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


# Υποστήριξη Λειτουργικότητας Σημασιολογικού Ιστού σε Περιβάλλον Διαδικτύου των Πραγμάτων και Υπολογιστικού Νέφους

# (Supporting Semantic Web of Things Functionality in the Cloud)

Διπλωματική Εργασία

Τζαβάρας Αιμίλιος

**Εξεταστική επιτροπή:**

Πετράκης Ευριπίδης, Καθηγητής (Επιβλέπων)

Σαμολαδάς Βασίλης, Αναπληρωτής Καθηγητής

Δεληγιαννάκης Αντώνιος, Αναπληρωτής Καθηγητής

Χανιά, Οκτώβριος 2019

# Περίληψη

Ο κόσμος κατευθύνεται προς την επικοινωνία τύπου μηχανής, όπου οτιδήποτε (από τους έξυπνους αισθητήρες μέχρι τα καθημερινά προϊόντα στα σουπερμάρκετ) θα είναι συνδεδεμένο στο Διαδίκτυο. Η τεχνολογία Σημασιολογικού Ιστού του Διαδικτύου των Πραγμάτων (Semantic Web of Things ή SWoT) προτείνει το σχεδιασμό διαλειτουργικών υπηρεσιών ή διαδικτυακών εφαρμογών του Ιστού των Πραγμάτων με τη χρήση τεχνολογιών του Σημασιολογικού Ιστού. Το SWoT αποτελεί μια σημασιολογική επέκταση του Διαδικτύου των Πραγμάτων που επιτρέπει στις εφαρμογές να μοιράζονται περιεχόμενο και υπηρεσίες πέραν των ορίων τους ή, ακόμα πιο σημαντικό, να αναπτύσσουν νέες εφαρμογές ως σύνθεση υφιστάμενων (π.χ. χρησιμοποιώντας mashups). Η βασική ιδέα είναι να πετύχουμε όλα αυτά να γίνονται αυτόματα ή με ελάχιστη ανθρώπινη παρέμβαση. Πρέπει να εφαρμοστούν εργαλεία που είναι ικανά να κατανοήσουν τη σημασία των εφαρμογών του Διαδικτύου (δηλαδή δεδομένα και υπηρεσίες) και να αιτιολογούν το περιεχόμενό τους, προκειμένου να υλοποιηθεί το όραμα της τεχνολογίας του Σημασιολογικού Ιστού του Διαδικτύου. Όπως και στο Σημασιολογικό ιστό, οι ορισμοί των εννοιών και των ιδιοτήτων τους σχηματίζουν οντολογίες, οι οποίες ορίζονται με τη χρήση των RDF, RDFS και OWL. Συγκεκριμένα, οι οντολογίες του Διαδικτύου των Πραγμάτων περιλαμβάνουν ορισμούς των εννοιών του IoT (π.χ. αισθητήρες, υπηρεσίες) και των ιδιοτήτων τους (π.χ. μετρήσεις αισθητήρα) μέσω δυαδικών σχέσεων. Οι γλώσσες ερωτήσεων όπως η SPARQL μπορούν να χρησιμοποιηθούν για την αναζήτηση πληροφοριών σε οντολογίες και οι μηχανισμοί αιτίασης όπως ο Pellet μπορούν να χρησιμοποιηθούν για την εύρεση ασυνεπειών και για την εξαγωγή νέων πληροφοριών από τις πληροφορίες που αντιπροσωπεύονται στις οντολογίες. Στην παρούσα εργασία σχεδιάζεται και υλοποιείται μια αρχιτεκτονική Σημασιολογικού Διαδικτύου των Πραγμάτων, με χρήση των παραπάνω τεχνολογιών και υιοθετώντας μια υπηρεσιοκεντρική αρχιτεκτονική. Επιπλέον, επιδεικνύεται η αποτελεσματικότητα της παραπάνω αρχιτεκτονικής και αποδεικνύεται ότι είναι δυνατή η υλοποίηση του οράματος της τεχνολογίας Σημασιολογικού Ιστού του Διαδικτύου με όσο το δυνατόν περισσότερο συνδυασμό των πλεονεκτημάτων των κλασικών και των νέων τεχνολογιών.

## Abstract

The world is moving towards machine-type communication, where anything (from smart sensors to everyday products in supermarkets) will be connected to the Internet. The Semantic Web of Things (SWoT) concept suggests the design of interoperable services or Web of Things applications using Semantic Web technologies. SWoT is a semantic extension of the Web of Things that allows applications to share content and services beyond their limits or, even more significant, to develop new applications as a composition of existing ones (e.g. using mashups). The main idea is to succeed the accomplishment of these tasks automatically or with minimum human intervention. Tools that are capable of understanding the meaning of IoT applications (i.e. data and services) and reason over their content must be applied, in order to enable the SWoT vision. Just as in the Semantic Web, formal definitions of concepts and their properties form ontologies, which are defined using the RDF, RDFS and the OWL language. In particular, IoT ontologies comprise definitions of IoT concepts (e.g. sensors, services) along with their properties (e.g. sensor observations) by means of binary relations. Query languages such as SPARQL can be used for querying information in ontologies and reasoners such as Pellet can be applied for locating inconsistencies or for inferring new information from information represented in ontologies. In this thesis, a Semantic Web of Things architecture is designed and implemented, using the above technologies and adopting a service-oriented architecture. What's more, the experimental results demonstrate the effectiveness of the above architecture and it is proved that the implementation of the SWoT vision is possible by combining the advantages of the classic and the new technologies as much as possible.

## 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, και ιδιαίτερα το συμφοιτητή μου, Στυλιανό Μποτωνάκη, με τον οποίο συνεργαστήκαμε με επιτυχία κατά την παράλληλη εκπόνηση των διπλωματικών μας εργασιών.

*Στους γονείς μου...*

# Περιεχόμενα

# Κεφάλαιο 1

# Εισαγωγή

## 1.1 Σκοπός της Εργασίας

### 1.1.1 Διαδίκτυο των Πραγμάτων και Υπολογιστικό Νέφος

Η ιδέα του Διαδικτύου των Πραγμάτων (Internet of Things ή IoT) και η τεχνολογία του Υπολογιστικού Νέφους συνδυάζονται και δημιουργούν καθημερινά νέους ορίζοντες στο πεδίο της συλλογής και ανάλυσης δεδομένων σε πραγματικό χρόνο. 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 πλεονεκτήματα αυτά καθιστούν το Υπολογιστικό Νέφος ιδανικό περιβάλλον ανάπτυξης εφαρμογών για το Διαδίκτυο των Πραγμάτων.

Το Υπολογιστικό Νέφος επιτυγχάνει τη διάθεση των υπολογιστικών πόρων μέσω διαδικτύου (π.χ. εξυπηρετητές, εφαρμογές κλπ.). Πολλοί χρήστες μπορούν να έχουν πρόσβαση ταυτόχρονα στην ίδια υπηρεσία ή υποδομή, χάρη στην ικανότητα του Νέφους να επιτρέπει την κατανάλωση των ίδιων πόρων σε ευρεία κλίμακα. Στην τεχνολογία του Υπολογιστικού Νέφους, οι πόροι προβλέπονται και διατίθενται κατά παραγγελία (on-demand), επομένως δίνεται η δυνατότητα

στους χρήστες να χρησιμοποιήσουν τους πόρους του Νέφους με βάση τις πραγματικές ανάγκες τους και να χρεωθούν αποκλειστικά για αυτή τη χρήση τους. Επιπλέον, η ιδιότητα της επεκτασιμότητας μιας υποδομής Νέφους καθιστά πιο εύκολη την εξυπηρέτηση των απαιτήσεων του συνεχώς αυξανόμενου πλήθους χρηστών και εφαρμογών.

Το Διαδίκτυο των Πραγμάτων γίνεται μέρα με τη μέρα όλο και πιο δημοφιλές, καθώς επιδιώκει να πετύχει την εσωτερική διασύνδεση των συσκευών των χρηστών και να καταστήσει ικανή την επεξεργασία της εξαιρετικά μεγάλης ποσότητας πληροφοριών, οι οποίες χρησιμοποιούνται καθημερινά από εκατομμύρια συσκευές που βρίσκονται συνδεδεμένες στο διαδίκτυο. Η ικανότητα σύνδεσης των συσκευών στο διαδίκτυο καθιστά πλέον τη χρήση τους αναπόσπαστο κομμάτι της καθημερινής ζωής, με αποτέλεσμα να γίνεται καθημερινά περισσότερο δημοφιλής η χρήση των συσκευών ως "φορετές" (wearable devices). Με αυτόν τον τρόπο, προσφέρονται σημαντικά πλεονεκτήματα στο πεδίο των εφαρμογών που απαιτούν γρήγορη και διαρκή παρακολούθηση των δεδομένων των χρηστών από οπουδήποτε. Χαρακτηριστικά παραδείγματα τέτοιων εφαρμογών αποτελούν αυτές που σχετίζονται με την απομακρυσμένη παρακολούθηση της υγείας (remote health monitoring), με τον έλεγχο της λειτουργίας ενός οικοσυστήματος, όπως ένα σπίτι, ένα εργοστάσιο ή μια ολόκληρη πόλη (smart cities), με την επίβλεψη γεωργικών εκτάσεων, κλπ. Με άλλα λόγια, το Διαδίκτυο των Πραγμάτων στοχεύει στη συλλογή και την ανάλυση μιας τεράστιας ποσότητας πληροφοριών, οι οποίες σχετίζονται με εφαρμογές που χρησιμοποιούνται για να εξυπηρετήσουν τις ανάγκες της καθημερινής ζωής (real-life applications). Συνεπώς, η δημιουργία εφαρμογών για το Διαδίκτυο των Πραγμάτων καθίσταται σήμερα αναγκαία σε πολλούς τομείς της καθημερινότητας.

### 1.1.2 Ο ορισμός του προβλήματος (Problem Definition)

Ο κόσμος πλέον κινείται προς την επικοινωνία τύπου μηχανής (machine-type communication), όπου όλες οι συσκευές θα είναι συνδεδεμένες στο διαδίκτυο. Ο Ιστός των Πραγμάτων (Web of Things) είναι μια πρωτοβουλία που αποσκοπεί στην ενοποίηση των διασυνδεδεμένων κόσμων των Πραγμάτων (π.χ. συσκευές, αισθητήρες) σε μια ενιαία αρχιτεκτονική. Το WoT απέχει πολύ από τη σημερινή πραγματικότητα, κυρίως λόγω του κατακερματισμού των τεχνολογιών που χρησιμοποιούνται σήμερα για το σχεδιασμό και την ανάπτυξη συστημάτων

Διαδικτύου των Πραγμάτων. Στην ουσία, αυτό που καθιστά το Διαδίκτυο των Πραγμάτων (IoT) ανεπαρκές σε σχέση με τον Ιστό των Πραγμάτων (WoT), είναι η αδυναμία ανεξαρτητοποίησης της επικοινωνίας από το πρωτόκολλο (επικοινωνίας), δηλαδή η δημιουργία μιας "καθολικής" επικοινωνίας, όπου όλες οι συσκευές θα μπορούν να επικοινωνούν με άλλες συσκευές και υπηρεσίες, χωρίς το πρωτόκολλο να δημιουργεί πρόβλημα. "Αφαιρώντας" την πολυπλοκότητα και την ποικιλία των πρωτοκόλλων χαμηλού επιπέδου (low-level protocols), και κρύβοντάς την πίσω από το απλό μοντέλο του Ιστού, προκύπτουν πολλά πλεονεκτήματα. Για παράδειγμα, ένα από αυτά είναι ότι οι προγραμματιστές εφαρμογών μπορούν να επικεντρωθούν στη λογική των εφαρμογών τους, χωρίς να τους απασχολεί πώς δουλεύει το κάθε πρωτόκολλο ή η συσκευή.

Μια κοινή τακτική γύρω από αυτό το πρόβλημα είναι να επιτραπεί σε κάθε συσκευή στο διαδίκτυο (όπως οι αισθητήρες ή τα κινητά τηλέφωνα) να γίνει μέρος του υπάρχοντος Παγκόσμιου Ιστού. Με αυτόν τον τρόπο, κάθε συσκευή έχει τη δυνατότητα να δημοσιευθεί στον Ιστό (δηλαδή να αναδείξει την ταυτότητα και το περιεχόμενό της), να ανακαλυφθεί από μηχανές αναζήτησης του Διαδικτύου και συνεπώς να χρησιμοποιηθεί από ανθρώπους ή εφαρμογές όπως κάθε άλλος ιστότοπος. Αμέσως μετά, μια υπηρεσία Mashup θα επιτρέπει στους προγραμματιστές εφαρμογών να συνθέτουν νέες εφαρμογές σε πολύ λιγότερο χρόνο ελαχιστοποιώντας την προσπάθεια που απαιτείται για τη συντήρηση του συστήματος κάθε φορά που γίνεται προσθήκη, αφαίρεση ή ενημέρωση μιας συσκευής ή μιας υπηρεσίας.

Παρά το γεγονός ότι οι ελαφρού βάρους (lightweight) διακομιστές Ιστού μπορούν να ενσωματωθούν σε μικρές συσκευές με σκοπό να ενεργοποιήσουν τη λειτουργικότητα του Ιστού των Πραγμάτων, οι πόροι που διαθέτουν είναι περιορισμένοι κι έτσι η λύση δεν είναι βέλτιστη όσον αφορά το χρόνο ζωής της μπαταρίας, την αυτονομία του αισθητήρα και το κόστος. Μια λύση γύρω από αυτό το πρόβλημα θα ήταν η ανάπτυξη ενός Διακομιστή μεσολάβησης στο Διαδίκτυο, ο οποίος θα "τρέχει" στο Υπολογιστικό Νέφος και θα διατηρεί την εικονική οντότητα του κάθε Πράγματος. Μέσω των Διακομιστών μεσολάβησης στο Web, τα Πράγματα έχουν τη δυνατότητα να επικοινωνούν με άλλες οντότητες ή με το Νέφος, ενώ παράλληλα τα στοιχεία τους (οι περιγραφές τους, τα δεδομένα και οι υπηρεσίες τους) γίνονται μέρος του Ιστού. Με αυτόν τον τρόπο, μπορούν να δημοσιεύονται, να καταναλώνονται, να συγκεντρώνονται, να ενημερώνονται αλλά και να αναζητούνται. Η χαρτογράφηση οποιασδήποτε

συσκευής σε ένα διακομιστή μεσολάβησης Ιστού κάνει πολύ πιο εύκολη τη διαδικασία διαμόρφωσης ή την επαναδημιουργία εφαρμογών που υπάρχουν ήδη, με τον ίδιο τρόπο που οι ιστότοποι μπορούν να δημιουργηθούν και να δημοσιευθούν στον Ιστό.

Ο Σημασιολογικός Ιστός του Διαδικτύου των Πραγμάτων ή Σημασιολογικός Ιστός των Πραγμάτων (Semantic Web of Things) αποτελεί μια σημασιολογική επέκταση του Ιστού των Πραγμάτων, δηλαδή επεκτείνει τη λειτουργικότητά του, αξιοποιώντας τα εργαλεία του Σημασιολογικού Ιστού. Ο στόχος του SWoT είναι τα εργαλεία αυτά να κατανοήσουν τη σημασία των διαδικτυακών εφαρμογών και να αιτιολογούν το περιεχόμενό τους, χρησιμοποιώντας μηχανισμούς όπως η σημασιολογική αιτίαση ή αλλιώς αυτοματοποιημένος συλλογισμός (semantic reasoning), που θα αναλυθεί παρακάτω. Χάρη στα εργαλεία αυτά, θα προσδίδεται στην πληροφορία μια σαφώς προσδιορισμένη σημασία, με αποτέλεσμα οι μηχανές να μπορούν πλέον να διαχειρίζονται καλύτερα την πληροφορία και να "κατανοούν" κατά κάποιο τρόπο τα δεδομένα, τα οποία ως τώρα απλά παρουσιάζουν. Βασικό συστατικό της παραπάνω τεχνολογίας αποτελούν οι οντολογίες, δηλαδή οι οργανωμένες συλλογές πληροφορίας, όπου φιλοξενείται η απαραίτητη γνώση (π.χ. ορισμοί εννοιών του Διαδικτύου των Πραγμάτων όπως οι αισθητήρες, τα χαρακτηριστικά τους, κλπ.).

## 1.2 Προτεινόμενη Λύση

Σκοπός της παρούσας εργασίας είναι η επίλυση του προβλήματος που αναφέρθηκε παραπάνω, με την πρότασή μας για την ανάπτυξη ενός περιβάλλοντος το οποίο επιτρέπει την αυτοματοποιημένη καταχώρηση και δημοσίευση συσκευών και συνεπώς την αυτοματοποιημένη κατασκευή εφαρμογών Διαδικτύου των Πραγμάτων με χρήση μηχανισμών Σημασιολογικού Ιστού. Η *έμφαση* της εργασίας είναι στην υποστήριξη λειτουργικότητας Καταχώρησης, Δημοσίευσης και Ανεύρεσης συσκευών του διαδικτύου καθώς και υπηρεσιών που υποστηρίζουν αυτήν την λειτουργικότητα. Το αποτέλεσμα της εργασίας είναι μια νέα προσέγγιση σχεδιασμού αρχιτεκτονικής Διαδικτύου των Πραγμάτων που επεκτείνει τη λειτουργικότητα υπαρχόντων αρχιτεκτονικών [1]. Με αυτόν τον τρόπο, υπάρχει κέρδος στο χρόνο ανάπτυξης καθώς και υποστήριξη αυξημένης λειτουργικότητας που εξειδικεύεται στην κάθε εφαρμογή. Η υποστήριξη λειτουργικότητας Σημασιολογικού Ιστού για την κατασκευή

διαδικτυακών εφαρμογών είναι αντικείμενο ξεχωριστής εργασίας που διεκπεραιώθηκε παράλληλα με την παρούσα εργασία [2].

Το σύστημα που υλοποιήθηκε βασίζεται στο Web Thing Model[1] της W3C. Πρόκειται για ένα κοινό μοντέλο που έχει ως στόχο την περιγραφή των φυσικών αντικειμένων με χρήση εικονικών οντοτήτων στον Ιστό των Πραγμάτων (βλ. Κεφάλαιο 2). Το εν λόγω μοντέλο έπαιξε καθοριστικό ρόλο για την εκπόνηση της παρούσας εργασίας. Υιοθετήθηκε η λογική του σχεδιασμού (π.χ. βασικοί κανόνες, υποθέσεις και παραδοχές που διέπουν το σύστημα) και κατ' επέκταση η υλοποίηση των λειτουργιών που προτείνει το μοντέλο. Επιπλέον, η ορολογία που χρησιμοποιήθηκε καθ' όλη τη διάρκεια εκπόνησης της εργασίας βασίστηκε σε μεγάλο βαθμό στο παραπάνω μοντέλο.

Στην εργασία αυτή, προτείνεται η Υπηρεσιοκεντρική Αρχιτεκτονική (Service Oriented Architecture, βλ. Κεφάλαιο 2), η οποία αποτελεί βασικό παράγοντα σχεδίασης για την πλατφόρμα που δημιουργήθηκε. Όλες οι λειτουργίες του συστήματος αποτελούν αυτόνομες υπηρεσίες που επικοινωνούν μεταξύ τους μέσω RESTful διεπαφών. Χάρη στην υπηρεσιοκεντρική αρχιτεκτονική, το σύστημα μπορεί να επεκταθεί πιο εύκολα, καθώς όλες οι υπηρεσίες μπορούν να αναβαθμιστούν ή ακόμα και να αντικατασταθούν, λαμβάνοντας υπόψιν μόνο την κοινή διεπαφή REST που θα πρέπει η υπηρεσία να εκθέτει. Με αυτή τη λογική, είναι δυνατή η προσθήκη στο σύστημα νέας λειτουργικότητας σε μορφή υπηρεσίας.

Τελικά, μια από τις υπηρεσίες του συστήματος επιτρέπει στους προγραμματιστές εφαρμογών τη σύνθεση νέων εφαρμογών σε πολύ λιγότερο χρόνο, ελαχιστοποιώντας την προσπάθεια που απαιτείται για τη συντήρηση του συστήματος κάθε φορά που γίνεται προσθήκη, αφαίρεση ή ενημέρωση μιας συσκευής ή μιας υπηρεσίας. Η υπηρεσία αυτή (Mashup Service), της οποίας ο ρόλος αναλύεται στο Κεφάλαιο 3, έχει υλοποιηθεί στην εργασία του συμφοιτητή μου, Μποτωνάκη Στυλιανού [2].

Το αποτέλεσμα της υλοποίησης είναι μια πλατφόρμα Ιστού των Πραγμάτων (WoT) στο Υπολογιστικό Νέφος, της οποίας οι λειτουργίες είναι διαθέσιμες μέσω γραφικών διεπαφών από ένα πρόγραμμα περιήγησης Διαδικτύου (Web browser). Όπως αποδεικνύεται και από τα πειράματα που περιγράφονται στο Κεφάλαιο 5, το σύστημα που υλοποιήθηκε έχει καλή απόδοση σε ρεαλιστική εφαρμογή.

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

## 1.3 Δομή της Εργασίας

- Στο Κεφάλαιο 2 γίνεται μια σύντομη περιγραφή σχετικών τεχνολογιών 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 περιγράφεται η τεχνική λύση που αναπτύχθηκε με βάση τις προδιαγραφές του Κεφαλαίου 3.
- Στο Κεφάλαιο 5 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 6 παραθέτουμε κάποια συμπεράσματα και προτάσεις για μελλοντική διερεύνηση.

# Κεφάλαιο 2

# Υπόβαθρο και Υπάρχουσες Τεχνολογίες

## 2.1 Web Thing Model

Το Web Thing Model αποτελεί ένα μοντέλο που ορίζεται από την Κοινοπραξία του Παγκόσμιου Ιστού (World Wide Web Consortium ή W3C)[2] 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

[2] https://el.wikipedia.org/wiki/WWW_Consortium

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. Συγκεκριμένα, προτείνεται η χρήση μιας σύντομης περιγραφής της μορφής JSON για κάθε Πράγμα (βλ. Ανάκτηση ενός Web Thing).

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). Οι λειτουργίες αυτές και η υλοποίησή τους, όπως αυτή έγινε κατά την εκπόνηση της παρούσας εργασίας, παρουσιάζονται εκτενώς στη συνέχεια του παρόντος Κεφαλαίου.

## 2.2 Υπολογιστικό Νέφος και SWoT

Η χρήση της τεχνολογίας του Υπολογιστικού Νέφους προσφέρει στους χρήστες του διαδικτύου πολλά πλεονεκτήματα όπως η απλότητα, η επεκτασιμότητα και το χαμηλό κόστος (αναφέρθηκαν και στο Κεφάλαιο 1). Ο συνδυασμός των κλασικών τεχνολογιών του Υπολογιστικού Νέφους με τις νέες τεχνολογίες που προτείνει ο Σημασιολογικός Ιστός του Διαδικτύου των Πραγμάτων μπορεί να πετύχει ενδιαφέροντα αποτελέσματα, καθώς οι υπάρχουσες εφαρμογές του Διαδικτύου των Πραγμάτων που "φιλοξενούνται" στο Νέφος θα αποκτήσουν σημασία και θα αρχίσουν να γίνονται σε ένα βαθμό κατανοητές και από τις μηχανές. Αυτό θα επιτευχθεί χάρη σε μεθόδους και τεχνολογίες όπως αυτή των *Διασυνδεδεμένων Δεδομένων (Linked Data)*[3]. Η μέθοδος αυτή χρησιμοποιείται για τη δημοσιοποίηση δομημένων δεδομένων και

[3] https://en.wikipedia.org/wiki/Linked_data

αξιοποιούνται γνωστές τεχνολογίες του Ιστού (όπως HTTP και URIs), με σκοπό, όχι μόνο την εξυπηρέτηση ιστοσελίδων για τους αναγνώστες, αλλά και την επέκταση των τεχνολογιών αυτών προκειμένου να πραγματοποιούν ανταλλαγή πληροφοριών. Η ανταλλαγή αυτή γίνεται με τέτοιο τρόπο ώστε οι υπολογιστές να μπορούν να διαβάζουν τις πληροφορίες αυτόματα. Επομένως, τα δεδομένα πλέον είναι αλληλένδετα και περισσότερο χρήσιμα, ενώ μπορούν πλέον δεδομένα από διαφορετικές πηγές να συνδεθούν και να αναζητηθούν. Στην αξιοποίηση των μεθόδων που παρέχει ο Σημασιολογικός Ιστός συντελεί και η χρήση σημασιολογικών ερωτημάτων (semantic queries), τα οποία εκμεταλλεύονται τη σύνδεση των δεδομένων για την άντληση της επιθυμητής πληροφορίας. Συγκεκριμένα, γλώσσες ερωτήσεων όπως η SPARQL επιστρατεύονται ούτως ώστε τα δεδομένα, οι υπηρεσίες και οι εφαρμογές που φιλοξενούνται στο Υπολογιστικό Νέφος να αποκτούν νόημα, όχι μόνο για τον άνθρωπο, αλλά και για τις μηχανές.

## 2.3 Σχετικές Εργασίες και Τεχνολογίες

Είναι γνωστό ότι υπάρχουν ήδη πολλές προσεγγίσεις για το σχεδιασμό συστημάτων Διαδικτύου των Πραγμάτων και μεγάλο μέρος από αυτές έχουν εφαρμοστεί σε εμπορικές πλατφόρμες IoT (διευκολύνοντας έτσι την ανάπτυξη εφαρμογών από προγραμματιστές) ή σε προσαρμοσμένες λύσεις εξυπηρετώντας τις ανάγκες ενός συγκεκριμένου πεδίου εφαρμογής. Μερικά παραδείγματα είναι οι εφαρμογές για την ηλεκτρονική υγεία (e-Health), την υποβοηθούμενη διαβίωση (ambient assisted living), την παρακολούθηση της δραστηριότητας των χρηστών, κλπ. Επιπλέον, έχουν δημοσιευθεί προσεγγίσεις που προτείνουν και χρησιμοποιούν τις τεχνολογίες του Σημασιολογικού Ιστού. Αρκετές από αυτές αναφέρονται στην χρήση των τεχνολογιών του Σημασιολογικών Τεχνολογιών για την αξιοποίηση της πληροφορίας που αφορά τους αισθητήρες, τις μετρήσεις τους και τις ενέργειές τους (actuations), για παράδειγμα χρησιμοποιώντας τις οντολογίες SSN και SOSA (αναλύονται στη συνέχεια του Κεφαλαίου). 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] , [6] και [7].

Ωστόσο, αναζητείται μια ολοκληρωμένη και αποδοτική αρχιτεκτονική, η οποία θα βασίζεται στην ιδέα του Διαδικτύου των Πραγμάτων και θα συνδυάζει

τις κλασικές τεχνολογίες του Υπολογιστικού Νέφους με τις νέες τεχνολογίες του Σημασιολογικού Ιστού. Ο σχεδιασμός ενός συστήματος που θα συνδυάζει όλα τα παραπάνω και θα επιτρέπει τη συλλογή και την ανάλυση δεδομένων σε πραγματικό χρόνο, αποτελεί πρόκληση που πρέπει να αντιμετωπίσει η μελλοντική δουλειά στα συστήματα IoT.

Η συνεισφορά της παρούσα εργασίας έγκειται στην πρόταση ενός περιβάλλοντος Σημασιολογικού Ιστού του Διαδικτύου των Πραγμάτων (Semantic Web of Things), δηλαδή ενός συστήματος που 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 RDF, τις οντολογίες, τη γλώσσα ερωτήσεων SPARQL, κ.ά. Συγκεκριμένα, προτείνεται η χρήση μιας Υπηρεσιοκεντρικής Αρχιτεκτονικής στο Υπολογιστικό Νέφος, η οποία ενσωματώνει τις τεχνολογίες του Σημασιολογικού Ιστού, προσδίδοντας στα δεδομένα μια σαφώς προσδιορισμένη σημασία, αξιοποιώντας τα μεταδεδομένα που αφορούν τις συσκευές / αισθητήρες, και προσφέροντας σε χρήστες σημαντικές λειτουργίες όπως η αποθήκευση, η ανάκτηση, η ενημέρωση και η διαγραφή των παραπάνω πληροφοριών. Ο σχεδιασμός αυτών των λειτουργιών βασίζεται στο μοντέλο "Web Thing Model" που αναπτύχθηκε από την Κοινοπραξία Παγκόσμια Ιστού (W3C) και αναλύεται παραπάνω αλλά και στη συνέχεια της εργασίας. Αυτό είναι το βασικό πλεονέκτημα που παρουσιάζει η αρχιτεκτονική που ακολουθείται στην εργασία, ωστόσο οι πειραματικές μετρήσεις αποδεικνύουν ότι πρόκειται και για μια αποδοτική αρχιτεκτονική, καθώς ο χρόνος απόκρισης του συστήματος και η κατανάλωση πόρων (CPU και μνήμης RAM) βρίσκονται σε χαμηλά επίπεδα. Επομένως, μέσω της πειραματικής διαδικασίας, καταγράφεται και αναδεικνύεται η αποτελεσματικότητα της προτεινόμενης αρχιτεκτονικής, η οποία συμπεραίνουμε ότι μπορεί να επιδείξει μια σειρά από σημαντικά πλεονεκτήματα απέναντι σε άλλες σχετικές προτεινόμενες λύσεις και αρχιτεκτονικές.

Στη συνέχεια, θα αναφερθούμε σε σχετικές τεχνολογίες που μπορούν να αποτελέσουν σημαντικά εργαλεία μιας αρχιτεκτονικής Σημασιολογικού Ιστού του Διαδικτύου των Πραγμάτων και αξιοποιούνται κατάλληλα στην παρούσα εργασία.

### 2.3.1 FIWARE

Το FIWARE αποτελεί μια πλατφόρμα middleware, η οποία βασίζεται στο λογισμικό Openstack και παρέχει ένα αρκετά απλό αλλά ισχυρό σύνολο διεπαφών προγραμματισμού εφαρμογών (APIs) που διευκολύνουν την ανάπτυξη εφαρμογών. Πρόκειται για ένα πλαίσιο πλατφόρμας ανοικτού κώδικα (open - source), του οποίου οι υπηρεσίες μπορούν να συναρμολογηθούν μεταξύ τους, καθώς και με άλλες υπηρεσίες "τρίτων" με σκοπό την ανάπτυξη της επιτάχυνσης της ανάπτυξης έξυπνων λύσεων. Συγκεκριμένα, το FIWARE προσφέρει υπηρεσίες γενικού σκοπού (GEs), οι οποίες περιλαμβάνουν απλές διεπαφές προγραμματισμού εφαρμογών. Χάρη στις υπηρεσίες αυτές, το FIWARE επιτρέπει την ανάπτυξη και τη διανομή εφαρμογών υπηρεσιοκεντρικής αρχιτεκτονικής στο Υπολογιστικό Νέφος.


*Εικόνα 2.1: Λογότυπο του FIWARE*

### 2.3.2 Υπηρεσίες στο FIWARE

Χάρη στους γενικούς ενεργοποιητές (Generic Enablers) του FIWARE, μπορούμε να πραγματοποιήσουμε μια σειρά λειτουργιών γενικού σκοπού, οι οποίες παρέχονται από σαφώς καθορισμένες διεπαφές προγραμματισμού εφαρμογών (REST APIs). Με αυτόν τον τρόπο, η ανάπτυξη έξυπνων εφαρμογών σε διάφορους τομείς καθίσταται σαφώς πιο εύκολη. Μια υπηρεσία γενικού σκοπού χαρακτηρίζεται από την απλότητα στη χρήση της αλλά και από τη δυνατότητα επαναχρησιμοποίησης, επομένως μπορεί να αποτελεί συστατικό για να δημιουργούνται σύνθετα συστήματα.

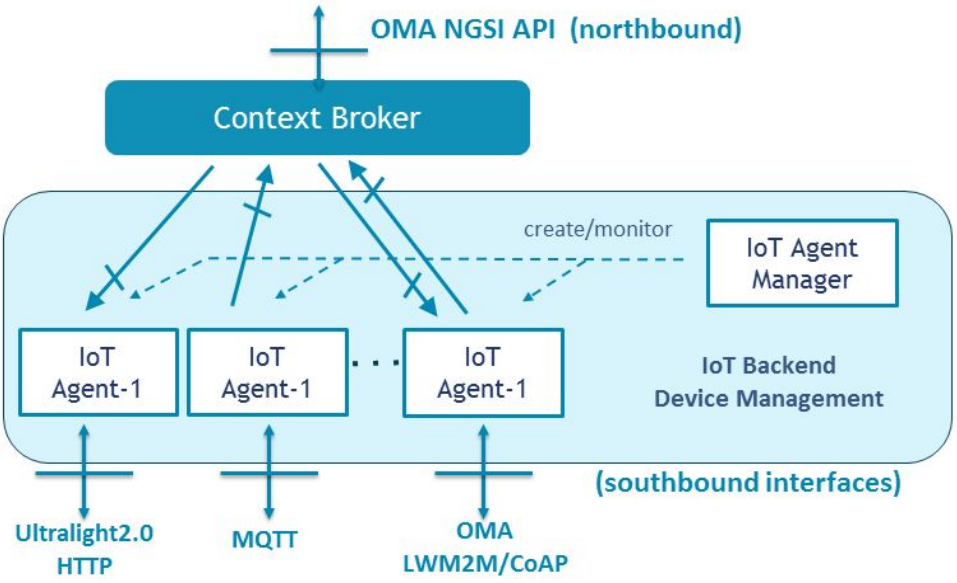
Ο κατάλογος υπηρεσιών του FIWARE περιλαμβάνει μια πλούσια συλλογή από γενικούς ενεργοποιητές (GE) που παρέχουν στους προγραμματιστές τη

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:

#### 2.3.2.1 FIWARE 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 NGSI2, που είναι το πρότυπο 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 NGSI2. 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. Στην παρακάτω εικόνα φαίνεται η χρήση διαφορετικών IoT Agents, οι οποίοι είτε μεταφράζουν τα αντίστοιχα πρωτόκολλα των συσκευών σε NGSI και τα στέλνουν στην υπηρεσία Context Broker είτε πραγματοποιούν την αντίστροφη διαδικασία (λαμβάνουν δεδομένα από τον Context Broker, τα μεταφράζουν και τα στέλνουν στις συσκευές).




***Εικόνα 2.2**: IoT Πράκτορες που μεταφράζουν διαφορετικά πρωτόκολλα σε πρωτόκολλο NGSI (και το αντίστροφο).*

#### 2.3.2.2 FIWARE - Cygnus

Πρόκειται για μια υπηρεσία του οικοσυστήματος FIWARE που έχει το ρόλο του συνδέσμου μεταξύ του Orion Context Broker (που συνιστά πηγή δεδομένων NGSI2) και πολλών τύπων βάσεων δεδομένων - αποθετηρίων όπως MongoDB, MySQL, CKAN, DynamoDB. Το Cygnus δέχεται ροές δεδομένων NGSI2 και τις αποθηκεύει στην προκαθορισμένη βάση δεδομένων που του έχει ανατεθεί. Έχει επιπλέον τη δυνατότητα να αποθηκεύει ακατέργαστα (Raw) και συγκεντρωτικά (Aggregated) δεδομένα, ενώ δεν δεσμεύει το σύστημα να χρησιμοποιήσει συγκεκριμένου τύπου βάση δεδομένων.

#### 2.3.2.3 FIWARE - Short Time Historic (STH) COMET

Το STH COMET αποτελεί μια υπηρεσία του FIWARE που πραγματοποιεί τη διαχείριση ιστορικών πληροφοριών χρονικής σειράς. Συγκεκριμένα, έχει τη δυνατότητα ανάκτησης της ιστορικών πληροφοριών από βάσεις δεδομένων (π.χ. από βάση δεδομένων MongoDB). Οι πληροφορίες αυτές προκύπτουν από τις αλλαγές στην τιμή των οντοτήτων που διατηρούνται στην υπηρεσία Orion Context Broker. Κάθε επικοινωνία μεταξύ του FIWARE-COMET και του Orion Context

Broker, καθώς και μεταξύ του FIWARE-COMET και οποιουδήποτε "τρίτου" (π.χ. για ανάκτηση δεδομένων), επιτυγχάνεται μέσω τυποποιημένων διεπαφών NGSI.

