ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Προγραμματισμού και
Τεχνολογίας Ευφυών Υπολογιστικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Management of Health Sensor Data in the Cloud

Διαχείρηση Δεδομένων Ιατρικών Αισθητήρων στο Υπολογιστικό Νέφος

Πελαγία Τσιαχρή-Ρέντα

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΠΤΡΟΠΗ

Ευριπίδης Πετράκης, Καθηγητής (Επιβλέπων)

Βασίλης Σαμολαδάς, Επίκουρος Καθηγητής

Δρ. Στέλιος Σωτηριάδης, Επιστημονικός Συνεργάτης

ΧΑΝΙΑ 2017

Abstract

The quality of health services can be significantly improved by supporting health care procedures with new technologies such as cloud computing and Internet of Things (IoTs). The need to monitor patient's health remotely in real time becomes a requirement especially for chronic patients and the elderly. A possible solution is the use wearable sensors connected to the Internet and capable of transmitting patient health status to the cloud and from there to the health care personnel. In this thesis, we will focus on the management of health related information on the cloud. Sensor information relates mainly to patients vital measurements like cardiac pulse rate and blood oxygen saturation and is acquired with the aid of special purpose medical sensors. We extend “Inteligate” (a mobile application that supports the collection of vital measurement collected from BLE sensors to Androids devices) with cloud functionality. Building upon principles of Service Oriented Architectures (SOA) and FIWARE, the application enables sensor data to connect to the Cloud where this data is processed, stored and communicated with the health care personnel. The cloud application is extended with new awareness services for both health care providers and patients minimizing the risk of data loss or late response to emergency conditions as defined by rules encoding patient specific reaction plans, especially in cases where normal medical measurement levels exceed the predefined limits. The performance of the system on the cloud has been experimentally evaluated using synthetic data loads that simulate the use of the system by hundreds of users who simultaneously send sensor data to the Cloud. The results of this evaluation reveal that the system on the cloud is able to respond close to real time even under heavy loads approaching the limits of a typical Apache Web server on the cloud that receives service request. Beyond this point, it a responsibility of the system designer to allocate more cloud resources (including Apache servers) to handle excessive data loads.

Περίληψη

Η ποιότητα των παρεχόμενων υπηρεσιών υγείας καθώς και ο βαθμός ικανοποίησης των ασθενών, μπορεί να βελτιωθεί σημαντικά εντάσσοντας την τεχνολογία του υπολογιστικού νέφους συνδυαστικά με την ιδεά του διαδικτύου των πραγμάτων, στις ιατρικές διαδικασίες που χρησιμοποιούνται στον ιατρικό τομέα. Ασθένειες που εξελίσσονται με το χρόνο χρειάζονται μια μέθοδο που θα κάνει την παρακολούθηση της κατάστασης της υγείας του ασθενούς πιο εύκολη, άμεση και αποτελεσματική. Η χρήση φορητών αισθητήρων, σε συνδυασμό με τη δυνατότητα συνδεσιμότητάς τους στο διαδίκτυο παρέχει σε ιατρικό προσωπικό ή αργανισμούς παροχής υπηρεσιών υγείας, τα παραπάνω οφέλη ακόμα και για ασθενείς που βρίσκονται μακρυά από ιατρικές μονάδες ή ιατρικό προσωπικό, διευκολύνοντας τη ζωή των ανθρώπων και σε μερικές περιπτώσεις συνεισφέρουν στη μείωση του κόστους παροχής υπηρεσιών υγείας.

Σε αυτή την εργασία, επικεντρωνόμαστε κυρίως σε μεθόδους και τεχνολογίες διαχείρισης πληροφορίας που αφορούν ζωτικής σημασίας μετρήσεις, όπως ο καρδιακός παλμός και η ποσότητα οξυγόνου στο αίμα. Αποτελεί επέκταση της έξυπνης εφαρμογης «Intelligate», για κινητές συσκευές android που συλλέγει δεδομένα που παράγονται από αισθητήρες Bluetooth Low Energy (BLE) και αποστέλλονται μέσω διαδικτύου για επεξεργασία και αρχειοθέτηση στο Υπολογιστικό Νέφος. Το σύστημα αναπτύσει υπηρεσίες που προσφέρουν συνδεσιμότητα και επικοινωνία με χρήστες και φορητές συσκευές αλλά επιπλέον διαχειρίζονται την κατάσταση της υγείας των ασθενών όταν οι μετρήσεις ξεπερνούν προκαθορισμένα όρια και με βάση κανόνες αντίδρασης που έχουν οριστεί. Το σύστημά μας έχει υλοποιηθεί με βάση τα πρότυπα της υπηρεσιο-κεντρικής αρχιτεκτονικής με τη χρήση υπηρεσιών γενικού σκοπού μέσω του FIWARE και της υποδομής Νέφους Intellicloud. Τέλος, προσφέρει υπηρεσίες διαχείρισης χρηστών και των δικαιωμάτων πρόσβασης σε υπηρεσίες ή δεδομένα, αλλά και υπηρεσίες διαχείρισης συσκευών ή αισθητήρων. Η απόδοση του συστήματος μετρήθηκε πειραματικά με χρήση συνθετικών δεδομένων που προσομειώνουν τη χρήση του από εκατοντάδες χρήστες που στέλνουν δεδομένα ταυτόχρονα για επεξεργασία στο υπολογιστικό νέφος. Οι πειραματικές μετρήσεις υπόσχονται καλή απόδοση του συστήματος σε πραγματικό χρόνο και πραγματικές συνθήκες.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ.Πετράκη για τις πολύτιμες συμβουλές και την καθοδήγησή του καθ’ όλη την διάρκεια της διπλωματικής μου εργασίας. Επίσης όλα τα μέλη του εργαστηρίου προπτυχιακούς, μεταπτυχιακούς και συνεργάτες για τη βοήθεια που προσέφερε ο καθένας ξεχωριστά και οδήγησε στην ολοκλήρωση της συγκεκριμένης εργασίας. Τέλος, ευχαριστώ την οικογένεια μου για την αμέριστη συμπαράσταση και στήριξή τους σε ολόκληρη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Περιεχόμενα Σχημάτων

Περιεχόμενα Πινάκων

1 Εισαγωγή

1.1 Περιγραφή του γενικού Προβλήματος

Τα τελευταία χρόνια τα προβλήματα που απαντώνται στον τομέα της υγείας, όπως η έλλειψη προσωπικού, η αύξηση του κόστους παροχής υπηρεσιών, η μείωση του προϋπολογισμού υγείας αλλά και η αύξηση του προσδόκημου ζωής, καθιστούν ολοένα και πιο επιτακτική την ανάγκη δημιουργίας συστημάτων που διαχειρίζονται καταστάσεις υγείας απο μακρυά.

Συγκεκριμένα, ασθενείς με κινητικά προβλήματα, με χρόνιες παθήσεις και άτομα τρίτης ηλικίας, όπου η επαφή με τα εκάστοτε νοσοκομεία ή ιδιώτη γιατρό πρέπει να είναι άμεση , υποχρεωτική και παράλληλα τακτική, χρήζουν ανάγκης ενός τέτοιου συστήματος. Παραδειγματικά μιλώντας, ας υποθέσουμε ότι ένας ασθενής

πάσχει από ταχυκαρδία, θα πρέπει να υπάρχει μια τακτική συνάντηση με τον εκάστοτε επιμελητή γιατρό του, για τον έλεγχο και την πρόοδο της ασθένειας του, την καταγραφή του ιστορικού συμπτωμάτων του καθώς και την επεξεργασία των δεδομένων που παράγουν τα κατάλληλα όργανα μέτρησης εντός αλλά και εκτός του χώρου νοσηλείας του οποιαδήποτε χρονική στιγμή.

Γίνεται λοιπόν λόγος για απομακρυσμένη παρακολούθηση της υγείας των ασθενών (Remote Health Monitoring). Η γενική ιδεά του Remote Health Monitoring, στηρίζεται στο γεγονός ότι μέσω μιας συσκευής (gateway) με δυνατότητα σύνδεσης στο διαδίκτυο, είναι εφικτή η λήψη σημάτων διαφορετικών αισθητήρων, που μεταδίδονται με κάποιο πρωτόκολλο επικοινωνίας όπως είναι το Bluetooth low energy[1] (BLE), και η αποστολή των δεδομένων αυτών σε διαφορετικές υπηρεσίες ανεπτυγμένες στο υπολογιστικό νέφος (cloud computing) [1].

1.2 Περιγραφή της λύσης

Για την αντιμετώπιση των παραπάνω προβλημάτων δημιουργήθηκε η εργασία με τίτλο «Monitoring and Processing of vital data through mobile platform in the “cloud”» (Intelligate) [2], που στόχο είχε τη δημιουργία μιας εφαρμογής ανεπτυγμένης σε λογισμικό android[3], η οποία είχε τη δυνατότητα να λαμβάνει και να στέλνει σήματα που παράγουν διαφορετικοί αισθητήρες με έξυπνο τρόπο.

Για την επικοινωνία μεταξύ της συσκευής και της πύλης δικτύου επιλέχθηκε το πρωτόκολλο BLE, εξαιτίας των παρακάτω χαρακτηριστικών:

- Απλότητα: Η απλότητα αυτή σημαίνει χαμηλή κατανάλωση ενέργειας, αφού απαιτείται εκτέλεση λιγότερων εντολών, αλλά και χαμηλότερο κόστος.

- Οικονομία σε κατανάλωση ενέργειας: Οι συσκευές BLE λειτουργούν ακόμα και για χρόνια με μπαταρίες.

[1] https://en.wikipedia.org/wiki/Bluetooth_low_energy
[2] http://www.intelligence.tuc.gr/publications.php?pub_author=All&pub_type=8&pub_subject=All
[3] https://el.wikipedia.org/wiki/Android

- Ισχυρή ασφάλεια: Πλήρης κρυπτογράφηση AES - 128[4] χρησιμοποιώντας CCM[5] για την παροχή ισχυρής κρυπτογράφησης και έλεγχου ταυτότητας των πακέτων δεδομένων

Ωστόσο, χρειάζονταν ένα πλήρες σύστημα διαχείρισης των δεδομένων, των χρηστών και των αισθητήρων καθώς και η δημιουργία μιας ολοκληρωμένης διεπαφής από την πλευρά του ιατρού, γιαυτό το λόγο, δημιουργήθηκε η παρούσα εργασία. Ο ασθενής λοιπόν, χρειάζεται να έχει στη διάθεσή του ένα tablet με εγκατεστημένη την παραπάνω εφαρμογή και τους απαραίτητους αισθητήρες που μετρούν ζωτικής σημασίας δεδομένα (καρδιακούς παλμούς, οξυγόνο στο αίμα). Φορώντας τους αισθητήρες αυτούς θα μπορεί μέσω της κινητής συσκευής να χρησιμοποιεί την εφαρμογή όπου θα του επιτρέπει να παρακολουθεί ο ίδιος τις μετρήσεις του αλλά και ταυτόχρονα όλες αυτές οι μετρήσεις να μπορούν να στέλνονται και να επιβλέπονται από έναν ιατρό μέσω της δικής του εφαρμογής, από οποιαδήποτε τοποθεσία και αν βρίσκεται. Με τον τρόπο αυτό πετυχαίνουμε, απομακρυνσμένη παρακολούθηση και ενημέρωση, καθώς ο γιατρός βλέπει την πρόοδο του ασθενούς του την ίδια στιγμή που γίνεται η μέτρηση και ειδοποιείται εαν κάτι δεν πάει καλά.

