεται το ερώτημα, το οποίο στη συνέχεια προωθεί στο τοπικό διακομιστή μεσολάβησης (Pep Proxy) της υπηρεσίας Orion-LD Context Broker. Ο Orion-LD αναλαμβάνει να εντοπίσει το εκάστοτε μοντέλο και να ενημερώσει τα αντίστοιχα χαρακτηριστικά του. Έπειτα, αφού το επιτρέπει ο διακομιστής μεσολάβησης, το ερώτημα δρομολογείται και στην Υπηρεσία της Οντολογίας - αρχικά στο Jena API και στη συνέχεια στο γράφο της οντολογίας στην Virtuoso (όπου βρίσκονται αποθηκευμένα όλα τα μοντέλα πραγμάτων σε μορφή τριπλετών). Εφόσον εντοπιστεί η τριπλέτα που πρέπει να υποστεί τροποποίηση, το αίτημα εκτελείται στο γράφο της οντολογίας και επιστρέφεται στο χρήστη ότι το αίτημα του πραγματοποιήθηκε με επιτυχία.

Αίτημα REST : POST
http://34.76.235.126:8060/web_app/wot/update_model/update.php
Το σώμα του αιτήματος περιλαμβάνει το όνομα/αναγνωριστικό του μοντέλου συσκευών που πρόκειται να ενημερωθεί, το όνομα του χαρακτηριστικού που υφίσταται ενημέρωση και τη νέα τιμή που παίρνει το χαρακτηριστικό.
Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στον Πίνακα 4.3. Παρατηρούμε ότι πάλι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (365,03 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (49,236 ms). Παρατηρούμε ότι το ποσοστό κατανάλωσης της RAM, βρίσκεται σε πολύ χαμηλά επίπεδα, ενώ της CPU σε ψηλότερα, χωρίς όμως να φτάνει το 100%. Παρατηρούμε ότι σε σχέση με τα προηγούμενα δύο πειραματα, οι χρόνοι απόκρισης εμφανίζουν σημαντική αύξηση. Ένας βασικός παράγοντας για αυτά τα αποτελέσματα είναι το γεγονός ότι πρόκειται για ερώτημα που απευθύνεται και απασχολεί περισσότερες υπηρεσίες.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	365.03	72	3
100	49.236	60	4,8
200	49.252	52	5,2

Πίνακας 4.3 : *Χρόνος απόκρισης σε 1000 ερωτήματα ενημέρωσης (update) ενός μοντέλου Πράγματος (Πείραμα 3).*

Πείραμα 4

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα για να προσθέσει ένα νέο μοντέλο της μορφής NGSI-LD στο σύστημα.
Υπηρεσίες : Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service). Το αίτημα ακολουθεί την ίδια πορεία με αυτήν στο Πείραμα 3, με την διαφορά ότι εδώ το ερώτημα πραγματοποιεί μια προσθήκη και όχι ενημέρωση συγκεκριμένου μοντέλου.
Αίτημα REST : POST
http://34.76.235.126:8060/web_app/wot/add_model/model.php
Το σώμα του αιτήματος περιλαμβάνει το μοντέλο NGSI-LD που δίνει ο χρήστης από το Web Application.

Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στον Πίνακα 4.4. Παρατηρούμε ότι ο μέγιστος από τους μέσους χρόνους εκτέλεσης εμφανίζεται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (286,994 ms) και είναι μειωμένος σε σχέση με αυτόν του πειράματος 3. Το παραπάνω μπορεί να θεωρηθεί λογικό αφού το αίτημα εκτελεί μια προσθήκη στις αντίστοιχες υπηρεσίες και δεν καθυστερεί κάνοντας κάποιου είδους αναζήτηση. Επιπλέον, ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (72,673 ms). Τέλος, τα ποσοστά κατανάλωσης CPU και RAM είναι σχεδόν σταθερά.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	286.994	75	5
100	72.673	70	5
200	73.025	60	5

Πίνακας 4.4 : *Χρόνος απόκρισης σε 1000 ερωτήματα εισαγωγής νέου μοντέλου Πράγματος (Πείραμα 4).*

Πείραμα 5

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα για εγγραφή μιας νέας συσκευή, προκειμένου το σύστημα να λαμβάνει δεδομένα - μετρήσεις από την συγκεκριμένη συσκευή.

Υπηρεσίες : Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το αίτημα δρομολογείται στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και έπειτα φτάνει στην προστατευόμενη υπηρεσία. Η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων επεξεργάζεται το ερώτημα, το οποίο στη συνέχεια προωθεί στο τοπικό διακομιστή μεσολάβησης (Pep Proxy) της υπηρεσίας IDAS. Η υπηρεσία IDAS αναλαμβάνει να εγγράψει την νέα συσκευή στη λίστα των εγγεγραμμένων συσκευών του συστήματος. Επιπλέον, η συγκεκριμένη ενέργεια απαιτεί την αποθήκευση της συσκευής στις βάσεις δεδομένων Cassandra, MySQL, στον Orion-LD και στην Οντολογία. Επομένως το αίτημα δρομολογείται σε όλες τις παραπάνω υπηρεσίες. Εφόσον η εγγραφή ολοκληρωθεί, επιστρέφεται στο χρήστη ότι το αίτημα του πραγματοποιήθηκε με επιτυχία. Μια πιο αναλυτική περιγραφή για την ροή εγγραφής νέας συσκευής διατυπώνεται στην ενότητα 3.3 (Διαγράμματα Ακολουθίας).

Αίτημα REST : POST
http://34.76.235.126:8060/web_app/wot/manageDevices/register_new_device/registerdevice.php

Το σώμα του αιτήματος περιλαμβάνει την JSON περιγραφή συσκευής που δίνει ο

χρήστης από το Web Application.

Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στον Πίνακα 4.5. Παρατηρούμε ότι πάλι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (391,025 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (108,866 ms). Παρατηρούμε ότι το ποσοστό κατανάλωσης της RAM, βρίσκεται σε πολύ χαμηλά επίπεδα, ενώ της CPU σε ψηλότερα και φτάνει το 100% στη τελευταία περίπτωση. Παρατηρούμε ότι σε σχέση με τα προηγούμενα δύο πειράματα, οι χρόνοι απόκρισης εμφανίζουν καθυστέρηση. Ένας βασικός παράγοντας για αυτά τα αποτελέσματα είναι το γεγονός ότι πρόκειται για ερώτημα που δρομολογείται και απασχολεί περισσότερες υπηρεσίες (IDAS, Orion-LD, Οντολογία, Cassandra, MySQL).

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	391.025	70	3
100	104.392	92	4
200	108.866	100	4

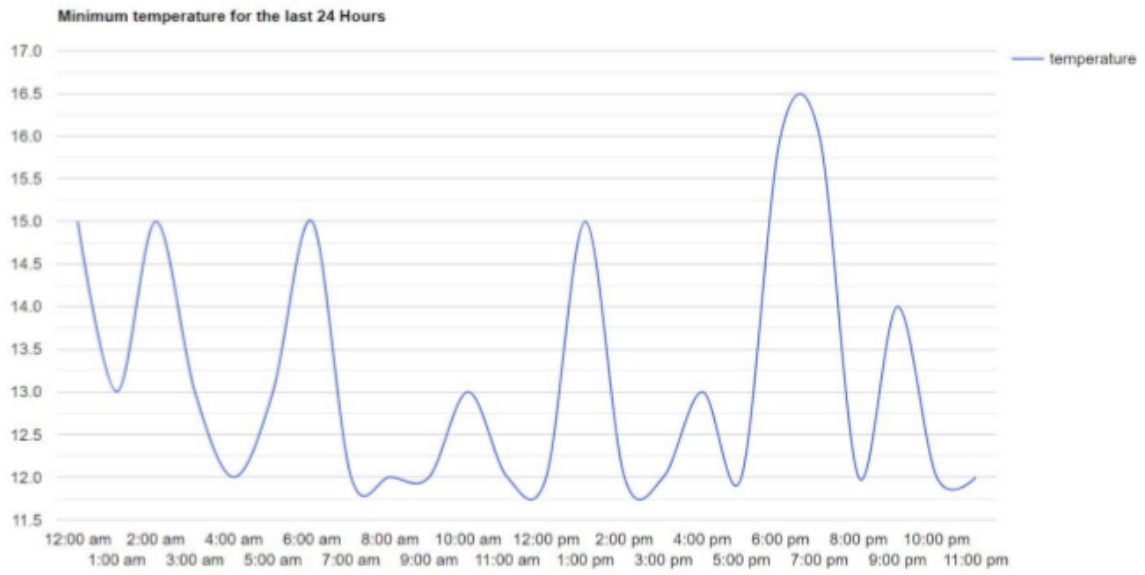
Πίνακας 4.5 : *Χρόνος απόκρισης σε 1000 ερωτήματα εγγραφής μιας νέας συσκευής (Πείραμα 5).*

Πείραμα 6

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής θέλει να δει την ελάχιστη τιμή της θερμοκρασίας ανά ώρα για 24 ώρες σε ένα σπίτι. Για τα πλαίσια του πειράματος, δημιουργήθηκε μία εφαρμογή με όνομα “Home150MinTemp”. Στην εικόνα 4.6.1 φαίνεται η JSON περιγραφή της εφαρμογής όπως είναι αποθηκευμένη στην υπηρεσία Node-RED, ενώ η γραφική απεικόνιση των τιμών παρουσιάζεται στην εικόνα 4.6.2.

```
{
    "appname": "Home150MinTemp",
    "appscope": "home",
    "info": [
        {
            "attributes": [
                "temperautre"
            ],
            "city": "Home150",
            "domains": [],
            "func": "MIN_24hr"
        }
    ]
}
```

Εικόνα 4.6.1 : *JSON περιγραφή της εφαρμογής Home150MinTemp.*



Εικόνα 4.6.2 : *Απεικόνιση εφαρμογής Home150MaxTemp που βλέπει ο χρήστης.*

Υπηρεσίες : Η συγκεκριμένη εφαρμογή αποτελείται μόνο από ιστορικά δεδομένα, επομένως η απόδοσή της εξαρτάται από την απόδοση της μη σχεσιακής βάσης δεδομένων PostgreSQL και της υπηρεσίας Node-red, καθώς εκεί είναι αποθηκευμένη η εφαρμογή.

Λεπτομέρειες : Το ερώτημα πραγματοποιήθηκε πάνω σε μια συλλογή από ***72.000*** μετρήσεις, οι οποίες φιλοξενούνται στη σχεσιακή βάση δεδομένων PostgreSQL και αναπαριστούν όλες τις τιμές συσκευών που προσθέσαμε στο σύστημα προκειμένου να εξετάσουμε την απόδοση του. Το ερώτημα πραγματοποιείται σε μια βάση με μεγάλο πλήθος δεδομένων - ώστε η απόκριση να είναι ρεαλιστική.

Αίτημα REST : GET http://34.76.235.126:1880/Home150MaxTemp

Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στον Πίνακα 4.6. Παρατηρούμε ότι πάλι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (96,941 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά 100 (55,712 ms). Επιπλέον, παρατηρούμε ότι η κατανάλωση της CPU και της RAM, βρίσκονται σε πολύ χαμηλά επίπεδα. Πρόκειται για ερώτημα που πραγματοποιεί αναζήτηση στην ιστορική βάση δεδομένων για συγκεκριμένη μέτρηση αισθητήρα. Στη βάση αυτή αποθηκεύονται μόνο οι μετρήσεις (οντότητες τύπου “Observation”), επομένως δεν έχει επιβαρυνθεί με πολλά επιπλέον δεδομένα όπως ο Orion Context Broker και η οντολογία. Ωστόσο, τα αποτελέσματα υπολογίζονται την στιγμή του αιτήματος, γεγονός που μπορεί να προσθέτει μια μικρή καθυστέρηση στο ερώτημα.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	96.941	10	1.8
100	55.712	15	1.4
200	56.243	20	1.4

Πίνακας 4.6 : Χρόνος απόκρισης σε 1000 ερωτήματα ανάκτησης-εκτέλεσης ιστορικής εφαρμογής (Πείραμα 6).