#### 2.3.2.4 FIWARE - Publish/Subscribe Context Broker

Ο Publish/Subscribe Context Broker (Orion Context Broker) του FIWARE συνιστά την υλοποίηση μιας Υπηρεσίας Διαχείρισης Συμβάντων και Συνδρομών και παρέχει για τη διαχείρισή τους RESTful διεπαφές NGSI (μοντέλο αναπαράστασης - ανταλλαγής δεδομένων στο FIWARE, βλ. 2.3.5). 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.

Η Εικόνα 2.3 δείχνει την επικοινωνία που αναπτύσσει η υπηρεσία Publish/Subscribe Context Broker μέσω των RESTful διεπαφών της, τόσο με χρήστες/άλλες υπηρεσίες (μέσω του port 1026) όσο και με τη βάση δεδομένων MongoDB, όπου αποθηκεύονται οι οντότητες που διαχειρίζεται η υπηρεσία.


***Εικόνα 2.3:*** *Η επικοινωνία του Orion Context Broker με χρήστες/υπηρεσίες (αριστερά) και με τη βάση δεδομένων MongoDB (δεξιά).*

#### 2.3.2.4 FIWARE Identity Manager (IdM) - Keyrock

Η υπηρεσία Keyrock IdM του οικοσυστήματος FIWARE καλύπτει ένα εύρος λειτουργιών - δυνατοτήτων που αφορούν την ασφάλεια του συστήματος,

για παράδειγμα η πρόσβαση των χρηστών σε υπηρεσίες, δίκτυα και εφαρμογές, η ασφάλεια και η πιστοποίηση από χρήστες σε συσκευές, δίκτυα και υπηρεσίες και η διαχείριση της εξουσιοδότησης και των προφίλ των χρηστών. Ακόμα, έχει τη δυνατότητα να διατηρεί προσωπικά δεδομένα και να παρέχει άδειες προς εφαρμογές. Όπως θα δούμε και στη συνέχεια, η υπηρεσία βασίζεται σε μεγάλο βαθμό στο πρωτόκολλο εξουσιοδότησης OAuth2.0

### 2.3.3 RESTful Υπηρεσίες Ιστού

Οι RESTful Υπηρεσίες Ιστού έχουν κατασκευαστεί με σκοπό να λειτουργούν γρήγορα και αποτελεσματικά στον παγκόσμιο ιστό. Το αρχιτεκτονικό πρότυπο *Representational State Transfer (REST)*[4], αξιοποιώντας τις πολύ σημαντικές ιδιότητές του (απόδοση, επεκτασιμότητα και τροποποιησιμότητα), προσφέρει στις Υπηρεσίες Ιστού τη δυνατότητα να λειτουργούν βέλτιστα στον παγκόσμιο ιστό. Σύμφωνα με το πρότυπο REST, τα δεδομένα και η λειτουργικότητα θεωρούνται πόροι (resources) και η πρόσβαση σε αυτούς επιτυγχάνεται από μια αυστηρά καθορισμένη κοινή διεπαφή που βασίζεται στη χρήση των πρότυπων μεθόδων HTTP. Συγκεκριμένα, κάθε πόρος είναι προσβάσιμος μέσω ενός Uniform Resource Identifier (URI), δηλαδή από έναν υπερσύνδεσμο του διαδικτύου. Επιπλέον, κάθε πόρος αποτελείται από μια ταυτότητα και έναν τύπο δεδομένων, ενώ υποστηρίζει και ένα σύνολο ενεργειών. Η ταυτότητα του πόρου είναι το προαναφερθέν URI και οι ενέργειες πραγματοποιούνται από τις τυπικές μεθόδους HTTP: την ανάκτηση / ανάγνωση (GET), την δημιουργία (POST), την ενημέρωση (PUT), και την διαγραφή (DELETE). Με τη λειτουργία GET είναι δυνατή 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.

### 2.3.4 Βάση Δεδομένων MongoDB

Το MongoDB[5] είναι ένα σύστημα διαχείρισης βάσεων δεδομένων (DMBS) ανοιχτού κώδικα. Χρησιμοποιεί ένα μοντέλο βάσης δεδομένων που προορίζεται

[4] https://en.wikipedia.org/wiki/Representational_state_transfer
[5] https://www.mongodb.com/

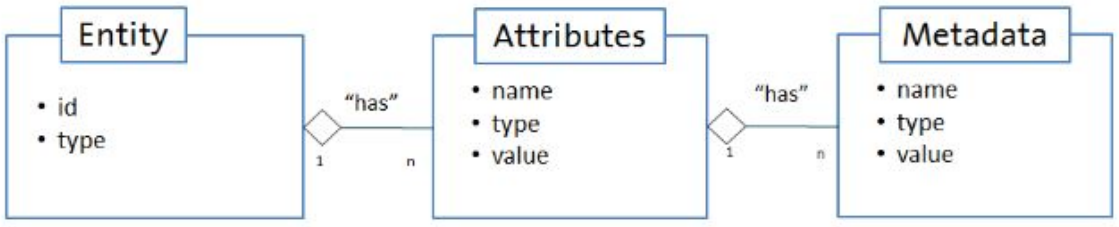
για έγγραφα και υποστηρίζει διάφορες μορφές δεδομένων όπως το μορφότυπο JSON. Συνιστά μια από τις πολυάριθμες μη σχεσιακές τεχνολογίες βάσεων δεδομένων που δημιουργήθηκαν με το banner "NoSQL" προκειμένου να χρησιμοποιηθούν σε μεγάλες εφαρμογές δεδομένων και άλλες εργασίες επεξεργασίας δεδομένων που δεν 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.

### 2.3.5 Μοντέλο πληροφορίας NGSI-2

Το API του FIWARE NGSI-2 (Next Generation Service Interface)[6] ορίζει:

- Ένα μοντέλο δεδομένων για πληροφορίες πλαισίου (context information), που βασίζεται σε ένα απλό πληροφοριακό μοντέλο και χρησιμοποιεί την έννοια των οντοτήτων πλαισίου (βλ. παρακάτω).
- Μια RESTful διεπαφή δεδομένων πλαισίου για την ανταλλαγή πληροφοριών μέσω επερωτήσεων, συνδρομών και ενημερώσεων.
- Μια RESTful διεπαφή διαθεσιμότητας πλαισίου για την ανταλλαγή πληροφοριών σχετικών με το πώς να αποκτήσεις δεδομένα πλαισίου (context data).

Το μοντέλο πληροφορίας NGSI βασίζεται σε ένα συγκεκριμένο πρότυπο αναπαράστασης, το οποίο φαίνεται στην εικόνα 2.4. Ακολουθεί μια εκτενής περιγραφή των βασικών συστατικών της NGSI πληροφορίας.




***Εικόνα 2.4:*** *Μοντέλο πληροφορίας NGSI*

[6] https://fiware.github.io/specifications/ngsiv2/stable/

**Entity (Οντότητα):** Πρόκειται για το κέντρο βάρους του μοντέλου πληροφοριών FIWARE NGSI. Μια οντότητα αντιπροσωπεύει ένα πράγμα, δηλαδή οποιοδήποτε φυσικό ή λογικό (π.χ. έναν αισθητήρα, μία μέτρηση, ένα σπίτι, κλπ.). Κάθε οντότητα έχει - υποχρεωτικά - ένα αναγνωριστικό οντότητας (id) και έναν τύπο (type), π.χ. "Sensor". Μια οντότητα μπορεί να αποτελείται από 1 έως n χαρακτηριστικά (Attributes). Ένα χαρακτηριστικό είναι "κομμάτι" (part of) μιας οντότητας.

**Attributes (Χαρακτηριστικά):** Τα χαρακτηριστικά αποτελούν ιδιότητες των οντοτήτων. Για παράδειγμα, η τρέχουσα μέτρηση θερμοκρασίας από έναν αισθητήρα θα μπορούσε να διαμορφωθεί ως "current_temperature" της οντότητας αισθητήρα "sensor_107". Στο μοντέλο πληροφορίας NGSI, τα χαρακτηριστικά έχουν ένα όνομα (name), έναν τύπο (type) και μια τιμή (value).

Το όνομα του χαρακτηριστικού χρησιμοποιείται για την περιγραφή του είδους της ιδιότητας που αντιπροσωπεύει η τιμή (value) του εκάστοτε χαρακτηριστικού της οντότητας. Παραδείγματος χάριν, η τιμή του "current_temperature" αφορά τη θερμοκρασία (σε δεδομένα, π.χ. 25 βαθμοί Κελσίου) που μέτρησε ο αισθητήρας. Ο τύπος (type) του χαρακτηριστικού υποδεικνύει τον τύπο δεδομένων της τιμής του χαρακτηριστικού (π.χ. float, integer, string, κλπ.).

Ένα χαρακτηριστικό μπορεί να περιλαμβάνει από 1 έως n μεταδεδομένα (Metadata). Τα μεταδεδομένα αποτελούν "κομμάτι" (Part of) ενός χαρακτηριστικού.

**Metadata (Μεταδεδομένα):** Τα μεταδεδομένα χρησιμοποιούνται για την περιγραφή της ιδιότητας της τιμής (value) του χαρακτηριστικού στο οποίο αναφέρονται, για παράδειγμα προσδιορισμός της μονάδας μέτρησής του (θα μπορούσε η τιμή των μεταδεδομένων να είναι "Celsius" για το χαρακτηριστικό "temperature"). Οι μεταβλητές όνομα (name), τύπος (type) και τιμή (value) των μεταδεδομένων βασίζονται στους ίδιους κανόνες που ακολουθούν και οι αντίστοιχες μεταβλητές του χαρακτηριστικού (Attribute).

Συνεπώς, μια οντότητα αποτελείται συνήθως από όλα τα παραπάνω στοιχεία (μεταδεδομένα δεν είναι απαραίτητο να υπάρχουν) και αντιπροσωπεύεται από ένα αντικείμενο JSON, του οποίου η σύνταξη βασίζεται στους κανόνες του προτύπου NGSI. Ένα απλό παράδειγμα οντότητας φαίνεται παρακάτω:

```
{
    "id": "Home215",
    "type": "Home",
    "pressure": {
        "type": "Integer",
        "value": 700,
        "metadata": {}
    },
    "temperature": {
        "type": "Float",
        "value": 15,
        "metadata": {}
    }
}
```


***Εικόνα 2.5**: Παράδειγμα οντότητας σπιτιού σε NGSI αναπαράσταση*

Στην εικόνα 2.5 φαίνεται η NGSI αναπαράσταση ενός σπιτιού (Home) με μοναδικό αναγνωριστικό "Home215". Το σπίτι έχει το χαρακτηριστικό της πίεσης (pressure), καθώς και αυτό της θερμοκρασίας (temperature). Η τιμή της πίεσης είναι 700 και πρόκειται για έναν ακέραιο αριθμό (Integer), ενώ αντίστοιχα η τιμή της θερμοκρασίας είναι 15 που είναι ένας "float" αριθμός.

### 2.3.6 OAuth2.0

Το πρωτόκολλο ταυτοποίησης και εξουσιοδότησης OAuth2.0[7] είναι ένα ανοιχτό πρότυπο για την ανάθεση της πρόσβασης και χρησιμοποιείται συνήθως από τους χρήστες του διαδικτύου ως ένας τρόπος για να επιτρέπουν σε ιστότοπους ή σε εφαρμογές να έχουν πρόσβαση σε άλλους ιστότοπους, χωρίς όμως να τους δίνουν τους κωδικούς πρόσβασης.

Γενικά, μέσω του πρωτοκόλλου OAuth2.0, 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 σύμφωνα με το πρωτόκολλο HTTP (Hypertext Transfer Protocol) και στην ουσία επιτρέπει την έκδοση ειδικών αναγνωριστικών πρόσβασης, που ονομάζονται Oauth2 tokens, σε τρίτους πελάτες. Η έκδοση αυτή επιτυγχάνεται μέσω μιας υπηρεσίας

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

εξουσιοδότησης (π.χ. Keyrock 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.

### 2.3.7 Οντολογίες

Οι οντολογίες[8] είναι 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. Οι οντολογίες αξιοποιούνται από γνωστές τεχνολογίες του Σημασιολογικού Ιστού όπως η γλώσσα ερωτήσεων SPARQL.

### 2.3.8 Οι οντολογίες SSN και SOSA

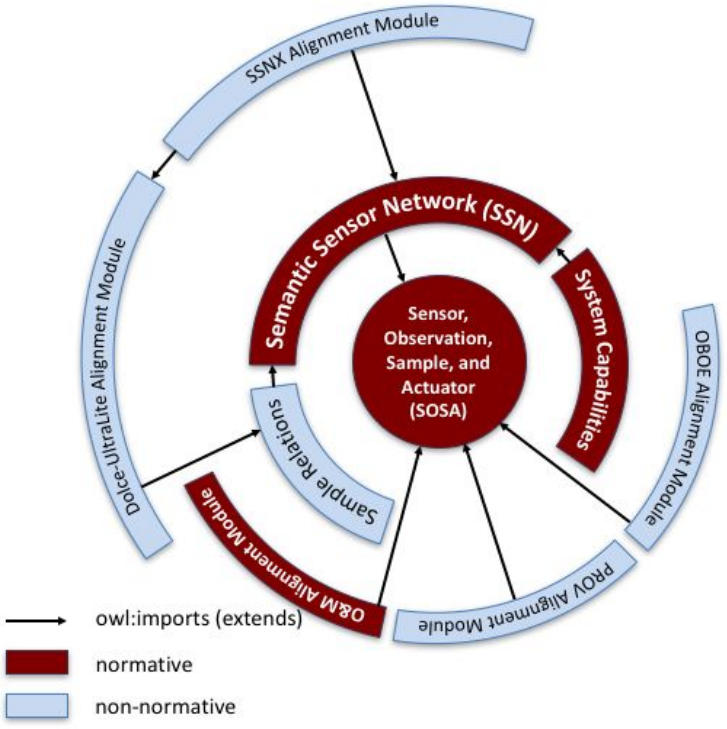
Σύμφωνα με την W3C, η οντολογία Semantic Sensor Network (SSN)[9] είναι μια οντολογία για την περιγραφή αισθητήρων και των μετρήσεών τους (observations), των διαδικασιών που σχετίζονται με αυτές, τα μελετώμενα χαρακτηριστικά ενδιαφέροντος (features of interest), τα δείγματα (samples) που αντιπροσωπεύουν όλα τα παραπάνω, και τις ιδιότητες που έχουν μετρηθεί, όπως και τους ενεργοποιητές (actuators). Με τον όρο "ενεργοποιητές" αναφερόμαστε στα συστατικά μιας μηχανής (π.χ. συσκευής) που είναι υπεύθυνα για την κίνηση και τον έλεγχο ενός μηχανισμού ή συστήματος, π.χ. για το άνοιγμα μιας βαλβίδας. Δηλαδή, πρόκειται για οντότητες που "υποκινούν" συγκεκριμένες ενέργειες.

Η οντολογία SSN ενσωματώνει στον πυρήνα της μια ελαφριά αλλά αυτοδύναμη οντολογία που ονομάζεται SOSA (Sensor, Observation, Sample, and Actuator), την οποία χρησιμοποιεί για τις βασικές κλάσεις και ιδιότητές της. Χάρη στο διαφορετικό πεδίο εφαρμογής τους αλλά και τους διαφορετικούς βαθμούς αξιοποίησης, η SSN και η SOSA είναι ικανές να υποστηρίξουν ένα μεγάλο εύρος εφαρμογών και σεναρίων χρήσης που συμπεριλαμβάνουν δορυφορικές απεικονίσεις, επιστημονική παρακολούθηση μεγάλης κλίμακας, βιομηχανικές και οικιακές υποδομές, πολιτικές επιστήμες, τεχνολογίες

[8] https://en.wikipedia.org/wiki/Ontology_(information_science)
[9] https://www.w3.org/TR/vocab-ssn/

κατασκευής οντολογιών και τον Ιστό των Πραγμάτων. Στο ευρετήριο (namespace) [10] για τους όρους της οντολογίας SSN που υπάρχει στο Διαδίκτυο είναι ορατές οι κλάσεις της οντολογίας, καθώς και οι ιδιότητες και οι συσχετίσεις τους.

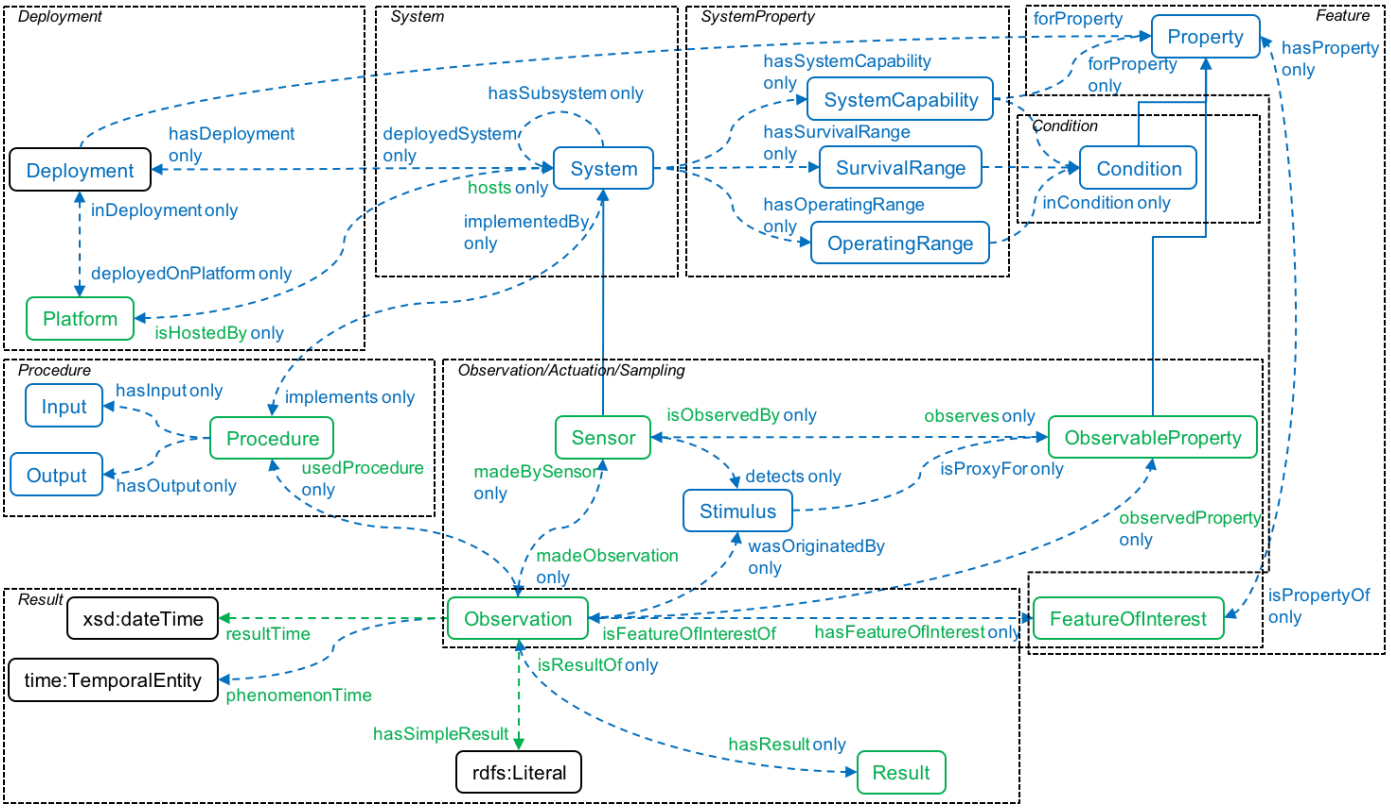
Το βασικό πλεονέκτημα της οντολογίας SSN σε σχέση με την SOSA είναι ότι έχει προσθέτει εκφραστικότητα στην SOSA, λόγω των παραπάνω κλάσεων και ιδιοτήτων που περιλαμβάνει, κι έτσι μπορεί να περιγράψει ένα μεγαλύτερο πλήθος οντοτήτων και με μεγαλύτερη ακρίβεια. Γι' αυτό το λόγο, έχει χρησιμοποιηθεί στις περισσότερες υλοποιήσεις που αφορούν το Διαδίκτυο των Πραγμάτων (IoT projects). Στην Εικόνα 2.6 αποτυπώνεται η δομή της οντολογίας SSN, η οποία περικλείει στον πυρήνα της και επεκτείνει την οντολογία SOSA.




***Εικόνα 2.6**: Μια γενική αποτύπωση της δομής των οντολογιών SOSA και SSN.*

[10] https://www.w3.org/ns/ssn/

Ακολουθεί μια σφαιρική εικόνα της οντολογίας SSN, όπου φαίνονται τα εννοιολογικά μοντέλα της. Στην εικόνα ωστόσο δε φαίνονται όλες οι έννοιες και οι ιδιότητες της οντολογίας.



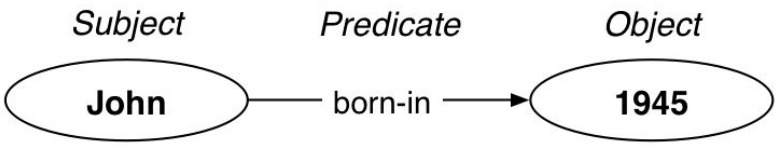

***Εικόνα 2.7**: Μια σφαιρική εικόνα από την οντολογία SSN και τα εννοιολογικά μοντέλα της.*

### 2.3.8 RDF - RDFS

Resource Description Framework (εν συντομία: RDF[11]) ονομάζεται μια οικογένεια προδιαγραφών (specifications) που αναπτύχθηκε από την Κοινοπραξία του Παγκόσμιου Ιστού και σχεδιάστηκε αρχικά ως ένα μοντέλο μεταδεδομένων (metadata data model). Πλέον χρησιμοποιείται ως μια γενική μέθοδος για την εννοιολογική περιγραφή ή για τη μοντελοποίηση των πληροφοριών που πραγματοποιείται στους πόρους του Διαδικτύου. Για την επίτευξη των παραπάνω στόχων, αξιοποιείται μια ποικιλία συντακτικών σημειογραφιών (syntax notations) και μορφών σειριοποίησης δεδομένων. Το πρότυπο RDF χρησιμοποιείται επίσης σε εφαρμογές διαχείρισης γνώσης (knowledge

[11] https://en.wikipedia.org/wiki/Resource_Description_Framework

management applications). Ως μοντέλο, έχει ομοιότητες με κλασικές προσεγγίσεις εννοιολογικής μοντελοποίησης (όπως το μοντέλο ER[12] ή τα διαγράμματα κλάσεων [13]). Βασίζεται στην ιδέα της δημιουργίας δηλώσεων (statements) που αφορούν πόρους - συγκεκριμένα πόρους τους 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.




***Εικόνα 2.8**: Το RDF σχήμα.*

Resource Description Framework Schema (εν συντομία: RDFS[14]) ονομάζεται ένα σύνολο κλάσεων με συγκεκριμένες ιδιότητες, οι οποίες χρησιμοποιούν το επεκτάσιμο RDF μοντέλο δεδομένων (data model) αναπαράστασης γνώσης, που παρέχει βασικά στοιχεία για την περιγραφή των οντολογιών, οι οποίες συναντώνται και με τον όρο "λεξιλόγια RDF" (RDF vocabularies) και 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. Πολλά συστατικά (components) της RDFS συμπεριλαμβάνονται στην Web Ontology Language (OWL), η οποία είναι περισσότερο εκφραστική.

### 2.3.9 OWL

Web Ontology Language (εν συντομία: OWL[15]) ονομάζεται μια οικογένεια γλωσσών αναπαράστασης γνώσης που χρησιμοποιούνται στο χώρο του Σημασιολογικού Ιστού για τη συγγραφή οντολογιών. Οι γλώσσες αυτές χτίζονται

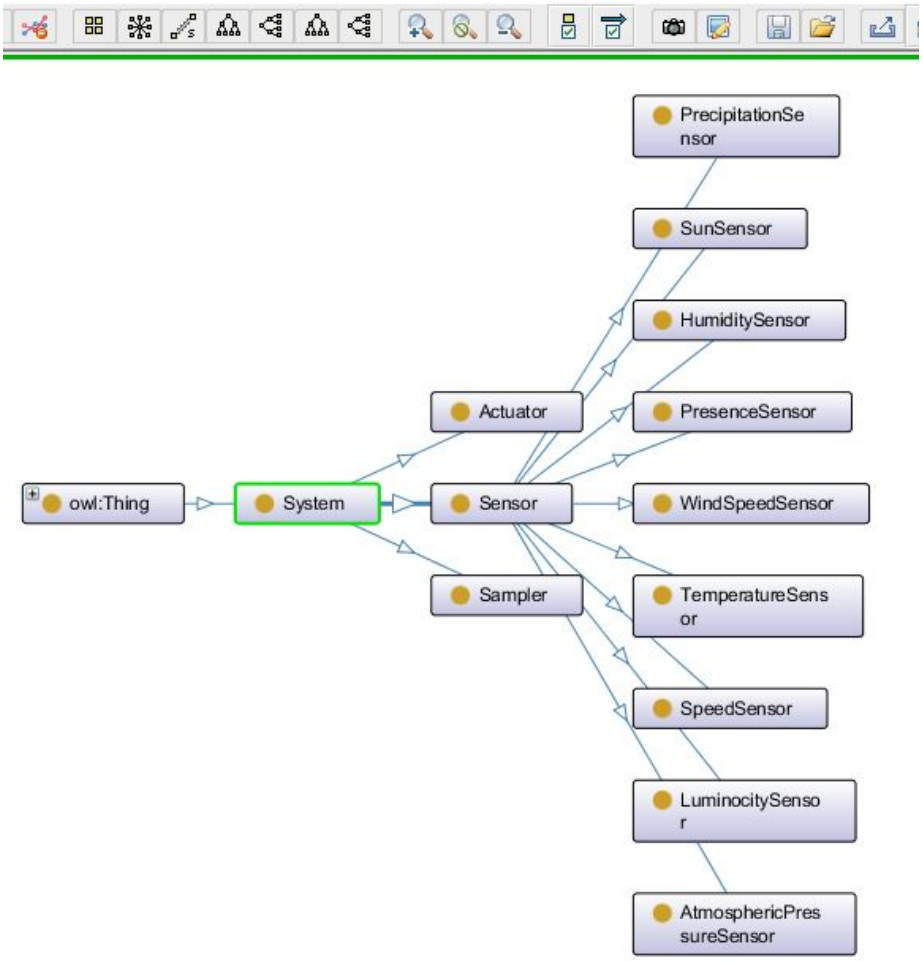
[12] https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model
[13] https://en.wikipedia.org/wiki/Class_diagram
[14] https://en.wikipedia.org/wiki/RDF_Schema
[15] https://en.wikipedia.org/wiki/Web_Ontology_Language

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. Η οικογένεια OWL περιέχει πολλά είδη, σειριοποιήσεις (serializations), συντακτικά και προδιαγραφές (specifications) με παρόμοια ονόματα. Η OWL και η OWL2 χρησιμοποιούνται για την αναφορά στις προδιαγραφές OWL του 2004 και του 2009, αντίστοιχα. Για την αναφορά σε συγκεκριμένη έκδοση προδιαγραφών χρησιμοποιούνται τα πλήρη ονόματα της έκδοσης, τα οποία περιλαμβάνουν και την έκδοση της εκάστοτε προδιαγραφής (π.χ. OWL2 EL). Ωστόσο, για πιο γενικές αναφορές, χρησιμοποιείται ο όρος *Οικογένεια OWL*.

### 2.3.11 Σημασιολογικός Γράφος

Ένας σημασιολογικός γράφος (ή γράφος RDF) είναι ένα δίκτυο κόμβων, το οποίο αναπαριστά σημασιολογικές σχέσεις μεταξύ εννοιών και απαρτίζεται από ένα σύνολο τριάδων RDF.

Στην παρακάτω εικόνα φαίνεται ένα στιγμιότυπο (από το περιβάλλον Protege, βλ. 2.3.19) που απεικονίζει σχηματικά το σημασιολογικό γράφο μιας οντολογίας - συγκεκριμένα αυτής που αναπτύχθηκε για τις ανάγκες των εφαρμογών του συστήματος - και εστιάζει στην κλάση System της οντολογίας (και τις υποκλάσεις της).




***Εικόνα 2.9**: Ένα μέρος του σημασιολογικού γράφου της οντολογίας του συστήματος (δείχνει την ιεραρχία ορισμένων κλάσεων).*

### 2.3.12 SPARQL

Η SPARQL είναι μια γλώσσα ερωτήσεων RDF (Resource Description Framework), δηλαδή μια σημασιολογική γλώσσα ερωτήσεων για βάσεις δεδομένων, η οποία χρησιμοποιείται 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, με τη χρήση της γλώσσας SPARQL, μπορούν να προστεθούν νέα δεδομένα σε γράφους, καθώς και να ανακτηθούν, να ενημερωθούν ή να διαγραφούν υπάρχοντα δεδομένα.

### 2.3.13 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. Η ελεύθερη και ανοικτού κώδικα έκδοση του Virtuoso Universal Server είναι επίσης γνωστή ως Openlink Virtuoso.

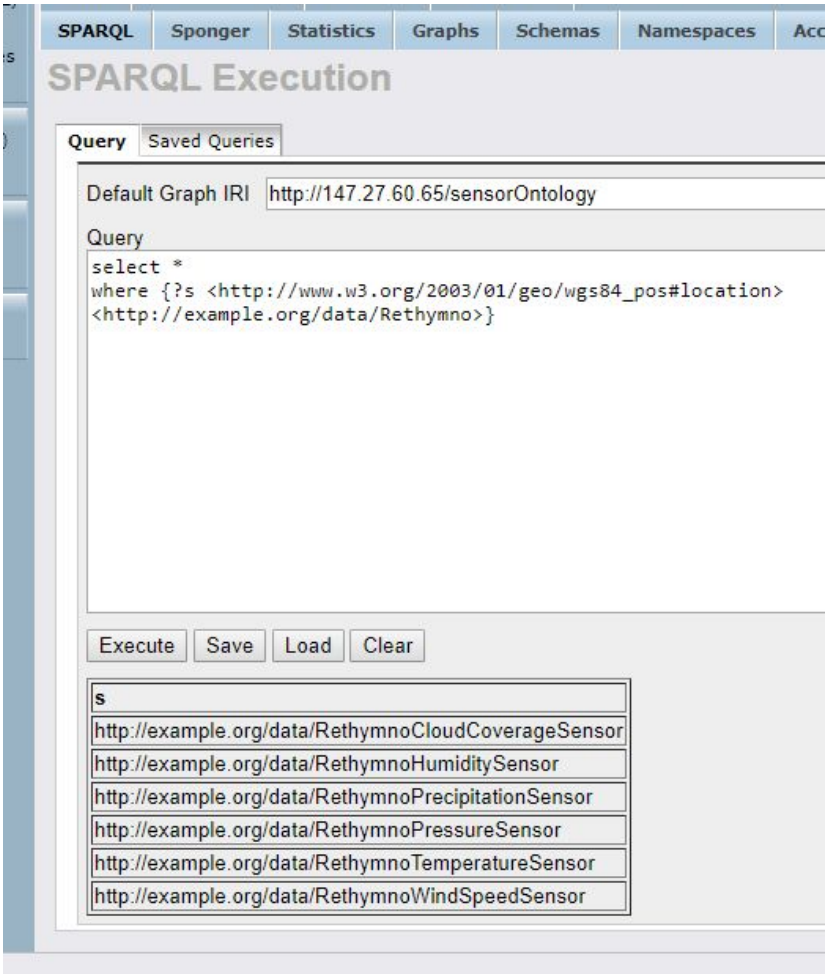

***Εικόνα 2.10**: Virtuoso Conductor (user interface της Virtuoso).*

Χάρη στο τερματικό της Virtuoso (Virtuoso SPARQL Endpoint), μπορούμε να εκτελέσουμε SPARQL ερωτήματα για οποιοδήποτε γράφο από αυτούς που φιλοξενούνται στην Virtuoso. Για παράδειγμα, μπορούμε να τρέξουμε ένα ερώτημα το οποίο θα ζητάει από το γράφο της οντολογίας όλες τις οντότητες (π.χ. αισθητήρες) που βρίσκονται στην τοποθεσία Ρέθυμνο. Αυτό επιτυγχάνεται

με ένα απλό SPARQL ερώτημα και προκύπτει το αποτέλεσμα που φαίνεται στην παρακάτω εικόνα, δηλαδή μια σειρά από αισθητήρες που βρίσκονται (σύμφωνα με τα δεδομένα που έχουν καταχωρηθεί στην οντολογία) στο Ρέθυμνο. Αυτή η πληροφορία φιλοξενείται στο γράφο σε μορφή τριπλετών, όπου, σε κάθε τριπλέτα που αναφέρεται στην τοποθεσία, έχουμε ότι:

- το υποκείμενο είναι το όνομα του αισθητήρα,
- το κατηγόρημα είναι ο υπερσύνδεσμος: "http://www.w3.org/2003/01/geo/wgs84_pos#location"
- και το αντικείμενο είναι ο υπερσύνδεσμος: "http://example.org/data/Rethymno".

Έτσι, παίρνουμε τα παρακάτω αποτελέσματα:




***Εικόνα 2.11**: Αναζήτηση μέσω SPARQL στην Virtuoso.*

### 2.3.12 Java EE (Enterprise Edition)

Η Java EE[16] είναι πλατφόρμα που βασίζεται στη γλώσσα προγραμματισμού Java και προορίζεται για προγραμματισμό εξυπηρετητών (server programming) με προδιαγραφές για enterprise features (?) όπως οι υπηρεσίες ιστού (web services). Η Java EE περιλαμβάνει πολλά specifications που εξυπηρετούν διαφορετικές σκοπιμότητες, όπως η δημιουργία ιστοσελίδων, το διάβασμα και το γράψιμο από μια βάση δεδομένων με ένα συναλλακτικό τρόπο (transactional way), η διαχείριση servlets για αιτήματα HTTP, η διεπαφή προγραμματισμού εφαρμογών Java για Restful υπηρεσίες Ιστού και η επεξεργασία μορφότυπου JSON.

### 2.3.13 Spring Boot

Πρόκειται για ένα πλαίσιο εφαρμογών ανοικτού κώδικα για την πλατφόρμα Java το οποίο παρέχει πολλές επεκτάσεις για τη δημιουργία διαδικτυακών εφαρμογών. Το Spring Boot είναι η convention-over-configuration[17] λύση του Spring[18] για να δημιουργήσουμε αυτόνομες εφαρμογές (βασισμένες σε Spring) που μπορούμε "απλά να τρέξουμε". Τα βασικά χαρακτηριστικά του Spring Boot είναι:

- Να δημιουργεί αυτόνομες εφαρμογές Spring
- Ο ενσωματωμένος Tomcat server
- Να διαμορφώνει αυτόματα (automatically configure) το Spring όταν είναι δυνατόν
- Να παρέχει δογματικά (opinionated) "starter" Project Object Models (POMs) για να απλοποιήσει τη Maven[19] διαμόρφωσή σου

[16]

[17] https://en.wikipedia.org/wiki/Convention_over_configuration
[18] https://spring.io/
[19] https://en.wikipedia.org/wiki/Apache_Maven

### 2.3.13 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, ή 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.

### 2.3.14 Μηχανισμοί συλλογισμού

Η σημασιολογική αιτίαση ή αλλιώς συλλογισμός (reasoning) αποτελεί μια διαδικασία αυτοματοποιημένου συλλογισμού, που χρησιμοποιείται για την εύρεση ασυνεπειών και για την εξαγωγή νέων πληροφοριών από τις πληροφορίες που αντιπροσωπεύονται σε μια οντολογία. Χαρακτηριστικοί μηχανισμοί αιτίασης ή συλλογισμού (reasoners) που χρησιμοποιούνται ευρέως σήμερα είναι ο Pellet, ο FaCT++ και ο HermiT.