Επιπλέον, η διαχείριση των χρηστών, των αισθητήρων και άλλων απαραίτητων λειτουργιών του συστήματος γίνονται με διάφορες υπηρεσίες που αναπτύχθηκαν στο υπολογιστικό νέφος, εκτενέστερη αναφορά του γίνεται στο κεφάλαιο 2.1. Πιο συγκεκριμένα, οι λειτουργίες που υποστηρίζονται από το υπολογιστικό νέφος είναι :

- Τα δικαιώματα πρόσβασης χρηστών: Για να αποκτήσει πρόσβαση στα δεδομένα και τις υπηρεσίες της εφαρμογής ο χρήστης πρέπει να πραγματοποιήσει εγγραφή στο σύστημα μέσω του Υπολογιστικού Νέφους και συγκεκριμένα μέσω της υποδομής Fiware Lab[6]. Ο διαχειριστής εγγράφει το νέο χρήστη στο σύστημα δίνοντάς του δικαίωμα πρόσβασης στις υπηρεσίες που δικαιούται ανάλογα με το ρόλο του στο σύστημα.

- Η διαχείριση χρηστών: Εισαγωγή-Επεξεργασία-Διαγραφή των χρηστών.

- Η διαχείριση αισθητήρων: Εισαγωγή-Επεξεργασία-Διαγραφή των ασιθητήρων.

[4] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[5] https://en.wikipedia.org/wiki/CCM_mode
[6] https://www.fiware.org/lab/

- Η πραγματοποίηση συνδρομών των χρήστων: Εγγραφή του χρήστη σε μια συσκευή που επιλέγει ώστε να παρακολουθεί τις αλλαγές των τιμών των αισθητήρων.

- Η αποθήκευση των δεδομένων: Βάση δεδομένων με το ιστορικό των μετρήσεων, τους κανόνες των ασθενών και όλα τα στοιχεία των χρηστών.

- Η διαχείριση γεγονότων: Όπως είναι για παράδειγμα, ο έλεγχος εαν βρίσκονται οι τιμές των ασθενών στα επιτρεπτά όρια που έχει ορίσει ο ιατρός του. Καθώς και η άμεση ενημέρωσή τους όταν κριθεί απαραίτητο.

Για την υλοποίηση της αρχιτεκτονικής του συστήματος βασιστήκαμε στην υπηρεσιο-κεντρική αρχιτεκτονική, αναφορά της οποίας γίνεται στο κεφάλαιο 2.5, και η οποία μας βοηθά να αντιμετωπίσουμε την πολυπλοκότητα του συστήματος διαιρώντας το σε μικρότερα κομμάτια. Αποτελείται από επαναχρησιμοποιήσιμες υπηρεσίες που δεν δεσμεύονται από το σύστημα και μπορούν να αντικατασταθούν άμεσα χωρίς σημαντικές αλλαγές στο μηχανισμό λειτουργίας της υπηρεσίας.

1.3 Πλεονεκτήματα εργασίας

Πλεονεκτήματα από την υλοποίηση της εργασίας, απαντώνται τόσο στην πλευρά των παρόχων υγειονομικής περίθαλψης όσο και στην πλευρά των ασθενών και είναι σημαντικα για διαφορετικούς λόγους στον καθένα:

Πλεονεκτήματα για τους Οργανισμούς Υγειονομικής Περίθαλψης (Νοσοκομεία-Γιατροί-Τεχνικοί διαχειριστές)

- Εξοικονόμιση χρημάτων για τα νοσοκομεία ή τους γιατρούς μιας και υπάρχει δυνατότητα εξυπηρέτησης περισσότερων ασθενών από οποιοδήποτε μέρος απλά και μόνο με τη χρήση του διαδικτύου.

- Περιορισμός γραφειοκρατίας καθώς η τεράστια λίστα των ασθενών στα νοσοκομεία αλλά και του ιστορικού τους θα γίνεται πλεον μέσω αποθήκευσής τους στο νέφος.

- Μείωση εξόδων των νοσοκομείων καθώς δεν είναι θα υπεύθυνα για την συντήρηση του συστήματος μιας και παρέχεται μέσω της τεχνολογίας νέφους όπου θα αναλυθεί αργότερα.

- Επίβλεψη σε πραγματικό χρόνο των μετρήσεων του ασθενούς από το γιατρό με την παράλληλη ειδοποίησή του εαν η κατάσταση του ασθενούς είναι κρίσιμη και την άμεση παρέμβασή του μέσω μηνύματος.

- Εισαγωγή οποιουδήποτε καινούργιου BLE αισθητήρα με την βοήθεια συμπλήρωσης φόρμας για το σχήμα του.

- Έλεγχος και παραμετροποίηση των κανόνων που διέπουν οι αισθητήρες για κάθε χρήστη, από το γιατρό.

Πλεονεκτήματα για τους Ασθενείς

- Ασφάλεια, μιας και η παρακολούθηση από το γιατρό τους είναι συνεχής άλλα και γιατί μπορούν και οι ίδιοι να παρακολουθήσουν την κατάστασή τους συνδέοντας απλά τους αισθητήρες με την εφαρμογή.

- Διαρκή επικοινωνία μέσω μηνυμάτων με τον φορέα περίθαλψής τους.

- Δυνατότητα χρήσης της εφαρμογής για χρονικό διάστημα που ο χρήστης επιθυμεί και κοστολόγηση σύμφωνα με αυτόν τον χρόνο.

Επιπλέον στα παραπάνω πλεονεκτήματα μπορούμε να προσθέσουμε και αυτά του Υπολογιστικού Νέφους, τα οποία περιγράφονται αναλυτικά στο κεφάλαιο 2.1.2, καθώς και τα πλεονεκτήματα της υπηρεσιο-κεντρικής αρχιτεκτονικής (κεφάλαιο 2.5).

1.4 Δομή της εργασίας

Στο **2ο κεφάλαιο** γίνεται εκτενέστερη αναφορά στην τεχνολογία Υπολογιστικού Νέφους. Δίνεται ο ορισμός του και αναλύονται τα πλεονεκτήματα και οι αδυναμίες της τεχνολογίας Νέφους. Επιπλέον, αναλύονται τα μοντέλα παροχής υπηρεσιών στο υπολογιστικό νέφος, τα μοντέλα εφαρμογών του και στο τι είναι υπηρεσίες γενικού και ειδικού σκοπού. Αναφερόμαστε στις τεχνολογίες νέφους που χρησιμοποιήθηκαν, ενώ στη συνέχεια, περιγράφεται η νέα τάση στο διαδίκτυο, το Διαδίκτυο των Πραγμάτων (Internet of Things), τι μας προσφέρει, πως συσχετίζεται άμεσα με το Υπολογιστικό Νέφος και τον τρόπο λειτουργίας του και ποιά παραδείγματα υλοποιήσεών του έχουν αρχίσει να εισέρχονται στην καθημερινότητά μας. Τέλος, γίνεται αναφορά στην Υπηρεσιοκεντρική Αρχιτεκτονική, στην οποία βασίζεται η αρχιτεκτονική του δικού μας συστήματος, με μια περιγραφή των βασικών χαρακτηριστικών της.

Στο **3ο κεφάλαιο ,** αρχικά παρουσιάζεται το σενάριο χρήσης το οποίο καλύπτεται από την εφαρμογή Intelligate και Back-End Intelligate. Γίνεται παρουσίαση των χρηστών του συστήματος και αναλύονται οι λειτουργικές απαιτήσεις των χρηστών αλλά και του συστήματος. Παρουσιάζονται τα διαγράμματα UML που μας βοήθησαν να υλοποιήσουμε τις παραπάνω λειτουργικές απαιτήσεις με βάση κάποιο σενάριο χρήσης του συστήματος.

Στο **4ο κεφάλαιο** , πλέον προχωράμε στην περιγραφή της υλοποίησης του συστήματος. Παρουσιάζεται το ειδικό διάγραμμα αρχιτεκτονικής του συστήματός μας και γίνεται περιγραφή των επιμέρους κομματιών της αρχιτεκτονικής σε επίπεδο διεπαφής χρήστη(Front end). Παρουσιάζεται το interface του ασθενή και το interface του γιατρού, καθώς και η διεπαφή του συστήματος(Back end) βάση της υπηρεσιο-κεντρικής αρχιτεκτονικής. Αναφέρεται αναλυτικά η υλοποίηση σε κάθε τομέα, με έμφαση στιι υπηρεσίες που χρησιμοποιήθηκαν. Τέλος, αναλύονται οι υπηρεσίες REST βάση των οποίων υλοποιήθηκαν οι υπηρεσίες του Υπολογσιτικού Νέφους.

Στο **5ο κεφάλαιο** , γίνεται η ανάλυση της απόδοσης του συστήματος. Αναφερόμαστε στα πειράματα που χρησιμοποιήσαμε για να μετρήσουμε την απόδοση αλλά και στα εργαλεία που μας βοήθησαν να βγάλουμε τα συμπεράσματα.

Τέλος, στο **6ο κεφάλαιο,** παραθέτουμε τα συμπεράσματα που προέκυψαν από την ενασχόληση μας με την υλοποίηση της εφαρμογής Back-end Intelligate καθώς και τις προτάσεις για μελλοντική δουλειά.

2 Υπόβαθρο και υπάρχουσες Τεχνολογίες

2.1 Τεχνολογία υπολογιστικού νέφους

Σε αυτο το κεφάλαιο θα ορίσουμε τι είναι το «υπολογιστικό νέφος», γιατί το προτιμούμε, ποιά ειναι τα πλεονεκτήμαυα και τα μειονεκτήματά του και από ποια μοντέλα παροχής υπηρεσιών αποτελείται με αναφορά σε ορισμένα παραδείγματα. Επίσης, θα αναλύσουμε και τα μοντέλα εφαρμογών του.

2.1.1 Ορισμός

Το υπολογιστικό νέφος είναι ένα μοντέλο που επιτρέπει εύκολη, ευέλικτη, κατά απαίτηση «on-demand» και απεριόριστη δικτυακή πρόσβαση σε μια συλλογή παραμετροποιήσιμων υπολογιστικών πόρων (δίκτυο, διακομιστές, αποθήκευση, εφαρμογές και υπηρεσίες). Η αποθήκευση , η επεξεργασία και η χρήση δεδομένων λογισμικού και υπηρεσιών γίνεται διαδικτυακά, μέσω απομακρυσμένων υπολογιστών και ο χρήστης δε χρειάζεται να γνωρίζει που βρίσκονται. Υπηρεσίες όπως η κατ' αίτηση παροχή εικονικών μηχανών, το διαδικτυακό ηλεκτρονικό ταχυδρομείο ή τα κοινωνικά δίκτυα βασίζονται στην τεχνολογία του. Αυτό το μοντέλο αποτελείται από τρία μοντέλα παροχής υπηρεσιών και τέσσερα μοντέλα ανάπτυξης [2].

2.1.2 Πλεονεκτήματα και Μειονεκτήματα του Υπολογιστικού Νέφους (cloud computing)

Τα πλεονεκτήματά του είναι:

- Οικονομία: Αυτό είναι από τα πιο βασικά πλεονεκτήματα του cloud computing. Το κόστος που μπορεί να έχει ένα λογισμικό ίσως να είναι απαγορευτικό για μία μικρή εταιρία. Με το «cloud» τα δεδομένα αυτά αλλάζουν καθώς η εταιρία δεν πληρώνει την εφαρμογή αλλά πληρώνει την χρήση της. Συνήθως σε cloud δίκτυα υπάρχουν πολλές δυνατότητες και «πακέτα» για την πληρωμή της χρήσης κάποιας εφαρμογής.

- Μεγάλος Αποθηκευτικός Χώρος**:** Η αποθήκευση των διαφόρων πληροφοριών είναι θέμα υψίστης σημασίας. Με το cloud computing έχουμε συνήθως όσο αποθηκευτικό χώρο θα χρειαστούμε.

- Πρόσβαση από οποιαδήποτε συσκευή διαθέτει σύνδεση στο ίντερνετ.