Πείραμα 7

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα εύρεσης όλων των διαφορετικών εφαρμογών που έχουν αποθηκευτεί στο σύστημα και συγκεκριμένα στην υπηρεσία Node Red (ώστε στη συνέχεια να επιλέξει μία από αυτές και να ανακτήσει πληροφορία).
Υπηρεσίες : Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το αίτημα δρομολογείται στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Node Red και έπειτα φτάνει στην προστατευόμενη υπηρεσία. Εκεί η Node Red επεξεργάζεται το ερώτημα και ανακτά από την βάση δεδομένων της όλες τις εφαρμογές που έχουν δημιουργηθεί. Η απάντηση επιστρέφει στο χρήστη ακολουθώντας την αντίθετη πορεία.
Λεπτομέρειες : Το ερώτημα πραγματοποιήθηκε πάνω σε μια συλλογή από ***3.000*** εφαρμογές, οι οποίες φιλοξενούνται στη βάση δεδομένων της υπηρεσίας Node Red.
Αίτημα REST : GET
http://34.76.235.126:8060/web_app/mashup/available_apps.php
Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στον Πίνακα 4.7. Παρατηρούμε ότι πάλι ο μέγιστος από τους μέσους χρόνους εκτέλεσης εμφανίζεται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (965,544 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά 100 (68,115 ms). Παρατηρούμε ότι και στο η κατανάλωση της CPU κυμαίνεται σε πολύ μεγάλα επίπεδα και της RAM σε μεσαία.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	965.544	100	12
100	68.115	100	12
200	73.058	100	12

Πίνακας 4.7 : Χρόνος απόκρισης σε 1000 ερωτήματα *εύρεσης όλων των διαφορετικών εφαρμογών.* (Πείραμα 7).

Αιτήματα σε τοπικό και απομακρυσμένο κόμβο

Πείραμα 8

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα εύρεσης όλων των διαφορετικών συσκευών που έχουν αποθηκευτεί στο σύστημα και συγκεκριμένα στην Οντολογία του συστήματος. Το αίτημα του χρήστη απευθύνεται τόσο στο τοπικό του κόμβο όσο και σε απομακρυσμένους.
Υπηρεσίες : Για την εκτέλεση του ερωτήματος σε τοπικό κόμβο, το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το Application Logic προωθεί το αίτημα στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και έπειτα φτάνει στην προστατευόμενη υπηρεσία. Εκεί η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων επεξεργάζεται το ερώτημα, το οποίο στη συνέχεια προωθεί στο τοπικό διακομιστή μεσολάβησης (Pep Proxy) της υπηρεσίας της Οντολογίας, ο οποίος με την σειρά του αναλαμβάνει να προωθήσει το ερώτημα στην Υπηρεσία Οντολογίας - αρχικά στο Jena API και στη συνέχεια στο γράφο της οντολογίας στην Virtuoso (όπου βρίσκονται αποθηκευμένα όλα τα μοντέλα συσκευών σε μορφή τριπλετών). Το ερώτημα εκτελείται από την οντολογία στην Virtuoso και η απάντηση επιστρέφει στο χρήστη ακολουθώντας την αντίθετη πορεία. Για την εκτέλεση του ερωτήματος σε απομακρυσμένο κόμβο, ο χρήστης θα πρέπει να διαλέξει σε ποιον κόμβο, από αυτούς που έχει δημιουργήσει συνδρομή και έχει άδεια πρόσβασης, θέλει να συνδεθεί. Αν η σύνδεση στο ξένο κόμβο πραγματοποιηθεί με επιτυχία, τότε ο χρήστης μπορεί να περιηγηθεί στη διαδικτυακή εφαρμογή του ξένου κόμβου και να επιλέξει να ανακτήσει μια λίστα με όλες τις συσκευές που είναι αποθηκευμένες στον εκάστοτε κόμβο. Η απομακρυσμένη διαδικτυακή εφαρμογή αναλαμβάνει να προωθήσει το παραπάνω αίτημα στην απομακρυσμένη εφαρμογή λογικής (Application Logic), η οποία με την σειρά της δρομολογεί το αίτημα στις κατάλληλες υπηρεσίες του ξένου κόμβου. Έπειτα το ερώτημα ανάκτησης συσκευών ακολουθεί την ίδια πορεία με αυτή που περιγράφηκε στο τοπικό κόμβο. Συγκεκριμένα, λαμβάνει χώρα ο μηχανισμός απόφασης (υπηρεσία AuthZForce), ο οποίος αν εγκρίνει το αίτημα (με βάση το ρόλο του χρήστη) η διαδικασία συνεχίζεται κατευθύνοντας το αίτημα στο διακομιστή μεσολάβησης (Pep Proxy) της απομακρυσμένης υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και από εκεί στη απομακρυσμένη υπηρεσία της οντολογίας στην οποία είναι αποθηκευμένα όλα τα μοντέλα συσκευών. Με αυτό τον τρόπο επιστρέφονται όλες οι συσκευές του απομακρυσμένου κόμβου και ο χρήστης μπορεί να επιλέξει όποια τον ενδιαφέρει και να ανακτήσει την NGSI-LD αναπαράσταση της.
Αίτημα REST : GET
http://34.76.235.126:8060/web_app/wot/ManageDevices/Search/devices.php

Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) τοπικού κόμβου αναγράφονται στον Πίνακα 4.8.1. Παρατηρούμε ότι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (41,107 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (30,201 ms). Το ποσοστό κατανάλωσης της RAM βρίσκεται σε χαμηλά επίπεδα, ενώ της CPU σε ψηλότερα και φτάνει το 100% στη τελευταία περίπτωση.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	41.107	56	8.3
100	30.201	65	8.3
200	31.549	100	1.5

Πίνακας 4.8.1: *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών συσκευών σε τοπικό κόμβο (Πείραμα 8).*