Ο Pellet Reasoner[20] αποτελεί έναν ανοικτού κώδικα OWL 2 μηχανισμό αιτίασης (reasoner), ο οποίος είναι βασισμένος σε Java. Μπορεί να χρησιμοποιηθεί σε συνδυασμό με τις βιβλιοθήκες του Apache Jena και του OWL API, καθώς και να ενσωματωθεί σε άλλες εφαρμογές. Όπως κάθε μηχανισμός αιτίασης, μπορεί να αξιοποιηθεί για την εύρεση ασυνεπειών αλλά και για την εξαγωγή νέων πληροφοριών από τις πληροφορίες που αντιπροσωπεύονται σε κάποια οντολογία.

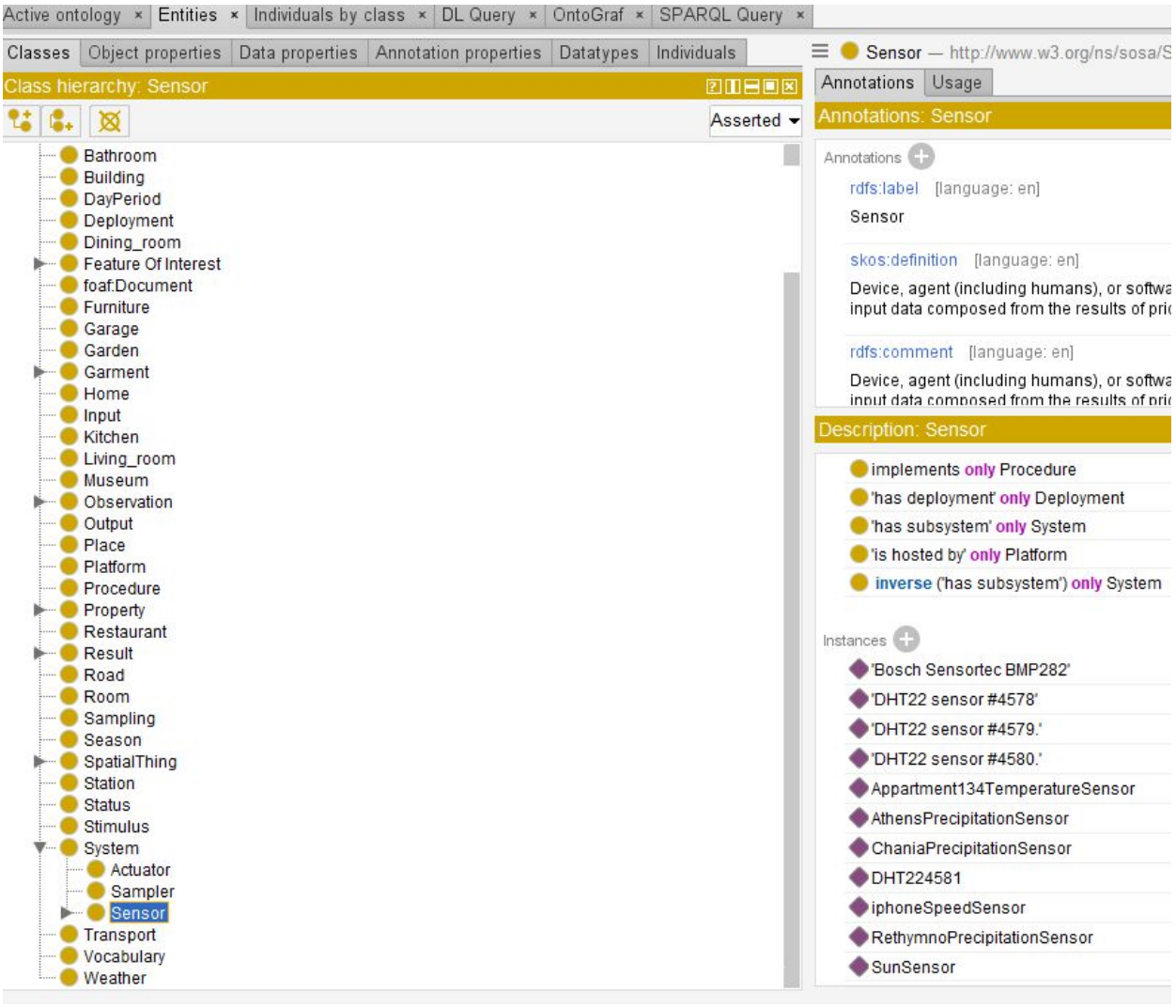
[20] https://www.w3.org/2001/sw/wiki/Pellet

### 2.3.16 Protege

Το Protege[21] είναι μια ελεύθερη, ανοικτού κώδικα πλατφόρμα δημιουργίας και επεξεργασίας οντολογιών και ένα σύστημα διαχείρισης γνώσεων. Διαθέτει μια γραφική διεπαφή χρήστη για ορισμό οντολογιών. Σε αυτήν τη διεπαφή, ο χρήστης μπορεί να εξερευνήσει μια ήδη υπάρχουσα οντολογία (να ανακαλύψει όλες τις κλάσεις και τις ιδιότητές της) και να την επεξεργαστεί ή να δημιουργήσει μια νέα οντολογία. Επιπλέον, το Protege έχει τη δυνατότητα ενσωμάτωσης επεκτάσεων όπως είναι οι μηχανισμοί αιτίασης/συλλογισμού (π.χ. Pellet reasoner) που έχουν σκοπό να επικυρώσουν ότι τα μοντέλα οντολογιών είναι έγκυρα καθώς και να εξαγάγουν νέες πληροφορίες και συμπεράσματα που βασίζονται στην ανάλυση των οντολογιών. Μετά τη σύνταξη μιας οντολογίας, υπάρχει η δυνατότητα εξαγωγής της (export) σε διαφορετικές μορφές αναπαράστασης όπως RDF, RDFS, OWL και XML.

Στην εικόνα 4.11 μπορούμε να δούμε ένα στιγμιότυπο από το περιβάλλον Protege, όπου φαίνεται η οντολογία που έχει δημιουργηθεί για τις ανάγκες των εφαρμογών του Mashup Service.

[21] https://protege.stanford.edu/




***Εικόνα 2.12**: Στιγμιότυπο από την πλατφόρμα Protege, στο οποίο διακρίνονται οι κλάσεις της οντολογίας (αυτής που αναπτύχθηκε για την παρούσα εργασία).*

### 2.3.17 PHP

Η PHP [22](Hypertext Processor) αποτελεί μια server-side γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Στο κομμάτι της υλοποίησης του Back-End της παρούσας εργασίας έγινε χρήση εξυπηρετητών Apache (διακομιστής του Παγκόσμιου Ιστού που είναι συμβατός με την PHP) για τη διαχείριση REST αιτημάτων μεταξύ υπηρεσιών. Συνεπώς, χρησιμοποιήθηκε η PHP, καθώς και ορισμένες επεκτάσεις

[22] https://www.php.net/

της, όπως η cURL[23]. Η cURL είναι μια βιβλιοθήκη της PHP που επιτρέπει την εκτέλεση HTTP αιτημάτων και συνεπώς τη μεταφορά δεδομένων μεταξύ των υπηρεσιών με τη χρήση διαφορετικών πρωτοκόλλων (DICT, FTP, FTPS, HTTP, HTTPS κ.ά.). Σε αυτήν την εργασία η βιβλιοθήκη cURL αξιοποιήθηκε για την κλήση μεθόδων του πρωτοκόλλου HTTP κατευθείαν μέσα από τον κώδικα της γλώσσας PHP.

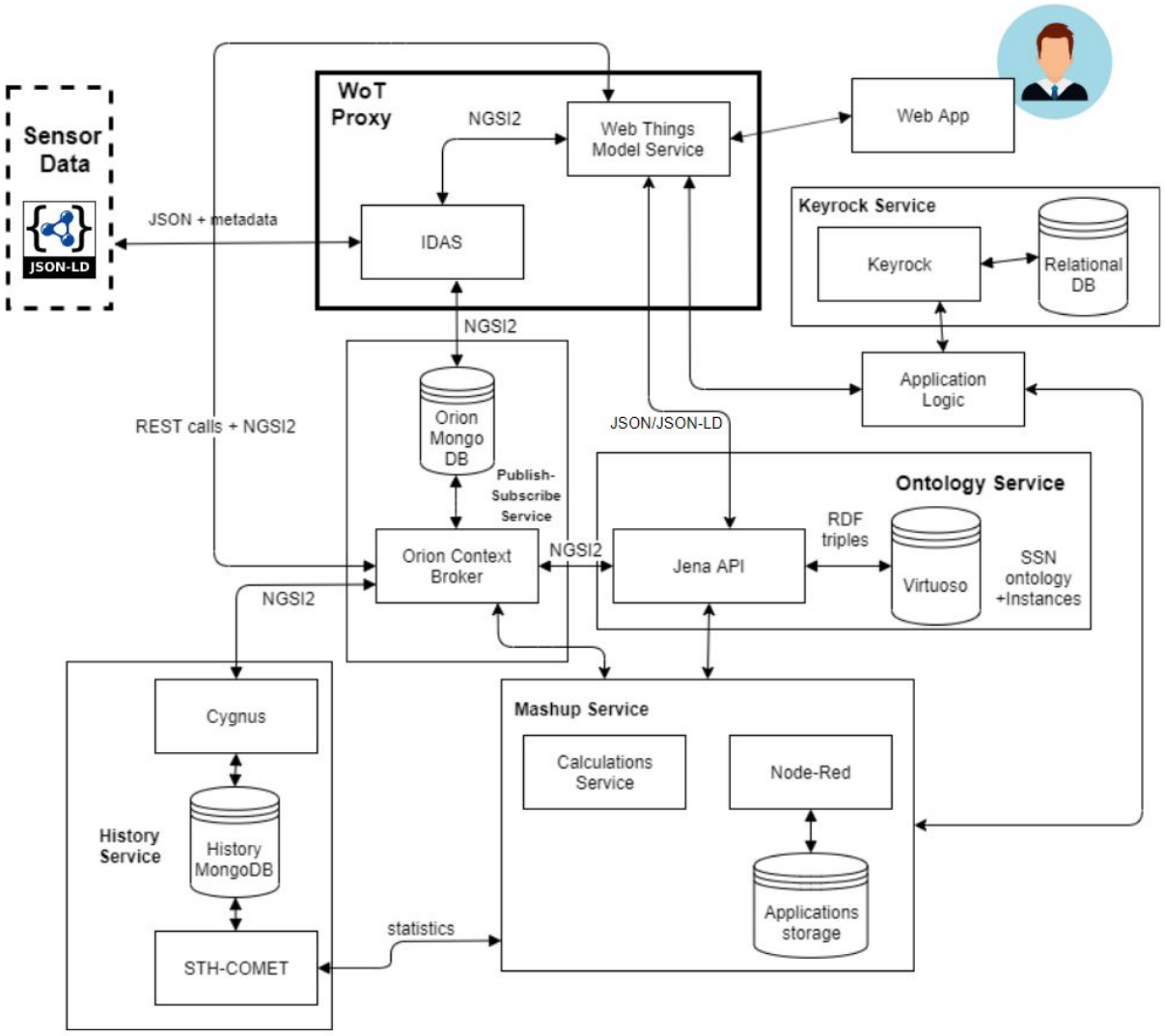
[23] https://curl.haxx.se/

# Κεφάλαιο 3

# Αρχιτεκτονική Συστήματος

## 3.1 Διάγραμμα Αρχιτεκτονικής

Ακολουθεί το διάγραμμα αρχιτεκτονικής του συστήματος, με βάση το οποίο αναπτύχθηκε στη συνέχεια η υλοποίησή του.




*Εικόνα 3.1: Διάγραμμα αρχιτεκτονικής συστήματος.*

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

## 3.2 Διαχείριση Δεδομένων

Η ιστορική βάση MongoDB (History MongoDB) είναι μια μη-σχεσιακή βάση δεδομένων κι έτσι μπορεί να εξυπηρετήσει το μεγάλο όγκο ιστορικών δεδομένων μετρήσεων (χρονικής σειράς) των αισθητήρων/συσκευών που έχουν συνδεθεί στο σύστημα. Ακόμα, όπως έχει αναφερθεί και παραπάνω, η MongoDB είναι μια απλή στη χρήση και γρήγορη βάση δεδομένων, ενώ είναι και συμβατή με δεδομένα της μορφής JSON (NGSI2). Επομένως, ικανοποιεί τις ανάγκες της ιστορικής βάσης δεδομένων. Χάρη στην υπηρεσία ανάκτησης πληροφοριών (COMET) και τις REST μεθόδους που αυτή παρέχει, η πληροφορία, που έχει αποθηκευτεί στην ιστορική βάση και αφορά το ιστορικό μετρήσεων, ανακτάται πολύ εύκολα.

Η σχεσιακή βάση δεδομένων του συστήματος (Relational DB) χρησιμοποιείται για την αποθήκευση πληροφοριών σχετικών με τους χρήστες. Λόγω της φύσης της πληροφορίας που αφορά το προφίλ ενός χρήστη, απαιτείται η χρήση σχεσιακής βάσης δεδομένων για την αποθήκευση αυτής της πληροφορίας.

Η βάση δεδομένων "Applications Storage" χρησιμοποιείται από την υπηρεσία Mashup (υλοποιείται στην εργασία του Στέλιου Μποτωνάκη [2]) για να αποθηκεύονται οι εφαρμογές που κατασκευάζουν οι προγραμματιστές. Πρόκειται για μη-σχεσιακή βάση δεδομένων MongoDB, διότι οι εφαρμογές αποθηκεύονται σε μορφή JSON.

Η βάση δεδομένων Orion MongoDB χρησιμοποιείται από την Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Orion Context Broker), με σκοπό αυτή να αποθηκεύει τις οντότητες πληροφορίας που πρέπει να διαχειριστεί. Πάλι πρόκειται για μη-σχεσιακή βάση δεδομένων MongoDB, αφού οι οντότητες πληροφορίας που διαχειρίζεται η υπηρεσία έχουν μορφή JSON.

Τέλος, στο Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) της Virtuoso περιέχεται ο γράφος RDF, στον οποίο έχει αποθηκευτεί η οντολογία του συστήματος. Η οντολογία αυτή 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 αισθητήρων. Όπως και οι υπόλοιπες υπηρεσίες, περιλαμβάνει μικρότερες υπηρεσίες, καθώς και μια βάση δεδομένων MongoDB (βλ. διάγραμμα αρχιτεκτονικής), που συνεργάζονται μεταξύ τους για να εξυπηρετήσουν τις ανάγκες του History Service.

Συγκεκριμένα, η παρούσα υπηρεσία αποτελείται από την υπηρεσία Cygnus (βλ. Κεφάλαιο 2), που αποτελεί σύνδεσμο μεταξύ του Orion Context Broker (στέλνει δεδομένα μετρήσεων σε μορφή NGSI2) και της βάσης δεδομένων MongoDB. Το Cygnus δέχεται τις ροές δεδομένων NGSI2 και τις αποθηκεύει στην History MongoDB, δηλαδή στην ιστορική βάση δεδομένων του συστήματος (βλ. *"Βάσεις δεδομένων στο σύστημα"*). Επιπλέον, έχει τη δυνατότητα να αποθηκεύει τα ιστορικά δεδομένα τόσο ακατέργαστα (Raw) όσο και συγκεντρωτικά (Aggregated), στην ιστορική βάση δεδομένων. Με αυτόν τον τρόπο, διατηρείται το ιστορικό των μετρήσεων των αισθητήρων.

Στη συνέχεια, η υπηρεσία COMET (βλ. Κεφάλαιο 2) χρησιμοποιείται για την ανάκτηση των δεδομένων που έχουν αποθηκευτεί στην ιστορική βάση MongoDB. Ο COMET έχει το πλεονέκτημα ότι παρέχει REST μεθόδους για εξειδικευμένα ερωτήματα (queries) που αφορούν στατιστικά (π.χ. για ανάκτηση του αθροίσματος, του μέγιστου ή του ελάχιστου από ένα πλήθος τιμών), καθώς και ερωτήματα για συγκεκριμένα χρονικά διαστήματα (π.χ. για ανάκτηση δεδομένων κατά το διάστημα ενός ή παραπάνω λεπτών, ωρών, ημερών ή ακόμα και μηνών). Τα δεδομένα που λαμβάνονται μέσω του COMET, και συγκεκριμένα τα στατιστικά, αξιοποιούνται στη συνέχεια από άλλη υπηρεσία (Mashup Service), όπως φαίνεται και στο διάγραμμα αρχιτεκτονικής του συστήματος. Το Mashup Service μπορεί να εκμεταλλευτεί κατάλληλα το γεγονός ότι λαμβάνει στατιστικά για επιλεγμένες χρονικές περιόδους, ούτως ώστε να εξυπηρετήσει τις ανάγκες των εφαρμογών.

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

Η υπηρεσία Διακομιστή μεσολάβησης του Ιστού των Πραγμάτων απαρτίζεται από δύο υπηρεσίες: την υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service) και την Υπηρεσία Διαχείρισης

Συσκευών Backend IDAS. Χάρη σε αυτές, ο Διακομιστής μεσολάβησης επικοινωνεί 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, αποτελώντας είσοδο δεδομένων από τον έξω κόσμο, καθώς και έξοδο προς αυτόν (π.χ. όταν δίνεται εντολή προς μια συσκευή ή κάποιος χρήστης ανακτά μια πληροφορία).

Συγκεκριμένα, η υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων είναι υπεύθυνη για την πραγματοποίηση 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 εξυπηρετήσει όλες τις λειτουργίες που επιλέγει ο χρήστης μέσω αυτής. Παράλληλα, αλληλεπιδρά τόσο με την Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών του συστήματος όσο και με την Υπηρεσία Οντολογίας, με τις οποίες έχει αμφίδρομη σχέση, καθώς και λαμβάνει και 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 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. Τέλος, 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 (επομένως ζητάει πληροφορία από την Υπηρεσία Διαχείρισης 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 εν λόγω υπηρεσία έχει έναν πολύ ειδικό ρόλο: λαμβάνει τα δεδομένα που στέλνει κάθε συσκευή, ανεξάρτητα από το πρωτόκολλο κάθε συσκευής, και μεταβιβάζει την πληροφορία στην Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών αποκλειστικά σε μορφή JSON-LD

(NGSI2). Επομένως, το έργο της υπηρεσίας IDAS είναι πολύ σημαντικό, διότι πετυχαίνει προσαρμογή πρωτοκόλλου (protocol adaptation) και καταφέρνει να καταστήσει την επικοινωνία ανεξάρτητη από το πρωτόκολλο μετάδοσης της συσκευής. Με αυτόν τον τρόπο, 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 Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Publish - Subscribe Service)

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

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

Οι οντότητες αυτές αποθηκεύονται σε μορφή JSON-LD (NGSI2) στη βάση δεδομένων Orion MongoDB (βλ. "Βάσεις Δεδομένων του Συστήματος"), που είναι συμβεβλημένη με την υπηρεσία Orion Context Broker του FIWARE. Ακολουθεί μια περιγραφή της πληροφορίας κάθε είδους οντότητας.

Η αναπαράσταση JSON-LD ενός στιγμιότυπου της οντότητας "Αισθητήρα" (που είναι στην ουσία ένα μοντέλο συσκευής, δηλαδή ένα Web Thing Model) περιέχει δεδομένα και μεταδεδομένα για τα χαρακτηριστικά μιας οποιασδήποτε συσκευής, όπως μοναδικό αναγνωριστικό (έχει το όνομα του αντίστοιχου Πράγματος), context (υπερσύνδεσμο σε τοποθεσία που προσφέρει πληροφορία για το Πράγμα), τοποθεσία όπου βρίσκεται τοποθετημένη η συσκευή (π.χ. σπίτι,

πόλη ή γεωγραφικό σημείο), χαρακτηριστικά μέτρησης (π.χ. ότι μετράει θερμοκρασία), κατασκευάστρια εταιρεία της συσκευής, κλπ.

Η αναπαράσταση JSON ενός στιγμιότυπου της οντότητας "Πράγματος" (Web Thing) περιέχει πληροφορία για τα χαρακτηριστικά μιας οποιασδήποτε συσκευής, όπως μοναδικό αναγνωριστικό (έχει το όνομα του Πράγματος), context (υπερσύνδεσμο σε τοποθεσία που προσφέρει πληροφορία για το Πράγμα), μια σύντομη περιγραφή του Πράγματος, κλπ.

Η αναπαράσταση JSON-LD ενός στιγμιότυπου της οντότητας "Μέτρηση" (Observation) περιέχει πληροφορία για τα χαρακτηριστικά μιας συγκεκριμένης μέτρησης ενός αισθητήρα, όπως την τιμή της μέτρησης, τη φύση της μέτρησης, κλπ.

Η αναπαράσταση JSON ενός στιγμιότυπου της οντότητας "Σπίτι" (Home) περιέχει πληροφορία για τα χαρακτηριστικά ενός συγκεκριμένου σπιτιού, όπως την τοποθεσία του, κλπ.

Η αναπαράσταση JSON ενός στιγμιότυπου της οντότητας "Εφαρμογή" (Application) αναφέρεται σε ένα συγκεκριμένο προγραμματιστή εφαρμογών και συνιστά μια περιγραφή της εφαρμογής που αυτός έχει δημιουργήσει. Περιλαμβάνει, δηλαδή, χαρακτηριστικά όπως μοναδικό αναγνωριστικό στιγμιότυπου, όνομα εφαρμογής και πεδίο ενδιαφέροντος εφαρμογής.

Η αναπαράσταση JSON-LD ενός στιγμιότυπου της οντότητας "Εκτέλεση ενέργειας" (Actuation) περιέχει πληροφορία για τα χαρακτηριστικά μιας συγκεκριμένης εκτέλεσης ενέργειας μια συσκευή. Περιλαμβάνει, δηλαδή, χαρακτηριστικά όπως μοναδικό αναγνωριστικό στιγμιότυπου, μια χαρακτηριστική τιμή που φανερώνει την ενέργεια (π.χ. 0 για "off" ή 1 για "on"), ένα σημασιολογικό πλαίσιο (context) για τη συγκεκριμένη εκτέλεση ενέργειας, καθώς και την ιδιότητα μιας οντότητας στην οποία έχει επίδραση η ενέργεια (π.χ. η εκτέλεση ενέργειας με αναγνωριστικό "Room245SwitchOnLights" έχει επίδραση στη φωτεινότητα της οντότητας Room245). Επιπλέον, μπορεί να περιλαμβάνει την πληροφορία για το ποια συσκευή εκτέλεσε την ενέργεια.

Η αναπαράσταση JSON ενός στιγμιότυπου της οντότητας "Χωρική οντότητα" περιγράφει χαρακτηριστικά μιας συγκεκριμένης τοποθεσίας (π.χ. πόλης), όπως είναι οι καιρικές και ατμοσφαιρικές συνθήκες της.

Χάρη στη RESTful διεπαφή που παρέχει η Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών, η πληροφορία που σχετίζεται με τις παραπάνω οντότητες είναι δυνατό να αποθηκεύεται - παραμετροποιείται, να ανακτάται, να ενημερώνεται αλλά και να διαγράφεται. Η λειτουργία συνδρομής (subscription)

προς οντότητες που διαχειρίζεται η υπηρεσία είναι μια αξιοσημείωτη μέθοδος της RESTful διεπαφής. Συγκεκριμένα, στην περίπτωση που υπάρξει κάποια αλλαγή στις τιμές των χαρακτηριστικών της οντότητας, πυροδοτείται μια ενημέρωση με περιεχόμενο αυτήν την αλλαγή, από την εν λόγω υπηρεσία στον κάτοχο της συνδρομής. Στην παρούσα εργασία, έγινε χρήση της λειτουργίας αυτής με σκοπό τη διατήρηση του ιστορικού χρονικής σειράς για τις αλλαγές στις μετρήσεις των συσκευών (καθώς η πληροφορία για κάθε αλλαγή τιμής φτάνει στην ιστορική βάση δεδομένων), ενώ αξιοποιήθηκε η ίδια λειτουργία και για την ενημέρωση της οντολογίας του συστήματος σχετικά με τις αλλαγές στις τιμές των συσκευών. Δηλαδή, κάθε φορά που μια μέτρηση (type: Observation) αλλάξει τιμή, ένα αίτημα HTTP με σώμα την πληροφορία αυτής της αλλαγής φτάνει στην Virtuoso κι ενημερώνει τις τιμές των συσκευών που έχουν αποθηκευτεί στο γράφο της οντολογίας.

## 3.5 Υπηρεσία Οντολογίας (Ontology Service)

Η Υπηρεσία Οντολογίας είναι υπεύθυνη για την αλληλεπίδραση της οντολογίας (ή των οντολογιών) του συστήματος με τις υπόλοιπες υπηρεσίες της πλατφόρμας. Περιλαμβάνει 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 (στην περίπτωσή μας του Pellet Reasoner).

Παρόλο που αρχικά μελετήθηκε η οντολογία SOSA και η χρήση της στο παρόν σύστημα, τελικά για τις ανάγκες περιγραφής των συσκευών / αισθητήρων επιλέχθηκε η οντολογία SSN, η οποία παρέχει ***μεγαλύτερη εκφραστικότητα*** στην περιγραφή των οντοτήτων (όπως εξηγήθηκε και στο Κεφάλαιο 2). Αυτό συμβαίνει επειδή η οντολογία SSN περιέχει περισσότερες κλάσεις και ιδιότητες, επομένως είναι ικανή να περιγράψει περισσότερες έννοιες και οντότητες (και με μεγαλύτερη ακρίβεια).

## 3.6 Υπηρεσία Σύνθεσης Εφαρμογών (Mashup Service)

Η Υπηρεσία Σύνθεσης Εφαρμογών επιτρέπει στους προγραμματιστές την κατασκευή IoT εφαρμογών και επιτρέπει στους χρήστες την πρόσβαση σε αυτές. Ειδικότερα, αξιοποιεί την πληροφορία που ανακτά από την υπηρεσία Διαχείρισης Δεδομένων (τα προαναφερόμενα στατιστικά), καθώς - και απαραίτητα για τις εφαρμογές - δεδομένα που προέρχονται τόσο από την Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών όσο και από την Υπηρεσία Οντολογίας (τα λαμβάνει μέσω του Jena API). Να σημειωθεί εδώ ότι η υπηρεσία αυτή υλοποιήθηκε εξ' ολοκλήρου από το συμφοιτητή μου, Μποτωνάκη Στυλιανό, και αναλύεται εκτενέστερα στην εργασία του [2].

## 3.7 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. Για τη σύνδεσή του στο σύστημα πληκτρολογεί στη σελίδα εισόδου το email του και τον κωδικό πρόσβασης. Το αίτημα σύνδεσης δρομολογείται στην Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών (μέσω της RESTful διεπαφής της), η οποία ελέγχει την ταυτότητα του χρήστη. Η ταυτότητα αυτή ελέγχεται χάρη στην αξιοποίηση του μηχανισμού OAuth2.0 και του OAuth2 token (η διαδικασία θα παρουσιαστεί αναλυτικά σε επόμενο κεφάλαιο).

# Κεφάλαιο 4

# Υλοποίηση Συστήματος

## 4.1 Υλοποίηση Υπηρεσιών

Παρακάτω παρατίθενται οι υπηρεσίες που αναπτύχθηκαν και συνιστούν μέρος της υλοποίησης του συστήματος, ενώ γίνεται και μια αναλυτική περιγραφή της χρήσης και της λειτουργίας για την κάθε υπηρεσία ξεχωριστά.

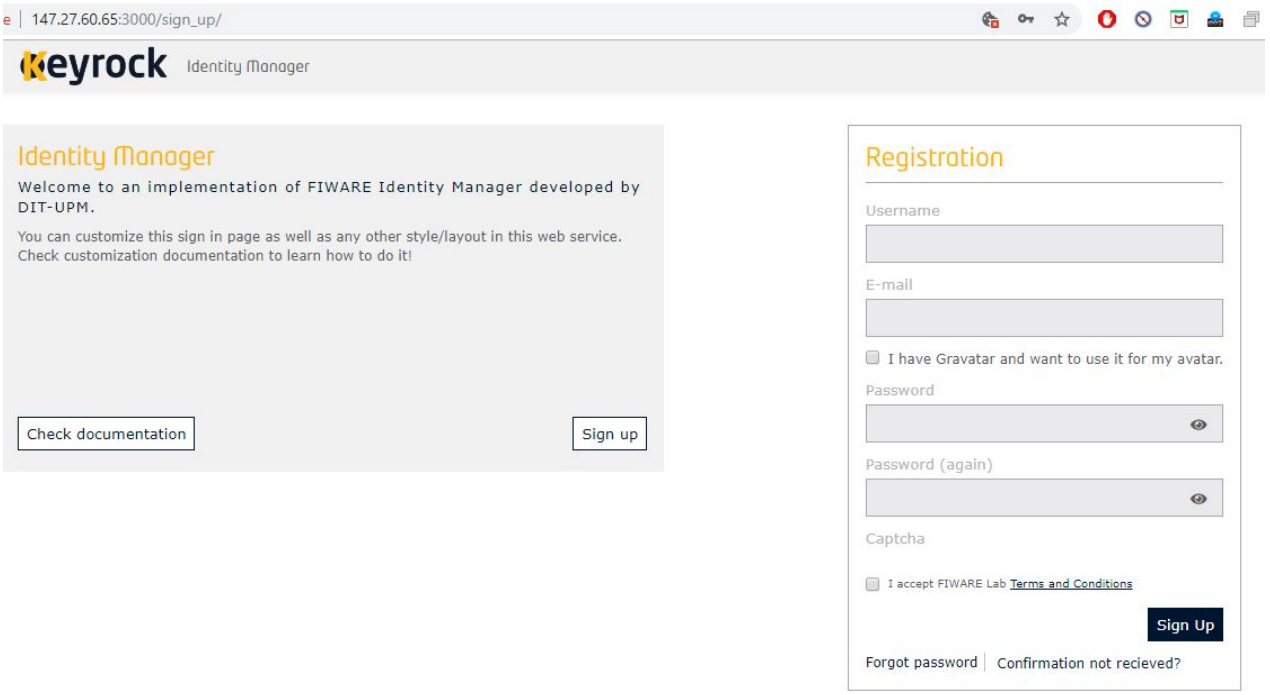
***Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών - Keyrock Identity Management (Keyrock IdM)***

Η υπηρεσία αυτή έχει αναπτυχθεί και παρέχεται από το FIWARE. Η λειτουργία της βασίζεται στη χρήση του πρωτοκόλλου εξουσιοδότησης OAuth2, χάρη στο οποίο γίνεται η ταυτοποίηση των χρηστών και η παροχή εξουσιοδότησης (authorization) πρόσβασης σε υπηρεσίες. Παρακάτω δίνεται μια περιγραφή του τρόπου λειτουργίας της υπηρεσίας, ο οποίος περιλαμβάνει τα εξής τρία στάδια: 1) την εγγραφή του συστήματος από κάποιο διαχειριστή (administrator) στην υπηρεσία Keyrock IdM, 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. Στη συνέχεια, αναλύονται τα τρία παραπάνω στάδια:

1) Αρχικά, πραγματοποιείται η διαδικασία εγγραφής του συστήματος στην υπηρεσία Keyrock IdM από το διαχειριστή. Συγκεκριμένα, ο πίνακας ελέγχου της υπηρεσίας Keyrock IdM παρέχει ένα γραφικό περιβάλλον, 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 συνδυάζονται με τη μέθοδο κωδικοποίησης base64 (client_id: client_secret) και οδηγούν στην παραγωγή ενός επιπλέον αναγνωριστικού που ονομάζεται "Authorization_Basic". Αυτό το αναγνωριστικό συνιστά για την υπηρεσία Keyrock IdM την ταυτότητα του συστήματος. Επομένως, κάθε αίτημα σύνδεσης στο σύστημα πρέπει οπωσδήποτε να περιλαμβάνει στην κεφαλίδα του (request header) το αναγνωριστικό "Authorization_Basic".

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) και το οποίο παρέχεται από τον πίνακα ελέγχου της υπηρεσίας Keyrock IdM, φαίνεται στην παρακάτω εικόνα:




*Εικόνα 4.1: Στιγμιότυπο από το γραφικό περιβάλλον όπου γίνεται η εγγραφή (sign up) ενός χρήστη στο σύστημα.*

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 Keyrock IdM. Η κεφαλίδα του αιτήματος περιέχει το διακριτικό "Authorization_Basic" και το σώμα του (request body) τα στοιχεία εισόδου του χρήστη, προκειμένου να γίνει η ταυτοποίηση. Μόλις αυτή ολοκληρωθεί επιτυχώς, επιστρέφεται ένα OAuth2 token από την υπηρεσία Keyrock 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. 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 της HTTP μεθόδου της υπηρεσίας Keyrock IdM που αναλύθηκε παραπάνω:

*Πίνακας 4.1: HTTP μέθοδος της υπηρεσίας Keyrock IdM.*

| Μέθοδος | URL Μεθόδου | Header | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|---|
| POST | /oauth2/token | Authorization: base64 (client_id: client_secret) | { &username = "username" &password = "user_password" } | Δίνεται ένα έγκυρο username και password και επιστρέφεται ένα OAuth2 token. |

### *Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών*

Ο τρόπος λειτουργίας της υπηρεσίας Orion Context Broker (η οποία επίσης είναι παροχή του FIWARE) βασίζεται στα πρότυπα του μοντέλου πληροφορίας NGSI-2 και στόχος της υπηρεσίας είναι η διαχείριση δεδομένων πλαισίου. Αυτή επιτυγχάνεται μέσω της RESTful διεπαφής του Orion Context Broker. Παρακάτω γίνεται μια περιγραφή της χρήσης και των λειτουργιών της υπηρεσίας στην υλοποίηση του συστήματος.

#### Orion Context Broker - Δημιουργία και ενημέρωση οντότητας

Στην παρούσα εργασία έχει γίνει η σχεδιαστική επιλογή να διατηρούνται στον Orion Context Broker οι οντότητες (entities) της μορφής NGSI οι οποίες περιγράφουν:

- Κάθε διαφορετικό “Πράγμα” (Thing) που έχει αποθηκευτεί στην πλατφόρμα.
- Κάθε διαφορετικό μοντέλο τύπου συσκευής (μοντέλο Πράγματος).
- Κάθε διαφορετική μέτρηση που έχει αποθηκευτεί στο σύστημα.
- Κάθε διαφορετικό σπίτι (που περιέχει αισθητήρες/συσκευές).

- Κάθε διαφορετική εφαρμογή που έχει δημιουργηθεί στο σύστημα από κάποιο προγραμματιστή εφαρμογών.
- Κάθε διαφορετική εκτέλεση ενέργειας (actuation) που έχει πραγματοποιηθεί από κάποια συσκευή.
- Κάθε διαφορετική μετρούμενη ιδιότητα (observable property) που έχει αποθηκευτεί στην πλατφόρμα.
- Κάθε διαφορετική χωρική οντότητα (Spatial Thing), δηλαδή τοποθεσία - για παράδειγμα, η πόλη του Ρεθύμνου - για την οποία έχουν καταγραφεί πληροφορίες στο σύστημα.

Προκειμένου να δημιουργηθεί μια νέα NGSI οντότητα στην υπηρεσία Orion Context Broker, πραγματοποιείται ένα HTTP αίτημα της μεθόδου POST, ενώ για την ενημέρωση μιας οντότητας χρησιμοποιείται η μέθοδος PUT.

Ακολουθεί ένα παράδειγμα της οντότητας "Αισθητήρα" (Sensor). Η οντότητα αυτή γενικά περιέχει πληροφορία για το μοντέλο ενός Πράγματος, δηλαδή ενός αισθητήρα ή γενικότερα μιας συσκευής. Στο συγκεκριμένο παράδειγμα, πρόκειται για το μοντέλο ενός υποθετικού αισθητήρα που μετράει την υγρασία στην πόλη των Χανίων.


```
{
    "id": "ChaniaHumiditySensor",
    "type": "Sensor",
    "context": "http://example.org/data/ChaniaHumiditySensor",
    "forProperty": {
        "type": "Relationship",
        "object": "http://example.org/data/ChaniaHumidty",
        "context": "http://www.w3.org/ns/ssn/forProperty"
    },
    "location": {
        "type": "Relationship",
        "object": "http://example.org/data/Chania",
        "context": "http://www.w3.org/2003/01/geo/wgs84_pos#location"
    },
    "madeObservation": {
        "type": "Relationship",
        "object": "http://example.org/data/ChaniaHumidityObservation",
        "context": "http://www.w3.org/ns/sosa/madeObservation"
    },
    "observes": {
        "type": "Relationship",
        "object": "http://example.org/data/ChaniaHumidty",
        "context": "http://www.w3.org/ns/sosa/observes"
    }
},
```


***Εικόνα 4.2:*** *Αναπαράσταση JSON-LD της οντότητας "ChaniaHumiditySensor".*

Τα χαρακτηριστικά που φαίνονται στην παραπάνω αναπαράσταση JSON-LD περιγράφονται παρακάτω:

**"id"**: Πρόκειται για το μοναδικό αναγνωριστικό του συγκεκριμένου μοντέλου συσκευών.

**"type"**: Δηλώνει τον τύπο της οντότητας, ο οποίος στην περίπτωση αυτή είναι "Sensor" επομένως πρόκειται για μοντέλο αισθητήρα (Πράγματος).