- Πολύ μεγάλη ευελιξία.

Τα μειονεκτήματά του είναι:

- Ασφάλεια δεδομένων: Είναι λογικό κάποιες φορές να έχουμε συγκεκριμένα δεδομένα σε δικό μας τοπικό εξυπηρετητή (server) και όχι στο νέφος (cloud).

- Αυξημένη πολυπλοκότητα: Αυτό συμβαίνει όταν έχουμε μία εφαρμογή αποθηκευμένη κάπου τοπικά, σε ένα δικό μας εξυπηρετητή διαδικτύου (webserver) και προσπαθούμε να την κάνουμε να επικοινωνήσει με μία άλλη στο νέφος (cloud) [3].

2.1.3 Μοντέλα παροχής υπηρεσιών νέφους

- **Υποδομή ως Υπηρεσία - Infrastructure as a service (IaaS)**

Αυτή η μορφή Νέφους δίνει πρόσβαση σε εικονικές πλατφόρμες υποδομών (hardware), που περιλαμβάνουν τα μηχανήματα, το δίκτυο και τα αποθηκευτικά μέσα. Με αυτή τη μορφή, οι αρχιτέκτονες διαδικτύου μπορούν να δημιουργήσουν τη δική τους Υπολογιστική Υποδομή πάνω στην οποία είναι οι ίδιοι υπεύθυνοι στο να εγκαταστήσουν, να διατηρήσουν και να τις παρέχουν σε άλλους εξαρτώμενους χρήστες. Η υποδομή είναι το χαμηλότερο επίπεδο και είναι ένα μέσο για να παρέχεται η επεξεργασία, η αποθήκευση, το δίκτυο και άλλοι βασικοί υπολογιστικοί πόροι σαν δεδομένες υπηρεσίες δια μέσω του διαδικτύου. Οι εξυπηρετητές, τα αποθηκευτικά συστήματα, οι δικτυακοί διακόπτες (network

switches), οι δρομολογητές (routers) και άλλα συστήματα, χειρίζονται διάφορους τύπους φόρτου εργασίας, από μια σειρά προγραμμάτων που εκτελούνται χωρίς παρέμβαση χρήστη μέχρι και την αποθήκευση στον εξυπηρετητή. Οι πάροχοι των υπηρεσιών Νέφους μπορούν να έχουν τα δικά τους λειτουργικά συστήματα και λογισμικό για την υποκείμενη υπολογιστική υποδομή που χρησιμοποιούν στο Νέφος. Παραδείγματα αυτού του μοντέλου: Amazon EC2[7], Rackspace[8] κ.α.

- **Λογισμικό ως Υπηρεσία - Software as a service (SaaS)**

Σε αυτό τον τύπο υπάρχει μια εφαρμογή η οποίο βρίσκεται σε ένα cloud server και ο χρήστης μπορεί να έχει πρόσβαση σε αυτό μέσω μίας απλή σύνδεσης στο ίντερνετ. Το λογισμικό αυτό ανήκει σε κάποιον κατασκευή και ο χρήστης το πληρώνει ανάλογα με τη χρήση που του κάνει και τους πόρους που χρειάζεται. Το βασικό πλεονέκτημα του μοντέλου «SaaS» είναι ότι ο κατασκευαστής αναλαμβάνει τα έξοδα συντήρησης του λογισμικού καθώς και τη φιλοξενία του σε κάποιον cloud server. Ο χρήστης πληρώνει μόνο την χρήση που κάνει. Επίσης το μοντέλο SaaS είναι δημιουργημένο με βασικό γνώμονα τη σωστή λειτουργία του λογισμικού με χρήση ενός προγράμματος περιήγησης. Όσον αφορά την ασφάλεια των διαφόρων εφαρμογών, συνήθως χρησιμοποιείται SSL (Secure Sockets Layer) το οποίο είναι παγκοσμίως αναγνωρισμένο. Έτσι, οι χρήστες μπορούν με ασφάλεια να χρησιμοποιήσουν την cloud εφαρμογή. Παραδείγματα αυτού του μοντέλου: Google App Engine[9], Windows Azure[10].

- **Πλατφόρμα ως Υπηρεσία - Platform as a service (PaaS)**

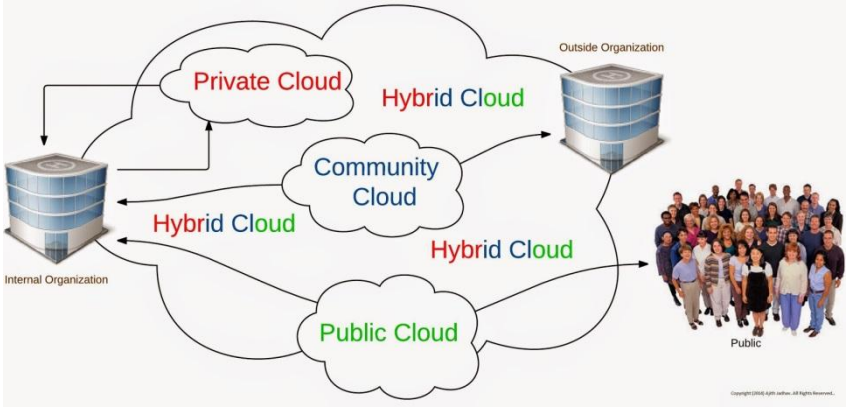
Αυτό το μοντέλο μοιάζει πολύ με το προηγούμενο. Το βασικό του στοιχείο είναι ότι παρέχει την πλατφόρμα την οποία χρησιμοποιεί ένας χρήστης για να

[7] http://aws.amazon.com/ec2/
[8] http://www.rackspace.com/managed-cloud/
[9] https://en.wikipedia.org/wiki/Google_App_Engine
[10] https://en.wikipedia.org/wiki/Microsoft_Azure

δημιουργήσει κάτι, για παράδειγμα μια εφαρμογή web, χωρίς να εγκαταστήσει τίποτα. Το «platform as service» μοντέλο, χρησιμοποιείται πιο πολύ για δημιουργία Διεπαφών στο Διαδίκτυο, Εφαρμογών κλπ. Ένα σημαντικό πρόβλημα που υπάρχει με αυτό το μοντέλο είναι ότι αυτή η εφαρμογή που δημιουργούμε βασίζεται σε μια συγκεκριμένη δομή και υπάρχει πιθανότητα αν θελήσουμε να τη μεταφέρουμε σε άλλον παροχέα cloud υπηρεσιών να μη λειτουργεί σωστά. Παραδείγματα αυτού του μοντέλου: Google Apps, Yahoo Mail, Dropbox κ.α.

2.1.4 Βασικά μοντέλα εφαρμογών του Cloud Computing

Τα τέσσερα μοντέλα του υπολογιστικού νέφους είναι το Δημόσιο Νέφος (Public Cloud), το Ιδιωτικό Νέφος (Private Cloud), το Κοινοτικό Νέφος (Community Cloud) και το Υβριδικό Νέφος (Hybrid Cloud), όπως φαίνεται και στο Σχήμα 1, και επικοινωνεί μέσω του διαδικτύου και ανάλογα με τον τύπο του, είτε με οργανισμούς είτε με χρήστες.



Σχήμα 1- Μοντέλα Ανάπτυξης Υπολογιστικού Νέφους[11]

- **Δημόσιο νέφος (Public Cloud)**

Αυτό το μοντέλο δημιουργείται από εκατοντάδες web servers που τρέχουν και πάρα πολλά κέντρα δεδομένων (datacenters) σε διάφορα σημεία του πλανήτη. Αυτό έχει ως αποτέλεσμα να μπορεί κάποιος να χρησιμοποιήσει μία υπηρεσία διαλέγοντας την τοποθεσία που θα βρίσκεται η εφαρμογή. Κοινώς διαλέγει το datacenter που είναι πιο κοντά του. Για παράδειγμα μία εταιρία στην Αμερική θα διαλέξει ένα cloud server που βρίσκεται στην Ν. Αμερική. Εταιρίες που προσφέρουν

[11] http://www.webopedia.com/TERM/C/CDN.html

το public cloud είναι οι: Google, Amazon, Rackspace κλπ. Αυτή η public εφαρμογή του cloud υποστηρίζεται από εταιρίες πολύ εύρωστες οικονομικά διότι η ανάπτυξη και συντήρηση των webserver και datacenter παγκοσμίως κοστίζει πολλά χρήματα. Μία εφαρμογή που χρησιμοποιεί το «public cloud» είναι τα CDN[12] (content delivery networks) μέσα από τα οποία το περιεχόμενο ενός site αποθηκεύεται σε κάποια datacenters παγκοσμίως και τα προσφέρει στους χρήστες της ιστοσελίδας όταν τα ζητάνε με πολύ μεγάλες ταχύτητες.

Το δημόσιο νέφος είναι μια ιδανική επιλογή όταν:

- Υπάρχει απασχόληση με τυποποιημένο φόρτο εργασίας για τις εφαρμογές που χρησιμοποιούνται από πολλούς ανθρώπους, όπως το ηλεκτρονικό ταχυδρομείο.
- Όταν θέλουμε να παράξουμε κώδικα για το στήσιμο και την δημιουργία μιας εφαρμογής με την βοήθεια υπηρεσιών που μας προσφέρει το δημόσιο νέφος.
- Χρήση επιπλέον υπολογιστικών πόρων σε ώρες αιχμής.
- Όταν θέλουμε να κάνουμε εργασίες συνεργασίας με άλλες εταιρίες.

- **Ιδιωτικό νέφος (Private Cloud)**

Αυτό το είδος της cloud τεχνολογίας εφαρμόζεται μέσα σε οργανισμούς-εταιρίες όπου δημιουργείται ένα cloud δίκτυο το οποίο όμως βρίσκεται στα όρια του οργανισμού αυτού. Το δίκτυο αυτό δημιουργείται κατά παραγγελία με βάση τις ανάγκες του οργανισμού.

Το ιδιωτικό νέφος είναι μια ιδανική επιλογή όταν:

- Το κέντρο δεδομένων(datacenter) θα πρέπει να γίνει πιο αποτελεσματικό σε πόρους (μνήμη, επεξεργαστική ισχύ κτλ).
- Είναι αναγκαία η παροχή ιδιωτικών υπηρεσιών νέφους.
- Είναι χρήσιμη η συνοχή σε όλες τις υπηρεσίες που προσφέρονται από τον οργανισμό.

- **Κοινοτικό Νέφος (Community Cloud)**

Η υποδομή Κοινοτικού Νέφους (Community Cloud) είναι διαμοιρασμένη σε πολλούς οργανισμούς και εξυπηρετεί μια συγκεκριμένη κοινότητα η οποία έχει σαν κοινό τόπο κάποιο ενδιαφέρον ή στόχο. Αυτή την

[12] http://integrationtechnologytrends.blogspot.gr/2014/10/cloudy-cloud-pay-as-you-go-upscale-as.html

υποδομή μπορεί να διαχειρίζεται κάποιος από τους αναφερθέντες οργανισμούς, ή από έναν τρίτο οργανισμό ή επιχείρηση.

Το κοινοτικό νέφος είναι μια ιδανική επιλογή όταν:

- Κυβερνητικοί οργανισμοί χρειάζονται να διαμοιράζονται πόρους.
- Είναι αναγκαία η δημιουργία ενός κοινού συστήματος πρόσβασης σε προφίλ πελατών και εργαζομένων για μια ομάδα νοσοκομείων ή κλινικών.