Το ίδιο ερώτημα μπορεί να εκτελεστεί και σε ξένο κόμβο, υπό την προϋπόθεση ότι ο χρήστης έχει άδεια πρόσβασης στον εκάστοτε κόμβο. Προκειμένου να εξετάσουμε την απόδοση του συστήματος σε αυτή την περίπτωση, δρομολογήσαμε το ερώτημα σε δύο, τρεις και τέσσερις κόμβους, ενώ ο αριθμός αιτημάτων μοιράστηκε ανάλογα με τον αριθμό κόμβων. Δηλαδή για δύο κόμβους προωθούνται 500 αιτήματα ανά κόμβο , σε τρεις κόμβους προωθούνται 333 αιτήματα ανά κόμβο και σε τέσσερις κόμβους προωθούνται 250 αιτήματα ανά κόμβο. Τα αποτελέσματα για τους μέσους χρόνους εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στους ακόλουθους πίνακες. Παρατηρούμε ότι αυξάνοντας τον αριθμό κόμβων, ο μέγιστος από τους μέσους χρόνους εκτέλεσης παραμένει στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα, ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό. Επιπλέον, παρατηρείται αύξηση του χρόνου εκτέλεσης σε σχέση με τον Πίνακα 4.8.1. Βασικός παράγοντας για αυτό αποτελεί η ταυτοποίηση και η εξουσιοδότηση του χρήστη, που μεσολαβεί όταν το αίτημα δρομολογείται σε ένα ξένο κόμβο. Πριν εκτελεστεί το αίτημα, πραγματοποιείται ένας έλεγχος εγκυρότητας του χρήστη και των δικαιωμάτων του με αποτέλεσμα να προστίθεται μια καθυστέρηση, η οποία δεν υπάρχει στην περίπτωση που το αίτημα εκτελείται σε τοπικό κόμβο.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	46.712	50	6
100	38.161	85	6
200	38.651	100	7

Πίνακας 4.8.2: *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών συσκευών σε δύο κόμβους(Πείραμα 8).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	43.110	70	5
100	40.009	60	6
200	40.526	63	6

Πίνακας 4.8.3: *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών συσκευών σε τρεις κόμβους(Πείραμα 8).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	44.547	65	6
100	40.400	72	6
200	40.501	70	6

Πίνακας 4.8.4: *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών συσκευών σε τέσσερις κόμβους(Πείραμα 8).*

Πείραμα 9

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα ανάκτησης τρέχουσας τιμής μιας συγκεκριμένης συσκευής. Το αίτημα του χρήστη απευθύνεται τόσο σε τοπικό κόμβο όσο και σε απομακρυσμένους.
Υπηρεσίες : Για την εκτέλεση του ερωτήματος σε τοπικό κόμβο, το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το Application Logic προωθεί το αίτημα στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και έπειτα φτάνει στην προστατευόμενη υπηρεσία. Η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων επεξεργάζεται το ερώτημα, το οποίο στη συνέχεια προωθεί στο τοπικό διακομιστή μεσολάβησης (Pep Proxy) της υπηρεσίας της Οντολογίας, ο οποίος με την σειρά του αναλαμβάνει να προωθήσει το ερώτημα στην Υπηρεσία Οντολογίας - αρχικά στο Jena API και στη συνέχεια στο γράφο της οντολογίας στην Virtuoso (όπου βρίσκονται αποθηκευμένα όλα τα μοντέλα συσκευών μαζί με τις τρέχουσες τιμές τους σε μορφή τριπλετών). Το ερώτημα εκτελείται από την οντολογία στην Virtuoso και η απάντηση επιστρέφει στο χρήστη ακολουθώντας την αντίθετη πορεία. Αντίστοιχα για την εκτέλεση του ερωτήματος σε απομακρυσμένο κόμβο, ο χρήστης θα πρέπει να διαλέξει σε ποιον κόμβο, από αυτούς που έχει δημιουργήσει συνδρομή και έχει άδεια πρόσβασης, θέλει να συνδεθεί. Αν η σύνδεση στο ξένο κόμβο πραγματοποιηθεί με επιτυχία, τότε ο χρήστης μπορεί να περιηγηθεί στη διαδικτυακή εφαρμογή του ξένου κόμβου και να επιλέξει να ανακτήσει την τρέχουσα τιμή μιας συγκεκριμένης συσκευής που διαθέτει ο απομακρυσμένος κόμβος. Έτσι, η απομακρυσμένη διαδικτυακή εφαρμογή αναλαμβάνει να προωθήσει το παραπάνω αίτημα στην απομακρυσμένη εφαρμογή λογικής (Application Logic), η οποία με την σειρά της δρομολογεί το αίτημα

στις κατάλληλες υπηρεσίες του ξένου κόμβου. Άφου λάβει χώρα ο μηχανισμός απόφασης (υπηρεσία AuthZForce) και το αίτημα εγκριθεί, το ερώτημα κατευθύνεται στο διακομιστή μεσολάβησης (Pep Proxy) της απομακρυσμένης υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και από εκεί στη απομακρυσμένη υπηρεσία της Οντολογίας - αρχικά στο Jena API και στη συνέχεια στο γράφο της οντολογίας στην Virtuoso (όπου βρίσκονται αποθηκευμένα όλα τα μοντέλα συσκευών μαζί με τις τρέχουσες τιμές τους σε μορφή τριπλετών). Με αυτό τον τρόπο επιστρέφεται στο χρήστη του τοπικο κόμβου, η τρέχουσα τιμή της εκάστοτε συσκευής του απομακρυσμένου κόμβου.