**"context"**: Αυτό το χαρακτηριστικό περιέχει σημασιολογική πληροφορία για το συγκεκριμένο μοντέλο αισθητήρα, δηλαδή το μοντέλο έχει τη σημασία που συνιστά ο διαδικτυακός σύνδεσμος "http://example.org/data/ChaniaHumiditySensor" (που είναι η τιμή του χαρακτηριστικού context). Χάρη στον παραπάνω υπερσύνδεσμο, με άλλα λόγια, μπορούμε να πάρουμε επιπλέον πληροφορίες για το παρόν μοντέλο.

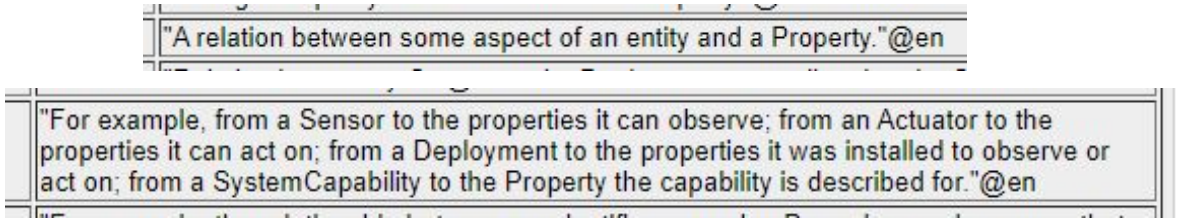
**"forProperty"**: Πρόκειται για ένα χαρακτηριστικό που δηλώνει ότι ο συγκεκριμένος τύπος αισθητήρα σχετίζεται με μια συγκεκριμένη ιδιότητα, η οποία στην περίπτωση αυτή είναι η υγρασία στην πόλη των Χανίων. Το "forProperty" συνιστά ένα αντικείμενο JSON, το οποίο θα μετατραπεί στη συνέχεια (στην Υπηρεσία Οντολογίας) σε μια σημασιολογική τριπλέτα και θα αποθηκευτεί στην Οντολογία. Όπως παρατηρούμε, στο αντικείμενο JSON "forProperty" (δηλαδή στα χαρακτηριστικά που περικλείονται στα άγκιστρα) περιέχονται το "type", το "object" και το "context". Το εσωτερικό αυτό "context" - διαφέρει από αυτό που αναφέρθηκε παραπάνω - υποδηλώνει μια σχέση ανάμεσα στο μοντέλο συσκευής και στο "object". Αν ανατρέξουμε στην οντολογία SSN, από την οποία προέρχεται το συγκεκριμένο "context" ("http://www.w3.org/ns/ssn/forProperty"), θα δούμε ότι πρόκειται για ένα Object Property της Οντολογίας που υποδηλώνει τη σχέση ανάμεσα σε μια πτυχή μιας οντότητας και σε μια ιδιότητα (βλ. Εικόνα 4.5). Επομένως, στην περίπτωση αυτή παίρνουμε την πληροφορία ότι το συγκεκριμένο μοντέλο συσκευής σχετίζεται με την ιδιότητα "υγρασία στην πόλη των Χανίων" (αυτή τη σημασία συνιστά ο διαδικτυακός σύνδεσμος που υποδηλώνει την τιμή του object, δηλαδή το "http://example.org/data/ChaniaHumidity"). Να σημειωθεί ωστόσο ότι το "forProperty", γενικά, δε δηλώνει απαραίτητα μια μετρήσιμη ιδιότητα, αλλά μια οποιασδήποτε φύσης ιδιότητα που αφορά αισθητήρες ή συσκευές.

**"location"**: Αυτό το χαρακτηριστικό δηλώνει την τοποθεσία που έχει ο συγκεκριμένος τύπος συσκευής. Με την ίδια λογική που περιγράφηκε παραπάνω, παίρνουμε την πληροφορία ότι αυτός ο τύπος συσκευής έχει ως τοποθεσία τα Χανιά.

**“madeObservation”**: Πρόκειται για ένα χαρακτηριστικό που δηλώνει τη σχέση ανάμεσα σε έναν τύπο / μοντέλο συσκευής και σε μια οντότητα μέτρησης (που επίσης είναι αποθηκευμένη στον Orion ως ChaniaHumidityObservation). Συμπεραίνουμε, ακολουθώντας ξανά την ίδια λογική, ότι ο αυτός ο τύπος συσκευής σχετίζεται με μια συγκεκριμένη μέτρηση της υγρασίας (από αισθητήρα) στα Χανιά, δηλαδή παίρνουμε την πληροφορία ότι η εν λόγω μέτρηση έγινε από συσκευή αυτού του τύπου.

**“observes”**: Δηλώνει μια μετρήσιμη ιδιότητα η οποία σχετίζεται με το παρόν μοντέλο αισθητήρα, δηλαδή αυτός ο τύπος αισθητήρα μετράει την ιδιότητα που υποδηλώνει εδώ το object. Όπως είναι φανερό, η ιδιότητα αυτή είναι η υγρασία στα Χανιά.

Ακολουθούν δύο στιγμιότυπα από την οντολογία SSN που σχετίζονται με τον διαδικτυακό πόρο: “http://www.w3.org/ns/ssn/forProperty”. Συγκεκριμένα, απεικονίζονται δύο αντικείμενα (predicates) από τριπλέτες που έχουν ως υποκείμενο (subject) το συγκεκριμένο διαδικτυακό σύνδεσμο. Τις πληροφορίες αυτές μπορούμε να τις αντλήσουμε επίσης αν μεταβούμε στην διαδικτυακή τοποθεσία που οδηγεί ο εν λόγω υπερσύνδεσμος.




***Εικόνα 4.3:*** *Στιγμιότυπα από την οντολογία SSN όπου βρίσκεται πληροφορία για τη σημασία του πόρου: “http://www.w3.org/ns/ssn/forProperty”.*

Στη συνέχεια, παρατίθενται στιγμιότυπα με παραδείγματα από τους υπόλοιπους τύπους οντοτήτων (μέτρηση, σπίτι κλπ) που φιλοξενούνται στον Orion Context Broker:

- Οντότητα Πράγματος (Thing):

```
{
    "id": "DHT224581",
    "type": "Thing",
    "context": "http://example.org/data/DHT224581",
    "description": "A Thing observing humidity of Room245"
}
```

***Εικόνα 4.4:*** *Στιγμιότυπο οντότητας Πράγματος (τύπου συσκευής) με αναγνωριστικό “DHT224581”.*

- Οντότητα σπιτιού (home):


```
{
    "id": "Apartment134",
    "type": "Home",
    "context": "http://example.org/data/Apartment134",
    "hasActuationState": {
        "type": "Relationship",
        "object": "http://example.org/data/SwitchOffAirConditioning",
        "context": "http://sensormeasurement.appspot.com/ont/m3/home#hasActuationState"
    },
    "hasHumidityState": {
        "type": "Relationship",
        "object": "http://example.org/data/VeryDryHumidity",
        "context": "http://sensormeasurement.appspot.com/ont/m3/home#hasHumidityState"
    },
    "hasLuminocityState": {
        "type": "Relationship",
        "object": "http://example.org/data/LowLighting",
        "context": "http://sensormeasurement.appspot.com/ont/m3/home#hasLuminocityState"
    },
    "hasPresenceState": {
        "type": "Relationship",
        "object": "http://example.org/data/SomeoneInTheRoom",
        "context": "http://sensormeasurement.appspot.com/ont/m3/home#hasPresenceState"
    },
    "hasProperty": {
        "type": "Relationship",
        "object": "http://example.org/data/Apartment134Presence",
        "context": "http://www.w3.org/ns/ssn/hasProperty"
    },
    "hasState": {
        "type": "Relationship",
        "object": "http://example.org/data/VeryDryHumidity",
        "context": "http://sensormeasurement.appspot.com/ont/m3/home#hasState"
    },
    "hasTemperatureState": {
        "type": "Relationship",
        "object": "http://example.org/data/BelowRoomTemperature",
        "context": "http://sensormeasurement.appspot.com/ont/m3/home#hasTemperatureState"
    },
    "isFeatureOfInterestOf": {
        "type": "Relationship",
        "object": "http://example.org/data/Apartment134SwitchOffDehydrator",
        "context": "http://www.w3.org/ns/sosa/isFeatureOfInterestOf"
    }
}
```


***Εικόνα 4.5:*** *Στιγμιότυπο οντότητας σπιτιού (Apartment134) όπου φαίνονται σημασιολογικές πληροφορίες σχετικά με τη θερμοκρασία του σπιτιού, την υγρασία του, την κατάσταση μιας συγκεκριμένης ενέργειας που έχει γίνει από συσκευή του σπιτιού, κλπ.*

- Οντότητα μετρήσιμης ιδιότητας (observable property):


```
{
    "id": "Apartment134Humidity",
    "type": "ObservableProperty",
    "context": "http://example.org/data/Apartment134Humidity",
    "isActedOnBy": {
        "type": "Relationship",
        "object": "http://example.org/data/Apartment134SwitchOffDehydrator",
        "context": "http://www.w3.org/ns/sosa/isActedOnBy"
    },
    "isObservedBy": {
        "type": "Relationship",
        "object": "http://example.org/data/Apartment134HumiditySensor",
        "context": "http://www.w3.org/ns/sosa/isObservedBy"
    },
    "isPropertyOf": {
        "type": "Relationship",
        "object": "http://example.org/data/Apartment134",
        "context": "http://www.w3.org/ns/ssn/isPropertyOf"
    }
}
```


***Εικόνα 4.6:*** *Στιγμιότυπο της οντότητας μετρήσιμης ιδιότητας (Apartment134Humidity).*

- Οντότητα μέτρησης (observation):


```
{
    "id": "Room145LuminocityObservation",
    "type": "Observation",
    "context": "http://example.org/data/Room145LuminocityObservation",
    "hasFeatureOfInterest": {
        "type": "Relationship",
        "object": "http://example.org/data/Room145",
        "context": "http://www.w3.org/ns/sosa/hasFeatureOfInterest"
    },
    "hasSimpleResult": 23,
    "madeBySensor": {
        "type": "Relationship",
        "object": "http://example.org/data/Room145LuminocitySensor",
        "context": "http://www.w3.org/ns/sosa/madeBySensor"
    },
    "resultTime": {
        "type": "DateTime",
        "value": "2019-03-07T10:20:00Z"
    },
    "roomLuminocityState": {
        "type": "Relationship",
        "object": "http://example.org/data/LowLighting",
        "context": "http://www.w3.org/ns/sosa/roomLuminocityState"
    },
    "roomState": {
        "type": "Relationship",
        "object": "http://example.org/data/LowLighting",
        "context": "http://www.w3.org/ns/sosa/roomState"
    }
}
```


***Εικόνα 4.7:*** *Στιγμιότυπο οντότητας μέτρησης (Room145LuminocityObservation).*

- Οντότητα (εκτέλεσης) ενέργειας:

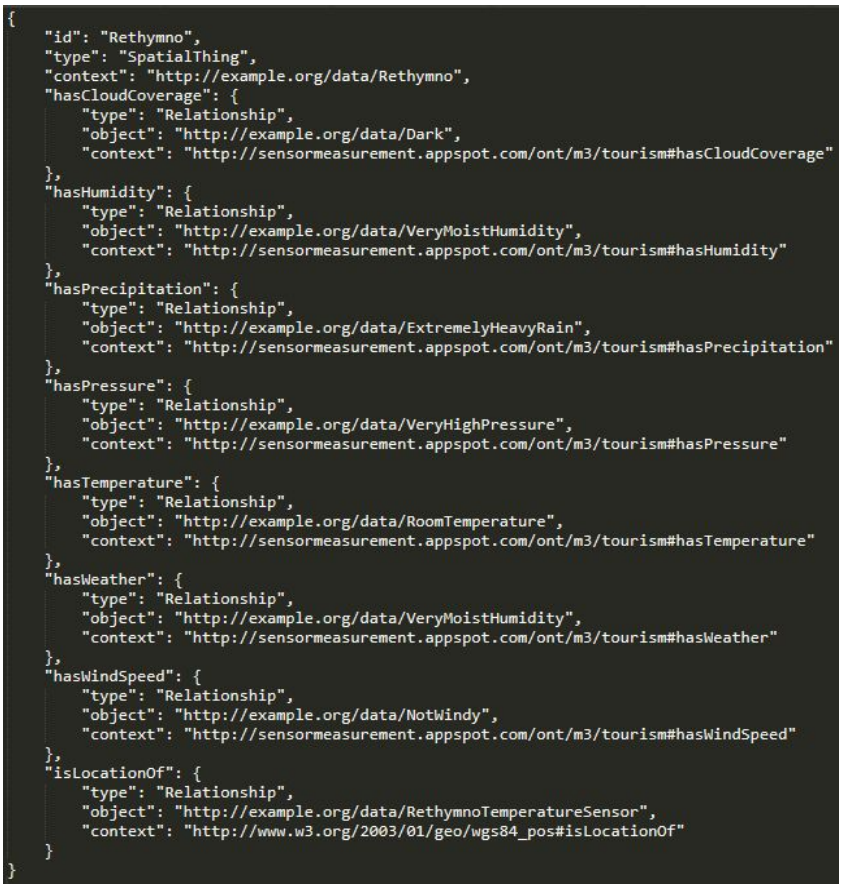
```
{
    "id": "iphoneNotification",
    "type": "Actuation",
    "actsOnProperty": {
        "type": "Relationship",
        "object": "http://example.org/data/iphoneSpeed",
        "context": "http://www.w3.org/ns/sosa/actsOnProperty"
    },
    "actuationEnabled": 0,
    "context": "http://example.org/data/iphoneNotification",
    "madeByActuator": {
        "type": "Relationship",
        "object": "http://example.org/data/iphoneNotificationSender",
        "context": "http://www.w3.org/ns/sosa/madeByActuator"
    }
}
```


***Εικόνα 4.8:*** *Στιγμιότυπο οντότητας εκτέλεσης ενέργειας (iphoneNotification).*

- Οντότητα εφαρμογής (application):

```
{
    "id": "room145live",
    "type": "application",
    "scope": {
        "value": "home",
        "type": "Text"
    },
```


***Εικόνα 4.9:*** *Στιγμιότυπο οντότητας εφαρμογής (room145live) που αφορά τις συνθήκες (όπως η θερμοκρασία και η φωτεινότητα) οι οποίες επικρατούν μέσα σε ένα σπίτι.*



```
{
    "id": "Rethymno",
    "type": "SpatialThing",
    "context": "http://example.org/data/Rethymno",
    "hasCloudCoverage": {
        "type": "Relationship",
        "object": "http://example.org/data/Dark",
        "context": "http://sensormeasurement.appspot.com/ont/m3/tourism#hasCloudCoverage"
    },
    "hasHumidity": {
        "type": "Relationship",
        "object": "http://example.org/data/VeryMoistHumidity",
        "context": "http://sensormeasurement.appspot.com/ont/m3/tourism#hasHumidity"
    },
    "hasPrecipitation": {
        "type": "Relationship",
        "object": "http://example.org/data/ExtremelyHeavyRain",
        "context": "http://sensormeasurement.appspot.com/ont/m3/tourism#hasPrecipitation"
    },
    "hasPressure": {
        "type": "Relationship",
        "object": "http://example.org/data/VeryHighPressure",
        "context": "http://sensormeasurement.appspot.com/ont/m3/tourism#hasPressure"
    },
    "hasTemperature": {
        "type": "Relationship",
        "object": "http://example.org/data/RoomTemperature",
        "context": "http://sensormeasurement.appspot.com/ont/m3/tourism#hasTemperature"
    },
    "hasWeather": {
        "type": "Relationship",
        "object": "http://example.org/data/VeryMoistHumidity",
        "context": "http://sensormeasurement.appspot.com/ont/m3/tourism#hasWeather"
    },
    "hasWindSpeed": {
        "type": "Relationship",
        "object": "http://example.org/data/NotWindy",
        "context": "http://sensormeasurement.appspot.com/ont/m3/tourism#hasWindSpeed"
    },
    "isLocationOf": {
        "type": "Relationship",
        "object": "http://example.org/data/RethymnoTemperatureSensor",
        "context": "http://www.w3.org/2003/01/geo/wgs84_pos#isLocationOf"
    }
}
```


***Εικόνα 4.10:*** *Στιγμιότυπο οντότητας χώρου που αφορά την πόλη του Ρεθύμνου.*

Όταν στο σύστημα εισέρχονται νέες τιμές που προέρχονται από συσκευές (π.χ. νέες μετρήσεις θερμοκρασίας) ή όταν κάποιος χρήστης κάνει - μέσω της Διαδικτυακής Εφαρμογής - ενημέρωση μιας οντότητας που υπάρχει ήδη στο σύστημα (π.χ. ενός μοντέλου συσκευής), οι αντίστοιχες οντότητες της υπηρεσίας Orion Context Broker υφίστανται τις αντίστοιχες αλλαγές, δηλαδή ενημερώνονται. Στην περίπτωση των οντοτήτων τύπου "Observation" που αφορούν τις μετρήσεις, κάθε φορά που εισέρχεται στο σύστημα η πληροφορία για μια νέα μέτρηση (που αφορά ήδη υπάρχουσα οντότητα), αλλάζει η τιμή του χαρακτηριστικού "hasSimpleResult", καθώς και η τιμή του "resultTime" που περιλαμβάνει την πληροφορία για τη χρονική στιγμή και την ημερομηνία που έγινε η μέτρηση. Επομένως, οι τιμές αυτές ενημερώνονται στον Orion Context Broker.

### Orion Context Broker - Δημιουργία συνδρομής σε οντότητες

Ο Orion Context Broker έχει - μεταξύ άλλων - μια πολύ σημαντική λειτουργία: μπορεί να δημιουργεί συνδρομή σε συγκεκριμένες (ή συγκεκριμένου τύπου) οντότητες, γεγονός που του δίνει τη δυνατότητα να πυροδοτεί ενημερώσεις για οποιαδήποτε αλλαγή συμβεί στα χαρακτηριστικά (attributes) κάποιας οντότητας. Η εκάστοτε ενημέρωση θα αποσταλεί από την υπηρεσία Orion 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-2.

Η δυνατότητα αυτή αξιοποιείται τόσο από την Υπηρεσία Διαχείρισης Δεδομένων (αποθήκευση και ανάκτηση 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 Context Broker).

Για να επιτευχθεί η καθεμία από τις παραπάνω συνδρομές, απαιτείται η αποστολή ενός αιτήματος στην υπηρεσία Orion 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 παρακάτω εικόνα:

```
{
    "description": "Notify Cygnus of all observation changes",
    "subject": {
        "entities": [
            {
                "idPattern": ".*",
                "type": "Observation"
            }
        ],
        "condition": {
            "attrs": [ "hasSimpleResult" ]
        }
    },
    "notification": {
        "http": {
            "url": "http://localhost:5050/notify"
        },
        "attrs": [ "hasSimpleResult" ],
        "attrsFormat": "legacy"
    }
}
```

***Εικόνα 4.11:*** *Στιγμιότυπο της συνδρομής της Υπηρεσίας Αποθήκευσης των Ιστορικών Δεδομένων προς τις οντότητες των μετρήσεων των αισθητήρων (σε μορφή JSON).*

Το χαρακτηριστικό "*description*" της παραπάνω συνδρομής συνιστά απλά μια σύντομη περιγραφή της συνδρομής. Το "subject" περιλαμβάνει τα χαρακτηριστικά "*entities*", δηλαδή τις οντότητες στις οποίες δημιουργείται η συνδρομή (στην περίπτωση αυτή επιλέχθηκε τύπος οντότητας), και "*condition*", που υποδηλώνει τη συνθήκη αλλαγής της οντότητας. Συγκεκριμένα, το "entities" υποδεικνύει μέσω του attribute "type" ότι μας ενδιαφέρει ο τύπος Observation και η τιμή "**.***" του "idPattern" υποδηλώνει ότι μας ενδιαφέρουν **όλα** τα διαφορετικά στιγμιότυπα μετρήσεων του συστήματος. Το "condition" εμπεριέχει ένα ακόμα χαρακτηριστικό, το "attributes" ("*attrs*"), το οποίο υποδεικνύει 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, η περιγραφή της οποίας περιέχει δεδομένα και μεταδεδομένα (σημασιολογικές πληροφορίες), απ' τα οποία μας ενδιαφέρει - για τις ανάγκες της ιστορικής βάσης - μόνο η αλλαγή στην τιμή της εκάστοτε μέτρησης. Δηλαδή, το attribute "hasSimpleResult" που κρατάει την τιμή της μέτρησης. Ακόμα, μέσω του attribute "notification" υποδηλώνεται το URI (http://localhost:5050/notify[24]) στο οποίο θα

[24] Πρόκειται για την ίδια εικονική μηχανή, στην οποία έχουν εγκατασταθεί και ο Orion και ο Cygnus, γι' αυτό το λόγο χρησιμοποιούμε το "localhost" στη θέση της διεύθυνσης IP.

γίνεται η αποστολή των ενημερώσεων - πρόκειται για το τελικό σημείο μέσω του οποίου θα λαμβάνει τις ενημερώσεις η υπηρεσία Διαχείρισης Δεδομένων και ειδικότερα η υπηρεσία Cygnus. Η συνδρομή θα μπορούσε ενδεχομένως να περιέχει και άλλα χαρακτηριστικά, όπως το "duration" (υποδηλώνει το χρονικό διάστημα για το οποίο ισχύει η συνδρομή).

Το αίτημα για να δημιουργηθεί η αντίστοιχη συνδρομή της Υπηρεσίας Οντολογίας είναι παρόμοιο με το προηγούμενο (εκείνο στην Εικόνα 4.11), αλλά έχει διαφορετικό "description", στα attributes του "condition" προστίθεται το "resultTime" (καθώς μας ενδιαφέρει η αλλαγή στη χρονική στιγμή της μέτρησης), ενώ αλλάζει και το URI στο οποίο αποστέλλονται οι ενημερώσεις. Συγκεκριμένα, αυτό αντικαθίσταται από ένα URI[25] που αποτελεί τελικό σημείο της Υπηρεσίας Οντολογίας και, μόλις η πληροφορία φτάσει σε αυτό, γίνεται η ενημέρωση των τιμών στην οντολογία. Ακόμα, στα attributes του "notification" προστίθεται το "id" (το μοναδικό αναγνωριστικό της κάθε οντότητας), γεγονός που διευκολύνει την υλοποίηση της Υπηρεσίας Οντολογίας.

```
{
    "description": "Observation sub",
    "subject": {
        "entities": [
            {
                "idPattern": ".*",
                "type": "Observation"
            }
        ],
        "condition": {
            "attrs": [
                "hasSimpleResult",
                "resultTime"
            ]
        }
    },
    "notification": {
        "attrs": [
            "id",
            "hasSimpleResult",
            "resultTime"
        ],
        "http": {
            "url": "http://147.27.60.182:8080/jena-examples-0.0.1-SNAPSHOT/update"
        }
    }
}
```

***Εικόνα 4.12:*** *Στιγμιότυπο της συνδρομής της Υπηρεσίας Οντολογίας προς τις οντότητες των μετρήσεων των αισθητήρων (σε μορφή JSON).*

[25] http://147.27.60.182:8080/jena-examples-0.0.1-SNAPSHOT/update

### Orion Context Broker - Διαγραφή Οντοτήτων

Η διαγραφή οντοτήτων (π.χ. η διαγραφή συνδρομών που προβλέπεται ως λειτουργία από την Υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων) επιτυγχάνεται μέσω ενός REST αιτήματος που χρησιμοποιεί τη μέθοδο DELETE του πρωτοκόλλου HTTP.

### Orion Context Broker - Ανάκτηση Οντοτήτων

Η ανάκτηση οντοτήτων πραγματοποιείται μέσω ενός REST αιτήματος που χρησιμοποιεί τη μέθοδο GET του πρωτοκόλλου HTTP (βλ πίνακα 4.2).

Οι REST μέθοδοι που υλοποιούν όλες τις λειτουργίες της υπηρεσίας Orion Context Broker που αναλύονται παραπάνω (αλλά και στη συνέχεια του παρόντος Κεφαλαίου, όπου παρουσιάζονται αναλυτικά οι λειτουργίες του Web Thing Model) παρατίθενται στον παρακάτω πίνακα:

*Πίνακας 4.2: HTTP μέθοδοι της υπηρεσίας Orion Context Broker.*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| POST | v2/entities | Εικόνα 4.1 | Δημιουργία μιας οποιασδήποτε από τις οντότητες που προαναφέρθηκαν, για παράδειγμα της οντότητας “Αισθητήρα” (μοντέλου συσκευής). |
| POST | v2/subscriptions | Εικόνα 4.6 / Εικόνα 4.7 | Δημιουργία συνδρομής προς όλες τις οντότητες μετρήσεων (είτε από την Υπηρεσία Αποθήκευσης των Ιστορικών Δεδομένων είτε από την Υπηρεσία Οντολογίας). |

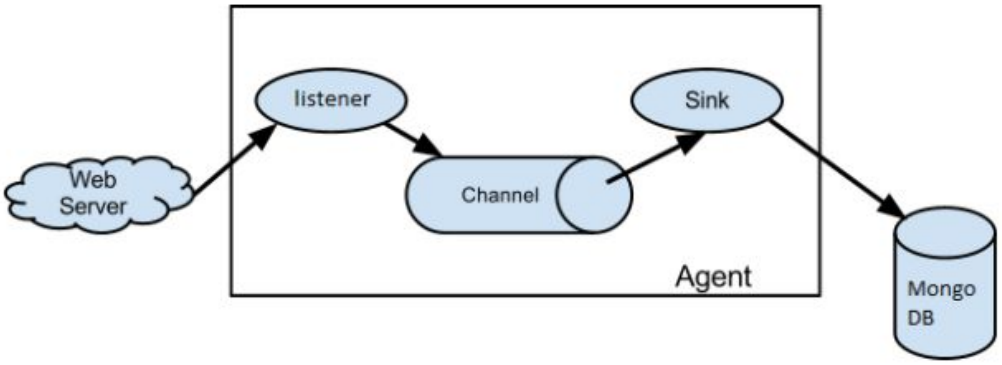
| | | | |
|---|---|---|---|
| GET | v2/entities/{entity_id}?type=Thing | | Ανάκτηση της περιγραφής συγκεκριμένης οντότητας Πράγματος (Web Thing). |
| GET | v2/entities?type=Sensor | | Ανάκτηση όλων των οντοτήτων "Αισθητήρων", δηλαδή όλων των μοντέλων συσκευών. |
| GET | v2/entities/{entity_id}/attrs?type=Sensor | | Ανάκτηση όλων των χαρακτηριστικών ιδιοτήτων (properties) ενός συγκεκριμένου μοντέλου αισθητήρα. |
| GET | v2/entities/{entity_id}/attrs?type=Sensor& attrs=observes | | Ανάκτηση μετρήσιμης ιδιότητας ενός συγκεκριμένου μοντέλου αισθητήρα. |
| GET | v2/entities/{entity_id}?type=Actuation& attrs=actuationEnabled,hasFeatureOfInterest, actuationMadeBy | | Ανάκτηση σημαντικών χαρακτηριστικών μιας συγκεκριμένης εκτέλεσης ενέργειας (actuation). |
| GET | v2/subscriptions/{entity_id} | | Ανάκτηση οντότητας συνδρομής. |
| DELETE | v2/entities/{entity_id}?type=Sensor | | Διαγραφή οντότητας "Αισθητήρα" (χρησιμοποιείται στη λειτουργία ενημέρωσης μοντέλου αισθητήρα). |
| DELETE | v2/entities/{entity_id}?type=Thing | | Διαγραφή οντότητας "Πράγματος" (χρησιμοποιείται στη λειτουργία ενημέρωσης οντότητας Πράγματος). |
| DELETE | v2/subscriptions/{entity_id} | | Διαγραφή συνδρομής. |

*Υπηρεσία Αποθήκευσης των Ιστορικών Δεδομένων*

Η υπηρεσία αυτή (όπως είδαμε και στο Κεφάλαιο 3) απαρτίζεται από την υπηρεσία Cygnus του FIWARE, καθώς και από την υπηρεσία STH Comet του FIWARE. Αμφότερες επικοινωνούν με την ιστορική βάση δεδομένων του συστήματος (MongoDB).

**FIWARE Cygnus**

Η υπηρεσία Cygnus είναι βασισμένη στην αρχιτεκτονική "Apache Flume" και παρέχει διάφορους πράκτορες (agents). Αυτοί είναι υπεύθυνοι για τη συλλογή ροών δεδομένων NGSI-2 και για την αποθήκευσή τους σε κάποια εξωτερική βάση δεδομένων που έχει προκαθοριστεί. Ένας πράκτορας (Εικόνα 4.9) 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).




***Εικόνα 4.13:*** *Apache Flume Agent.*

Το Cygnus παρέχει εξειδικευμένους πράκτορες οι οποίοι έχουν τη δυνατότητα να υποστηρίξουν τη συλλογή και τη διατήρηση NGSI δεδομένων στις εξής υπηρεσίες βάσης δεδομένων - αποθετηρίων:

- **HDFS**, το σύστημα κατανομής αρχείων Hadoop.
- **MySQL**, το γνωστό σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων.
- **CKAN**, μια πλατφόρμα Open data.
- **MongoDB**, η NoSQL βάση δεδομένων για έγγραφα.
- **Kafka**, ο μεσίτης μηνυμάτων δημοσίευσης - εγγραφής.
- **DynamoDB**, μια NoSQL βασισμένη στο υπολογιστικό νέφος βάση δεδομένων της Amazon Web Services,
- **PostgreSQL**, το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων.
- **CarTo,** μια βάση δεδομένων που ειδικεύεται σε γεωγραφικά δεδομένα.

Για παράδειγμα, στο σενάριο κατά το οποίο πρέπει να αποθηκεύσουμε ροές δεδομένων NGSI σε μία βάση δεδομένων MongoDB (όπως συμβαίνει στην παρούσα εργασία), απαιτείται η χρήση ενός εξειδικευμένου "MongoDB πράκτορα" της υπηρεσίας Cygnus. Ομοίως, για το σενάριο αποθήκευσης σε βάση δεδομένων MySQL, θα γίνει χρήση ενός εξειδικευμένου "MySQL πράκτορα".

Στα πλαίσια της παρούσας εργασίας, έχει τεθεί σε λειτουργία ένας εξειδικευμένος πράκτορας του Cygnus, με σκοπό την αποθήκευση τόσο ακατέργαστων (Raw) όσο και συγκεντρωτικών (Aggregated) δεδομένων στην ιστορική βάση MongoDB του συστήματος. Αξιοποιώντας τη λειτουργία συνδρομής που παρέχει η υπηρεσία Orion Context Broker, ο πράκτορας αποτελεί συνδρομητή για όλες τις οντότητες μετρήσεων του Orion. Με αυτόν τον τρόπο, όπως αναλύθηκε και παραπάνω, οποιαδήποτε αλλαγή συμβαίνει στο χαρακτηριστικό (attribute) "hasSimpleResult" σε οντότητα του τύπου Observation - δηλαδή οποιαδήποτε νέα μέτρηση - πυροδοτεί μια ενημέρωση από τον Orion 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 αυτές ως ακατέργαστα και συγκεντρωτικά δεδομένα (σημαντική λειτουργία που *επιτυγχάνεται από τον Cygnus*) στην ιστορική βάση δεδομένων, διατηρώντας έτσι ένα ιστορικό δεδομένων χρονικής σειράς για τις μετρήσεις όλων των αισθητήρων που έχουν συνδεθεί στο σύστημα.

Προκειμένου να αποθηκευτούν οι συγκεντρωτικές πληροφορίες στην ιστορική βάση δεδομένων, διατηρούνται σε αυτήν ξεχωριστές μεταβλητές που αφορούν τα εξής δεδομένα:

1) Την μέγιστη τιμή των μετρήσεων του κάθε αισθητήρα για συγκεκριμένα χρονικά διαστήματα (την τελευταία ώρα / μέρα / μήνα).

2) Την ελάχιστη τιμή των μετρήσεων του κάθε αισθητήρα για συγκεκριμένα χρονικά διαστήματα (την τελευταία ώρα / μέρα / μήνα).
3) Την αθροιστική τιμή των μετρήσεων του κάθε αισθητήρα για συγκεκριμένα χρονικά διαστήματα (την τελευταία ώρα / μέρα / μήνα), η οποία μπορεί στη συνέχεια να αξιοποιηθεί από το Mashup Service για εύρεση του μέσου όρου των τιμών των μετρήσεων.

Στον παρακάτω πίνακα περιγράφεται η REST μέθοδος που παρέχεται από την υπηρεσία Cygnus για τη λήψη και την αποθήκευση ροών δεδομένων NGSI-2:

***Πίνακας 4.1:*** *HTTP μέθοδος της υπηρεσίας Cygnus.*

| Μέθοδος | URL Μεθόδου | Περιγραφή μεθόδου |
|---|---|---|
| POST | /notify | Πρόκειται για το τελικό σημείο (URI) του συνδρομητή πράκτορα της υπηρεσίας Cygnus στο οποίο γίνεται η αποστολή ροών δεδομένων NGSI προκειμένου να αποθηκευτούν. Αυτό φαίνεται και στην εικόνα 4.7 που αναλύσαμε παραπάνω. |

## FIWARE - COMET

Όπως έχει ήδη επισημανθεί, η υπηρεσία COMET παρέχεται επίσης από το FIWARE και συνδέεται τοπικά με την ιστορική βάση δεδομένων MongoDB του συστήματος (History MongoDB). Η υπηρεσία φιλοξενείται μάλιστα στην ίδια εικονική μηχανή με τη βάση. Χρησιμοποιώντας τη RESTful διεπαφή της, η υπηρεσία επιτρέπει την ανάκτηση ακατέργαστων (Raw) και συγκεντρωτικών (Aggregated) δεδομένων που έχουν αποθηκευτεί στην ιστορική βάση δεδομένων. Η αποθήκευση αυτών των δεδομένων, όπως αναφέρεται και παραπάνω, έχει επιτευχθεί από τον πράκτορα του Cygnus. Στον πίνακα που ακολουθεί περιγράφεται η μέθοδος της υπηρεσίας COMET που χρησιμοποιήθηκε κατά την υλοποίηση της παρούσας εργασίας:

***Πίνακας 4.1:*** *HTTP μέθοδος της υπηρεσίας FIWARE - COMET.*

| Μέθοδος | URL Μεθόδου | Περιγραφή μεθόδου |
|---|---|---|
| GET | /STH/v1/contextEntities/ type / Observation / id / {Observation_id} / attributes/ {hasSimpleResult}? aggrMethod={max}&aggrPeriod=}day} | Αίτημα ανάκτησης συγκεντρωτικής ιστορικής πληροφορίας, που αφορά τη μέγιστη μέτρηση με αναγνωριστικό {Observation_id} ανά ημέρα. Σημείωση: Το "aggrMethod" μπορεί να λάβει τις τιμές: "max", "min", "sum". Το "aggrPeriod" μπορεί να λάβει τις τιμές "month", "day", "hour". |

## *Υπηρεσία Οντολογίας*

Η Υπηρεσία Οντολογίας, όπως έχει ήδη αναφερθεί, αποτελείται από το Jena API και την οντολογία που είναι αποθηκευμένη στο ΣΔΒΔ Virtuoso. Προκειμένου να εξυπηρετήσει τα αιτήματα που δέχεται από τις υπόλοιπες υπηρεσίες, βασίζεται στο Jena API, που είναι υπεύθυνο για κάθε είδους επικοινωνία της οντολογίας με τις υπηρεσίες του υπόλοιπου συστήματος - το Jena API αποτελεί μια "γέφυρα" μεταξύ της οντολογίας και των υπόλοιπων υπηρεσιών. Χάρη σε αυτό, μπορεί να γίνει ανάκτηση πληροφοριών από την οντολογία, εισαγωγή νέων πληροφοριών σε αυτήν, ενημέρωση ή ακόμα και διαγραφή τους. Ακόμα, το Jena είναι υπεύθυνο για την πραγματοποίηση της σημασιολογικής αιτίασης / συλλογισμού (reasoning), το οποίο γίνεται κυρίως κατά την προσθήκη νέων δεδομένων, ούτως ώστε να βρεθούν όλες οι πιθανές ασυνέπειες που μπορεί να προκύψουν, καθώς η εξαγωγή νέων πληροφοριών από αυτές που ήδη υπάρχουν στην οντολογία.