- **Υβριδικό Νέφος (Hybrid Cloud)**

Το Υβριδικό Νέφος είναι το μοντέλο που συνδυάζει όλα τα παραπάνω Νέφη, ώστε να δημιουργηθεί μια υποδομή. Η χρήση αυτού του μοντέλου παρέχει σε ορισμένες περιπτώσεις μεγαλύτερη ευελιξία σε επιχειρήσεις, δεδομένου ότι είναι εφικτή μια ποικιλία συνδυασμών των πόρων. Οι συνδυασμοί αυτοί επιτρέπουν την αποτελεσματικότερη αντιμετώπιση φόρτου εργασίας και μεγάλων απαιτήσεων υπολογιστικών πόρων. Μια ιδανική επιλογή για το υβριδικό νέφος είναι το γεγονός μιας εταιρίας να θέλει να χρησιμοποιήσει μια εφαρμογή SaaS αλλά ανησυχεί για την ασφάλεια που της παρέχει [4].

2.2 Τεχνολογίες νέφους

Εδώ γίνεται αναφορά στις υπάρχουσες τεχνολογίες νέφους και στο τι μας παρέχει η καθεμιά, αλλά και στις υπηρεσίες κοινού σκοπού που διανέμονται μέσω διαδικτύου.

2.2.1 *Fiware*

Το FIWARE είναι μια μη εμπορική πλατφόρμα που παρέχει ένα απλό αλλά ισχυρό σύνολο από APIs (Application Programming Interfaces) με σκοπό τη διευκόλυνση και την ανάπτυξη έξυπνων εφαρμογών. Έτσι, δίνεται η δυνατότητα στον χρήστη να αποκτήσει την κατάλληλη τεχνογνωσία ώστε να εξοικειωθεί με τις υπηρεσίες, εφόσον παρέχονται πειράματα και αποτελέσματα αυτών των υπηρεσιών από χρήστες που τις έχουν ήδη χρησιμοποιήσει. Οι υπηρεσίες που

διανέμονται είναι δωρεάν και δημόσιες ώστε το κοινό να έχει πρόσβαση σε αυτές. Γεγονός που το κάνει ακόμα πιο ελκυστικό στους χρήστες που θέλουν να δημιουργήσουν τις δικές τους εφαρμογές. Οι παροχές του FIWARE δεν σταματάνε στην διανομή υπηρεσιών με την μορφή λογισμικού αλλά δίνει την δυνατότητα σε προγραμματιστές να αποκτήσουν υπηρεσίες ως υποδομή (IaaS) δημιουργώντας εικονικές μηχανές και δεσμεύοντας υπολογιστικούς πόρους (μνήμη, επεξεργαστική ισχύ, αποθηκευτικό χώρο) [5][6].

2.2.1.1 Fiware Enablers

Οι Generic Enablers είναι υπηρεσίες κοινού σκοπού οι οποίες διανέμονται μέσω διαδικτύου με τη μορφή SaaS από παρόχους Υπολογιστικού Νέφους για την δημιουργία εφαρμογών. Η αρχιτεκτονική στην οποία υπακούουν τέτοιου είδους υπηρεσίες και συμβαδίζουν με τους κανόνες της, είναι η REST και ακολουθείται από ένα API το οποίο επιτρέπει την πρόσβαση σε αυτές. Ο FIWARE κατάλογος περιέχει μια πλούσια βιβλιοθήκη από Enablers (Generic Enablers). Διαθέτει παραδείγματα υλοποιήσεων που επιτρέπουν στους προγραμματιστές να συνδεθούν με το Ίντερνετ των πραγμάτων (IoT), κάνοντας τον προγραμματισμό πολύ ευκολότερο. Όλοι τους είναι δημόσιοι , ανοικτού κώδικα και απλοί στη χρήση τους [7].

Μερικοί από αυτούς είναι:

- **Identity Management – KeyRock**

  Ο keyrock καλύπτει μια σειρά από ζητήματα που αφορούν την πρόσβαση των χρηστών στα δίκτυα, υπηρεσίες και εφαρμογές. Στόχος του είναι η δημιουργία λογαριασμού για συσκευές, δίκτυα και υπηρεσίες, συμπεριλαμβανομένης της ασφαλούς πρόσβασης και της διατήρησης των προσωπικών στοιχείων των χρηστών.

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

  Ο Publish / Subscribe Context Broker GE, παρέχει τις διασυνδέσεις NGSI9 και NGSI10. Χρησιμοποιώντας τες, οι πελάτες μπορούν να κάνουν διάφορες εργασίες όπως η εγγραφή σε διάφορες οντότητες, π.χ. εγγραφή σε έναν αισθητήρα θερμοκρασίας μέσα σε ένα δωμάτιο, με ενημέρωση για τις αλλαγές στη θερμοκρασία ή αποστολή θερμοκρασίας κάθε λεπτό.

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

Ο CEP GE αναλύει δεδομένα συμβάντων σε πραγματικό χρόνο και αντιδρά σε καταστάσεις και όχι σε μεμονωμένα γεγονότα. Μια κατάσταση που βασίζεται σε μια σειρά από γεγονότα που έχουν συμβεί μέσα σε πλαίσιο επεξεργασίας. Οι καταστάσεις αυτές συνήθως περιλαμβάνουν σύνθετα γεγονότα.

Η επικοινωνία και η σύνδεση παραπάνω του ενός Generic Enabler δημιουργεί έναν Specific Enabler που έχει ως στόχο την υλοποίηση μιας πιο σύνθετης λειτουργίας.

2.2.2 Intellicloud

Η υποδομή Υπολογιστικού Νέφους Intellicloud, σχεδιάστηκε, υλοποιήθηκε και συντηρείται από το εργαστήριο Ευφυών Συστημάτων. Σκοπός της υποδομής αυτής είναι η παροχή υπολογιστικών πόρων, για την ανάπτυξη εφαρμογών στο Νέφος. Η υποδομή φιλοξενείται εξολοκλήρου στο Πολυτεχνείο Κρήτης και αυτή τη στιγμή περιλαμβάνει 128 πυρήνες επεξεργαστικής ισχύς, 284 GB RAM και 12 TB σε σκληρό δίσκο. Το λογισμικό που είναι υπεύθυνο για τη λειτουργία του Intellicloud είναι βασισμένο σε Openstack Grizzly.

2.3 Διαδίκτυο των Πραγμάτων –Υπάρχουσες Υλοποιήσεις

Γίνεται αναφορά στο τι είναι το Διαδίκτυο των Πραγμάτων, γιατί το χρησιμοποιούμε, ποιές υλοποιήσεις συναντάμε που βασίζονται στο IoT, με ορισμένα παραδείγματα. Επίσης, επισημαίνεται η σχέση του με το υπολογιστικό νέφος, το οποίο αναλύθηκε παραπάνω.

2.3.1 Διαδικτύο των πραγμάτων - Internet of things (IoT)

Η ιδέα πίσω από το Internet of Things, είναι η σύνδεση όλων των ηλεκτρονικών συσκευών μεταξύ τους ή/και με το Internet. Το Internet όπως το γνωρίζουμε αυτή τη στιγμή αποτελεί τη ραχοκοκαλιά του Internet of Things, ωστόσο δεν είναι απαραίτητο οι συσκευές να έχουν απευθείας πρόσβαση σε αυτό. Για

παράδειγμα, ένα fitness band συλλέγει αμέτρητα δεδομένα για τη φυσική κατάσταση και την υγεία κάποιου, τα μεταδίδει σε μια έξυπνη συσκευή (smartphone ή tablet) μέσω Bluetooth και στη συνέχει αυτά περνάνε online, στην cloud υπηρεσία που χρησιμοποιεί για την καταγραφή τους. Πρακτικά, δηλαδή, μιλάμε για ένα περιβάλλον συλλογής δεδομένων από οποιαδήποτε ηλεκτρονική συσκευή ή μικροσκοπικό αισθητήρα υπάρχει γύρω μας. Το Ίντερνετ των πραγμάτων επιτρέπει αντικείμενα να ελέγχονται εξ αποστάσεως σε όλη την υπάρχουσα υποδομή του δικτύου, αλλά και τη δημιουργία ευκαιριών για πιο άμεση ενσωμάτωση του φυσικού κόσμου σε συστήματα που βασίζονται σε υπολογιστές. Στόχος είναι η βελτίωση της ποιότητας ζωής των ανθρώπων και η αποφυγή απρόβλεπτων γεγονότων.

2.3.2 Υλοποιήσεις στο Διαδίκτυο των πραγμάτων

Οι εταιρίες έχουν ήδη αρχίσει να μετακινούνται προς την ιδέα του Διαδικτύου των πραγμάτων και να επενδύουν τεράστια χρηματικά ποσά με στόχο την εξέλιξη σημαντικών τομέων που αποτελούν την καθημερινότητα των ανθρώπων.

LoRaWAN: Είναι ένα χαμηλής ισχύος δικτύο ευρείας περιοχής με χαρακτηριστικά που υποστηρίζουν χαμηλού κόστους, κινητά, ώστε να εξασφαλίσει την αμφίδρομη επικοινωνία για Ίντερνετ των πραγμάτων (IoT), machine-to-machine (M2M), έξυπνη πόλη, και βιομηχανικές εφαρμογές. Το LoRaWAN αποτελεί τον προάγγελο για την ανάπτυξη αυτής της τεχνολογίας και έχει πολλές διαφορετικές κατηγορίες συσκευών για την αντιμετώπιση των διαφόρων αναγκών με ένα ευρύ φάσμα εφαρμογών.

Εφαρμογές του Ίντερνετ των πραγμάτων:

Έξυπνο Σπίτι (Smart Home): Το έξυπνο σπίτι είναι πιθανόν η πιο δημοφιλής εφαρμογή IoT αυτή τη στιγμή, διότι είναι η εφαρμογή που είναι πιο προσιτή και εύκολα διαθέσιμη στους καταναλωτές. Για παράδειγμα, ας φανταστούμε ένα σπίτι με συνδεδεμένες συσκευές που όταν το πρωί χτυπάει το ξυπνητήρι του smartphone μας, θα ανάβει αυτόματα το φως στο υπνοδωμάτιο και, ταυτόχρονα, θα ενεργοποιείται η καφετιέρα για να ετοιμάσει ένα ζεστό espresso.

Φορητές Συσκευές (Wearables): Τα Ρολόγια δεν είναι πλέον μόνο για να λένε την ώρα. Η Apple Watch και άλλες smartwatches στην αγορά έχουν μετατρέψει τους καρπούς μας σε smartphone θήκες επιτρέποντας την ανταλλαγή μηνυμάτων κειμένου, τηλεφωνήματα, και πολλά άλλα με μία μόνο κίνηση. Και συσκευές όπως Fitbit και Jawbone βοήθησαν να έρθει η επανάσταση στον κόσμο του γυμναστηρίου, δίνοντας στους ανθρώπους περισσότερα στοιχεία σχετικά με τις προπονήσεις τους.

Έξυπνες Πόλεις (Smart Cities): Το Ίντερνετ των πραγμάτων έχει τη δυνατότητα να μετατρέψει ολόκληρες πόλεις, με την επίλυση πραγματικών προβλημάτων που αντιμετωπίζουν οι πολίτες κάθε μέρα. Με τις κατάλληλες συνδέσεις και τα στοιχεία, το Ίντερνετ των πραγμάτων μπορεί να λύσει τα προβλήματα της κυκλοφοριακής συμφόρησης και τη μείωση του θορύβου, της εγκληματικότητας και της ρύπανσης.

Συνδεδεμένα Αυτοκίνητα (Connected Cars): Το Internet of Things πηγαίνει πολύ πιο πέρα από τους «αυτοματισμούς» ενός σπιτιού. Συνδεδεμένα αυτοκίνητα μπορούν να «επικοινωνούν» μεταξύ τους και να κρατάνε αποστάσεις ασφαλείας, καθώς και να ενημερώνουν την τροχαία και τους άλλους οδηγούς για την κίνηση στους δρόμους. Συνδεδεμένα μέσα μαζικής μεταφοράς επιτρέπουν να γνωρίζουμε το πότε ακριβώς θα βρίσκονται στη στάση.