Αίτημα REST : GET http://34.76.235.126:8060/web_app/wot/retrieve_current_value.php?q=ChaniaTemperatureObservation

Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) τοπικού κόμβου αναγράφονται στον Πίνακα 4.9.1. Παρατηρούμε ότι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται πάλι στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (20,485 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (11,756 ms). Η κατανάλωση της CPU φτάνει το 100% σε όλες τις περιπτώσεις, ενώ το ποσοστό κατανάλωσης της RAM βρίσκεται σε χαμηλά επίπεδα. Πρόκειται για ένα γρήγορο ερώτημα που δεν απαιτεί μεγάλη κατανάλωση μνήμης RAM παρά τις συνθήκες φόρτου που επικρατούν στο σύστημα.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	20.485	100	5
100	11.756	100	6
200	11.875	100	6

Πίνακας 4.9.1: *Χρόνος απόκρισης σε 1000 ερωτήματα ανάκτησης τρέχουσας τιμής από συσκευή τοπικού κόμβου (Πείραμα 9).*

Το ίδιο ερώτημα μπορεί να εκτελεστεί και σε ξένο κόμβο, υπό την προϋπόθεση ότι ο χρήστης έχει άδεια πρόσβασης στον εκάστοτε κόμβο. Όπως και στο Πείραμα 8, δρομολογήσαμε το ερώτημα σε δύο, τρεις και τέσσερις κόμβους, ενώ ο αριθμός αιτημάτων μοιράστηκε ανάλογα με τον αριθμό κόμβων. Τα αποτελέσματα για τους μέσους χρόνους εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στους ακόλουθους πίνακες. Παρατηρούμε ότι αυξάνοντας τον αριθμό κόμβων, ο μέγιστος από τους μέσους χρόνους εκτέλεσης παραμένει στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα, ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό. Και σε αυτή την περίπτωση, παρατηρείται αύξηση του χρόνου εκτέλεσης σε σχέση με τον Πίνακα 4.9.1 γεγονός που οφείλεται στη ταυτοποίηση και

εξουσιοδότηση του χρήστη, που μεσολαβεί όταν το αίτημα δρομολογείται σε ένα ξένο κόμβο. Ωστόσο για αριθμό κόμβων μεγαλύτερο του δύο, ο χρόνος εκτέλεσης κυμαίνεται στο ίδιο εύρος και δεν παρουσιάζει μεγάλες αλλαγές.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	27.340	65	5
100	24.621	68	6
200	25.114	66	6

Πίνακας 4.9.2: *Χρόνος απόκρισης σε 1000 ερωτήματα ανάκτησης τρέχουσας τιμής συσκευής σε δύο κόμβους (Πείραμα 9).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	30.945	70	6
100	26.562	72	6
200	26.646	70	6

Πίνακας 4.9.3: *Χρόνος απόκρισης σε 1000 ερωτήματα ανάκτησης τρέχουσας τιμής συσκευής σε τρεις κόμβους (Πείραμα 9).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	32.297	75	6
100	28.759	78	6
200	28.842	76	6

Πίνακας 4.9.4: *Χρόνος απόκρισης σε 1000 ερωτήματα ανάκτησης τρέχουσας τιμής συσκευής σε τέσσερις κόμβους (Πείραμα 9).*

Πείραμα 10

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα συνδρομής σε υπάρχουσα συσκευή του συστήματος. Το αίτημα του χρήστη απευθύνεται τόσο σε τοπικό κόμβο όσο και σε απομακρυσμένους.

Υπηρεσίες : Για την εκτέλεση του ερωτήματος σε τοπικό κόμβο, το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το Application Logic προωθεί το αίτημα στον διακομιστή μεσολάβησης Pep Proxy της Υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και έπειτα φτάνει στην προστατευόμενη υπηρεσία. Η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων επεξεργάζεται το ερώτημα, το οποίο στη συνέχεια προωθεί στο τοπικό διακομιστή μεσολάβησης (Pep Proxy) της υπηρεσίας της Διαχείρισης Συμβάντων και Συνδρομών, η οποία είναι υπεύθυνη για την δημιουργία συνδρομών σε οντότητες του

συστήματος. Αντίστοιχα για την εκτέλεση του ερωτήματος σε απομακρυσμένο κόμβο, ο χρήστης θα πρέπει να διαλέξει σε ποιον κόμβο, από αυτούς που έχει δημιουργήσει συνδρομή και έχει άδεια πρόσβασης, θέλει να συνδεθεί. Αν η σύνδεση στο ξένο κόμβο πραγματοποιηθεί με επιτυχία, τότε ο χρήστης μπορεί να περιηγηθεί στη διαδικτυακή εφαρμογή του ξένου κόμβου και να επιλέξει να δημιουργήσει συνδρομή σε συγκεκριμένη συσκευή του απομακρυσμένου κόμβου. Έτσι, η απομακρυσμένη διαδικτυακή εφαρμογή αναλαμβάνει να προωθήσει το παραπάνω αίτημα στην απομακρυσμένη εφαρμογή λογικής (Application Logic), η οποία με την σειρά της δρομολογεί το αίτημα στο διακομιστή μεσολάβησης (Pep Proxy) της απομακρυσμένης υπηρεσίας Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), αφού γίνει έλεγχος των δικαιωμάτων του χρήστη (υπηρεσία AuthZForce). Έπειτα, όπως και στο τοπικό κόμβο, το ερώτημα δρομολογείται στον απομακρυσμένο διακομιστή μεσολάβησης (Pep Proxy) της υπηρεσίας της Διαχείρισης Συμβάντων και Συνδρομών. Με αυτό τον τρόπο ο χρήστης μπορεί να χρησιμοποιεί την εκάστοτε “ξένη” συσκευή και να λαμβάνει ειδοποιήσεις σε περίπτωση νέας μέτρησης.

Αίτημα REST : POST
http://34.76.235.126:8060/web_app/wot/subscribe.php?q=ChaniaTemperatureObservation

Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) τοπικού κόμβου αναγράφονται στον Πίνακα 4.10.1. Παρατηρούμε ότι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται πάλι στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (25,679 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (14,570 ms). Το ποσοστό κατανάλωσης της RAM βρίσκεται σε χαμηλά επίπεδα, ενώ της CPU σε ψηλοτερα χωρίς όμως να φτάνει το 100%. Το συγκεκριμένο ερώτημα εξετάζει την απόδοση της υπηρεσίας Orion-LD στην περίπτωση που δρα ως συνδρομητής και αποδεικνύει ότι πρόκειται για μια γρήγορη υπηρεσία του συστήματος.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	25.679	44	3.1
100	14.570	62	3.1
200	15.762	88	3.1

Πίνακας 4.10.1: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε συσκευή τοπικού κόμβου (Πείραμα 10).*