Η Υπηρεσία Οντολογίας, και συνεπώς όλες οι λειτουργίες που πραγματοποιούνται μέσω του Jena API, υλοποιήθηκαν με τη χρήση του JavaEE και του Spring Framework. Η εν λόγω υπηρεσία έχει ενσωματώσει έναν Tomcat server (εξυπηρετητή) και για το τελικό σημείο του Tomcat ορίζουμε ένα ειδικό port στην εικονική μηχανή που φιλοξενεί το Jena API (το port 8080).

Οι REST μέθοδοι που υλοποιούν όλες τις λειτουργίες της υπηρεσίας Οντολογίας που αναλύονται παραπάνω παρατίθενται στον παρακάτω πίνακα:

***Πίνακας 4.1:*** *HTTP μέθοδος της Υπηρεσίας Οντολογίας (Jena API).*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| POST | /jena-project-0.0.1-SNAPSHOT/ create_wt_add_to_virt | Η περιγραφή JSON του Πράγματος, η οποία δίνεται από το χρήστη μέσω του Web Application. | Αίτημα εισαγωγής περιγραφής JSON ενός Πράγματος (Web Thing) στην οντολογία. |
| POST | /jena-project-0.0.1-SNAPSHOT/ update_wt | Η περιγραφή JSON του Πράγματος, η οποία δίνεται από το χρήστη μέσω του Web Application. | Αίτημα ενημέρωσης περιγραφής JSON ενός Πράγματος (Web Thing) στην οντολογία. |
| POST | /jena-project-0.0.1-SNAPSHOT/ create_model_add_to_virt | Η περιγραφή JSON-LD του μοντέλου του Πράγματος, η οποία δίνεται από το χρήστη μέσω του Web Application. | Αίτημα εισαγωγής περιγραφής JSON-LD του μοντέλου ενός Πράγματος (Web Thing Model) στην οντολογία. |
| POST | /jena-project-0.0.1-SNAPSHOT/ update_model | Η περιγραφή JSON-LD του μοντέλου του Πράγματος, η οποία δίνεται από το χρήστη μέσω του Web Application. | Αίτημα ενημέρωσης περιγραφής JSON-LD του μοντέλου ενός Πράγματος (Web Thing Model) στην οντολογία. |
| POST | /jena-project-0.0.1-SNAPSHOT/ update | Οι τιμές που αλλάζουν και στέλνονται από τον Orion (βλ. Εικόνα 4.8). | Αίτημα προσθήκης των νέων τιμών των μετρήσεων αισθητήρων στην οντολογία (μετά την ενημέρωση που πυροδοτείται από τον Orion Context Broker). |
| POST | /jena-project-0.0.1-SNAPSHOT/ execute_action | Η περιγραφή JSON της ενέργειας που πρόκειται να εκτελεστεί. | Αίτημα εκτέλεσης ενέργειας (σε μια συσκευή). Η πληροφορία της ενέργειας που θα εκτελεστεί αποθηκεύεται στην οντολογία. |

| | | | |
|---|---|---|---|
| GET | /jena-project-0.0.1-SNAPSHOT/ properties | | Αίτημα εύρεσης όλων των μετρήσιμων ιδιοτήτων των μοντέλων αισθητήρων που έχουν αποθηκευτεί στην οντολογία. |
| GET | /jena-project-0.0.1-SNAPSHOT/ action | | Αίτημα εύρεσης όλων των δυνατών ενεργειών που πραγματοποιούνται από αισθητήρες. |
| GET | /jena-project-0.0.1-SNAPSHOT/ actuators | | Αίτημα εύρεσης όλων των ενεργοποιητών (actuators) που έχουν αποθηκευτεί στην οντολογία. |

***Υπηρεσία Διακομιστή μεσολάβησης του Ιστού των Πραγμάτων (WoT Proxy)***

### Υπηρεσία Διαχείρισης Συσκευών Backend IDAS

Η υπηρεσία IDAS, όπως αναφέρθηκε και στο Κεφάλαιο 3, πραγματοποιεί τη διασύνδεση των αισθητήρων - συσκευών με το Υπολογιστικό Νέφος, χρησιμοποιώντας διαφορετικούς IoT Agents οι οποίοι μεταφράζουν τα αντίστοιχα πρωτόκολλα των συσκευών σε NGSI2 και τα στέλνουν στην υπηρεσία Context Broker (ή, αντίστροφα, μεταφράζουν και στέλνουν δεδομένα στις συσκευές). Κατά την υλοποίηση της παρούσας εργασίας, για την υπηρεσία IDAS χρησιμοποιήθηκαν εξειδικευμένοι IoT Agents, όπως αυτός που προορίζεται για το πρωτόκολλο επικοινωνίας MQTT.

Οι REST μέθοδοι που υλοποιούν τις λειτουργίες της υπηρεσίας IDAS παρατίθενται στον παρακάτω πίνακα:

*Πίνακας 4.2: HTTP μέθοδοι της υπηρεσίας IDAS.*

| Μέθοδος | URL Μεθόδου | Σώμα αιτήματος | Περιγραφή μεθόδου |
|---|---|---|---|
| POST | iot/devices | Μια περιγραφή JSON με τα απαιτούμενα χαρακτηριστικά της συσκευής (την εισάγει ο χρήστης). | Εγγραφή συσκευής στη λίστα συσκευών του IDAS. |
| GET | iot/devices/{device_id} | | Ανάκτηση περιγραφής συσκευής από τη λίστα συσκευών του IDAS (για να ελεγχθεί αν υπάρχει στη λίστα και να διαβαστούν τα χαρακτηριστικά της). |

### Υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service)

Η υλοποίηση της πλατφόρμας, όπως προαναφέρθηκε, βασίστηκε στο μοντέλο *Web Thing Model της W3C* και στις λειτουργίες - υπηρεσίες που αυτό προτείνει. Οι υπηρεσίες αυτές πραγματοποιούνται από την εν λόγω υπηρεσία (Web Thing Model Service).

Οι REST μέθοδοι που δρομολογούνται από την Υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (προκειμένου να πραγματοποιηθούν οι λειτουργίες του Web Thing Model) περιγράφονται στις ενότητες των υπόλοιπων υπηρεσιών.

Παρακάτω παρουσιάζεται αναλυτικά ο τρόπος με τον οποίο πραγματοποιούνται οι λειτουργίες αυτές στα πλαίσια της πλατφόρμας που αναπτύχθηκε. Συγκεκριμένα, έχουμε:

- ***Retrieve a Web Thing (Ανάκτηση ενός Πράγματος του Ιστού):***

Όταν ένας χρήστης επιλέγει την ανάκτηση ενός Πράγματος, του επιστρέφεται μια σύντομη περιγραφή της μορφής JSON, η οποία περιέχει ζευγάρια ιδιοτήτων - τιμών (key - value pairs) που χρησιμοποιούνται για το

χαρακτηρισμό του Πράγματος. Η περιγραφή αυτή έχει εισαχθεί στο σύστημα από κάποιον χρήστη με τη λειτουργία της προσθήκης Πράγματος (βλ. "Add a Web Thing"). Ένα παράδειγμα περιγραφής JSON φαίνεται στην παρακάτω εικόνα:

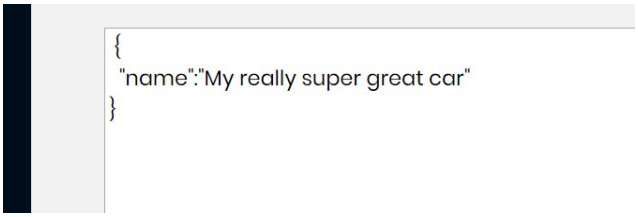
```
{
  "id": "MyLamp",
  "type": "Thing",
  "context": "https://mywebthingserver.com/things/lamp",
  "description": "A web connected lamp"
}
```

***Εικόνα 4.14:*** *Περιγραφή JSON για την αναπαράσταση ενός Πράγματος*

Όπως παρατηρούμε στην εικόνα 2.1, στην περιγραφή JSON φαίνεται το "identifier" (id), δηλαδή ένα αναγνωριστικό του Πράγματος, που βρίσκεται υποχρεωτικά σε κάθε περιγραφή οντότητας, όπως και ο τύπος της οντότητας που επίσης πρέπει να υπάρχει σε κάθε οντότητα (στην περίπτωση αυτή έχει την τιμή "Thing" επειδή πρόκειται για Πράγμα). Επίσης, στην περιγραφή JSON, φαίνεται η τιμή του "context", που είναι ένας υπερσύνδεσμος σε κάποια τοποθεσία του Διαδικτύου, από την οποία μπορούμε να αντλήσουμε πληροφορίες σχετικές με το Πράγμα. Τέλος, στην περιγραφή υπάρχει και το "description", που αποτελεί μια σύντομη περιγραφή του Πράγματος.

- ***Update a Web Thing (Ενημέρωση ενός Πράγματος του Ιστού):***

Όταν ένας χρήστης επιλέγει την ενημέρωση ενός Πράγματος (Web Thing), του δίνεται η δυνατότητα να αλλάξει μία ή περισσότερες από τις παραπάνω τιμές των χαρακτηριστικών της περιγραφής του Πράγματος, δίνοντας νέες τιμές. Έτσι, η εικονική αναπαράσταση του Πράγματος ενημερώνεται και αποθηκεύεται εκ νέου στο σύστημα (στη βάση δεδομένων MongoDB αλλά και στην οντολογία του συστήματος).




***Εικόνα 4.15:*** *Το χαρακτηριστικό ("name") που ο χρήστης επιθυμεί να ενημερώσει και η νέα τιμή του.*

- ***Retrieve the model of a Thing (Ανάκτηση του μοντέλου ενός Πράγματος):***

Όταν ένας χρήστης επιλέγει την ανάκτηση του μοντέλου ενός Πράγματος, του επιστρέφεται μια περιγραφή JSON-LD, η οποία περιέχει ζευγάρια κλειδιών-τιμών όπως η περιγραφή JSON, καθώς και υπερσυνδέσμους που χρησιμοποιούνται για τη σημασιολογική περιγραφή του Πράγματος. Το JSON-LD αποτελεί μια μορφή διασυνδεδεμένων δεδομένων, τα οποία μπορούν να "μεταφραστούν" σε δεδομένα RDF, δηλαδή τριπλέτες της μορφής "υποκείμενο - κατηγόρημα - αντικείμενο" (subject - predicate - object). Έτσι, ο χρήστης ανακτά μια αναλυτική σημασιολογική αναπαράσταση των χαρακτηριστικών του Πράγματος. Ένα παράδειγμα μοντέλου ενός Πράγματος με όνομα "DHT224580" (που αποτελεί την εικονική οντότητα ενός αισθητήρα θερμοκρασίας) φαίνεται στην παρακάτω εικόνα:


```
{
  "id": "DHT224580",
  "type": "Sensor",
  "context": "http://example.org/data/DHT224580",
  "forProperty": {
    "type": "Relationship",
    "object": "http://example.org/data/Room245Temperature",
    "context": "http://www.w3.org/ns/ssn/forProperty"
  },
  "isHostedBy": {
    "type": "Relationship",
    "object": "http://example.org/data/PCBBoard3",
    "context": "http://www.w3.org/ns/sosa/isHostedBy"
  },
  "madeObservation": {
    "type": "Relationship",
    "object": "http://example.org/data/Room245TemperatureObservation",
    "context": "http://www.w3.org/ns/sosa/madeObservation"
  },
  "observes": {
    "type": "Relationship",
    "object": "http://example.org/data/Room245Temperature",
    "context": "http://www.w3.org/ns/sosa/observes"
  }
}
```


***Εικόνα 4.16:*** *Στιγμιότυπο περιγραφής JSON-LD για την αναπαράσταση του μοντέλου ενός Πράγματος.*

Όπως παρατηρούμε στην εικόνα 2.3, η περιγραφή JSON-LD του μοντέλου, ξεκινάει (όπως και στην περιγραφή JSON) με το "identifier" του Πράγματος, ενώ

περιέχει και τον τύπο της οντότητας. Αυτός έχει την τιμή "Sensor" επειδή πρόκειται για Μοντέλο αισθητήρα/συσκευής. Όμοια με την περιγραφή JSON, φαίνεται η τιμή του "context", η οποία εξηγήθηκε και παραπάνω.

Στη συνέχεια, παρατηρούμε αντικείμενα JSON (JSON objects)[26], δηλαδή χαρακτηριστικά των οποίων η τιμή περικλείεται από άγκιστρα και αποτελείται από άλλα χαρακτηριστικά (εμφωλευμένα key-value pairs). Αυτά μεταφράζονται στη συνέχεια - με ένα μηχανισμό που θα εξηγηθεί αργότερα - σε τριπλέτες δεδομένων, χάρη στο χαρακτηριστικό "context" του JSON αντικειμένου, που χρησιμοποιείται ως κατηγόρημα (predicate) για τη δημιουργία των τριπλετών. Για παράδειγμα, στο παραπάνω μοντέλο, η τιμή του χαρακτηριστικού "observes" βρίσκεται μέσα σε άγκιστρα και αποτελείται από τρία ζεύγη χαρακτηριστικών - τιμών (key-value pairs). Το χαρακτηριστικό "type" υποδηλώνει τον τύπο του χαρακτηριστικού "observes", ενώ το "context", ως κατηγόρημα, συνδέει το χαρακτηριστικό "object", δηλαδή τον υπερσύνδεσμο:

➔ *"http://example.org/data/Room245Temperature"*

με το υποκείμενο (subject), το οποίο για κάθε τριπλέτα είναι το "context" της JSON-LD περιγραφής που δηλώνεται ακριβώς κάτω από τον τύπο της οντότητας. Επομένως, από το attribute "observes", θα δημιουργηθεί η εξής τριπλέτα:

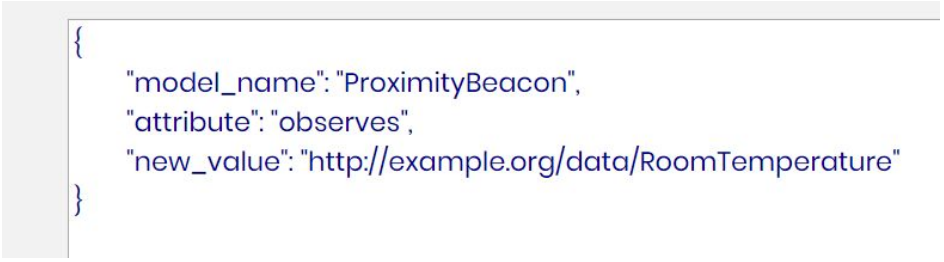
- ❏ υποκείμενο (subject): http://example.org/data/DHT224580
- ❏ κατηγόρημα (predicate): http://www.w3.org/ns/sosa/observes
- ❏ αντικείμενο (object): http://example.org/data/Room245Temperature

Από τα παραπάνω συμπεραίνουμε ότι ο χρήστης μαθαίνει μια πληροφορία που αφορά το υποκείμενο (αισθητήρας τύπου DHT224580). Το κατηγόρημα δείχνει μια ιδιότητα (Object Property) της οντολογίας SOSA, η οποία γενικά φανερώνει τη σχέση μεταξύ ενός αισθητήρα και κάποιας αισθητής ιδιότητας (Observable Property), δηλαδή κάποιας ιδιότητας που μπορεί να μετρηθεί από αισθητήρα. Επομένως, η παραπάνω τριπλέτα υποδηλώνει ότι ο αισθητήρας αυτού του τύπου χρησιμοποιείται για τη μέτρηση θερμοκρασίας και συγκεκριμένα για τη θερμοκρασία ενός δωματίου με αναγνωριστικό "Room245". Η πληροφορία που προσφέρει ο υπερσύνδεσμος του αντικειμένου αντλείται από την αντίστοιχη τοποθεσία - οντολογία στο Διαδίκτυο, η οποία στην περίπτωση αυτή είναι υποθετική (ο υπερσύνδεσμος δημιουργήθηκε για τις ανάγκες της υλοποίησης).

- ***Update the model of a Thing (Ενημέρωση του μοντέλου ενός Πράγματος):***

[26] https://www.w3schools.com/js/js_json_objects.asp

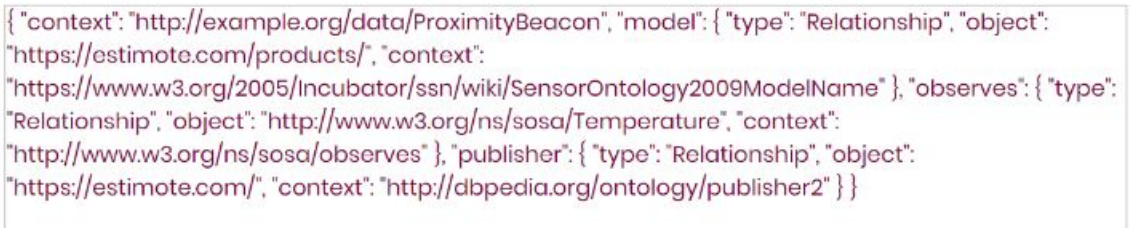
Όταν ένας χρήστης επιλέγει την ενημέρωση ενός μοντέλου, του δίνεται η δυνατότητα να ενημερώσει έναν ή περισσότερους υπερσυνδέσμους (ή οποιαδήποτε άλλη τιμή) της αναπαράστασης JSON-LD του Πράγματος, δίνοντας νέες τιμές σε αυτούς. Έτσι, η περιγραφή του μοντέλου του Πράγματος ενημερώνεται και αποθηκεύεται εκ νέου στο σύστημα (στη βάση δεδομένων MongoDB αλλά και στην οντολογία).




***Εικόνα 4.17:*** *Στιγμιότυπο περιγραφής JSON που περιέχει το όνομα της οντότητας στην οποία πρόκειται να γίνει ενημέρωση, το όνομα του χαρακτηριστικού που θα ενημερωθεί και τη νέα τιμή του χαρακτηριστικού.*

- ***Retrieve a list of properties (Ανάκτηση μιας λίστας ιδιοτήτων ενός Πράγματος):***

Όταν ένας χρήστης επιλέγει την ανάκτηση των ιδιοτήτων ενός Πράγματος, του επιστρέφεται μια περιγραφή JSON-LD που περιέχει όλους τους υπερσυνδέσμους που χαρακτηρίζουν το Πράγμα, οι οποίοι προέρχονται από το προαναφερθέν μοντέλο του Πράγματος. Έτσι, ο χρήστης έχει τη δυνατότητα να μάθει όλες τις ιδιότητες (properties) που αφορούν τη συγκεκριμένη οντότητα. Ένα παράδειγμα JSON-LD με τις ιδιότητες αυτές φαίνεται παρακάτω:




***Εικόνα 4.18:*** *Στιγμιότυπο περιγραφής JSON-LD με τους υπερσυνδέσμους που φανερώνουν όλες τις ιδιότητες οι οποίες χαρακτηρίζουν ένα συγκεκριμένο Πράγμα.*

- ***Retrieve the value of a property (Ανάκτηση της τιμής μιας ιδιότητας ενός Πράγματος):***

Όταν ένας χρήστης επιλέγει την ανάκτηση της τιμής μιας συγκεκριμένης ιδιότητας του Πράγματος, του επιστρέφεται μια σύντομη περιγραφή JSON-LD που πληροφορεί το χρήστη σχετικά με την τιμή που έχει αυτή η ιδιότητα. Για παράδειγμα, η ιδιότητα *"observes"* δίνει στο χρήστη την πληροφορία για το τι μετράει ο αντίστοιχος αισθητήρας, χρησιμοποιώντας έναν υπερσύνδεσμο που παραπέμπει σε μια σχετική οντολογία στο Διαδίκτυο (οντολογία SOSA). Ένα τέτοιο παράδειγμα, πάλι για το Πράγμα με όνομα "DHT224580", φαίνεται στην παρακάτω εικόνα:

```
{
  "id": "DHT224580",
  "type": "Sensor",
  "observes": {
    "type": "Relationship",
    "object": "http://example.org/data/Room245Temperature",
    "context": "http://www.w3.org/ns/sosa/observes"
  }
}
```

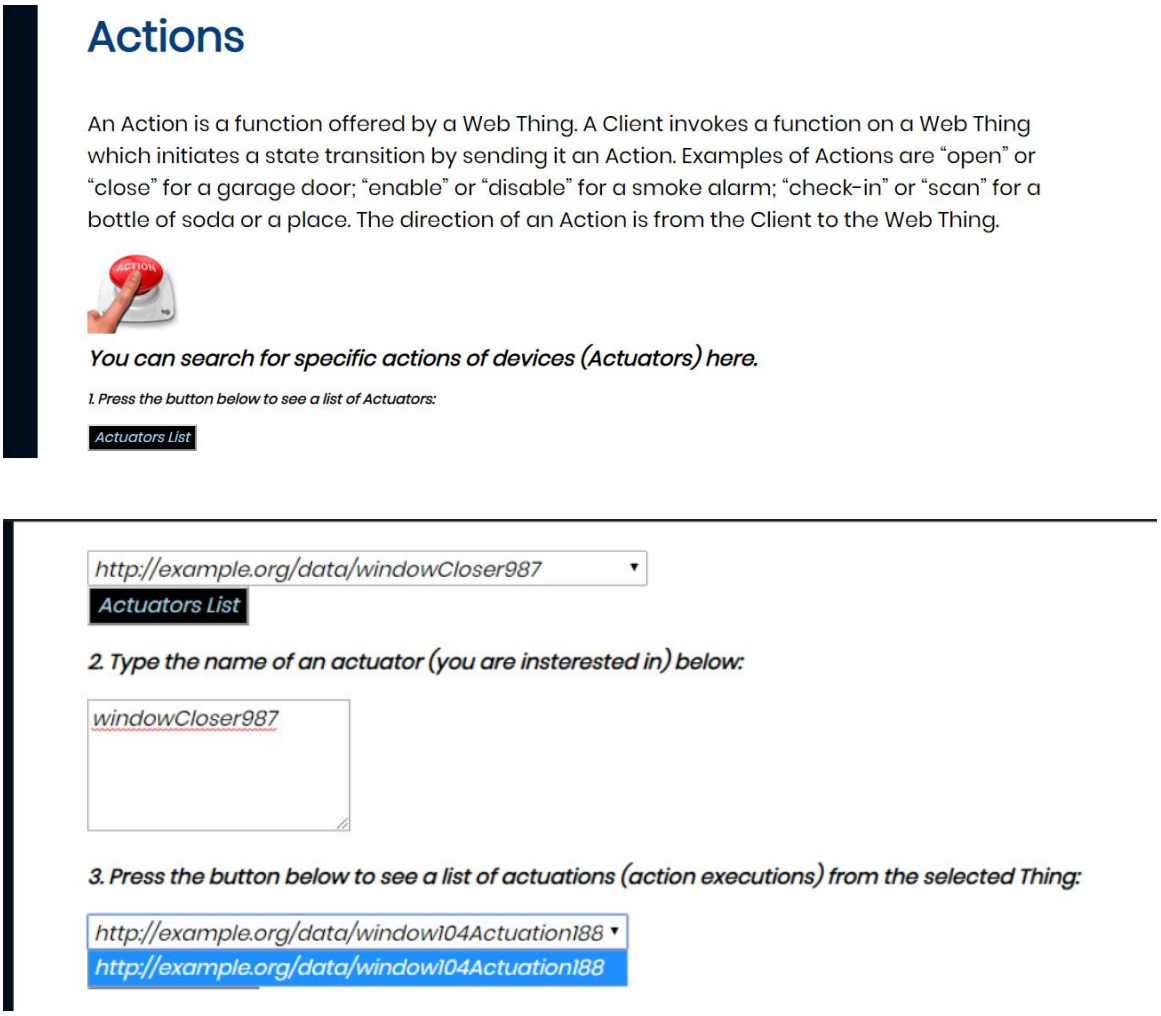
***Εικόνα 4.19:*** *Στιγμιότυπο περιγραφής JSON-LD με τον υπερσύνδεσμο που φανερώνει την τιμή της ιδιότητας "observes".*

- ***Update a specific property (Ενημέρωση μιας συγκεκριμένης ιδιότητας ενός Πράγματος):***

Η λειτουργία αυτή επιτρέπει στο χρήστη να ενημερώσει την τιμή μιας συγκεκριμένης ιδιότητας του Πράγματος, επομένως στην ουσία ταυτίζεται με τη λειτουργία *Update the model of a Thing* (βλ. παραπάνω) και δεν υλοποιήθηκε ως ξεχωριστή λειτουργία.

- ***Retrieve a list of actions (Ανάκτηση όλων των ενεργειών ενός Πράγματος):***

Όταν ένας χρήστης επιλέγει την ανάκτηση όλων των ενεργειών ενός Πράγματος, του επιστρέφεται μια λίστα με όλους τους υπερσυνδέσμους που υποδηλώνουν είτε τις ενέργειες (actions) που μπορεί να κάνει το Πράγμα είτε τις εκτελέσεις συγκεκριμένων ενεργειών (actuations) που ενδέχεται να έχει κάνει. Αυτοί οι υπερσύνδεσμοι προέρχονται επίσης από το μοντέλο του Πράγματος, το οποίο έχει τη μορφή JSON-LD. Ένα τέτοιο παράδειγμα φαίνεται στην παρακάτω εικόνα:




***Εικόνα 4.20:*** *Στιγμιότυπο της λίστας που φανερώνει όλες τις εκτελέσεις ενεργειών (στην περίπτωση αυτή ήταν μόνο μία) που έχει κάνει η οντότητα windowCloser987.*

- ***Retrieve recent executions of a specific action (Ανάκτηση όλων των πρόσφατων εκτελέσεων μιας συγκεκριμένης ενέργειας ενός Πράγματος):***

Όταν ένας χρήστης επιλέγει την ανάκτηση όλων των πρόσφατων εκτελέσεων μιας συγκεκριμένης ενέργειας ενός Πράγματος, του επιστρέφεται μια σύντομη JSON-LD περιγραφή, όπου φαίνεται η τιμή της πιο πρόσφατης εκτέλεσης της ενέργειας (π.χ. 0 ή 1 για κάποια ενέργεια του τύπου on/off). Ένα τέτοιο παράδειγμα φαίνεται στην παρακάτω εικόνα:

```
{
  "id": "window104Actuation188",
  "type": "Actuation",
  "actuationEnabled": 0,
  "actuationMadeBy": {
    "type": "Relationship",
    "object": "http://example.org/data/windowCloser987",
    "context": "http://www.w3.org/ns/sosa/actuationMadeBy"
  }
}
```


***Εικόνα 4.21:*** *Στιγμιότυπο περιγραφής JSON-LD όπου φαίνεται η τιμή της εκτέλεσης της ενέργειας με id "window104Actuation888" μαζί με την πληροφορία για το ποια συσκευή έκανε την ενέργεια.*

- ***Execute an action (Εκτέλεση μια ενέργειας):***

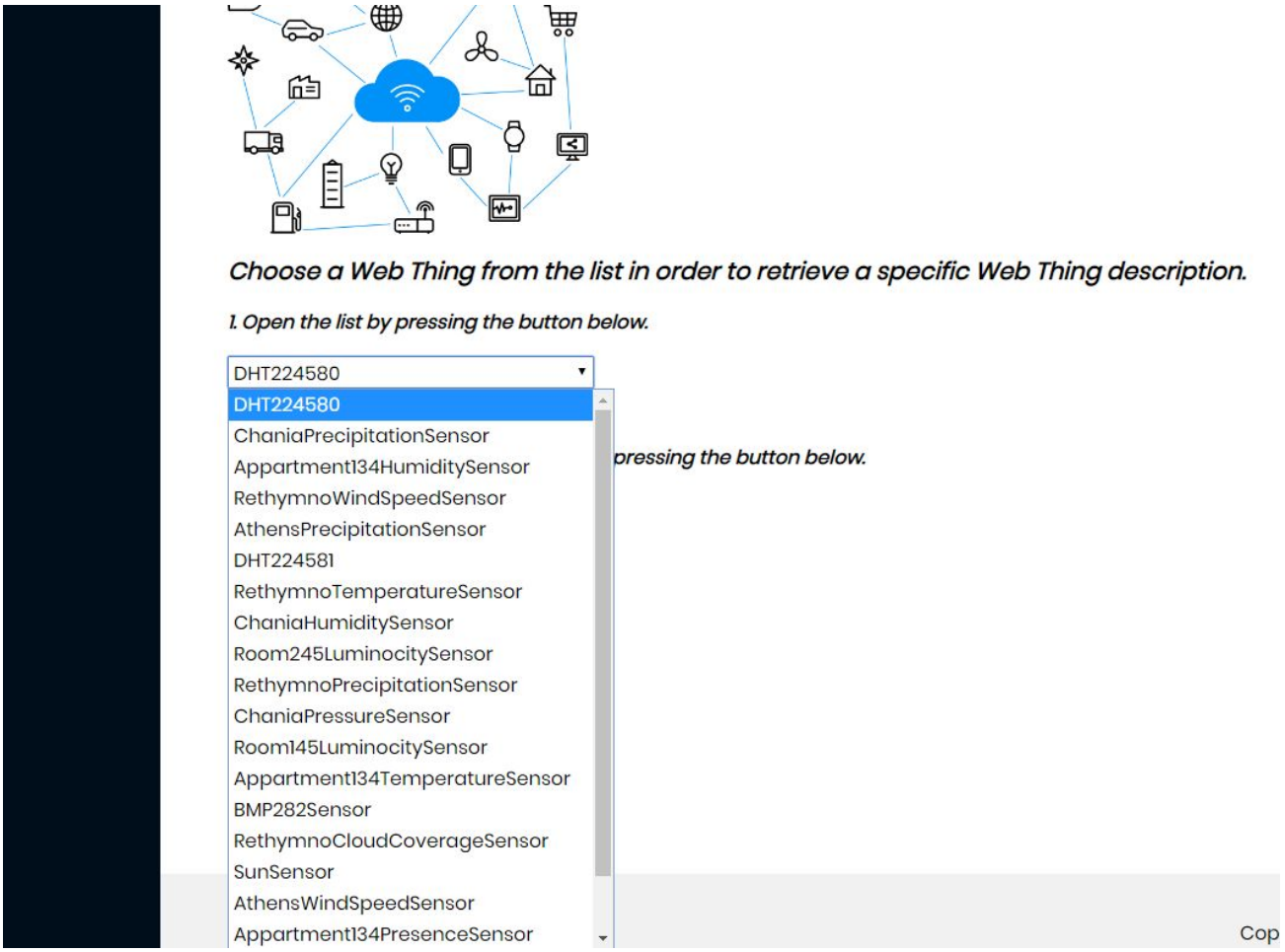
Όταν ένας χρήστης επιλέγει την εκτέλεση μιας ενέργειας, έχει τη δυνατότητα να εισαγάγει τις απαιτούμενες για την ενέργεια πληροφορίες, ούτως ώστε αυτές να αποθηκευτούν στο σύστημα (στη βάση MongoDB του Orion Context Broker καθώς και στην οντολογία) και να δρομολογηθεί η επιθυμητή ενέργεια. Με αυτόν τον τρόπο, ένας χρήστης μπορεί να στείλει μια εντολή (command) σε μια συσκευή. Ωστόσο, το σύστημα που υλοποιήθηκε υποστηρίζει μόνο τη δρομολόγηση μιας ενέργειας (για συσκευές που μπορούν να εκτελέσουν μια ενέργεια), καθώς δεν έχει γίνει προσομοίωση με χρήση πραγματικών συσκευών στα πλαίσια της παρούσας εργασίας.

- ***Retrieve the status of an action (Ανάκτηση της κατάστασης μιας συγκεκριμένης ενέργειας ενός Πράγματος):***

Η λειτουργία αυτή επιτρέπει στο χρήστη να ανακτήσει την τιμή μιας συγκεκριμένης ενέργειας του Πράγματος, επομένως στην ουσία ταυτίζεται με τη λειτουργία *Retrieve recent executions of a specific action* (βλ. παραπάνω) και δεν υλοποιήθηκε ως ξεχωριστή λειτουργία.

- ***Retrieve a list of Web Things (Ανάκτηση μιας λίστας όλων των Πραγμάτων):***

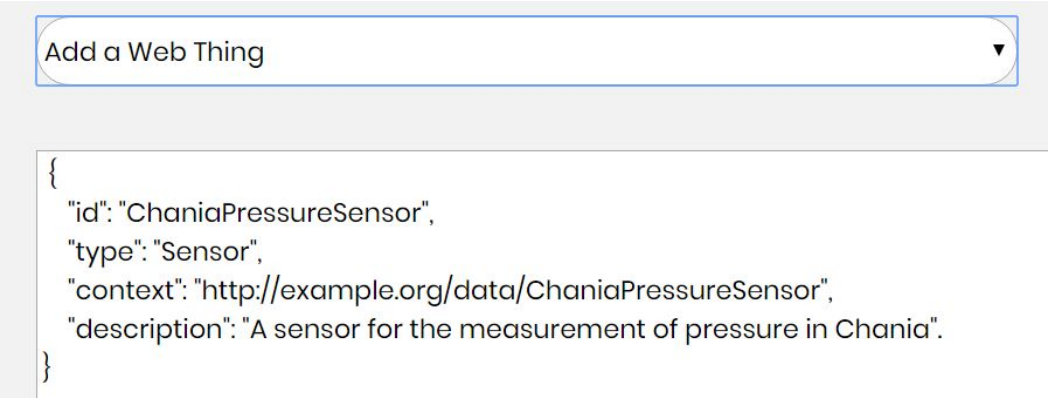
Η λειτουργία αυτή επιτρέπει στο χρήστη να ανακτήσει μια λίστα με όλα τα Πράγματα που βρίσκονται αποθηκευμένα στο σύστημα.




***Εικόνα 4.22:*** *Στιγμιότυπο με τη λίστα όλων των Πραγμάτων που βρίσκονται αποθηκευμένα στο σύστημα.*

- ***Add a Web Thing (Εισαγωγή ενός Πράγματος):***

Η λειτουργία αυτή επιτρέπει στο χρήστη να προσθέσει ένα νέο Πράγμα στο σύστημα, δηλαδή μια νέα εικονική οντότητα (ενός αισθητήρα ή μιας άλλης συσκευής). Έτσι, ο χρήστης εισάγει ένα νέο τύπο συσκευής στο σύστημα. Για να συμβεί αυτό, πρέπει να προσθέσει μια JSON αναπαράσταση του Πράγματος. Στη συνέχεια, μέσω της λειτουργίας *Add a Web Thing Model*, ο χρήστης έχει τη δυνατότητα να προσθέτει μια JSON-LD περιγραφή του μοντέλου της συσκευής.