Έξυπνοι Αισθητήρες (Smart Sensors): Έξυπνα πιεσόμετρα ή συσκευές μέτρησης των επιπέδων της γλυκόζης στο αίμα και αισθητήρες γενικοτερα, θα μπορούν να στέλνουν τις μετρήσεις απευθείας στον γιατρό για να έχει μια καλύτερη εικόνα της κατάστασης της υγείας του ασθενούς, με αναλυτικές μετρήσεις σε καθημερινή βάση [8].

2.3.3 Διαδίκτυο των Πραγμάτων (IoT) και Υπολογιστικό Νέφος (Cloud Computing)

Το IoT δημιουργεί μια άνευ προηγουμένου ποσότητα δεδομένων, το οποίο με τη σειρά του βάζει μια τεράστια πίεση στην υποδομή του Διαδικτύου. Ως αποτέλεσμα, οι εταιρείες να ψάχνουν να βρουν τρόπους για να ανακουφιστεί αυτή η πίεση και να λυθεί το πρόβλημα των δεδομένων.

Το cloud computing βοηθά σημαντικά στο πρόβλημα αυτό καθιστώντας δυνατή τη συνεργασία όλων των συνδεδεμένων συσκευών. Το Ίντερνετ των πραγμάτων και το cloud computing είναι διαφορετικά, αλλά ο καθένας θα έχει τη δική του δουλειά στην αντιμετώπιση αυτού του νέου κόσμου των δεδομένων.

Το IoT παράγει τεράστιες ποσότητες δεδομένων και το cloud computing παρέχει ένα μονοπάτι για να ταξιδέψουν τα δεδομένα στον προορισμό τους. Τα οφέλη που παρουσιάζονται από το Cloud Computing στο IoT, ωθούν μεγάλους πωλητές cloud, όπως η Amazon και η Google να έχουν αρχίσει να προσφέρουν τις υπηρεσίες cloud με στόχο τη στήριξη συσκευών και εφαρμογών IoT από την άποψη των δυνατοτήτων της πληροφορικής, της ανάλυσης δεδομένων, της ελαστικότητας των πόρων, και της επεκτασιμότητας που αυτό προσφέρει.

2.4 Υπηρεσιοκεντρική Αρχιτεκτονική

Πριν αναφερθούμε στον ορισμό της Υπηρεσιοκεντρικής Αρχιτεκτονικής και στα πλεονεκτήματά της, θα κάνουμε μια αναφορά στο τι είναι υπηρεσίες και γιατί τις προτιμούμε.

Ως υπηρεσίες ορίζονται διάφορα συστήματα λογισμικού που είναι σχεδιασμένα για να υποστηρίζουν διαλειτουργική αλληλεπίδραση υπολογιστών μέσω δικτύου. Ο κύριος σκοπός τους είναι να παρέχουν λειτουργικότητα από τον ιδιοκτήτη ή πάροχο της υπηρεσίας προς άλλες εφαρμογές. Υπάρχουν πολλά πλεονεκτήματα από τη χρήση τέτοιου είδους υπηρεσιών. Το κύριο πλεονέκτημα είναι η διαλειτουργικότητα στην επικοινωνία μεταξύ των διαφόρων μηχανημάτων/εφαρμογών. Δύο ακόμη πλεονεκτήματα των υπηρεσιών είναι η επαναχρησιμοποίηση λογισμικού και η διασύνδεση υπάρχοντων συστημάτων.

Η Υπηρεσιοκεντρική Αρχιτεκτονική (Service Oriented Architecture) είναι ένα είδος αρχιτεκτονικής που έχει ως στόχο την δημιουργία λογισμικού μέσω υπηρεσιών που είναι διαθέσιμες σε κάποιο δίκτυο όπως ο παγκόσμιος ιστός (web). Η βασική φιλοσοφία αυτής της αρχιτεκτονικής είναι η χαλαρή σύζευξη μεταξύ των τμημάτων λογισμικού έτσι ώστε να μπορούν να επαναχρησιμοποιηθούν. Η Υπηρεσιοκεντρική Αρχιτεκτονική υιοθετεί το προτέρημα της επαναχρησιμοποίησης μιας υπηρεσίας ιστού που μπορεί να δημιουργηθεί από μια υπάρχουσα υποδομή ενός πληροφοριακού συστήματος. Επιτρέπει δηλαδή στις επιχειρήσεις και τους οργανισμούς να επαναχρησιμοποιήσουν υπάρχουσες εφαρμογές διαφορετικών τεχνολογιών και ετερογενών εφαρμογών.

Η λογική της υπηρεσιο-κεντρικής αρχιτεκτονικής προβλέπει διαφορετικούς παρόχους υπηρεσιών Νέφους να αναπτύσσουν εφαρμογές και υπηρεσίες, ο καθένας ακολουθώντας τη δικιά του υλοποίηση και τα δικά του μέσα (λειτουργικό σύστημα, γλώσσα προγραμματισμού, φυσικούς πόρους κλπ.) και οι υπηρεσίες αυτές να προσφέρονται μέσω διαδικτύου σε οργανισμούς που θέλουν να αναπτύξουν πιο σύνθετα υπολογιστικά συστήματα, ικανοποιώντας τις ανάγκες τους από τις ήδη υπάρχουσες υπηρεσίες που διατίθενται.

Η υπηρεσιοκεντρική αρχιτεκτονική είναι ιδιαίτερα χρήσιμη και ορισμένα από τα πλεονεκτήματα που την κάνουν γνωστή και ελκυστική είναι:

- Οι υπηρεσίες είναι επαναχρησιμοποιήσιμες και μπορούν να διατεθούν σε μεγάλη κλίμακα.
- Γρηγορότερη και αποτελεσματικότερη αποσφαλμάτωση.
- Μικρότερος χρόνος διάθεσης νέων προϊόντων και εφαρμογών.
- Οι υπηρεσίες αυτές δεν δεσμεύονται από το σύστημα, αλλά είναι δυνατή η αντικατάστασή τους.
- Σε ένα υπάρχον σύστημα η ένταξη και η ενσωμάτωση μιας νέας υπηρεσίας δεν απαιτεί αλλαγές στον μηχανισμό λειτουργίας της υπηρεσίας [11].

2.5 Αρχιτεκτονική REST

Το REST (Representational State Transfer) αποτελεί ένα σύνολο από αρχές σχεδίασης μιας δικτυακής υπηρεσίας που επικεντρώνει στους πόρους (π.χ δεδομένα) ενός συστήματος. Το REST πρωτοεμφανίστηκε το 2000 από τον Roy Fielding στην ακαδημαϊκή του διατριβή με τίτλο «Architectural Styles and the Design of Network-based Software Architectures»[13].

Τα βασικά χαρακτηριστικά του REST είναι τα εξής:

- **Αποκλειστική χρήση HTTP αιτημάτων/μεθόδων για την επικοινωνία του χρήστη με τον παροχέα της δικτυακής υπηρεσίας, όπως φαίνεται και στο Σχήμα 2.**

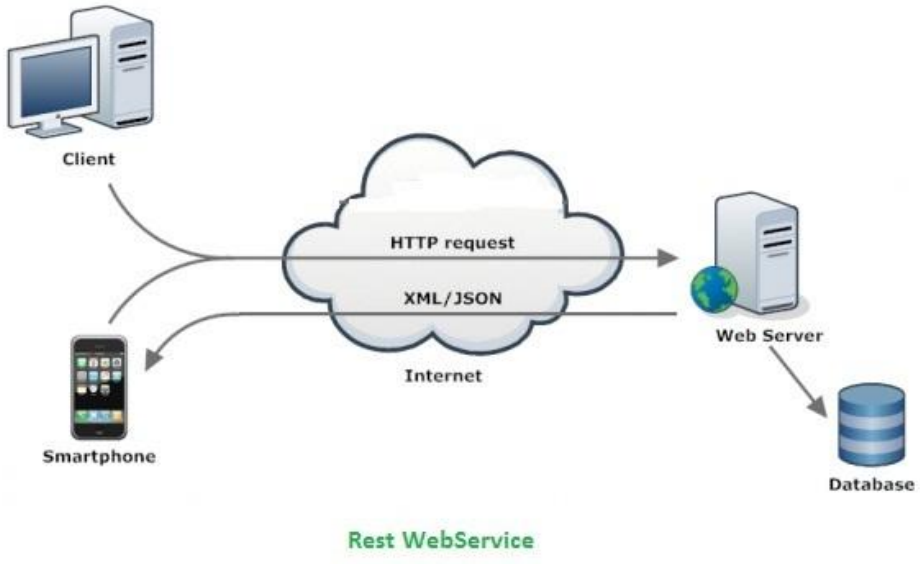
Η βασική αρχή σχεδίασης του REST είναι η ένα-προς-ένα αντιστοίχιση μεταξύ λειτουργιών CRUD (create, read, update, delete) και HTTP μεθόδων. Σύμφωνα με αυτή την αντιστοίχιση:

  - Για τη δημιουργία ενός πόρου στον server, χρησιμοπούμε την μέθοδο POST.
  - Για την ανάσυρση ενός πόρου, χρησιμοποιούμε την GET.
  - Για την αλλαγή της κατάστασης ενός πόρου ή την ενημέρωσή του, χρησιμοποιούμε την PUT.
  - Για την απομάκρυνση ή διαγραφή ενός πόρου, χρησιμοπουούμε την DELETE.

[13] https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Με βάση το REST το URI δεν χρησιμοποιείται πια για την περιγραφή της ενέργειας που θέλουμε να εκτελέσουμε αλλά μόνο για τον εντοπισμό του πόρου επί του οποίου θα ασκηθεί η ενέργεια. Σε μια υπηρεσία REST, ένα URI εκφράζει ένα αντικείμενο στο οποίο παρέχει πρόσβαση η υπηρεσία μέσω ενός HTTP αιτήματος. Το είδος του αιτήματος καθορίζει την ενέργεια που θέλουμε να εφαρμόσουμε στο αντικείμενο αυτό και το περιεχόμενο του αιτήματος περιέχει διάφορες εξειδικεύσεις τις ενέργειας. Τα δεδομένα του αιτήματος μεταφέρονται από τον πάροχο της υπηρεσίας προς τον καταναλωτή με τη μορφή XML[14] ή JSON[15].

Οπότε ο σχεδιασμός μιας υπηρεσίας REST περιλαμβάνει κυρίως δύο βήματα. Πρώτον, να αποφασίσουμε τι αντικείμενα θα διαθέτουμε δια μέσω της υπηρεσίας και δεύτερον, τι ενέργεια θα εφαρμόζεται στο κάθε αντικείμενο για κάθε μία από τις GET, POST, PUT και DELETE μεθόδους.



Σχήμα 2- RestFul WebServices[16]

- **Απουσία μνήμης κατάστασης (stateless)**

Η ουσία της «έλλειψης κατάστασης» είναι ότι οποιαδήποτε κλήση σε μια υπηρεσία REST δε θα πρέπει να αναφέρεται σε άλλη προγενέστερη κλήση. Όλες οι

[14] https://www.w3schools.com/xml

[15] https://www.tutorialspoint.com/json/json_quick_guide.htm

[16] http://phpflow.com/php/web-service-types-soapxml-rpcrestful/

κλήσεις πρέπει να είναι ανεξάρτητες. Ο server δε θα πρέπει να γνωρίζει τι έγινε με κάποια προηγούμενη κλήση τη στιγμή που επεξεργάζεται την τρέχουσα κλήση. Μία υπηρεσία ή χρήστης του REST πρέπει να περιλαμβάνει εντός του HTTP αιτήματος (header και body) όλη την πληροφορία (action, parameters, context,...) που είναι απαραίτητη από τον server για την παραγωγή μιας απόκρισης στο αίτημα αυτό. Τα αιτήματα πρέπει να είναι ολοκληρωμένα και αυτόνομα έτσι ώστε κατά την επεξεργασία τους από τον server να μην απαιτούν την ανάσυρση κάποιας άλλης πληροφορίας η οποία να επηρεάζει την απόκριση στο ερώτημα.