Αντίστοιχα με τα προηγούμενα πειράματα της τρέχουσας κατηγοριας, το ερώτημα πραγματοποιήθηκε και σε απομακρυσμένους κόμβους. Ακολουθώντας την ίδια διαδικασία, μοιράσαμε τα αιτήματα ανάλογα τον αριθμό κόμβων. Τα αποτελέσματα για τους μέσους χρόνους εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στους ακόλουθους πίνακες. Παρατηρούμε ότι αυξάνοντας τον αριθμό κόμβων, ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται πάλι

στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα, ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό. Και σε αυτή την περίπτωση, παρατηρείται αύξηση του χρόνου εκτέλεσης σε σχέση με τον Πίνακα 4.10.1 γεγονός που οφείλεται στη ταυτοποίηση και εξουσιοδότηση του χρήστη, που μεσολαβεί όταν το αίτημα δρομολογείται σε ένα ξένο κόμβο. Ωστόσο για αριθμό κόμβων μεγαλύτερο του δύο, τα ποσοστά κατανάλωσης CPU και RAM, παρουσίαζουν παρόμοια συμπεριφορά.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	48.505	22	2.7
100	46.647	62	3
200	48.228	60	3

Πίνακας 4.10.2: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε συσκευή δύο κόμβων (Πείραμα 10).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	54.087	15	3
100	52.517	66	3
200	53.916	70	3

Πίνακας 4.10.3: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε συσκευή τριων κόμβων (Πείραμα 10).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	62.103	14	3.5
100	60.738	75	3.5
200	61.726	73	3.5

Πίνακας 4.10.4: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε συσκευή τεσσάρων κόμβων (Πείραμα 10).*

Πείραμα 11

Σενάριο : Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα συνδρομής σε υπάρχουσα εφαρμογή του συστήματος. Το αίτημα του χρήστη απευθύνεται τόσο σε τοπικό κόμβο όσο και σε απομακρυσμένους.

Υπηρεσίες : Για την εκτέλεση του ερωτήματος σε τοπικό κόμβο, το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στη Λογική Εφαρμογής (Application Logic), η οποία είναι υπεύθυνη να προωθήσει το αίτημα στη κατάλληλη υπηρεσία. Το αίτημα ακολουθεί την ίδια πορεία με αυτήν στο Πείραμα 10.

Για την εκτέλεση του ερωτήματος σε απομακρυσμένο κόμβο, ο χρήστης θα πρέπει να διαλέξει σε ποιον κόμβο, από αυτούς που έχει δημιουργήσει συνδρομή και έχει άδεια πρόσβασης, θέλει να συνδεθεί. Αν η σύνδεση στο ξένο κόμβο πραγματοποιηθεί με επιτυχία, τότε ο χρήστης μπορεί να περιηγηθεί στη διαδικτυακή εφαρμογή του ξένου κόμβου και να επιλέξει να δημιουργήσει συνδρομή σε συγκεκριμένη εφαρμογή του απομακρυσμένου κόμβου. Στη συνέχεια το αίτημα ακολουθεί την ίδια πορεία με αυτήν που περιγράφεται στο Πείραμα 10 για απομακρυσμένο κόμβο.
Αίτημα REST : POST
http://34.76.235.126:8060/web_app/wot/application_subscribe.php?q=Home150MinTemp
Αποτελέσματα και σχολιασμός : Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) τοπικού κόμβου αναγράφονται στον Πίνακα 4.11.1. Παρατηρούμε ότι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται πάλι στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (40,561 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (30,587 ms). Το ποσοστό κατανάλωσης της RAM βρίσκεται σε χαμηλά επίπεδα, ενώ της CPU σε ψηλοτερα φτάνοντας το 100%. Το συγκεκριμένο ερώτημα εξετάζει την απόδοση της υπηρεσίας Orion-LD στην περίπτωση που δρα ως συνδρομητής σε οντότητες τύπου εφαρμογή και αποδεικνύει ότι πρόκειται για μια γρήγορη υπηρεσία του συστήματος.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	40.561	66	1
100	30.587	100	1
200	30.640	100	1

Πίνακας 4.11.1: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε εφαρμογή τοπικού κόμβου (Πείραμα 11).*

Αντίστοιχα με τα προηγούμενα πειράματα της τρέχουσας κατηγοριας, το ερώτημα πραγματοποιήθηκε και σε απομακρυσμένους κόμβους. Ακολουθώντας την ίδια διαδικασία, μοιράσαμε τα αιτήματα ανάλογα τον αριθμό κόμβων. Τα αποτελέσματα για τους μέσους χρόνους εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) αναγράφονται στους ακόλουθους πίνακες. Παρατηρούμε ότι αυξάνοντας τον αριθμό κόμβων, ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται πάλι στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα, ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό. Και σε αυτή την περίπτωση, παρατηρείται αύξηση του χρόνου εκτέλεσης σε σχέση με τον Πίνακα 4.11.1 γεγονός που οφείλεται στη ταυτοποίηση και εξουσιοδότηση του χρήστη, που μεσολαβεί όταν το αίτημα δρομολογείται σε ένα ξένο κόμβο. Ωστόσο για αριθμό κόμβων μεγαλύτερο του δύο, ο χρόνος εκτέλεσης κυμαίνεται στο ίδιο εύρος και δεν παρουσιάζει μεγάλες αλλαγές, ενώ τα ποσοστά κατανάλωσης CPU και RAM, παρουσίαζουν παρόμοια συμπεριφορά.