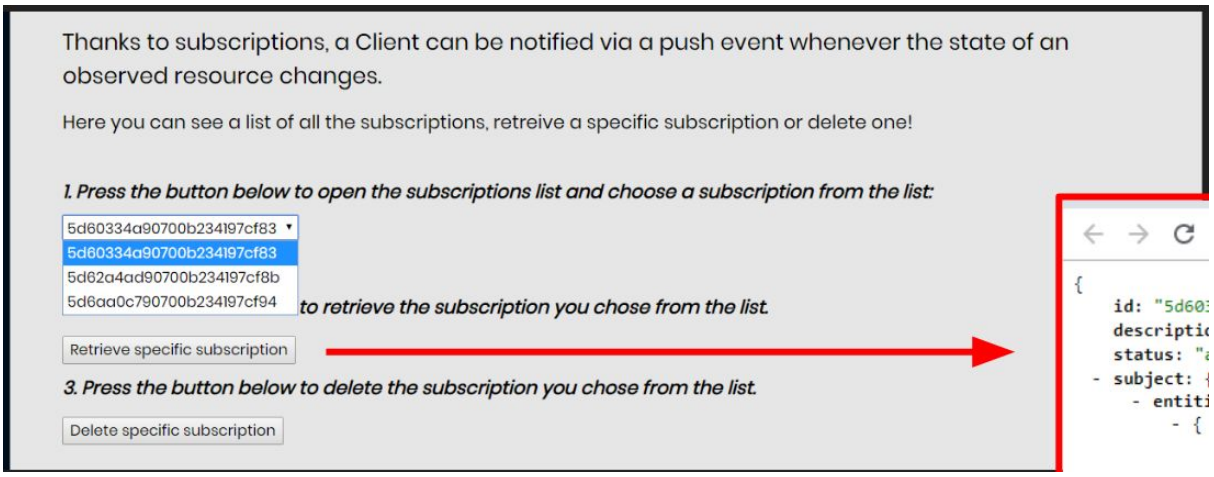

***Εικόνα 4.23**: Στιγμιότυπο της JSON περιγραφής ενός νέου Πράγματος, το οποίο ο χρήστης επιχειρεί να προσθέσει στο σύστημα (μέσω του Web Application).*

- ***Create a subscription (Δημιουργία μιας νέας συνδρομής):***

Η λειτουργία αυτή επιτρέπει στο χρήστη να δημιουργήσει μία νέα συνδρομή σε κάποιον πόρο του συστήματος, π.χ. σε μια συγκεκριμένη συσκευή. Παραδείγματα συνδρομών έχουν παρουσιαστεί παραπάνω.

- ***Retrieve a list of subscriptions (Ανάκτηση της λίστας όλων των συνδρομών):***

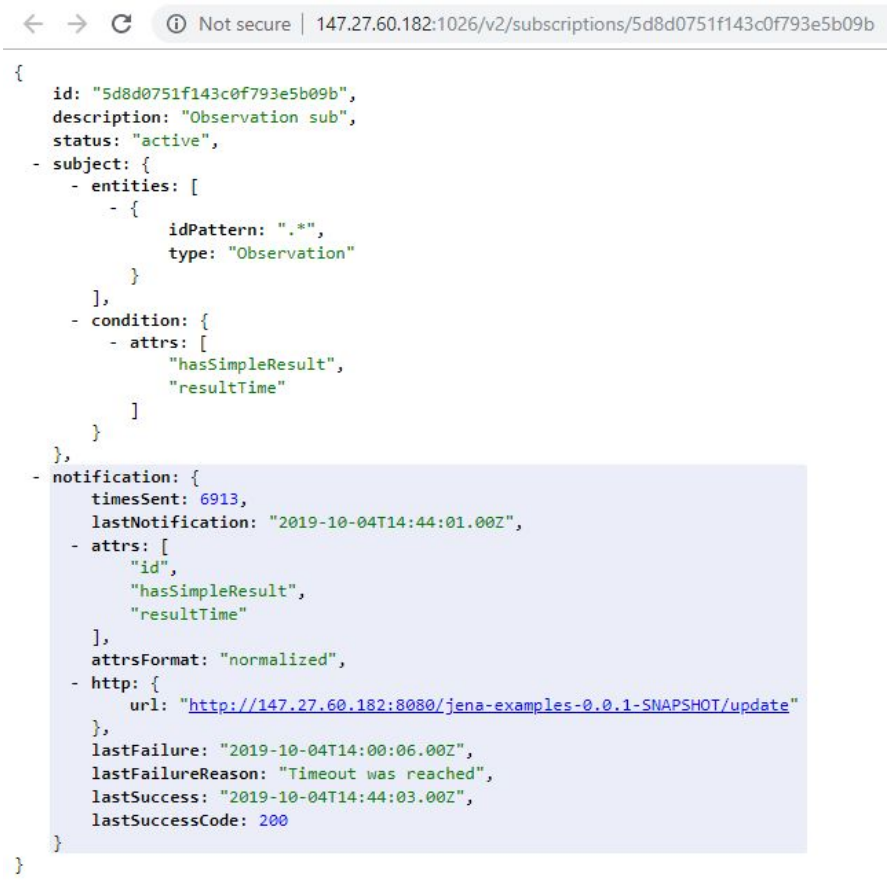
Η λειτουργία αυτή επιτρέπει στο χρήστη να δει μια λίστα με όλες τις συνδρομές που έχουν αποθηκευτεί στο σύστημα, προκειμένου να ανακτήσει όποιες από αυτές επιθυμεί ή να διαγράψει κάποιες από αυτές (βλ. επόμενες λειτουργίες). Κάθε συνδρομή έχει ένα χαρακτηριστικό identifier (ένα τυχαίο αλφαριθμητικό που δίνεται αυτόματα από την Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών) και προαιρετικά μια πολύ σύντομη περιγραφή που μπορεί να δώσει ο χρήστης στη συνδρομή, επομένως η λίστα μπορεί να περιέχει τα identifiers όλων των συνδρομών ή τις περιγραφές τους. Στην παρακάτω εικόνα φαίνεται μια λίστα με τα identifiers των συνδρομών που υπάρχουν στο σύστημα:




***Εικόνα 4.24:*** *Στιγμιότυπο της λίστας των συνδρομών (από το Web Application της πλατφόρμας).*

- ***Retrieve information about a specific subscription (Ανάκτηση πληροφοριών που αφορούν μια συγκεκριμένη συνδρομή):***

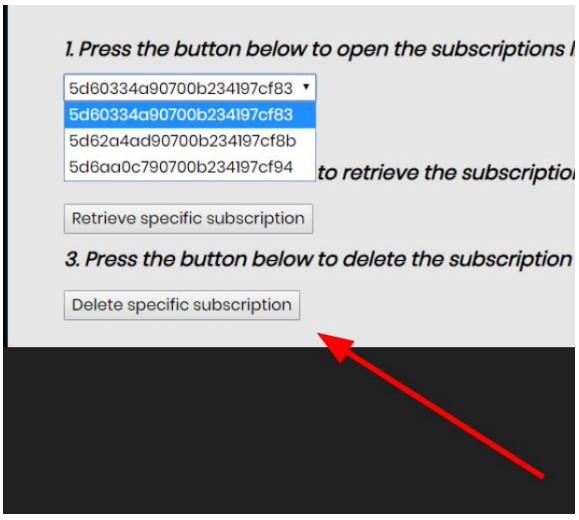
Η λειτουργία αυτή επιτρέπει στο χρήστη να επιλέξει μια συγκεκριμένη συνδρομή και να ανακτήσει την περιγραφή JSON της συνδρομής, η οποία υπάρχει αποθηκευμένη στο σύστημα. Στο Web Application της πλατφόρμας, ο χρήστης επιλέγει μια συνδρομή από τη λίστα και η περιγραφή της συνδρομής εμφανίζεται σε μια νέα καρτέλα του φυλλομετρητή.




*Εικόνα 4.25*

- ***Delete a subscription (Διαγραφή μιας συνδρομής):***

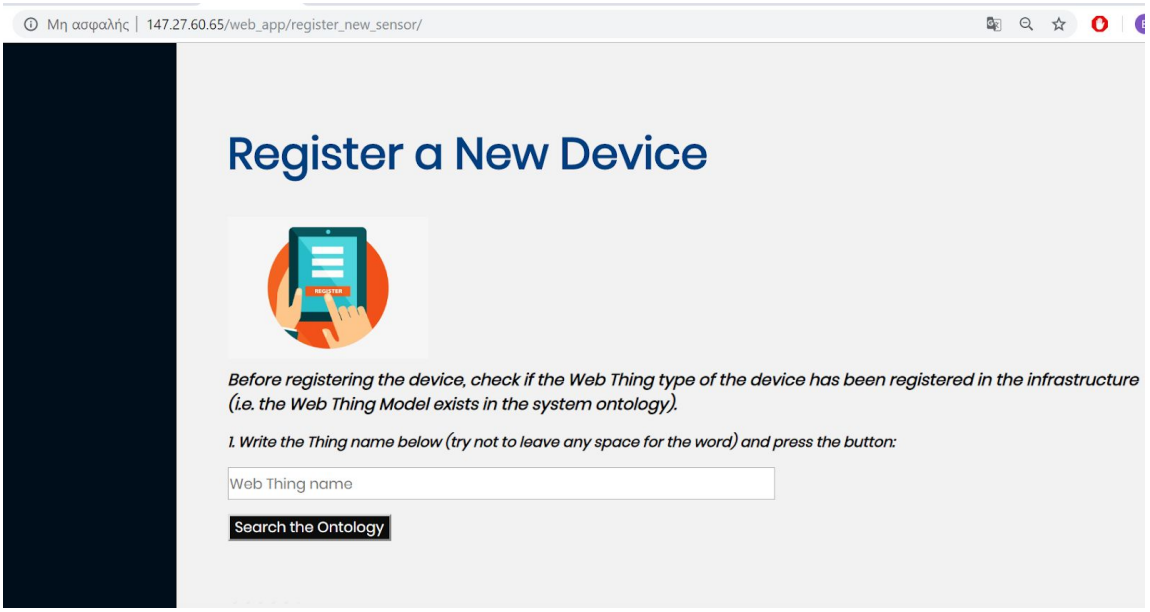
Η λειτουργία αυτή επιτρέπει στο χρήστη να διαγράψει οποιαδήποτε συνδρομή επιθυμεί, επιλέγοντας στο Web Application τη συνδρομή από τη λίστα και πατώντας ένα κουμπί για τη διαγραφή της συνδρομής, όπως φαίνεται στην παρακάτω εικόνα:



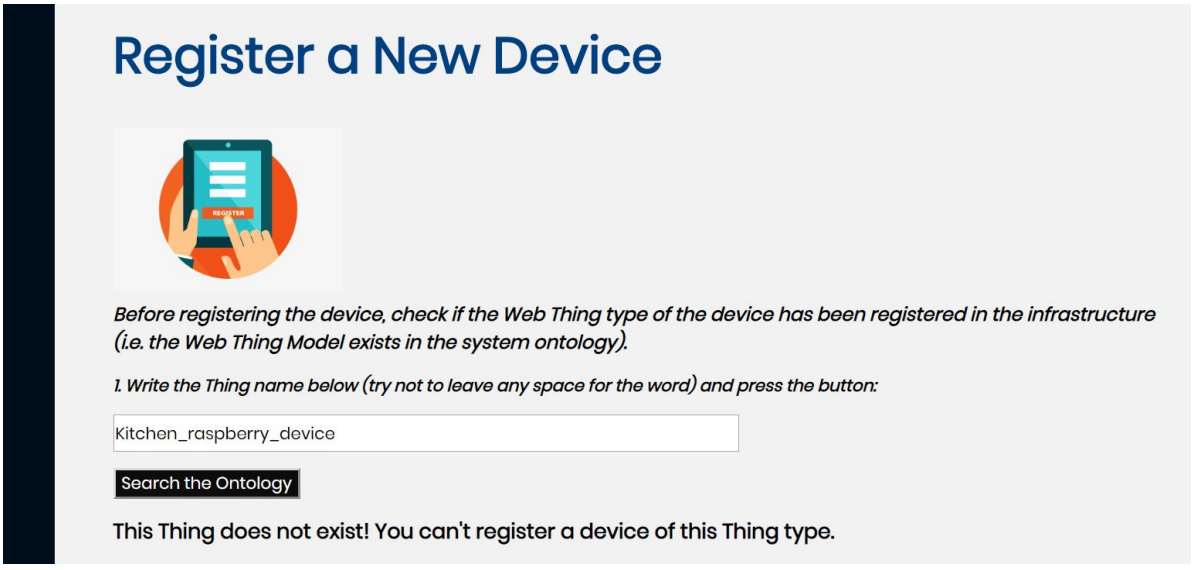

***Εικόνα 4.26:****: Στιγμιότυπο όπου φαίνεται η λίστα των συνδρομών και το κουμπί με το οποίο ο χρήστης μπορεί να διαγράψει την επιλεγμένη συνδρομή (από το Web Application της πλατφόρμας).*

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

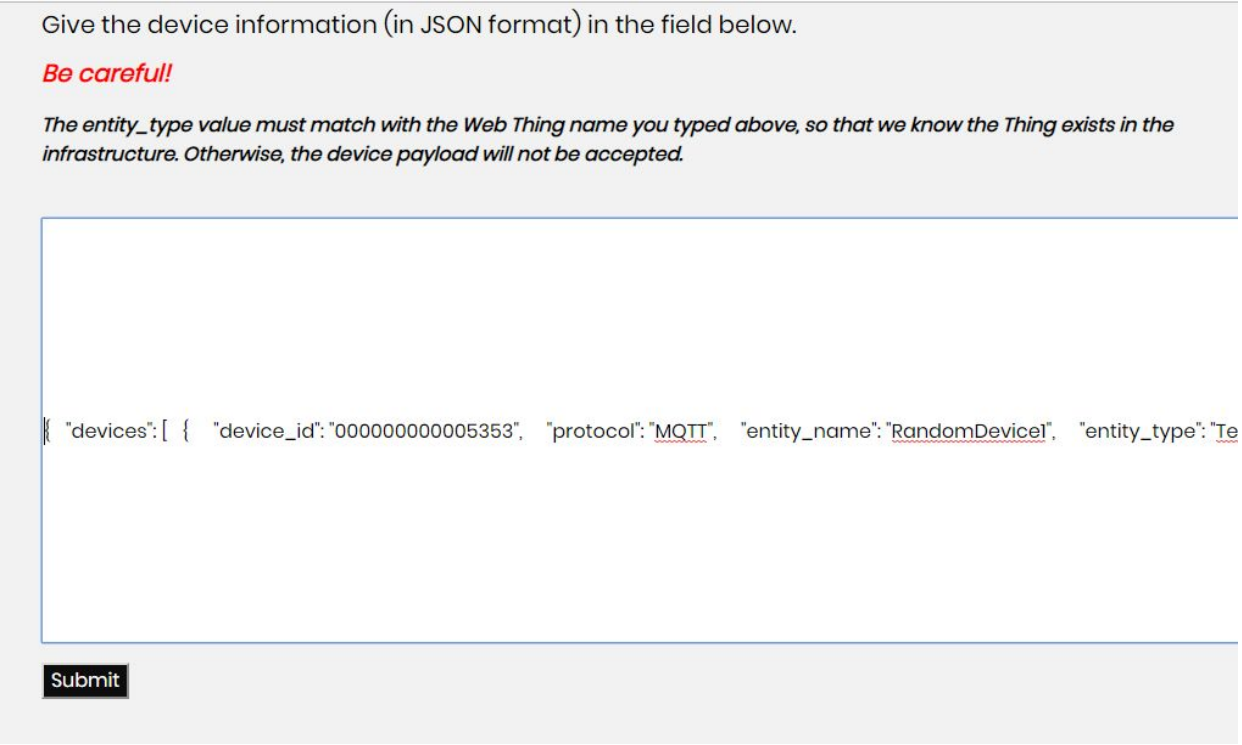
Η λειτουργία αυτή επιτρέπει στο χρήστη να εγγράψει μέσω του Web Application μια συγκεκριμένη συσκευή στον πίνακα συσκευών της υπηρεσίας Διαχείρισης Συσκευών - Backend IDAS (μέσω αυτής γίνεται η διασύνδεση των συσκευών με το σύστημα). Όπως αναφέρθηκε και στο Κεφάλαιο 2, η υπηρεσία IDAS διαθέτει το δικό της μηχανισμό προστασίας και έτσι λαμβάνει δεδομένα μόνο από τις εγγεγραμμένες φυσικές συσκευές. Επομένως, ο χρήστης κάνει εγγραφή μιας συγκεκριμένης συσκευής, εισάγοντας το όνομά της και άλλα στοιχεία που κρίνονται απαραίτητα, όπως είναι το αναγνωριστικό συσκευής (device id). Προτού, όμως, εισάγει τα στοιχεία της συσκευής, 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. Αν ο χρήστης λάβει θετική απάντηση, μπορεί να προχωρήσει στην εγγραφή της συσκευής, συνεπώς θα μπορεί πλέον η εν λόγω συσκευή να στέλνει δεδομένα στο σύστημα. Η διαδικασία που περιγράφηκε φαίνεται και στα παρακάτω στιγμιότυπα:




***Εικόνα 4.27:*** *Στιγμιότυπο της σελίδας εγγραφής νέας συσκευής στον IDAS.*




***Εικόνα 4.28:*** *Στιγμιότυπο όπου φαίνεται το μήνυμα που εκτυπώνεται σε περίπτωση που ο χρήστης εισάγει όνομα Πράγματος του οποίου το μοντέλο δεν έχει καταχωρηθεί στην οντολογία.*




***Εικόνα 4.29:*** *Στιγμιότυπο της φόρμας στην οποία ο χρήστης εισάγει μια JSON περιγραφή με τα χαρακτηριστικά που απαιτούνται για την εγγραφή μιας συσκευής στην υπηρεσία IDAS.*

## 4.2 Σύστημα διεπαφής χρηστών

Με τον όρο "Σύστημα διεπαφής χρηστών" αναφερόμαστε στην γραφική διεπαφή στην οποία ο χρήστης έχει πρόσβαση από το πρόγραμμα περιήγησής του, καθώς και τον κώδικα που τη δημιούργησε και τρέχει τοπικά σε αυτήν.

Προκειμένου να υλοποιηθούν οι γραφικές διεπαφές, χρησιμοποιήθηκαν γνωστές τεχνολογίες Ιστού: HTML[27], CSS[28], Javascript[29] και AJAX[30].

### *Γραφική διεπαφή εισόδου*

Όταν κάποιος χρήστης προσπαθήσει να μεταβεί για πρώτη φορά στην κεντρική σελίδα της Διαδικτυακής Εφαρμογής, ανακατευθύνεται αυτόματα στη σελίδα εισόδου του συστήματος, όπου καλείται να εισάγει τα στοιχεία (username

[27] https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
[28] https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps
[29] https://www.javascript.com/
[30] https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

και password) του λογαριασμού του, όπως φαίνεται στην εικόνα 4.2. Δεν επιτρέπεται στο χρήστη να έχει οποιαδήποτε αλληλεπίδραση με το σύστημα αν δεν περάσει επιτυχώς από αυτό το βήμα. Μόλις επιτευχθεί η είσοδος, δημιουργείται μια συνεδρία (session) με τα στοιχεία του χρήστη. Η παραπάνω διαδικασία δεν επαναλαμβάνεται όσο η συνεδρία αυτή είναι ενεργή. Οποιαδήποτε προσπάθεια του χρήστη για μετάβαση σε κάποια γραφική διεπαφή του συστήματος, χωρίς την ύπαρξη ενεργής συνεδρίας εισόδου στο πρόγραμμα περιήγησης, έχει ως αποτέλεσμα την επιστροφή του χρήστη στη σελίδα εισόδου.


***Εικόνα 4.30:*** *Σελίδα εισόδου του συστήματος*

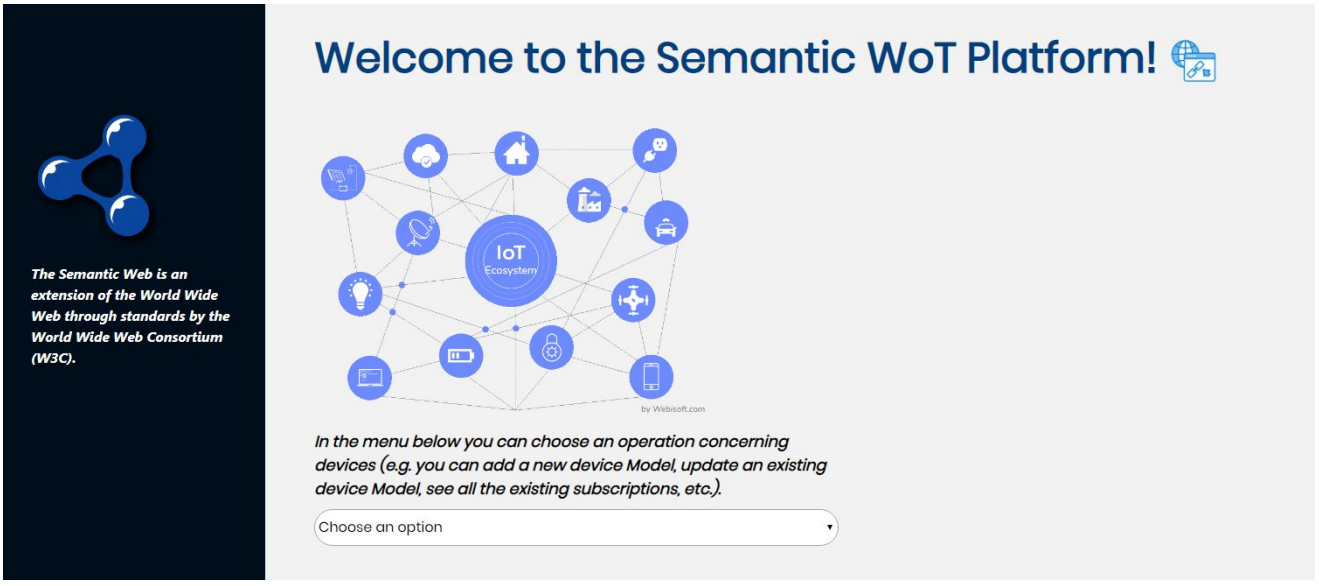
## *Γραφική διεπαφή χρήστη*

Η γραφική διεπαφή που αναπτύχθηκε αποτελεί αφετηρία για το χρήστη του συστήματος (Διαδικτυακή Εφαρμογή). Στη σελίδα αυτή υπάρχει ένα κεντρικό μενού που περιλαμβάνει όλες τις λειτουργίες που περιγράφονται στο Web Thing Model της W3C, καθώς και κάποιες που προστέθηκαν στα πλαίσια της παρούσας εργασίας.

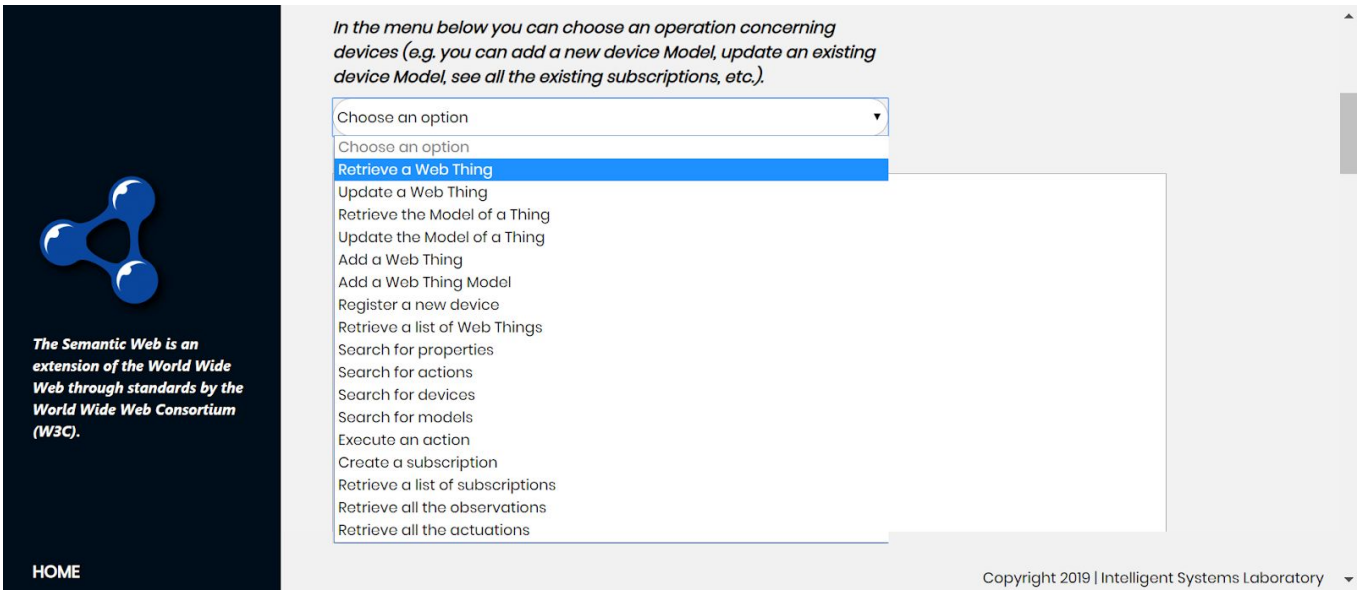
Ο χρήστης, για παράδειγμα, μπορεί να επιλέξει την εισαγωγή της περιγραφής JSON ενός νέου Πράγματος ή την εισαγωγή του μοντέλου ενός νέου Πράγματος (σε μορφή JSON-LD). Αφού επιλέξει την επιθυμητή ενέργεια, μπορεί να εισαγάγει την αντίστοιχη περιγραφή στο πεδίο που του υποδεικνύεται από το γραπτό κείμενο που εξηγεί τη λειτουργικότητα των δύο παραπάνω ενεργειών. Όλες οι επιλογές που βρίσκονται στο κεντρικό μενού δημιουργούν ανακατεύθυνση σε μια νέα σελίδα στην οποία ο χρήστης μπορεί είτε να πραγματοποιήσει την ενέργεια που επέλεξε (π.χ. την ανάκτηση μιας περιγραφής ενός Πράγματος, την ενημέρωση μιας περιγραφής, κλπ.) είτε να δει κατευθείαν το αποτέλεσμα της επιλογής του - ανάλογα με την επιλογή που έκανε στο μενού.

Ακολουθούν χαρακτηριστικά στιγμιότυπα της γραφικής διεπαφής του συστήματος, απ' τα οποία το καθένα αντιστοιχεί σε συγκεκριμένη λειτουργία (οι λειτουργίες αναγράφονται πάνω από κάθε στιγμιότυπο).

Στην εικόνα που ακολουθεί φαίνεται η αφετηρία της Διαδικτυακής Εφαρμογής και στη συνέχεια το κεντρικό μενού της:

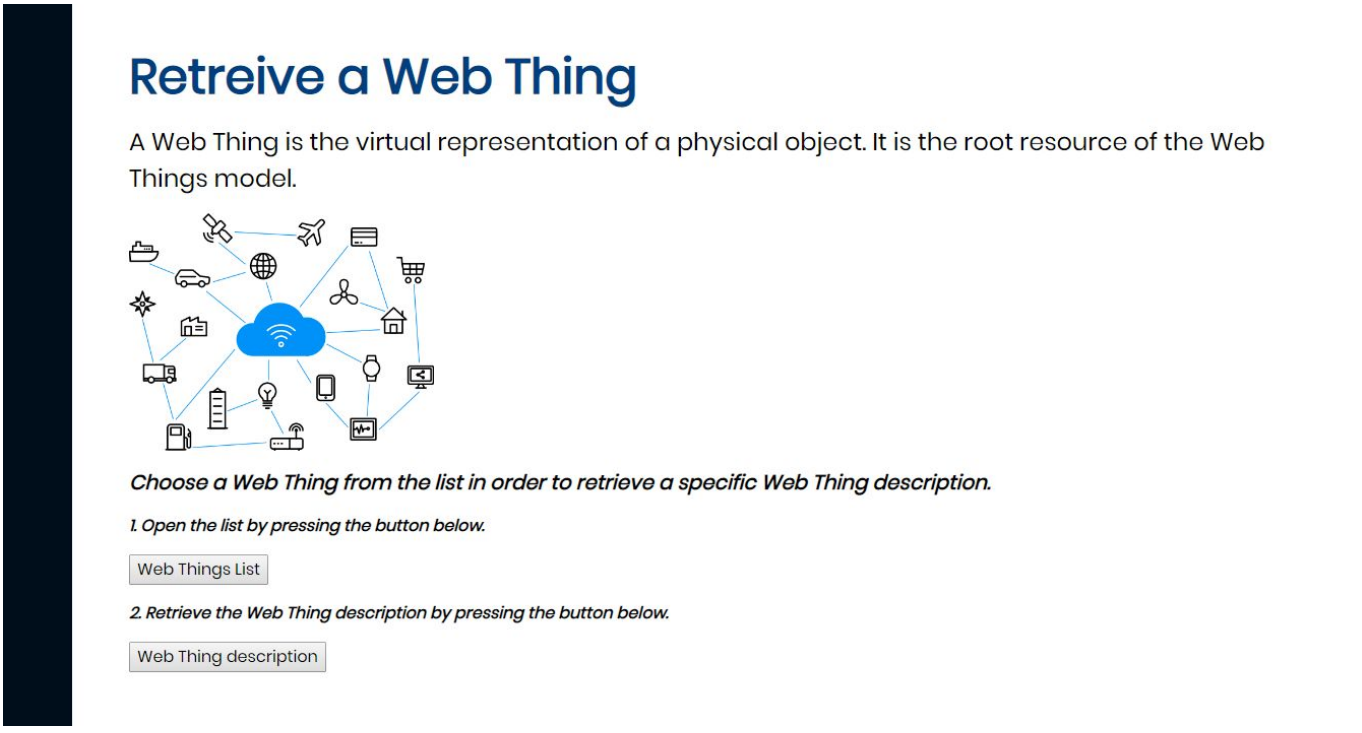



*Εικόνα 4.31*




*Εικόνα 4.32*

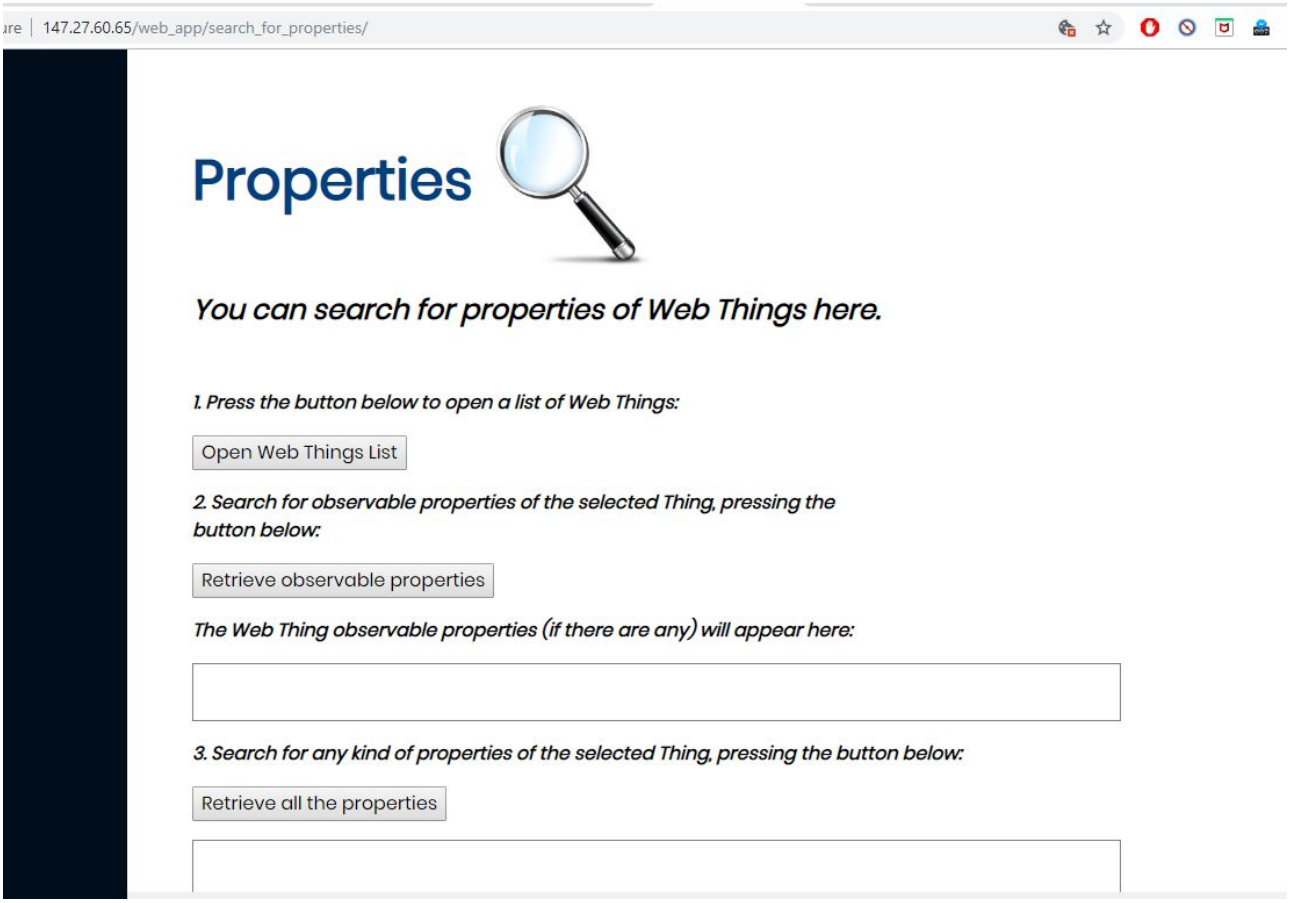
## Λειτουργία: Retrieve a Web Thing



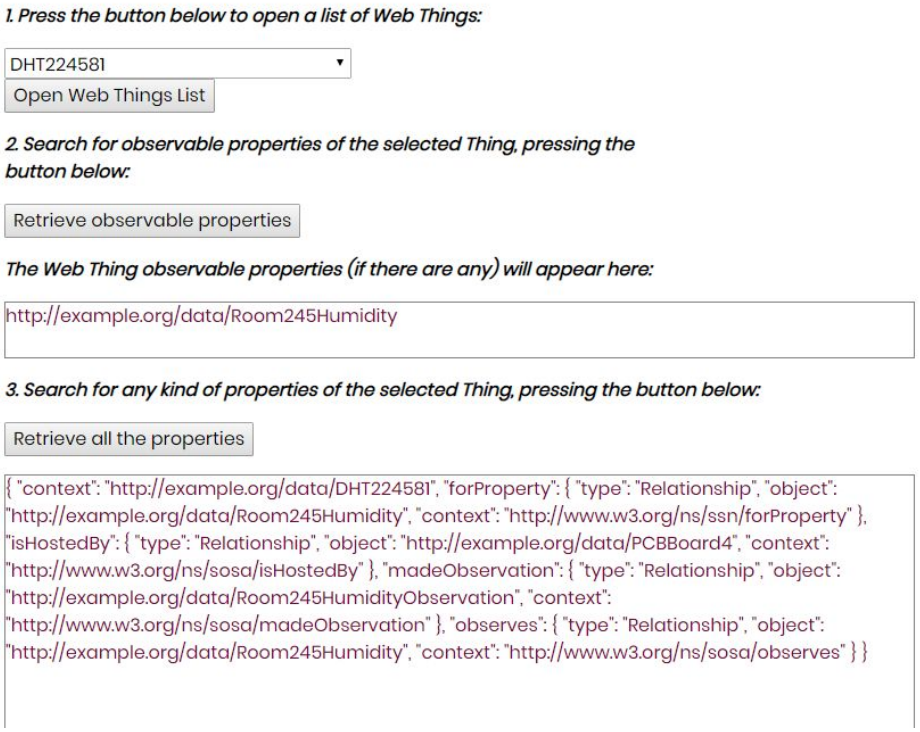

*Εικόνα 4.33*


*Εικόνα 4.34*

## Λειτουργία: Retrieve the properties of a Web Thing



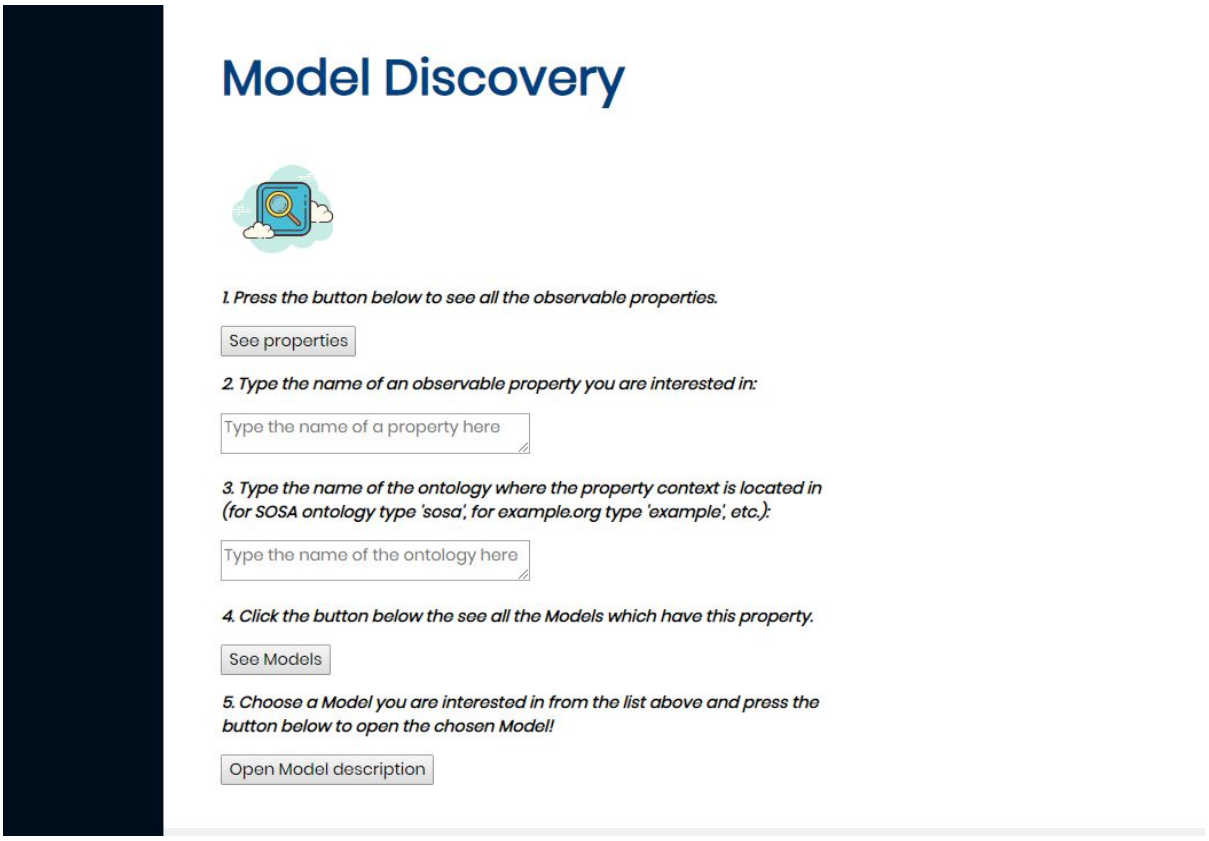

*Εικόνα 4.35*



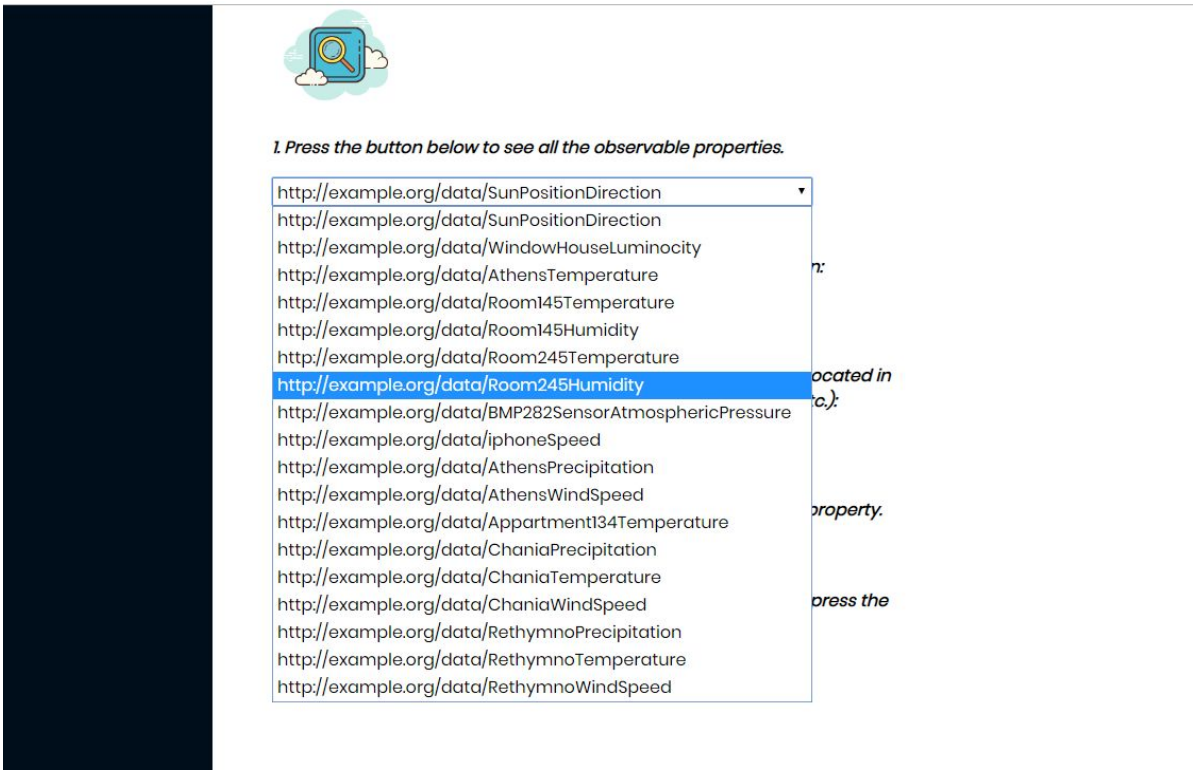

*Εικόνα 4.36*

## Λειτουργία: Search for models (Model Discovery ή IoT Model Discovery)

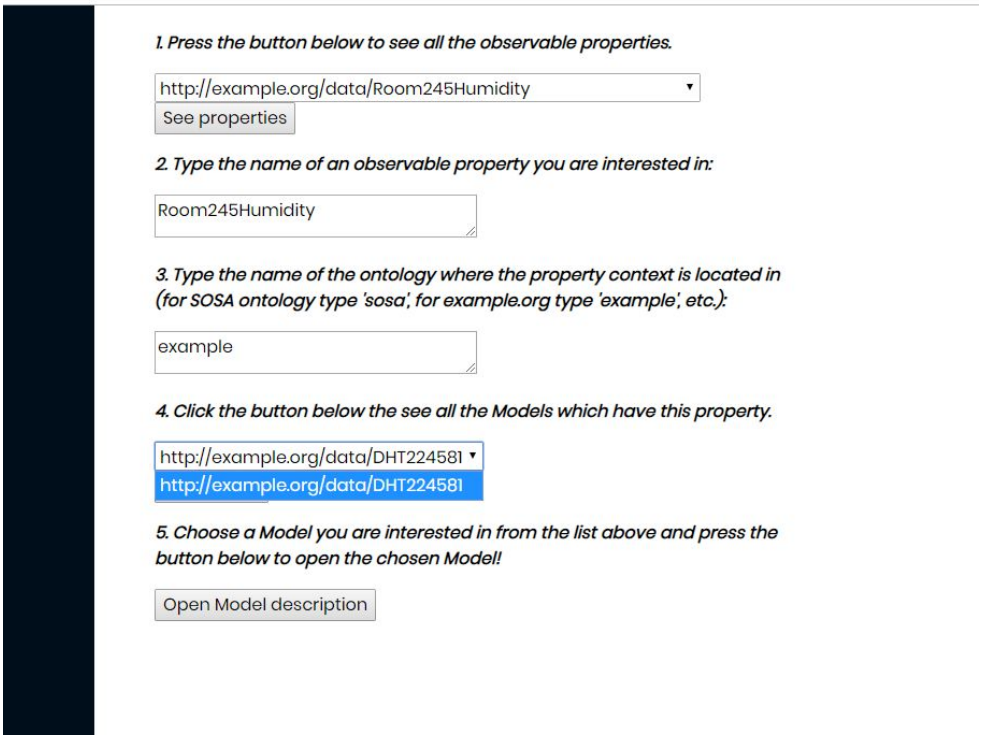
Η λειτουργία αυτή δεν υπάρχει στο Web Thing Model, αλλά προστέθηκε κατά την εκπόνηση της παρούσας εργασίας. Ο χρήστης μπορεί να αναζητήσει μοντέλα από την οντολογία με βάση τις μετρήσιμες ιδιότητές τους (observable properties).


Model Discovery
1. Press the button below to see all the observable properties.
See properties
2. Type the name of an observable property you are interested in:
Type the name of a property here
3. Type the name of the ontology where the property context is located in (for SOSA ontology type 'sosa', for example.org type 'example', etc.):
Type the name of the ontology here
4. Click the button below the see all the Models which have this property.
See Models
5. Choose a Model you are interested in from the list above and press the button below to open the chosen Model!
Open Model description

*Εικόνα 4.37*




*Εικόνα 4.38*



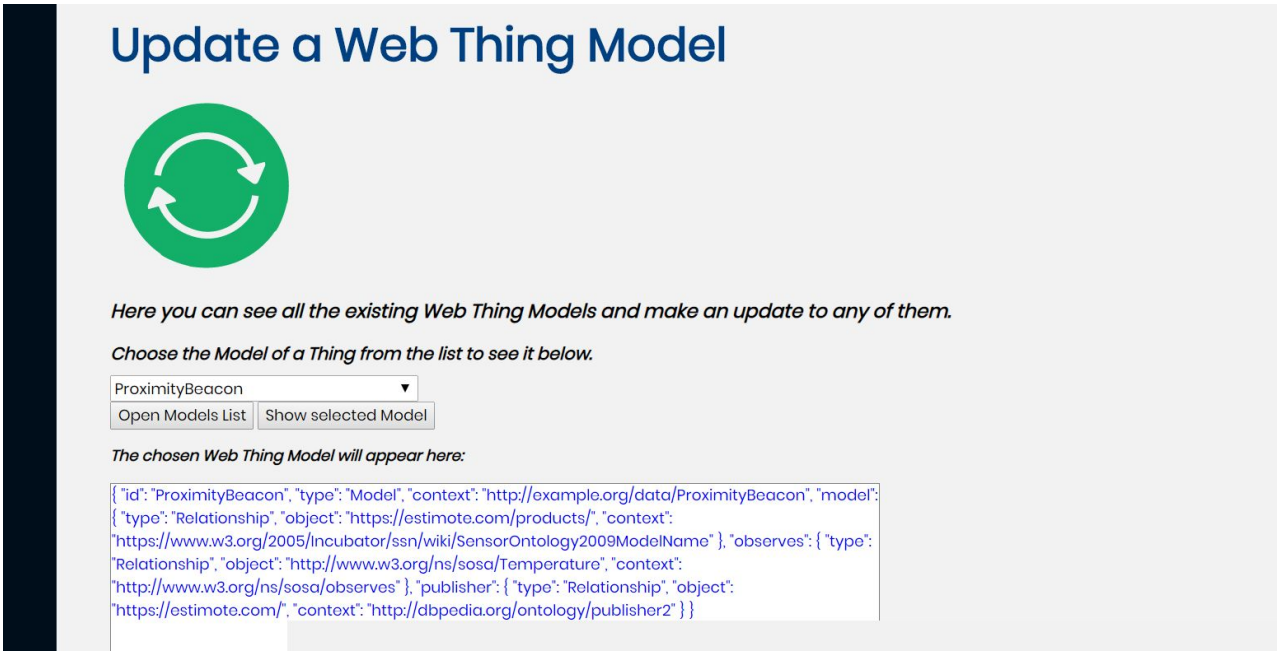

*Εικόνα 4.39*

Not secure | 147.27.60.182:1026/v2/entities/DHT224581?type=Sensor&options=keyValues


```
{
    id: "DHT224581",
    type: "Sensor",
    context: "http://example.org/data/DHT224581",
  - forProperty: {
        type: "Relationship",
        object: "http://example.org/data/Room245Humidity",
        context: "http://www.w3.org/ns/ssn/forProperty"
    },
  - isHostedBy: {
        type: "Relationship",
        object: "http://example.org/data/PCBBoard4",
        context: "http://www.w3.org/ns/sosa/isHostedBy"
    },
  - madeObservation: {
        type: "Relationship",
        object: "http://example.org/data/Room245HumidityObservation",
        context: "http://www.w3.org/ns/sosa/madeObservation"
    },
  - observes: {
        type: "Relationship",
        object: "http://example.org/data/Room245Humidity",
        context: "http://www.w3.org/ns/sosa/observes"
    }
}
```


*Εικόνα 4.40*

## Λειτουργία: Update the Model of a Thing




*Εικόνα 4.41*

Η φόρμα στην οποία γίνεται η ενημέρωση του μοντέλου του Πράγματος δε φαίνεται εδώ αλλά στην Εικόνα 4.17.


*Εικόνα 4.42*

# Κεφάλαιο 5

# Ανάλυση απόδοσης

Η υποδομή του συστήματος που υλοποιήθηκε στην παρούσα εργασία στεγάζεται στο περιβάλλον Intellicloud του Πολυτεχνείου Κρήτης και απαρτίζεται από **5** εικονικές μηχανές (virtual machines).

Στην **πρώτη εικονική μηχανή (1)** εκτελούνται:

- Η Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών (Orion Context Broker).
- Η υπηρεσία Cygnus, που χρησιμοποιείται με σκοπό την αποθήκευση ιστορικού ακατέργαστων / συγκεντρωτικών μετρήσεων χρονικής σειράς στην ιστορική βάση δεδομένων.
- Η Υπηρεσία Οντολογίας και συγκεκριμένα το Jena API, που φιλοξενείται στον Apache Tomcat (Tomcat Server).

Τα τεχνικά χαρακτηριστικά της παραπάνω εικονικής μηχανής (η μόνη με 2 CPU cores) είναι τα εξής:

| CPU | 2 CPU cores, x86_64@2.8GHz |
|---|---|
| Memory | 4096 MB |
| HDD | 40GB |
| OS | CentOS 7 |

Στη **δεύτερη εικονική μηχανή (2)** εκτελείται η υπηρεσία Διαχείρισης Συσκευών (Device Management) Backend IDAS. Τα τεχνικά χαρακτηριστικά της παραπάνω εικονικής μηχανής είναι τα εξής:

| CPU | x86_64@2.8GHz |
|---|---|
| Memory | 2048 MB |
| HDD | 20GB |
| OS | CentOS 7 |

Στην **τρίτη εικονική μηχανή (3)** εκτελούνται:

- Η Υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service).
- Η υπηρεσία Λογικής Εφαρμογής (Application Logic).

Στην **τέταρτη εικονική μηχανή (4)** εκτελείται η υπηρεσία FIWARE-COMET που χρησιμοποιείται για την ανάκτηση ιστορικών δεδομένων χρονικής σειράς από την ιστορική βάση δεδομένων.

Στην **πέμπτη εικονική μηχανή (5)** εκτελείται η υπηρεσία Keyrock IDM, που είναι η υπηρεσία ταυτοποίησης και εξουσιοδότησης χρηστών.

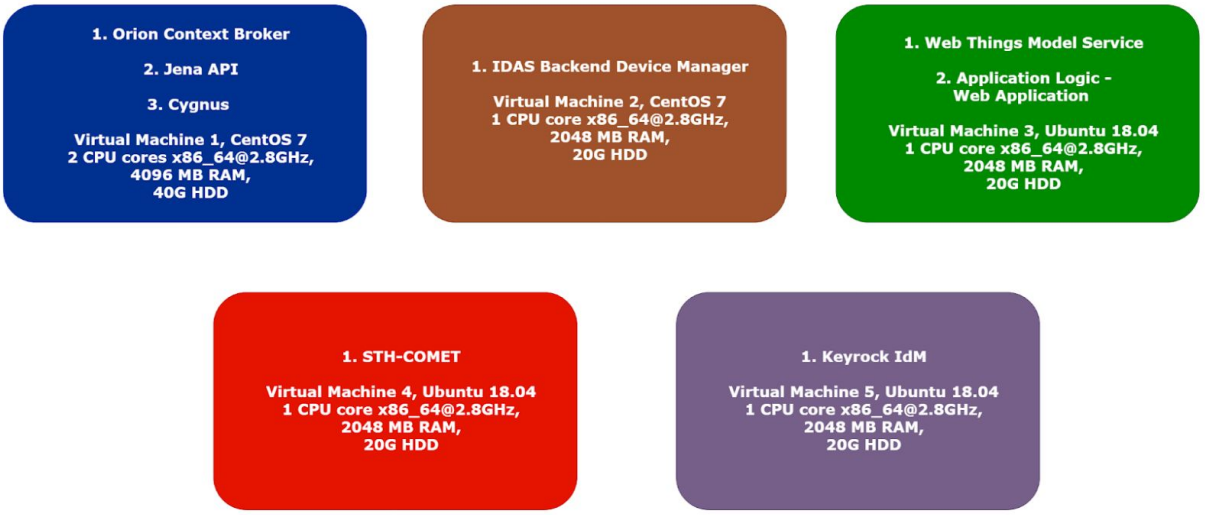
Τα τεχνικά χαρακτηριστικά των παραπάνω εικονικών μηχανών είναι τα εξής:

| CPU | x86_64@2.8GHz |
|---|---|
| Memory | 2048 MB |
| HDD | 20GB |
| OS | Ubuntu 18.04 |

Ο στόχος της πειραματικής διαδικασίας είναι να αποδειχθεί ότι το σύστημα που υλοποιήθηκε είναι αποδοτικό σε πραγματικές συνθήκες. Για αυτό το λόγο, χρησιμοποιήθηκε το εργαλείο Apache Bench[31], το οποίο έχει τη δυνατότητα δημιουργίας αρκετών ταυτόχρονων αιτημάτων (concurrent requests) και παράλληλα υπολογισμού σημαντικών παραμέτρων (όπως ο χρόνος απόκρισης για συγκεκριμένο αριθμό αιτημάτων) σε ένα διακομιστή HTTP. Χάρη στο Apache Bench, μπορούμε να δημιουργήσουμε συνθήκες φόρτου σε κάθε υπηρεσία του συστήματος ξεχωριστά, ενώ παράλληλα μας δίνεται η δυνατότητα να καθορίσουμε τον αριθμό των αιτημάτων που θα πρέπει να εξυπηρετήσει η κάθε υπηρεσία, καθώς και τον αριθμό αυτών που θα εκτελεστούν ταυτόχρονα. Κάθε εντολή του Apache Bench πρέπει να περιέχει την IP που πρόκειται να "χτυπήσει", το συνολικό αριθμό των αιτημάτων, τον αριθμό των ταυτόχρονων αιτημάτων, αλλά και την είσοδο που θα έχει η εκάστοτε εφαρμογή της υπηρεσίας.

Τα τεχνικά χαρακτηριστικά συνολικά των 5 εικονικών μηχανών φαίνονται συγκεντρωμένα στην παρακάτω εικόνα:

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




***Εικόνα 5.1:*** *Υποδομή Συστήματος.*

Προκειμένου να παρακολουθήσουμε τους φυσικούς πόρους των εικονικών μηχανών κατά τη διάρκεια των πειραμάτων, αξιοποιήσαμε το εργαλείο HTOP. Το πρόγραμμα αυτό 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.

Κάθε πείραμα πραγματοποιείται στο ίδιο μοτίβο, δηλαδή περιλαμβάνει **1000 αιτήματα** με σκοπό να εξεταστεί η λειτουργία της εκάστοτε υπηρεσίας του συστήματος. Όλα τα πειράματα επαναλαμβάνονται με χρήση διαφορετικού αριθμού ταυτόχρονων αιτημάτων κάθε φορά. Οι μετρήσεις δείχνουν το μέσο χρόνο εξυπηρέτησης ανά αίτημα και διακρίνονται σε κατηγορίες ανάλογα με τον ταυτοχρονισμό που χρησιμοποιήθηκε για να σταλούν τα αιτήματα. Συγκεκριμένα, τα αιτήματα στέλνονται ανά ένα, ανά πενήντα, ανά εκατό, ανά εκατόν πενήντα και ανά διακόσια. Με ταυτοχρονισμό ανά ένα σημαίνει ότι τα αιτήματα στέλνονται το ένα μετά το άλλο, διαδοχικά, με ταυτοχρονισμό ανά πενήντα σημαίνει ότι στέλνονται διαδοχικά σε ομάδες των πενήντα αιτημάτων, με ταυτοχρονισμό ανά εκατό σημαίνει ότι στέλνονται διαδοχικά σε ομάδες των εκατό αιτημάτων, και ούτω καθεξής. Για κάθε κατηγορία ταυτοχρονισμού εξετάζεται - εκτός από το το μέσο χρόνο εξυπηρέτησης ανά αίτημα - και η κατανάλωση σε CPU και μνήμη RAM σε μορφή ποσοστού. Τα αποτελέσματα αυτά έχουν καταγραφεί τόσο σε πίνακες όσο και διαγράμματα, τα οποία φαίνονται στη συνέχεια του Κεφαλαίου.

Για να δημιουργήσουμε συνθήκες φόρτου στο σύστημα, φτιάξαμε ένα υποθετικό σενάριο με εικονικές οντότητες 1050 σπιτιών, οι οποίες “φιλοξενούν”

αισθητήρες, και βρίσκονται σε μία συγκεκριμένη πόλη. Άρα πρόκειται για μια πόλη, στην οποία έχουμε 1050 σπίτια, με 3 είδη αισθητήρων (υγρασίας, θερμοκρασίας, φωτεινότητας) και σε όλα τα σπίτια 1 αισθητήρα από κάθε είδος. Οι οντότητες αυτές αποθηκεύτηκαν τόσο στην οντολογία όσο και στη βάση δεδομένων του Orion Context Broker, που φιλοξενεί τις πιο πρόσφατες τιμές.

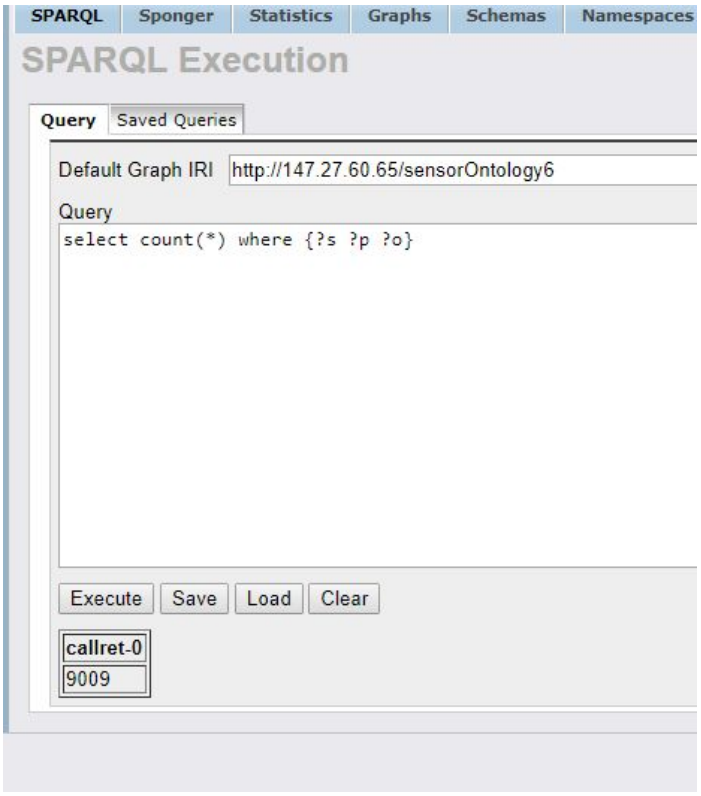
Λόγω του γεγονότος ότι ο αποθηκευτικός χώρος ενός γράφου δεν επαρκούσε για την αποθήκευση όλων των δεδομένων (μετρήσεις και άλλες πληροφορίες) και των 1050 σπιτιών, χωρίσαμε την οντολογία σε 7 διαφορετικούς γράφους, οι οποίοι όμως συντελούν μια ενιαία οντολογία που αφορά μία πόλη (υποθέσαμε την Αθήνα).

Στην Εικόνα 5.2 φαίνεται ένα παράδειγμα ερωτήματος που χρησιμοποιείται στα πειράματα και πώς αυτό (μέσω της γλώσσας SPARQL) κάνει αναζήτηση *και στους 7 γράφους* της οντολογίας που έχει δημιουργηθεί με τις συνθήκες φόρτου. Αυτό το ερώτημα ελέγχει εάν υπάρχει (δηλαδή εάν έχει αποθηκευτεί) ένα συγκεκριμένο μοντέλο συσκευής στην οντολογία. Πρόκειται για ερώτημα τύπου ASK της SPARQL, επομένως αυτό που επιστρέφει είναι "true" (σε περίπτωση ύπαρξης του μοντέλου στην οντολογία) ή "false" (σε περίπτωση που δεν υπάρχει το μοντέλο).

```
PREFIX ontol0: <http://147.27.60.65/sensorOntology>
PREFIX ontol1: <http://147.27.60.65/sensorOntology1>
PREFIX ontol2: <http://147.27.60.65/sensorOntology2>
PREFIX ontol3: <http://147.27.60.65/sensorOntology3>
PREFIX ontol4: <http://147.27.60.65/sensorOntology4>
PREFIX ontol5: <http://147.27.60.65/sensorOntology5>
PREFIX ontol6: <http://147.27.60.65/sensorOntology6>
ASK { <DHT224585> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Sensor> }
```

***Εικόνα 5.2:*** *Ερώτημα SPARQL που κάνει αναζήτηση και στους 7 γράφους.*

Στην Εικόνα 5.8 φαίνεται ο υπολογισμός (πάλι με χρήση της γλώσσας SPARQL) των συνολικών τριπλετών που φιλοξενεί ο κάθε γράφος της οντολογίας, συγκεκριμένα 9009 τριπλέτες. Εφόσον οι γράφοι είναι όλοι ίδιοι σε μέγεθος, συνολικά στους 7 γράφους έχουμε: 7 x 9009 = ***63063 τριπλέτες δεδομένων***.




***Εικόνα 5.3:*** *Στιγμιότυπο όπου φαίνεται ο υπολογισμός των τριπλετών του ενός γράφου της οντολογίας. στο πεδίο εκτέλεσης ερωτημάτων SPARQL που παρέχει το Virtuoso.*

## 5.1 Πείραμα 1

**Σενάριο:** Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα ανεύρεσης ιδιοτήτων (χαρακτηριστικών) συγκεκριμένου μοντέλου αισθητήρα, δηλαδή ψάχνει όλα τα attributes μια συγκεκριμένης οντότητας του Orion Context Broker.
**Υπηρεσίες**: Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στην Υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), η οποία επικοινωνεί και με τη Λογική Εφαρμογής. Οι δύο αυτές υπηρεσίες επεξεργάζονται το ερώτημα, το οποίο στη συνέχεια προωθείται στη βάση δεδομένων MongoDB του Orion Context Broker (όπου βρίσκονται αποθηκευμένα όλα τα μοντέλα αισθητήρων ως οντότητες JSON-LD). Το ερώτημα εκτελείται από τη βάση δεδομένων και η απάντηση επιστρέφει στο χρήστη ακολουθώντας "ανάποδα" την ίδια πορεία. Για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιούνται οι εικονικές μηχανές (1) και (3).

**Λεπτομέρειες:** Το αίτημα που εξετάζεται στο παρόν πείραμα αφορά στην εύρεση όλων των ιδιοτήτων (properties) ενός συγκεκριμένου μοντέλου αισθητήρων, δηλαδή επιστρέφει όλα τα attributes της JSON-LD οντότητας του εν λόγω μοντέλου. Το ερώτημα πραγματοποιήθηκε πάνω σε μια συλλογή από 1050 εικονικές οντότητες σπιτιών, από τα οποία το καθένα "φιλοξενεί" υποθετικά τρεις εικονικούς αισθητήρες που ανήκουν ο καθένας σε έναν διαφορετικό τύπο αισθητήρα (δηλαδή κάθε σπίτι έχει 3 διαφορετικά είδη αισθητήρων). Δηλαδή, η βάση δεδομένων έχει μια συλλογή από 1050 υποθετικά σπίτια και 3184 υποθετικές μετρήσεις, οι οποίες προέρχονται από τους παραπάνω αισθητήρες κάθε σπιτιού (οι αισθητήρες είναι 3150 στο πλήθος, αλλά υπήρχαν ήδη κάποιες λίγες μετρήσεις στη βάση και προστέθηκαν σε αυτές των 3150 αισθητήρων). Έτσι, η αναζήτηση γίνεται σε μια βάση με μεγάλο πλήθος δεδομένων - ώστε η απόκριση να είναι ρεαλιστική - όπου μόνο ένα μοντέλο πληροί τις προϋποθέσεις της αναζήτησης. Αυτό που αναζητείται από το ερώτημα είναι το συγκεκριμένο JSON-LD μοντέλο αισθητήρων και στη συνέχεια όλα τα χαρακτηριστικά του, που περιλαμβάνονται σε αυτό το μοντέλο.

**Αίτημα REST:**

GET

*http://147.27.60.65/web_app/search_for_properties/readpayload1.php?q=DHT224581*

**Αποτελέσματα και σχολιασμός:** Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) φαίνονται στο Σχήμα 5.1 και αναγράφονται στον Πίνακα 5.1. Παρατηρούμε ότι ο μέγιστος - με διαφορά - από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (9,261 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (5,942 ms). Η κατανάλωση CPU φτάνει το 100% σε όλες τις περιπτώσεις, ενώ η κατανάλωση της μνήμης RAM είναι κι αυτή ίδια στις πέντε περιπτώσεις (2,4%). Το ερώτημα, δηλαδή, είναι γρήγορο και δεν απαιτεί μεγάλη κατανάλωση μνήμης RAM παρά τις συνθήκες φόρτου που επικρατούν στο σύστημα.

## 5.2 Πείραμα 2

**Σενάριο:** Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα εύρεσης όλων των διαφορετικών μοντέλων των Πραγμάτων (Thing models) που έχουν

αποθηκευτεί στο σύστημα και συγκεκριμένα στον Orion Context Broker (ώστε στη συνέχεια να διαλέξει κάποιο από αυτά και να ανακτήσει κάποια πληροφορία για ένα από αυτά).

**Υπηρεσίες**: Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στην Υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), επομένως ακολουθείται η ίδια διαδικασία με αυτήν στο Πείραμα 1, με τη διαφορά ότι εδώ ανακτώνται όλα τα μοντέλα αισθητήρων που έχουν αποθηκευτεί στον Orion ως οντότητες JSON-LD. Ομοίως με πριν, για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιούνται οι εικονικές μηχανές (1) και (3).

**Λεπτομέρειες**: Το αίτημα που εξετάζεται στο παρόν πείραμα αφορά στην 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 id όλων των (διαφορετικών) μοντέλων, δηλαδή το μοναδικό αναγνωριστικό της οντότητας του κάθε μοντέλου στον Orion Context Broker.

**Αίτημα REST:**

GET

*http://147.27.60.65/web_app/retrieve_a_webthing_model/things.php*

**Αποτελέσματα και σχολιασμός:** Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) φαίνονται στο Σχήμα 5.2 και αναγράφονται στον Πίνακα 5.2. Παρατηρούμε ότι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται πάλι στην περίπτωση όπου τα αιτήματα στέλνονται ανά ένα (9,611 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά πενήντα (4,579 ms). Η κατανάλωση CPU φτάνει το 100% σε όλες τις περιπτώσεις εκτός από την πρώτη (όπου τα αιτήματα στέλνονται ανά ένα), όπου φτάνει μόνο μέχρι το 51%. Η κατανάλωση της μνήμης RAM είναι ίδια στις πέντε περιπτώσεις (2%). Το ερώτημα, όπως και στο Πείραμα 1, είναι γρήγορο και δεν απαιτεί μεγάλη κατανάλωση μνήμης RAM παρά τις συνθήκες φόρτου που επικρατούν στο σύστημα. Παρατηρούμε δηλαδή παρόμοια συμπεριφορά ανάμεσα στα αποτελέσματα των δύο ερωτημάτων και αυτό είναι λογικό, αφού πρόκειται για ερωτήματα που γίνονται αμφότερα στη βάση δεδομένων του Orion Context Broker και ακολουθούν την ίδια πορεία μέσα στο σύστημα (όσον αφορά τις υπηρεσίες).

## 5.3 Πείραμα 3

**Σενάριο:** Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα ερώτημα εύρεσης όλων των διαφορετικών ενεργοποιητών (actuators) που έχουν αποθηκευτεί στο σύστημα και συγκεκριμένα στην οντολογία του συστήματος, ώστε να του επιστραφούν όλα τα ονόματα των ενεργοποιητών.
**Υπηρεσίες:** Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στην Υπηρεσία Υλοποίησης του Μοντέλου του Ιστού των Πραγμάτων (Web Things Model Service), η οποία επικοινωνεί και με τη Λογική Εφαρμογής. Οι δύο αυτές υπηρεσίες επεξεργάζονται το ερώτημα, το οποίο στη συνέχεια προωθείται στην Υπηρεσία Οντολογίας - αρχικά στο Jena API και στη συνέχεια στους γράφους της οντολογίας στην Virtuoso (όπου βρίσκονται αποθηκευμένοι όλοι οι ενεργοποιητές σε μορφή τριπλετών). Το ερώτημα πραγματοποιήθηκε και σε αυτό το πείραμα πάνω σε μια πολύ μεγάλη συλλογή που έχει διαμορφωθεί από τα 1050 υποθετικά σπίτια, τους 3150 υποθετικούς αισθητήρες κλπ, ώστε η απόκριση να είναι ρεαλιστική, για την αναζήτηση μιας ελάχιστης ποσότητας στοιχείων που αποτελούν οι ενεργοποιητές. Προκειμένου το ερώτημα να πραγματοποιήσει αναζήτηση σε ολόκληρη την οντολογία του συστήματος, θα πρέπει να "απευθυνθεί" και στους 7 γράφους της οντολογίας (63063 τριπλέτες), γεγονός που είναι λογικό ότι θα προκαλέσει μία - έστω και μικρή - καθυστέρηση. Το ερώτημα, λοιπόν, εκτελείται από την οντολογία στην Virtuoso και η απάντηση επιστρέφει στο χρήστη ακολουθώντας "ανάποδα" την ίδια πορεία. Για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιούνται ξανά οι εικονικές μηχανές (1) και (3).

**Αίτημα REST:**
GET
*http://147.27.60.65/web_app/search_for_actions/actuators.php*

**Αποτελέσματα και σχολιασμός:** Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) φαίνονται στο Σχήμα 5.3 και αναγράφονται στον Πίνακα 5.3. Παρατηρούμε ότι ο μέγιστος - με διαφορά - από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (11,558 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (2,791 ms). Η κατανάλωση της CPU φτάνει μέχρι και το 60,3%

(μέγιστη τιμή), ενώ στην πρώτη περίπτωση φτάνει μόνο μέχρι 15,7%, άρα συμπεραίνουμε ότι έχουμε μικρή κατανάλωση CPU σε σχέση με τα δύο πρώτα πειράματα. Αντιθέτως, η κατανάλωση της μνήμης RAM έχει αυξηθεί σημαντικά σε σχέση με τα 2 πρώτα Πειράματα (γεγονός που ίσως οφείλεται στο "φόρτωμα" της οντολογίας στη μνήμη) αλλά δεν ξεπερνάει το 36,4%. Και αυτό το ερώτημα, δηλαδή, που πραγματοποιεί αναζήτηση στην οντολογία, είναι γρήγορο και δεν απαιτεί πολύ μεγάλη κατανάλωση μνήμης RAM, παρά τις συνθήκες φόρτου που επικρατούν στο σύστημα.