Αυτή η «έλλειψη κατάστασης» σε μια υπηρεσία REST απαλλάσει τον server από την ανάγκη συγχρονισμού των δεδομένων μιας συνόδου με μια εξωτερική εφαρμογή. Ακόμα περισσότερο επιτρέπει στις υπηρεσίες REST να προσαρμόζονται σε περιπτώσεις όπου απαιτούνται υψηλές επιδόσεις χρησιμοποιώντας clusters of servers με διαμοιρασμό φορτίου (load-balancing) με σκοπό την ελάττωση του συνολικού χρόνου απόκρισης στην κλήση μιας δικτυακής υπηρεσίας.

- **Απεικόνιση της δομής των καταλόγων σαν URIs.**

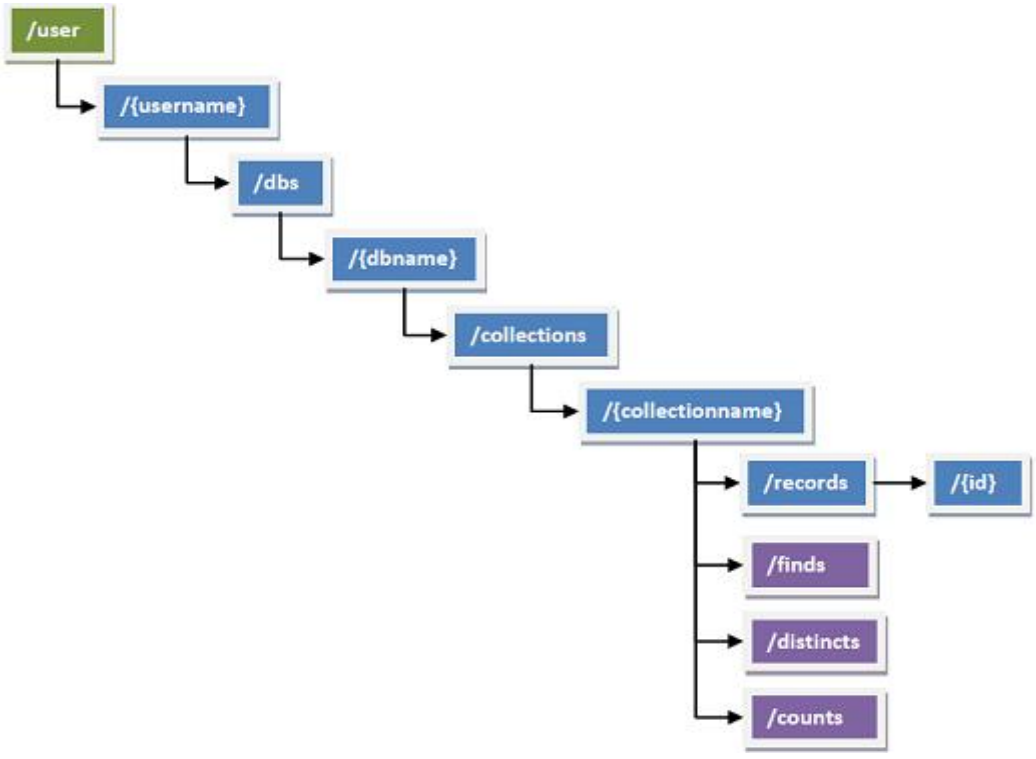
Ο σχηματισμός των URI μιας υπηρεσίας REST πρέπει να είναι προβλέψιμος και ευνόητος για τον χρήστη της υπηρεσίας και να απαιτείται από λίγη έως καθόλου τεκμηρίωση ή εξήγηση σχετικά με τον πόρο προς τον οποίο δείχνει κάθε URI. Ένας τρόπος να το πετύχουμε αυτό είναι να ορίσουμε URI παρόμοια με τον τρόπο δόμησης των καταλόγων σε ένα σύστημα αρχείων. Για παράδειγμα η δομή ενός URI που δείχνει όλους του εγγεγραμένους ασθενείς για κάποιον γιατρό θα μπορούσε να είναι: http://../users/doctors/{id_doctor}/patients

- **Μεταφορά δεδομένων, και προς τις δύο διευθύνσεις, μέσω XML, JSON ή και των δύο.**

Για να δώσουμε στον client την δυνατότητα να επιλέξει τη μορφή των περιεχομένων της απόκρισης του server στην πιο κατάλληλη γι αυτόν δομή, δεν έχουμε παρά να φτιάξουμε την υπηρεσία μας έτσι ώστε να κάνει χρήση του ενσωματομένου HTTP Accept header όπου η τιμή του θα είναι ένας τύπος MIME. Οι πιο συνηθισμένοι τύποι μορφοποίησης δεδομένων στις REST υπηρεσίες είναι JSON (application/json), XML (application/xml) και XHTML(application/xhtml+xml).[9][10]

2.6 JSON storage GE και Τρόπος λειτουργίας του

Ο JSON Storage GE είναι η υπηρεσία που υποστηρίζει την αποθήκευση πληροφορίας σε μορφή JSON και βρίσκεται στο περιβάλλον του Intellicloud. Η υπηρεσία συνοδεύεται από API σχεδιασμένο με βάση την αρχιτεκτονική REST. Οι βασικές λειτουργίες που υποστηρίζει είναι η δημιουργία (POST), η ανανέωση (PUT) και η διαγραφή (DELETE) χρηστών όπως επίσης και η δημιουργία βάσεων δεδομένων, συλλογών και εγγραφών. Βρίσκεται σε μια κοινόχρηστη εικονική μηχανή και ο χρήστης δεσμεύει χώρο, έτσι ώστε να αποθηκεύει δεδομένα, δίνοντας όνομα χρήστη και κωδικό πρόσβασης, χωρίς να επιτρέπεται η προσπέλαση προσωπικών δεδομένων από τρίτους. Ο κάθε χρήστης δημιουργεί βάσεις δεδομένων που περιέχουν συλλογές και αυτές με τη σειρά τους περιλαμβάνουν εγγραφές. Η δομή της υπηρεσίας όπως φαίνεται και στο Σχήμα 3, αποτελείται από ένα χρήστη (user) με το όνομά του (username), ο κάθε χρήστης δημιουργεί βάσεις δεδομένων (dbs), οι οποίες με τη σειρά τους περιέχουν συλλογές (collections) και αυτές οι συλλογές περιλαμβάνουν ορισμένες εγγραφές (records). Οι εγγραφές αυτές χαρακτηρίζονται από ενα μοναδικό αναγνωριστικό με το οποίο είναι δυνατή η ταυτοποίησή τους. Ο JSON Storage GE αποτελεί μια πιο ευέλικτη λύση για αποθήκευση δεδομένων καθώς δέχεται μεγαλύτερο όγκο δεδομένων.



Σχήμα 3- Δομή οργάνωσης JSON Storage GE

3 Σχεδιασμός Συστήματος

3.1 Απαιτήσεις Συστήματος

Πριν περάσουμε στο σχεδιασμό του συστήματός μας, θα αναφερθούμε στις βασικές ομάδες των χρηστών που το απαρτίζουν καθώς και στις λειτουργικές απαιτήσεις τους. Σε μια εφαρμογή σημαντικό ρόλο παίζουν οι λειτουργικές και μη λειτουργικές απαιτήσεις που θα πρέπει να ικανοποιεί ένα σύστημα. Ας δούμε πρώτα τον ορισμό κάθε μιας από αυτές και στη συνέχεια θα παρουσιαστούν ειδικά για το σύστημά μας.

Με τον όρο λειτουργικές απαιτήσεις εννοούμε τι πρέπει να κάνει ένα σύστημα, με άλλα λόγια ποιός είναι ο λόγος ύπαρξης του συστήματος και τι ενέργειες εξυπηρετεί για κάθε χρήστη του συστήματος ξεχωριστά. Οι μη λειτουργικές απαιτήσεις, είναι τα χαρακτηριστικά που πρέπει να έχει το σύστημα, δηλαδή οι λόγοι σύμφωνα με τους οποίους θα θεωρήσουμε ένα σύστημα επιτυχημένο. Γνωρίζοντας τις απαιτήσεις του συστήματος μπορούμε πιο εύκολα να δημιουργήσουμε υπηρεσίες που τις ικανονποιούν.

Οι ομάδες χρηστών του δικού μας συστήματος είναι οι εξής:

- Ιατρός (Καταναλώνει και Παράγει)
- Ασθενής (Παράγει και Καταναλώνει)
- Διαχειριστής (Διαχειρίζεται)

Ο διαχειριστής είναι ένας και μοναδικός για το σύστημά μας. Πρέπει να είναι εγγεγραμμένος στο FIWARE LAB[17] και είναι ο πρώτος χρήστης που συνδέεται με username και password.

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

Παρακάτω διατυπώνονται οι λειτουργικές απαιτήσεις, δηλαδή οι διαδικασίες που πρέπει να υλοποιεί το σύστημά μας, για κάθε ομάδα-χρήστη. Στη συνέχεια αυτές οι απαιτήσεις θα παρουσιαστούν με τη βοήθεια διαγραμμάτων UML.

Λειτουργικές Απαιτήσεις Συστήματος για τον Διαχειριστή

- Εξουσιοδότηση χρήστη με username, password.
- Ταυτοποίηση ασθενών-ιατρών με συγκεκριμένο username, password.
- Εισαγωγή/Επεξεργασία/Διαγραφή χρηστών.
- Εισαγωγή/Επεξεργασία/Διαγραφή xml σχήματος αισθητήρα.
- Ανάθεση ασθενούς σε ιατρό.
- Συσχέτιση ασθενή με ιατρό και αισθητήρα.
- Ανάκτηση συγκεκριμένου χρήστη.
- Ανάκτηση όλων των χρηστών.
- Δυνατότητα ανάκτησης xml σχημάτων αισθητήρων.
- Ανάθεση αισθητήρα σε ασθενή.

Λειτουργικές Απαιτήσεις Συστήματος για τον Ασθενή

- Αποστολή δεδομένων που παράγουν οι αισθητήρες.
- Λήψη ενημερώσεων/μηνυμάτων από γιατρό.
- Τροποποίηση τρόπου αποστολής δεδομένων.
- Ανάκτηση ιστορικού δεδομένων όλων των αισθητήρων του.

Λειτουργικές Απαιτήσεις Συστήματος για τον Ιατρό

- Έλεγχος μετρήσεων αισθητήρων.
- Ορισμός/ αλλαγή κανόνων ασθενούς.
- Αποστολή κανόνα στο σύστημα.
- Ανάκτηση πρόσφατων κανόνων ασθενούς.
- Ανάκτηση ιστορικών δεδομένων όλων των αισθητήρων συγκεκριμένου ασθενούς.
- Προβολή όλων των ασθενών του.
- Προβολή κατάστασης ασθενών του.
- Αποστολή μηνυμάτων σε ασθενή.

Έπειτα παρουσιάζουμε τις μη λειτουργικές απαιτήσεις, που όπως αναφέραμε είναι τα χαρακτηριστικά που πρέπει να έχει το σύστημά μας και η ικανοποίησή τους οδηγεί σε ένα καλής ποιότητας σύστημα.

Οι κατηγορίες των μη λειτουργικών απαιτήσεων

- Ασφάλεια: Σχετίζονται με την ασφαλή πρόσβαση σε δεδομένα αλλά και σε hardware π.χ μόνο εγγεγραμμένοι χρήστες έχουν πρόσβαση στο σύστημα. Το κριτησιο αυτό ικανοποιείται στο σύστημά μας με τη χρήση του KeyRock IDM GE για τον οποίο αναφερόμαστε αναλυτικά στο Κεφάλαιο 4.2.4 που αναλύουμε τις υπηρεσίες του Υπολογιστικού Νέφους Back-End.