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	54.374	100	6
100	50.951	100	6
200	51.657	100	6

Πίνακας 4.11.2: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε εφαρμογή δύο κόμβων (Πείραμα 11).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	57.014	100	6
100	53.628	100	6
200	53.734	100	6

Πίνακας 4.11.3: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε εφαρμογή τριων κόμβων (Πείραμα 11).*

concurrency	time/req (ms)	CPU load (%)	RAM usage (%)
1	57.502	100	6
100	54.258	100	6
200	54.366	100	6

Πίνακας 4.11.4: *Χρόνος απόκρισης σε 1000 ερωτήματα δημιουργίας συνδρομής σε εφαρμογή τεσσάρων κόμβων (Πείραμα 11).*

Κεφάλαιο 5

Συμπεράσματα - Μελλοντικές Επεκτάσεις

5.1 Συμπεράσματα

Η παρούσα εργασία, σχεδιάζει και παρουσιάζει μια Κατανεμημένη Αρχιτεκτονική Υπηρεσιών Νέφους για το Σημασιολογικό Δίκτυο των Πραγμάτων. Το Fi-SWoT υιοθετεί μια Υπηρεσιοκεντρική Αρχιτεκτονική, διευκολύνοντας σε μεγάλο βαθμό την επικοινωνία μεταξύ των υπηρεσιών και συνεπώς την ανάπτυξη της δομής που είναι υπεύθυνη για την ενορχήστρωση των υπηρεσιών (Application Logic) σε κάθε σύστημα cloud. Μεγάλο πλεονέκτημα της συγκεκριμένης αρχιτεκτονικής αποτέλεσε η ευελιξία στη χρήση διαφορετικών γλωσσών προγραμματισμού για να υλοποιηθούν διαφορετικές λειτουργίες του συστήματος καθώς και ευκολία στην παρέμβαση - τροποποίηση υπηρεσιών μεμονωμένα, δίχως να επηρεάζεται το συνολικό σύστημα. Επομένως, το σύστημα βασίζεται σε μια επεκτάσιμη και εύκολα τροποποιήσιμη αρχιτεκτονική.

Η ανάπτυξη κάθε κόμβου του Fi-SWoT πραγματοποιήθηκε μέσω του οικοσυστήματος FIWARE, προσφέροντας το πλεονέκτημα της χρήσης έτοιμων υπηρεσιών (που παρέχει το FIWARE). Η υλοποίηση της εργασίας ξεκίνησε πάνω σε μια πολύ σημαντική βάση, καθώς υπήρχαν διαθέσιμες υπηρεσίες όπως η Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών, η Υπηρεσία Cygnus-LD, την υπηρεσία ελέγχου εξουσιοδότησης (AuthZforce PDP), κ.ά.

Αξίζει να αναφερθεί ότι η υπηρεσία Ιστορικών Δεδομένων, που απαρτίζεται από την υπηρεσία Cygnus-LD και τη σχεσιακή βάση δεδομένων PostgreSQL, δεν αποτελεί μοναδική λύση. Προκειμένου η υπηρεσία Διαχείρισης Ιστορικών Δεδομένων να είναι συμβατή με τον μοντέλο πληροφορίας NGSI-LD, ως δεύτερη λύση προτείνεται η χρήση ενός Apache Flume, το οποίο λειτουργεί σαν “ουρά” και μεταφέρει τα δεδομένα ανεξαρτήτως της μορφής τους σε μια προκαθορισμένη βάση δεδομένων. Με αυτόν τον τρόπο, τα δεδομένα να αποθηκευτούν σε μια μη- σχεσιακή βάση δεδομένων MongoDB ενώ η ανάκτηση ιστορικών τιμών να πραγματοποιηθεί με MongoDB Query Generator.

Αξιολογώντας την υπηρεσία Οντολογίας μέσω της πειραματικής διαδικασίας, και συγκεκριμένα την υπηρεσία JENA API και την επικοινωνία της με την οντολογία στο ΣΔΒΔ Virtuoso, αυτή κρίνεται πολύ ικανοποιητική σχετικά με το χρόνο απόκρισης και την κατανάλωση πόρων. Πέραν αυτού, επιτυγχάνει να καταστήσει γρήγορη και απλή την επικοινωνία της οντολογίας του συστήματος με όλο το υπόλοιπο σύστημα κι επομένως να αξιοποιήσει τις τεχνολογίες του Σημασιολογικού Ιστού για τις ανάγκες της πλατφόρμας.

Ένα από τα κύρια χαρακτηριστικά του Fi-SWoT, είναι η χρήση της βάσης δεδομένων Apache Cassandra. Η τελευταία συμβάλλει στη δημιουργία κατανεμημένου συστήματος, ενώ είναι υπεύθυνη για την αποθήκευση και την ανακάλυψη κόμβων που είναι καταχωρημένα στο Fi-SWoT. Ωστόσο μια επέκταση στη σχεδίαση του παρόντος συστήματος, αποτελεί η προσθήκη υπηρεσίας η οποία θα είναι υπεύθυνη για την εισαγωγή νέου κόμβου στη Cassandra. Σε αυτή την περίπτωση, αρκεί το αίτημα προσθήκης νέου κόμβου να δρομολογηθεί στο διαχειριστή ενός υπάρχοντος κόμβου περιλαμβάνοντας στοιχεία του εκάστοτε κόμβου που ζητείται να συνδεθεί. Αν ο διαχειριστής εγκρίνει το αίτημα τότε η λειτουργία ολοκληρώνεται.

Επιπλέον, το Fi-SWoT υποστηρίζει μηχανισμούς ασφαλείας. Συγκεκριμένα, η η ταυτοποίηση και η προστασία πιστοποιητικών των χρηστών μειώνει την πιθανότητα έκθεσης “ευαίσθητων” πληροφοριών. Επίσης, η χρήση της υπηρεσίας PEP Proxy του FIWARE, που επιτρέπει την πρόσβαση σε πόρους του συστήματος μόνο σε συγκεκριμένους εγγεγραμμένους - εξουσιοδοτημένους χρήστες και υπηρεσίες, συμβάλλει στον ασφαλή σχεδιασμό του συστήματος.

Συνοψίζοντας τα παραπάνω συμπεράσματα, το σύστημα που αναπτύχθηκε είναι σε θέση να διαχειρίζεται μεγάλο αριθμό συσκευών,χρηστών ενώ οι λειτουργίες που παρέχει εκτελούνται ικανοποιητικά σε πραγματικό χρόνο. Επιπλέον, το Fi-SWoT επιτρέπει την αλληλεπίδραση συστημάτων Σημασιολογικού Ιστου των Πραγμάτων με ασφάλεια και την ανταλλαγή δεδομένων προκειμένου να αυξηθούν οι δυνατότητες δημιουργίας εφαρμογών, μεγαλύτερης κλίμακας, από συσκευές και υπηρεσίες ενός ή περισσότερων κόμβων.