## 5.4 Πείραμα 4

**Σενάριο:** Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα ενημέρωσης (update) ενός μοντέλου που έχει αποθηκευτεί στο σύστημα (λειτουργία "Update the model of a Thing") και συγκεκριμένα στην οντολογία του συστήματος. Ο χρήστης με αυτήν την ενημέρωση αλλάζει μόνο ένα attribute ενός συγκεκριμένου μοντέλου που έχει αποθηκευτεί στην οντολογία σε μορφή τριπλετών.
**Υπηρεσίες:** Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή (Web Application) στην υπηρεσία Web Thing Model Service του Διακομιστή μεσολάβησης (Web Proxy), η οποία επικοινωνεί και με τη Λογική Εφαρμογής. Οι δύο αυτές υπηρεσίες επεξεργάζονται το ερώτημα, το οποίο στη συνέχεια προωθείται στην Υπηρεσία Οντολογίας - αρχικά στο Jena API και στη συνέχεια στο γράφο της οντολογία στην Virtuoso. Το ερώτημα απευθύνεται σε συγκεκριμένο γράφο όπου βρίσκεται αποθηκευμένο το μοντέλο - επομένως δεν ψάχνει και στους 7 - και εντοπίζει τη συγκεκριμένη τριπλέτα που πρέπει να υποστεί ενημέρωση. Επομένως, το ερώτημα αυτό είναι πιο "ελαφρύ" σε σχέση με το προηγούμενο. Για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιούνται ξανά οι εικονικές μηχανές (1) και (3).

**Αίτημα REST:**
POST
*http://147.27.60.65/web_app/update_webthing_model*
Το σώμα του αιτήματος περιλαμβάνει το όνομα/αναγνωριστικό του μοντέλου συσκευών που πρόκειται να ενημερωθεί, το όνομα του χαρακτηριστικού (attribute) που υφίσταται ενημέρωση και τη νέα τιμή που παίρνει το χαρακτηριστικό (όπως φαίνεται και στην Εικόνα 4.5).

**Αποτελέσματα και σχολιασμός:** Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) φαίνονται στο Σχήμα 5.4 και αναγράφονται στον Πίνακα 5.4. Παρατηρούμε ότι πάλι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (2,086 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (0,163 ms). Παρατηρούμε ότι ο μέσος χρόνος απόκρισης, καθώς και η κατανάλωση της CPU και της RAM, βρίσκονται σε πολύ χαμηλά επίπεδα. Ένας βασικός παράγοντας για αυτά τα αποτελέσματα μπορούμε να θεωρήσουμε ότι είναι το γεγονός ότι πρόκειται για ερώτημα που απευθύνεται *σε ένα μόνο γράφο* και αντικαθιστά μόνο μια συγκεκριμένη τριπλέτα.

## 5.5 Πείραμα 5

**Σενάριο:** Μία εφαρμογή του συστήματος "ζητάει" τη μέγιστη μέτρηση θερμοκρασίας ενός αισθητήρα για το διάστημα μιας ώρας. Στο πείραμα αυτό μελετάται μόνο η λειτουργία ανάκτησης της πληροφορίας από την υπηρεσία Comet, χωρίς να συνυπολογίζονται οι χρόνοι της Υπηρεσίας Σύνθεσης Εφαρμογών (Mashup Service), η οποία υλοποιείται στην εργασία του Στυλιανού Μποτωνάκη [2].

**Υπηρεσίες:** Το αίτημα ανάκτησης δεδομένων γίνεται από την υπηρεσία FIWARE COMET η οποία ανακτά τα ζητούμενα δεδομένα (συγκεντρωτική πληροφορία) από την ιστορική βάση και επιστρέφει την απάντηση στην εφαρμογή που τη "ζήτησε". Για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιείται η εικονική μηχανή (4).

**Αίτημα REST:**

GET

*http://147.27.60.79:8666/STH/v1/contextEntities/type/Observation/id/Room112TemperatureObservation/attributes/hasSimpleResult?aggrMethod=max&aggrPeriod=day*

**Αποτελέσματα και σχολιασμός:** Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) φαίνονται στο Σχήμα 5.4 και αναγράφονται στον Πίνακα 5.4. Παρατηρούμε ότι πάλι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση, όπου τα αιτήματα στέλνονται ανά ένα (2,086 ms), ενώ ο ελάχιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην περίπτωση όπου τα αιτήματα στέλνονται ανά εκατό (0,163 ms). Παρατηρούμε ότι και στο παρόν πείραμα ο μέσος χρόνος

απόκρισης, καθώς και η κατανάλωση της CPU και της RAM, βρίσκονται σε πολύ χαμηλά επίπεδα. Ένα βασικό παράγοντα για αυτά τα αποτελέσματα μπορούμε να θεωρήσουμε το γεγονός ότι πρόκειται για ερώτημα που πραγματοποιεί αναζήτηση σε μία υπηρεσία (σε μία μόνο εικονική μηχανή), για συγκεκριμένη μέτρηση αισθητήρα. Ακόμα, παρόλο που πρόκειται για την ιστορική βάση δεδομένων (που διατηρεί ιστορικό μετρήσεων χρονικής σειράς), σε αυτήν αποθηκεύονται μόνο οι μετρήσεις (οντότητες τύπου "Observation"), επομένως δεν έχει επιβαρυνθεί με πολλά επιπλέον δεδομένα όπως ο Orion Context Broker και η οντολογία.

## 5.6 Πείραμα 6

**Σενάριο:** Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα για εγγραφή μιας συγκεκριμένης συσκευής (device) στη λίστα των συσκευών της υπηρεσίας IDAS που μπορούν να στέλνουν δεδομένα στο σύστημα - δηλαδή μετά την ολοκλήρωση της εγγραφής, η υπηρεσία IDAS (Υπηρεσία Διαχείρισης Συσκευών Backend) θα αρχίσει να διαβάζει δεδομένα από τη συγκεκριμένη συσκευή.
**Υπηρεσίες**: Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή στην υπηρεσία Web Thing Model Service του Διακομιστή μεσολάβησης (Web Proxy), η οποία επικοινωνεί και με τη Λογική Εφαρμογής. Οι δύο αυτές υπηρεσίες επεξεργάζονται το ερώτημα, το οποίο στη συνέχεια προωθείται στην Υπηρεσία IDAS. Το ερώτημα αυτό είναι πολύ "ελαφρύ", καθώς γίνεται μόνο μια λειτουργία εγγραφής (μέσω αιτήματος POST), κατευθείαν στη λίστα των συσκευών και δεν πραγματοποιείται καμία αναζήτηση. Για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιούνται οι εικονικές μηχανές (2) και (3).
**Αίτημα REST:**
POST
*http://147.27.60.202:4041/iot/devices/*

**Αποτελέσματα και σχολιασμός:** Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) φαίνονται στο Σχήμα 5.6 και αναγράφονται στον Πίνακα 5.6. Παρατηρούμε ότι και στο παρόν πείραμα ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση. Ακόμα, ο μέσος χρόνος απόκρισης καθώς και τα ποσοστό κατανάλωσης της μνήμης RAM βρίσκονται σε πολύ χαμηλά επίπεδα, ενώ αυτό της CPU σε ψηλότερα, ωστόσο δε φτάνει το 100%. Πρόκειται για ερώτημα που πραγματοποιεί μια απλή εγγραφή στη λίστα συσκευών, επομένως είναι ένα

αίτημα που εκτελείται πολύ γρήγορα και δεν απαιτεί ιδιαίτερα μεγάλη κατανάλωση πόρων.

## 5.6 Πείραμα 7

**Σενάριο:** Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα για να ελέγξει εάν υπάρχει ένα συγκεκριμένο μοντέλο (Thing Model) στο σύστημα - συγκεκριμένα το αίτημα γίνεται στην οντολογία. Για την ακρίβεια, είναι το ερώτημα που φαίνεται στην Εικόνα 5.7.
**Υπηρεσίες:** Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή στην υπηρεσία Web Thing Model Service του Διακομιστή μεσολάβησης (Web Proxy), η οποία επικοινωνεί και με τη Λογική Εφαρμογής. Οι δύο αυτές υπηρεσίες επεξεργάζονται το ερώτημα, το οποίο στη συνέχεια προωθείται στην Υπηρεσία Οντολογίας, επομένως ακολουθεί την ίδια "πορεία" με τα προηγούμενα ερωτήματα που απευθύνονταν στην οντολογία του συστήματος. Το ερώτημα αυτό πραγματοποιεί αναζήτηση και στους 7 γράφους της οντολογίας, γεγονός που προκαλεί σίγουρα κάποια μικρή καθυστέρηση. Όπως αναφέρθηκε και παραπάνω, πρόκειται που επιστρέφει "true" ή "false", γεγονός που συμβαίνει μόλις εντοπίσει τη συγκεκριμένη τριπλέτα που ψάχνει (μέσω του ερωτήματος ASK). Επομένως, το ερώτημα μπορεί να απαντηθεί γρήγορα, με το που βρεθεί η τριπλέτα - δεν καθυστερεί αναζητώντας κι άλλες τριπλέτες. Για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιούνται οι εικονικές μηχανές (2) και (3).
**Αίτημα REST:**
GET
*http://147.27.60.65/web_app/register_new_sensor/check_if_thing_exists.php?q=DHT224585*

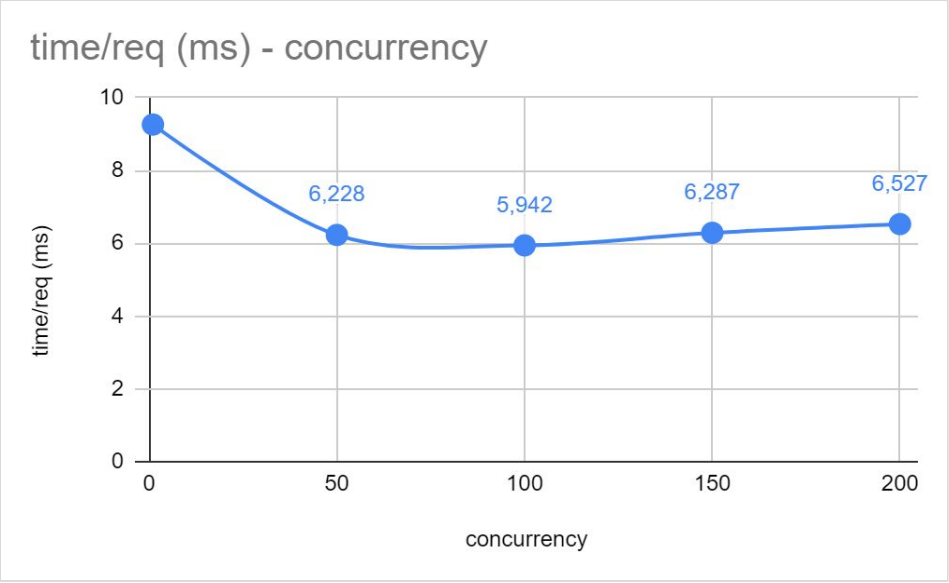
**Αποτελέσματα και σχολιασμός:** Τα αποτελέσματα για το μέσο χρόνο εκτέλεσης και την κατανάλωση πόρων (CPU και μνήμης RAM) φαίνονται στο Σχήμα 5.7 και αναγράφονται στον Πίνακα 5.7. Παρατηρούμε ότι και στο παρόν πείραμα ο μέγιστος από τους μέσους χρόνους εκτέλεσης (με διαφορά από τους υπόλοιπους) παρατηρείται στην πρώτη περίπτωση. Ακόμα, ο μέσος χρόνος απόκρισης βρίσκεται σε χαμηλά επίπεδα, ενώ δεν απαιτείται και πολύ υψηλή κατανάλωση πόρων RAM και CPU (δε φτάνουν το 100%).

## 5.6 Πείραμα 8

**Σενάριο:** Ένας χρήστης μέσω γραφικής διεπαφής κάνει ένα αίτημα για να προσθέσει ένα νέο μοντέλο της μορφής JSON-LD στο σύστημα (λειτουργία "Add a Web Thing Model") - συγκεκριμένα το αίτημα γίνεται στην οντολογία, όπου αποθηκεύεται το μοντέλο σε μορφή τριπλετών.
**Υπηρεσίες:** Το αίτημα δρομολογείται από τη Διαδικτυακή Εφαρμογή στην υπηρεσία Web Thing Model Service του Διακομιστή μεσολάβησης (Web Proxy), η οποία επικοινωνεί και με τη Λογική Εφαρμογής. Οι δύο αυτές υπηρεσίες επεξεργάζονται το ερώτημα, το οποίο στη συνέχεια προωθείται στην Υπηρεσία Οντολογίας, επομένως ακολουθεί και αυτό την ίδια "πορεία" με τα προηγούμενα ερωτήματα που απευθύνονταν στην οντολογία του συστήματος. Το ερώτημα αυτό πραγματοποιεί κατευθείαν μία προσθήκη, επομένως δεν καθυστερεί κάνοντας κάποιου είδους αναζήτηση. Για την ολοκλήρωση της παραπάνω λειτουργίας, ενεργοποιούνται οι εικονικές μηχανές (2) και (3).
**Αίτημα REST:**
POST
*http://147.27.60.182:8080/jena-project-0.0.1-SNAPSHOT/create_model_add_to_virt*
Το σώμα του αιτήματος είναι το μοντέλο JSON-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) φαίνονται στο Σχήμα 5.8 και αναγράφονται στον Πίνακα 5.8. Παρατηρούμε ότι πάλι ο μέγιστος από τους μέσους χρόνους εκτέλεσης παρατηρείται στην πρώτη περίπτωση. Ακόμα, τόσο ο μέσος χρόνος απόκρισης, όσο και η κατανάλωση πόρων RAM και CPU είναι σχεδόν σταθερά και κυμαίνονται σε πολύ χαμηλά επίπεδα.

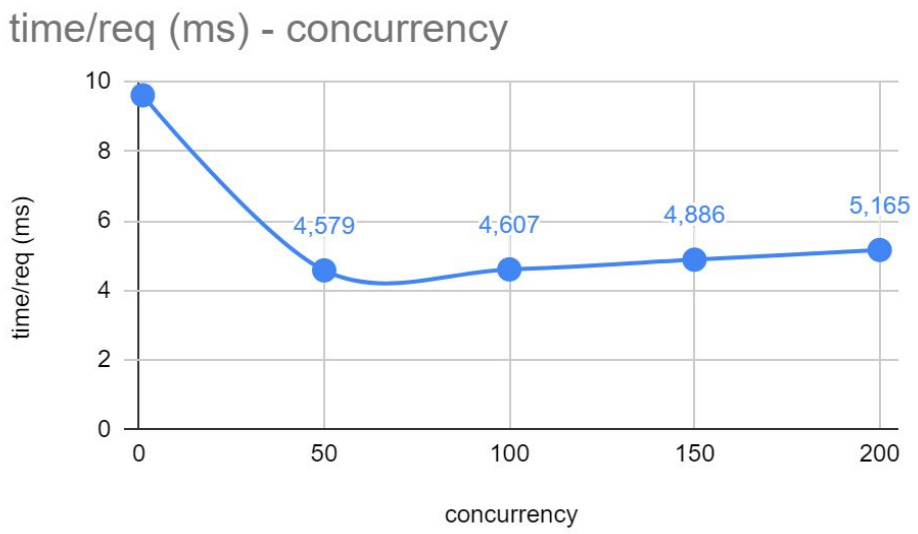
## Πίνακες - Διαγράμματα μετρήσεων




***Σχήμα 5.1:*** *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης ιδιοτήτων συγκεκριμένου μοντέλου αισθητήρα (Πείραμα 1).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 9,261 | 100 | 2,4 |
| 50 | 6,228 | 100 | 2,4 |
| 100 | 5,942 | 100 | 2,4 |
| 150 | 6,287 | 100 | 2,4 |
| 200 | 6,527 | 100 | 2,4 |

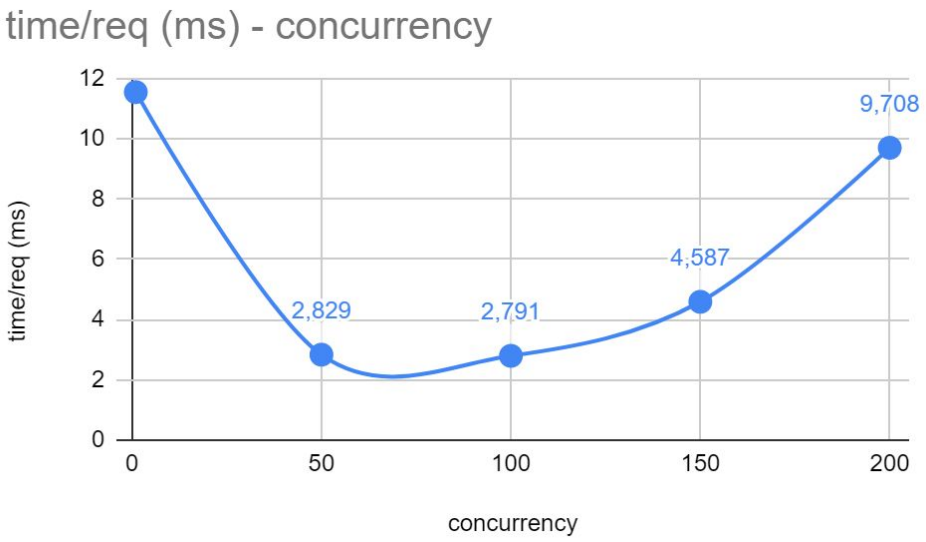
***Πίνακας 5.1:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα εύρεσης ιδιοτήτων συγκεκριμένου μοντέλου αισθητήρα (Πείραμα 1).*




***Σχήμα 5.2:*** *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών μοντέλων των Πραγμάτων (Πείραμα 2).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 9,611 | 51 | 2 |
| 50 | 4,579 | 100 | 2 |
| 100 | 4,607 | 100 | 2 |
| 150 | 4,886 | 100 | 2 |
| 200 | 5,165 | 100 | 2 |

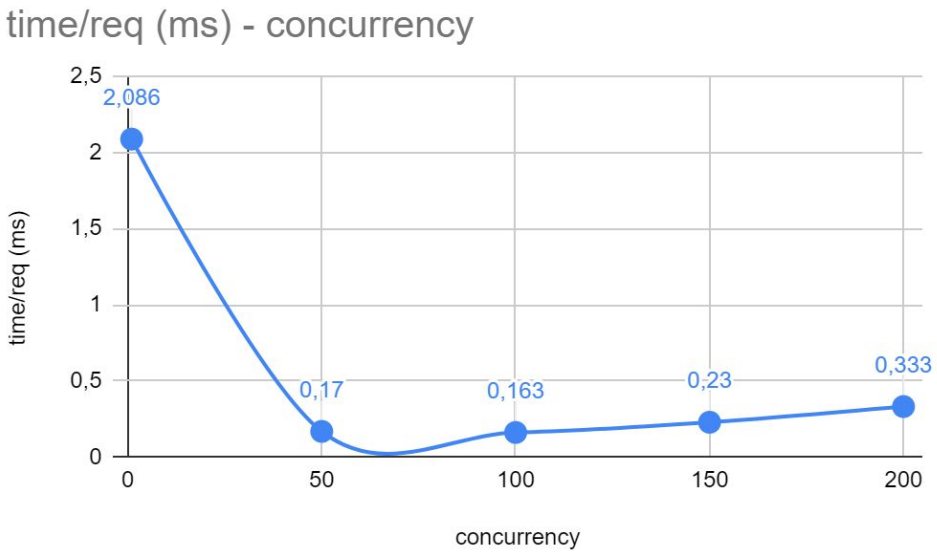
***Πίνακας 5.2:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών μοντέλων των Πραγμάτων (Πείραμα 2).*




***Σχήμα 5.3:*** *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών ενεργοποιητών / actuators (Πείραμα 3).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 11,558 | 15,7 | 36,4 |
| 50 | 2,829 | 53 | 36,4 |
| 100 | 2,791 | 51,4 | 36,4 |
| 150 | 4,587 | 50 | 36,4 |
| 200 | 9,708 | 60,3 | 36,4 |

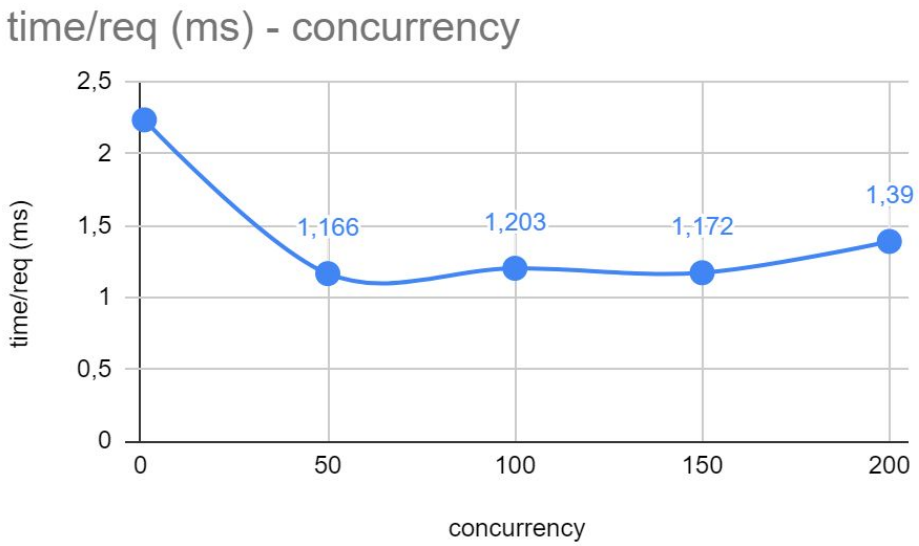
***Πίνακας 5.3:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα εύρεσης όλων των διαφορετικών ενεργοποιητών / actuators (Πείραμα 3).*




***Σχήμα 5.4:*** *Χρόνος απόκρισης σε 1000 ερωτήματα ενημέρωσης (update) ενός μοντέλου Πράγματος (Πείραμα 4).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 2,086 | 56,4 | 29,6 |
| 50 | 0,17 | 18,1 | 29,6 |
| 100 | 0,163 | 20,1 | 29,6 |
| 150 | 0,23 | 26,8 | 29,6 |
| 200 | 0,333 | 24,8 | 29,6 |

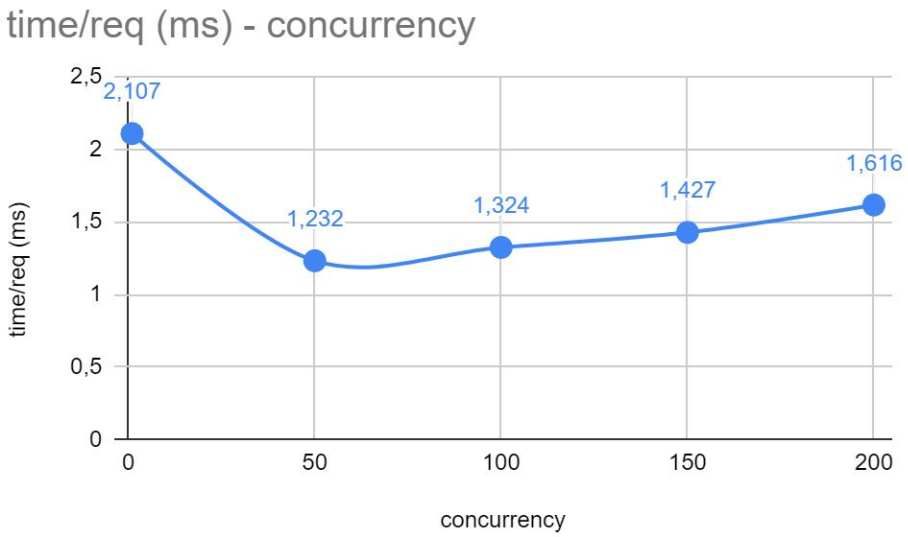
***Πίνακας 5.4:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα ενημέρωσης (update) ενός μοντέλου Πράγματος (Πείραμα 4).*




***Σχήμα 5.5:*** *Χρόνος απόκρισης σε 1000 ερωτήματα εύρεσης μέγιστης μέτρησης θερμοκρασίας ενός αισθητήρα για το διάστημα μιας ώρας (Πείραμα 5).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 2,236 | 54,8 | 3,1 |
| 50 | 1,166 | 37,5 | 3,2 |
| 100 | 1,203 | 45,8 | 3,4 |
| 150 | 1,172 | 52 | 3,2 |
| 200 | 1,39 | 45 | 3,6 |

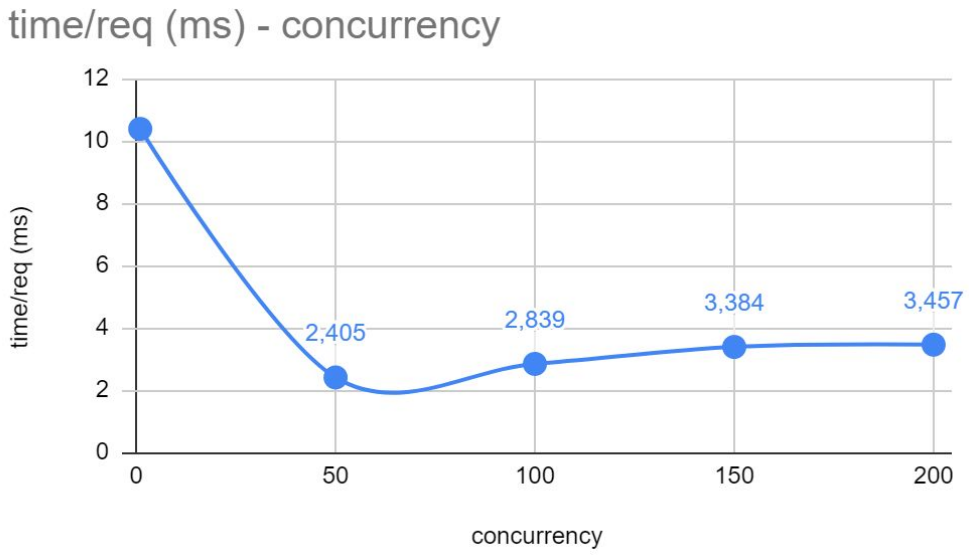
***Πίνακας 5.5:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα εύρεσης μέγιστης μέτρησης θερμοκρασίας ενός αισθητήρα για το διάστημα μιας ώρας (Πείραμα 5).*




***Σχήμα 5.6:*** *Χρόνος απόκρισης σε 1000 ερωτήματα εγγραφής μιας συγκεκριμένης συσκευής στη λίστα συσκευών της υπηρεσίας IDAS (Πείραμα 6).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 2,107 | 63 | 5,2 |
| 50 | 1,232 | 68,8 | 4,4 |
| 100 | 1,324 | 70,9 | 4,6 |
| 150 | 1,427 | 73,8 | 5,1 |
| 200 | 1,616 | 82,9 | 5,3 |

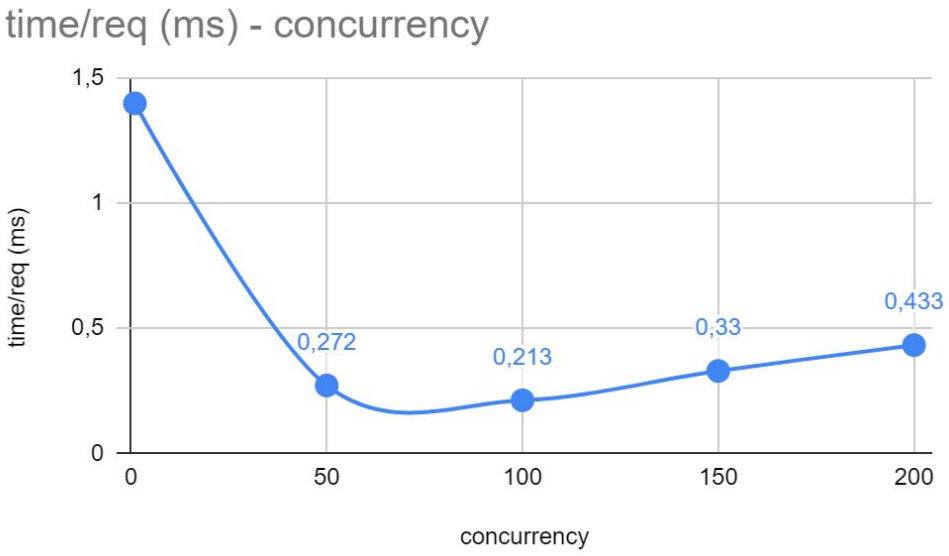
***Πίνακας 5.6:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα εγγραφής μιας συγκεκριμένης συσκευής στη λίστα συσκευών της υπηρεσίας IDAS (Πείραμα 6).*




***Σχήμα 5.7:*** *Χρόνος απόκρισης σε 1000 ερωτήματα ελέγχου ύπαρξης μοντέλου Πράγματος στην οντολογία (Πείραμα 7).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 10,383 | 62 | 31 |
| 50 | 2,405 | 61 | 31 |
| 100 | 2,839 | 79 | 31 |
| 150 | 3,384 | 79 | 31 |
| 200 | 3,457 | 91 | 31 |

***Πίνακας 5.7:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα ελέγχου ύπαρξης μοντέλου Πράγματος στην οντολογία (Πείραμα 7).*




***Σχήμα 5.8:*** *Χρόνος απόκρισης σε 1000 ερωτήματα εισαγωγής νέου μοντέλου Πράγματος στην οντολογία (Πείραμα 7).*

| concurrency | time/req (ms) | CPU load (%) | RAM usage (%) |
|---|---|---|---|
| 1 | 1,398 | 21 | 31 |
| 50 | 0,272 | 22 | 31 |
| 100 | 0,213 | 22 | 31 |
| 150 | 0,33 | 22 | 31 |
| 200 | 0,433 | 25 | 31 |

***Πίνακας 5.8:*** *Χρόνος απόκρισης, κατανάλωση CPU και κατανάλωση RAM σε 1000 ερωτήματα εισαγωγής νέου μοντέλου Πράγματος στην οντολογία. (Πείραμα 7).*

# Κεφάλαιο 6

# Συμπεράσματα - Μελλοντικές Επεκτάσεις

## 6.1 Συμπεράσματα

Η υιοθέτηση Υπηρεσιοκεντρικής Αρχιτεκτονικής, και συγκεκριμένα των υπηρεσιών που βασίζονται στο πρότυπο REST, διευκόλυνε σε μεγάλο βαθμό την επικοινωνία μεταξύ των υπηρεσιών και συνεπώς τον προγραμματισμό της δομής ενορχήστρωσης των υπηρεσιών (App logic). Μεγάλο πλεονέκτημα αυτής της αρχιτεκτονικής αποτέλεσε η ευελιξία στη χρήση διαφορετικών γλωσσών προγραμματισμού για να πραγματοποιηθούν διαφορετικές λειτουργίες του συστήματος καθώς και ευκολία στην παρέμβαση - τροποποίηση υπηρεσιών μεμονωμένων υπηρεσιών, δίχως να επηρεάζεται το συνολικό σύστημα. Επομένως, το σύστημα βασίζεται σε μια επεκτάσιμη και εύκολα τροποποιήσιμη αρχιτεκτονική.

Το γεγονός ότι το σύστημα αναπτύχθηκε μέσω του οικοσυστήματος FIWARE, προσέφερε 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, η Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών (Keyrock), κ.ά.

Η υπηρεσία IDAS του FIWARE παρείχε στο σύστημα τη ζητούμενη ανεξαρτησία από τα πρωτόκολλα επικοινωνίας των συσκευών, έτσι ώστε οι προγραμματιστές εφαρμογών να λαμβάνουν κατευθείαν την πληροφορία σε JSON / JSON-LD και να μην ενδιαφέρονται για το εκάστοτε πρωτόκολλο. Συνεπώς, ικανοποιείται αυτή η προδιαγραφή που τέθηκε αρχικά - για ανεξαρτησία από το πρωτόκολλο επικοινωνίας - η οποία ορίζεται από τον Ιστό των Πραγμάτων (Web of Things).

Αξιολογώντας την υπηρεσία Οντολογίας, 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.

Συνοψίζοντας τα παραπάνω συμπεράσματα, η πλατφόρμα που αναπτύχθηκε είναι σε θέση να διαχειρίζεται μεγάλο αριθμό συσκευών, ενώ οι λειτουργίες που παρέχει εκτελούνται ικανοποιητικά σε πραγματικό χρόνο. Ακόμα, ο σημασιολογικός χαρακτήρας της υλοποίησης προσδίδει στο σύστημα ένα σημαντικό πλεονέκτημα που εκλείπει από άλλες αρχιτεκτονικές Διαδικτύου των Πραγμάτων.

## 6.2 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 περιβάλλοντος Διαδικτύου των Πραγμάτων για την καταχώρηση, διαχείριση και δημοσίευση συσκευών, που επιτρέπει και διευκολύνει την δημιουργία εφαρμογών με σημασιολογικό χαρακτήρα. Ωστόσο, καθώς η εργασία εκπονήθηκε μέσα σε ένα συγκεκριμένο χρονικό πλαίσιο, είναι αναπόφευκτο να υπάρχουν ορισμένες αδυναμίες που θα περιγραφούν παρακάτω συνοδευόμενες από πιθανές προτάσεις για τη λύση τους.

Μια βελτίωση που επιδέχεται το σύστημα είναι σίγουρα η προσθήκη επιπλέον υπηρεσιών Ασφάλειας (Security services), για παράδειγμα η χρήση της

υπηρεσίας PEP Proxy του FIWARE, που επιτρέπει την πρόσβαση σε πόρους του συστήματος μόνο σε συγκεκριμένους εγγεγραμμένους - εξουσιοδοτημένους χρήστες και υπηρεσίες, κάτι που λείπει από το παρόν σύστημα. Η προσθήκη μιας τέτοιας υπηρεσίας ***προτείνεται ως μια σημαντική μελλοντική επέκταση***, καθώς κρίνεται απαραίτητη η ασφάλεια του συστήματος από αυτήν.

Τα αιτήματα μεταξύ των υπηρεσιών του συστήματος πραγματοποιούνται με τη χρήση του πρωτοκόλλου HTTP. Σημαντική βελτίωση στον τομέα της ασφάλειας του συστήματος θα ήταν η αλλαγή του πρωτοκόλλου επικοινωνίας σε HTTPS καθώς είναι - λόγω της αρχιτεκτονικής του - πιο ασφαλές πρωτόκολλο όσον αφορά στη μετάδοση "ευαίσθητων" πληροφοριών όπως είναι τα OAuth2 tokens.

Ακόμα, εντοπίστηκε το πρόβλημα σχετικά με την επεκτασιμότητα των γράφων της Virtuoso, καθώς παρατηρήσαμε ότι ο αριθμός των δεδομένων που μπορούν να αποθηκεύσουν είναι περιορισμένος, με αποτέλεσμα να χωρίσουμε την οντολογία σε διαφορετικούς γράφους. Μία λύση στο παραπάνω πρόβλημα θα ήταν να κάνουμε μια καλύτερη κατανομή της πληροφορίας σε γράφους, για παράδειγμα να "διακρίνουμε" την κάθε πόλη που μελετάται από το σύστημα σε μικρότερες περιοχές, με αποτέλεσμα την "ελάφρυνση" του κάθε γράφου της οντολογίας. Αυτό θα είναι πολύ σημαντικό για να πραγματοποιείται πιο εύκολα και αποδοτικά ο μηχανισμός συλλογισμού (reasoning) της Υπηρεσίας Οντολογίας του συστήματος.

# Βιβλιογραφία - Αναφορές

[1] X. Koundourakis: Design and Implementation of Service Oriented Architecture for Deploying IoT Applications in the Cloud, Diploma Thesis, School of Electrical and Computer Engineering, Technical University of Crete, 2019.
https://dias.library.tuc.gr/view/81121

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

[3] Euripides G.M. Petrakis, Stelios Sotiriadis, Theodoros Soultanopoulos, Pelagia Tsiachri Rentaa, Rajkumar Buyyac, Nik Bessis, Internet of Things as a Service (iTaaS): Challenges and solutions for management of sensor data on the cloud and the fog, September 2018.
doi:10.1016/j.iot.2018.09.009.
http://www.intelligence.tuc.gr/~petrakis/publications/iTaaS.pdf

[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] Farzad Khodadadi, Richard O. Sinnott, A Semantic-aware Framework for Service Definition and Discovery in the Internet of Things Using CoAP, in: The 8th

International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN).
https://www.sciencedirect.com/science/article/pii/S1877050917317441

[6] 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

[7] 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

[8] Vlad Trifa, Dominique Guinard, David Carrera, Web Thing Model, W3C Member Submission 24 August 2015.
https://www.w3.org/Submission/wot-model/