- Απόδοση: Ορίζουν χαρακτηριστικά που έχουν να κάνουν με την ταχύτητα του συστήματος και την ανταπόκριση του. Για την απόδοση του συστήματος και τα πειράματα που πραγματοποιήσαμε γίνεται αναλυτική περιγραφή στο Κεφάλαιο 5.

- Επεκτασιμότητα: Σχετίζεται με το πόσο εύκολα ή δύσκολα προστίθενται περισσότερες δυνατότητες και λειτουργίες στην εφαρμογή. Για την ικανοποίηση αυτού του κριτηρίου φροντίσαμε το σύστημά μας να δέχεται οποιοδήποτε σχήμα xml αισθητήρα. Ωστόσο, η επεκτασιμότητα υποστηρίζεται και από τον τύπο της αρχιτεκτονικής που επιλέξαμε να βασιστούμε, μιας και η υπηρεσιοκεντρική αρχιτεκτονική έχει ως ενα βασικό χαρακτηριστικό της την επεκτασιμότητα και αναφέρεται αναλυτικά στο Κεφάλαιο 2.5.

3.2 Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams)

Έχοντας παρουσίασει τις λειτουργικές απαιτήσεις των χρηστών στην ενότητα 3.1.1, θα προσπαθήσουμε σε αυτή την ενότητα να τις ομαδοποιήσουμε και να τις παρουσιάσουμε σε μορφή διαγραμμάτων, με σκοπό να καθορίσουμε τις υπηρεσίες που χρησιμοποιήθηκαν στο σύστημά μας. Αρχικά θα παρουσιάσουμε τις περιπτώσεις χρήσης (use cases) για κάθε κατηγορία χρηστών ξεχωριστά και στη συνέχεια τα διάφορα UML διαγράμματα που χρησιμοποιήθηκαν και βοήθησαν στην υλοποίηση συγκεκριμένων ενεργειών του συστήματος.

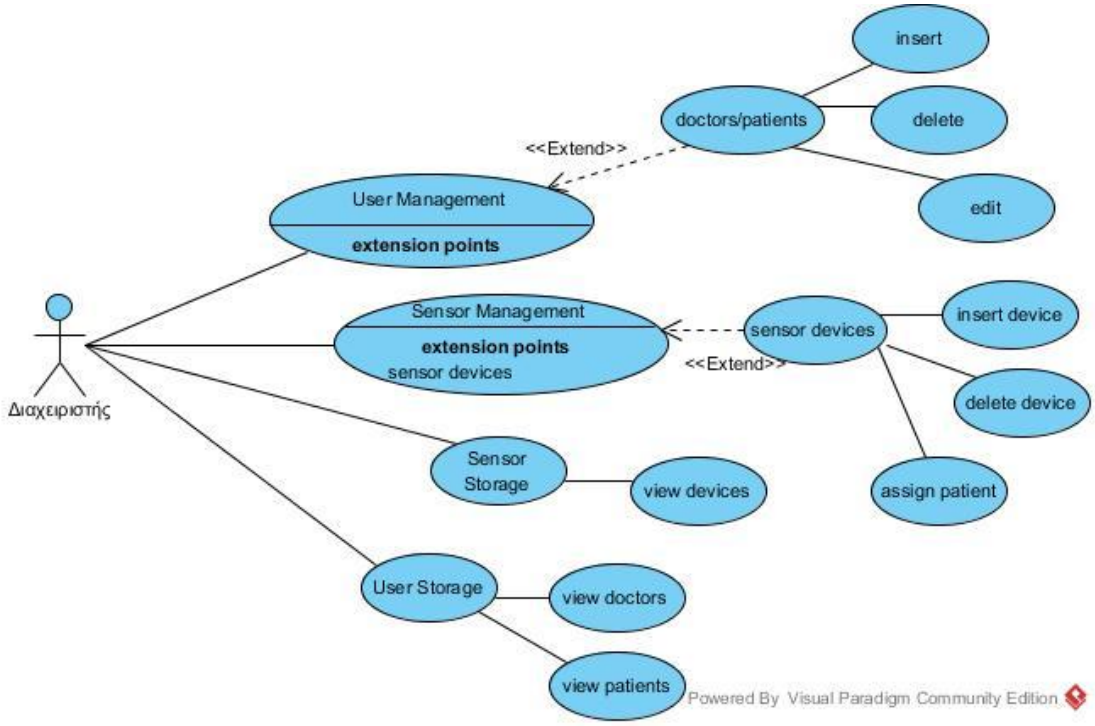
Για το διαχειριστή του συστήματος, δημιουργήσαμε δύο υπηρεσίες με βάση τις λειτουργικές απαιτήσεις του:

- Υπηρεσία Διαχείρισης Χρηστών, που περιλαμβάνει την
    - Εισαγωγή/Επεξεργασία/Διαγραφή χρήστη
    - Ανάθεση ασθενούς σε ιατρό

- Υπηρεσία Διαχείρισης Αισθητήρων , που περιλαμβάνει την
    - Εισαγωγή/Επεξεργασία/Διαγραφή xml σχήματος αισθητήρα
    - Ανάθεση αισθητήρα σε ασθενή
    - Συσχέτιση ασθενή με ιατρό και αισθητήρα

Για την προβολή όλων των χρηστών ή μεμονομένων, γίνεται ανάκτησή τους από τη βάση των χρηστών User Storage, και για την προβολή των διαθέσιμων αισθητήρων αντίστοιχα γίνεται ανάκτηση από τη βάση τους, Sensor Storage.

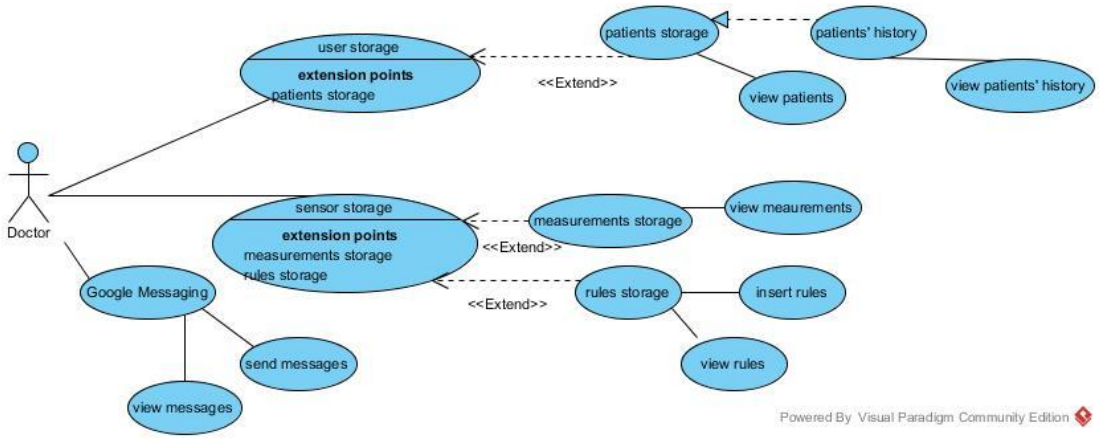
Έτσι καταλήγουμε στο Σχήμα 4, παρακάτω.



Σχήμα 4- Διάγραμμα Περιπτώσεων Χρήσης (Use Case) Διαχειριστή

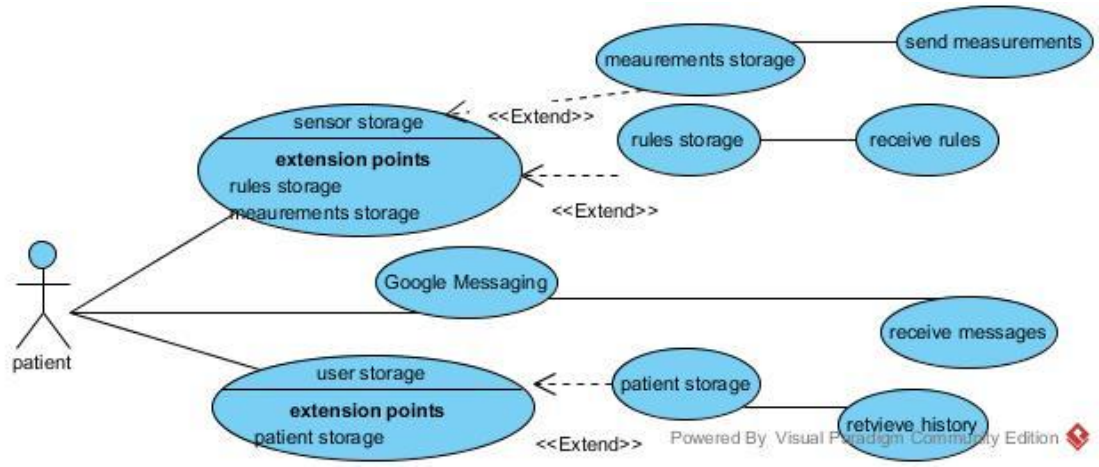
Ομοίως, για τον ιατρό με βάση τις απαιτήσεις του έχουμε την Υπηρεσία Διαχείρισης Χρηστών, στην οποία περιλαμβάνονται η προβολή των ασθενών του, του ιστορικού τους και όλων των στοιχείων τους. Η Υπηρεσία Διαχείρισης

Μετρήσεων και Κανόνων, η οποία περιλαμβάνει τον ορισμό των κανόνων των ασθενών του, και την προβολή των μετρήσεων τους. Μια επιπλέον υπηρεσία είναι αυτή της αποστολής Μηνυμάτων προς τους ασθενείς του, η Google Messaging, όπως φαίνεται στο Σχήμα 5.



Σχήμα 5- Διάγραμμα Περιπτώσεων Χρήσης (Use Case) Ιατρού

Αλλά και για τον ασθενή, έχουμε την αποστολή των μετρήσεων μετά τη συλλογή του από την Υπηρεσία Συλλογής Δεδομένων του Front-End, όπου στέλνονται οι μετρήσε στο Υπολογιστικό Νέφος, τη λήψη των κανόνων όπου πρόκειται για τα όρια των τιμών των μετρήσεων ορισμένα από τον ιατρό του, την προβολή του ιστορικού του όπου μπορεί να βλέπει το ιστορικό των μετρήσεών του και τη λήψη μηνυμάτων από τον ιατρό του, όπως φαίνεται και στο Σχήμα 6.