5.2 Μελλοντικές Επεκτάσεις

Η υλοποίηση του παρόντος συστήματος, όπως αυτό σχεδιάστηκε κατά την εκπόνηση της εργασίας, μπορεί να θεωρηθεί ότι εν τέλει ικανοποιεί τις προδιαγραφές που τέθηκαν, με αποτέλεσμα την ανάπτυξη ενός πλήρως επεκτάσιμου κατανεμημένου συστήματος Σημασιολογικού Ιστού στο Υπολογιστικό Νέφος. Ωστόσο, η εργασία διεκπεραιώθηκε μέσα σε ένα συγκεκριμένο χρονικό πλαίσιο, με αποτέλεσμα να τίθενται ζητήματα επέκτασης τα οποία περιγράφονται παρακάτω :

- **Ανάπτυξη του συστήματος Fi-SWoT σε Kubernetes :** Το Kubernetes προσφέρει αυτοματοποιημένη ανάπτυξη, κλιμάκωση και διαχείριση πολλαπλών εφαρμογών σε εικονοποιημένα περιβάλλοντα χρησιμοποιώντας Containers. Αποτελεί δημοφιλή μηχανισμό διαχείρισης και ενορχήστρωσης Containers ενώ ταυτόχρονα συμβάλλει στην διανομή υπολογιστικών πόρων στις εφαρμογές μέσω ενός μηχανισμού χρονοδρομολόγησης. Επομένως, η ανάπτυξη του Fi-SWoT σε Kubernetes θα προσφέρει καλύτερη διαχείριση και αξιοποίηση υπολογιστών πόρων, καθώς και περαιτέρω κλιμάκωση.
- **Εφαρμογή Serverless Computing σε υπηρεσίες του Cloud:** Το Serverless Computing ή αλλιώς πάροχος χωρίς διακομιστή, είναι ένα μοντέλο εκτέλεσης για το Cloud στο οποίο ένας πάροχος cloud εκχωρεί δυναμικά - και στη συνέχεια χρεώνει τον χρήστη - μόνο τους υπολογιστικούς πόρους και τον αποθηκευτικό χώρο που απαιτούνται για την εκτέλεση ενός συγκεκριμένου κώδικα. Οι συναρτήσεις Serverless Computing βασίζονται σε συμβάντα, που σημαίνει ότι ο κώδικας καλείται μόνο όταν ενεργοποιείται από ένα αίτημα. Σε αυτή την περίπτωση ο πάροχος χρεώνει μόνο τον υπολογιστικό χρόνο που χρησιμοποιείται η συνάρτηση Serverless Computing, αντί για μια σταθερή μηνιαία χρέωση. Οι συναρτήσεις αυτές μπορούν να αποτελούν τμήματα μιας μεγαλύτερης εφαρμογής, και να αλληλεπιδρούν με άλλο κώδικα που εκτελείται σε ένα Container. Επομένως, προσφέρει στους χρήστες χαμηλό κόστος, ενώ οι προγραμματιστές δεν χρειάζεται να ανησυχούν για την υποκείμενη υποδομή εκτέλεσης του κώδικα.
- **Automatic scaling :** Η αυτόματη κλιμάκωση είναι μια δυνατότητα του υπολογιστικού νέφους που επιτρέπει στους οργανισμούς να κλιμακώνουν αυτόματα υπηρεσίες cloud, όπως είναι οι εικονικές μηχανές, με βάση καθορισμένες καταστάσεις όπως τα επίπεδα χρήσης και κίνησης. Το Google Cloud Platform (GCP) που χρησιμοποιείται στη παρούσα εργασία, παρέχει εργαλεία αυτόματης κλιμάκωσης, τα οποία προσφέρουν δυναμική διαχείριση και κατανομή πόρων, με στόχο η εφαρμογή να εξυπηρετεί το φορτίο της.

Βιβλιογραφία - Αναφορές

[1] Α. Τζαβάρας, Υποστήριξη Λειτουργικότητας Σημασιολογικού Ιστού σε Περιβάλλον Διαδικτύου των Πραγμάτων και Υπολογιστικού Νέφους,Διπλωματική Εργασία, Πολυτεχνείο Κρήτης, Οκτώβριος 2019.

[2] Σ. Μποτωνάκης, Σύνθεση Υπηρεσιών για την Δημιουργία Εφαρμογών σε Περιβάλλον Σημασιολογικού Διαδικτύου των Πραγμάτων, Διπλωματική Εργασία, Πολυτεχνείο Κρήτης, Οκτώβριος 2019.

[3] Ν. Ζαχαρία, LINCA: Service Oriented Architecture for Supporting Multi-Cloud Ecosystems, Διπλωματική Εργασία, Πολυτεχνείο Κρήτης, Οκτώβριος 2019.

[4] Dominique Guinard, Vlad Trifa, Towards the Web of Things: Web Mashups for Embedded Devices, in: MEM 2009 in Proceedings of WWW 2009. ACM
https://webofthings.org/wp-content/uploads/2009/03/guinard_trifa_webofthings_mashup.pdf

[5] Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi, Kerry Taylor, IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics.
http://epubs.surrey.ac.uk/841613/1/IoT-Lite.pdf

[6] Payam Barnaghi, Frieder Ganz, Hamidreza Abangar, Mirko Presser, and Klaus Moessner, Sense2Web: A Linked Data Platform for Semantic Sensor Networks, Centre for Communication Systems Research, University of Surrey, Guildford, GU2 7XH, United Kingdom.
http://semantic-web-journal.org/sites/default/files/swj189.pdf

[7] MongoDB Vs PostgreSQL: A comparative study on performance aspects
https://link.springer.com/article/10.1007/s10707-020-00407-w

[8] Comparing MongoDB vs PostgreSQL
https://www.mongodb.com/compare/mongodb-postgresql