Σχήμα 6- Διάγραμμα Περιπτώσεων Χρήσης (Use Case) Ασθενούς

3.3 Αρχιτεκτονικά Διαγραμματα

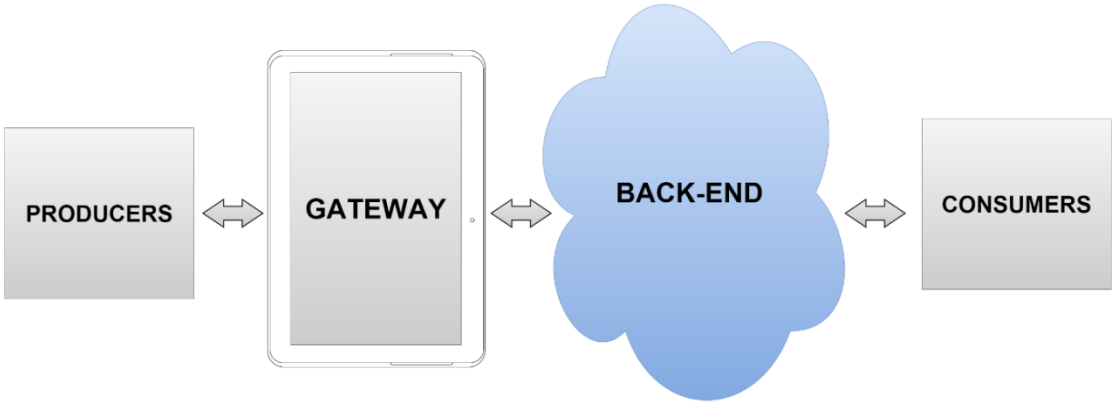
Στη συνέχεια, παρουσιάζονται τα διαγράμματα που μας βοήθησαν στην υλοποίηση του συστήματός μας. Αρχικά, παρουσιάζεται το γενικό αρχιτεκτονικό διάγραμμα που δείχνει τα μικρότερα τμήματα/υπηρεσίες από τα οποία αποτελείται το σύστημά μας, στην ενότητα 3.3.1. Έπειτα παρουσιάζεται το διάγραμμα που δείχνει τη ροή εργασιών, δηλαδή τη σειρά των βημάτων που πρέπει να ακολουθήσει ο χρήστης για την εκπλήρωση μιας συγκεκριμένης ενέργειας (ενότητα 3.3.2). Τέλος, παρατίθενται και τα UML διαγράμματα και συγκεκριμένα: 1)Διάγραμμα κλάσεων (Class diagram), που περιγράφει τη δομή του συστήματος, 2)Διάγραμμα δραστηριοτήτων (Activity diagram), που παρουσιάζει τη σειρά των βημάτων για μια επιλογή, μια επανάληψη ή ένα συγχρονισμό και 3)Διάγραμμα αλληλεπίδρασης (Interaction UML diagram), που δείχνει τη λειτουργία ενός αντικειμένου με κάποιο άλλο, αλλά και τη σειρά με την οποία γίνεται (ενότητα 3.3.2).

3.3.1 Γενικό Αρχιτεκτονικό Διάγραμμα

Η υλοποίηση του συστήματός μας βασίζεται στη λογική της Υπηρεσιο-κεντρικής Αρχιτεκτονικής (Service Oriented Architecture - SoA) όπως αναλύθηκε και στο δεύτερο κεφάλαιο. Δηλαδή στην αρχιτεκτονική που θέλει το σύστημά μας να αποτελείται από μικρότερα υπο-συστήματα/υπηρεσίες. Οι υπηρεσίες αυτές με τον κατάλληλο προγραμματισμό, μπορούν να επικοινωνούν μεταξύ τους και να παράγουν πληροφορία η οποία μπορεί να αξιοποιηθεί κατάλληλα. Τα τμήματα αυτής της αρχιτεκτονικής είναι:

- Παραγωγοί (Producers)
- Πύλη εφαρμογή (Application Gateway)
- Διεπαφή διαχείρισης Συστήματος νέφους (Back-End)
- Καταναλωτές (Consumers)

και φαίνονται στο Σχήμα 7 παρακάτω.



Σχήμα 7 – Λογική της Υπηρεσιο-κεντρικής Αρχιτεκτονικής

(Service Oriented Architecture - SoA) [18]

- **Παραγωγοί (Producers):** Διάφοροι χρήστες έχουν στην διάθεση τους αισθητήρες οι οποίοι παράγουν πληροφορία για τα ζωτικά τους δεδομένα όπως για παράδειγμα μέτρηση του καρδιακού ρυθμού (heart rate). Σε αυτό το στάδιο λειτουργίας στην δική μας περίπτωση οι αισθητήρες είναι τύπου Bluetooth low energy και στην περίπτωση τους κάθε φορά που ενεργοποιούνται ξεκινάει η διαδικασία του advertising προκειμένου να συνδεθούν με μια κεντρική συσκευή.

- **Πύλη Εφαρμογή (Gateway):** Αποτελεί μια android Εφαρμογή η οποία χρησιμοποιεί της ιδιότητες μιας πύλης δικτύου (gateway) μέσω ενός κινητού η tablet για να μεταφέρει τα δεδομένα από τους αισθητήρες στο νέφος.

- **Διεπαφή Συστήματος Νέφους (Back-end):** Αποτελεί το βασικό κομμάτι της διπλωματικής μας εργασίας μιας και λαμβάνει τα δεδομένα από το Front-end και πραγμοτοποιεί τις απαραίτητες ενέργειες-υπηρεσίες. Τα δεδομένα αυτά επεξεργάζονται και αποθηκεύονται κατάλληλα με βάση το σενάριο χρήσης της εφαρμογής μας με στόχο την εξυπηρέτηση των καταναλωτών.

- **Καταναλωτές (Consumers):** Το τελευταίο υποσύστημα της γενικής αρχιτεκτονικής μας. Σε αυτό το τμήμα βρίσκονται οι χρήστες οι οποίοι θέλουν να έχουν πρόσβαση στις πληροφορίες που παράγει το σύστημά μας, καθώς επίσης και η εφαρμογή διεπαφής μέσω της οποίας έχει πρόσβαση ο

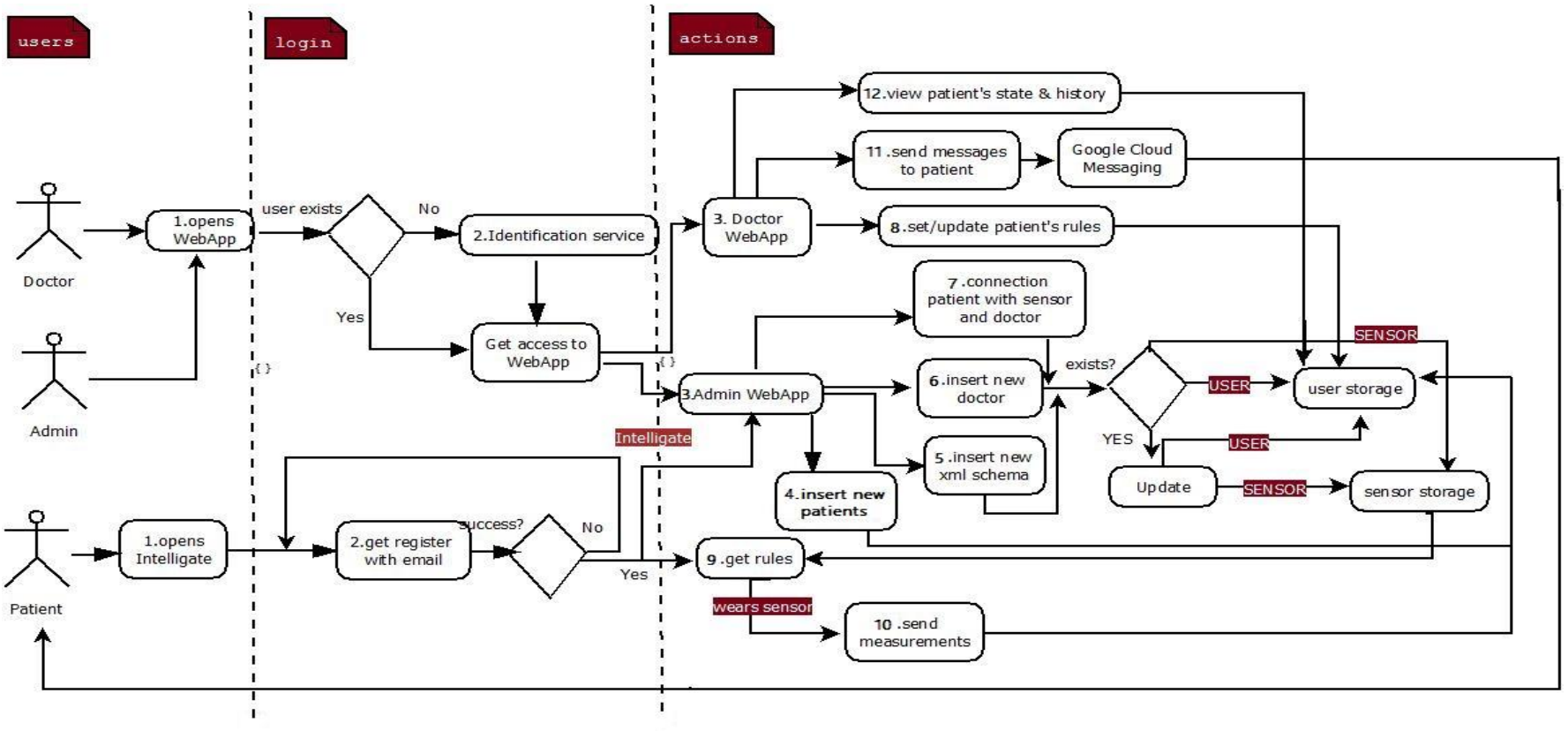
[18] http://www.intelligence.tuc.gr/publications.php?pub_author=All&pub_type=8&pub_subject=All

χρήστης-ασθενής και ο χρήστης-ιατρός. Σε αυτό το κομμάτι υλοποίησης αναπτύσσεται και το γραφικό περιβάλλον(graphical user interface – GUI) το οποίο διευκολύνει την πρόσβαση του χρήστη στις παρεχόμενες πληροφορίες και υπηρεσίες, διαθέτοντας του ένα σύνολο από εργαλεία [12].

3.3.2 UML Διαγράμματα του Συστήματος

Αρχικά θα παρουσιάζουμε με ένα διάγραμμα ροής[19], που φαίνεται στο Σχήμα 8, την πορεία των ενεργειών στο σύστημά μας και έπειτα θα δείξουμε με διαγράμματα UML ορισμένες ενέργειες ξεχωριστά. Τα διαγράμματα που θα παρουσιάσουμε είναι: 1)Διάγραμμα κλάσεων (Class diagram), που περιγράφει τη δομή του συστήματος με τη μορφή πινάκων, 2)Διάγραμμα δραστηριοτήτων (Activity diagram), που παρουσιάζει τη σειρά των βημάτων για μια επιλογή, μια επανάληψη ή ένα συγχρονισμό και 3)Διάγραμμα αλληλεπίδρασης (Interaction UML diagram), που δείχνει τη λειτουργία ενός αντικειμένου με κάποιο άλλο, αλλά και τη σειρά με την οποία γίνεται.

[19] https://en.wikipedia.org/wiki/Flowchart



Σχήμα 8- Διάγραμμα Ροής Εργασιών

Το παραπάνω διάγραμμα δείχνει τα εξής βήματα για την εκτέλεση ενεργειών:

- Ο ασθενής εγγράφεται στο Intellligate στο tablet επιλέγοντας «Register Device», με βάση το email του.
- Φορτώνεται το email του ασθενούς στο Json Storage .
- διαχειριστής του συστήματος από το WebApp επιλέγοντας «insert new patient» φορτώνει τον/τους νεους ασθενείς που έκαναν εγγραφή από το tablet τους, στη βάση του Intelligate Back –End (MySQL DB).
- Ο διαχειριστής συσχετίζει τον ασθενή , με τον ιατρό αλλά και με τη συσκευή/ες που χρειάζεται .
- Ο ιατρός μέσω του δικού του WebApp όρίζει τους κανόνες για τις συσκευές και τις μετρήσεις που ενδιαφέρουν τον ασθενή.
- Οι κανόνες αποθηκεύονται στη MySql DB , καθώς και στο Json Storage.
- Ο ασθενής απο το tablet του επιλέγοντας «Get Rules» λαμβάνει τους κανόνες του και μπορεί να πάρει κάποια μέτρηση.
- Ο ιατρός παρακολουθεί το ιστορικό του ασθενούς του, αλλά και ανανεώσει/ορίζει τους κανόνες για τις μετρήσεις του εαν το κρίνει απαραίτητο.
- Ο ιατρός επίσης μπορεί να στείλει ένα μήνυμα στον ασθενή του .
- Τέλος , ο ιατρός μπορεί να δει εαν η κατάσταση κάποιου ασθενούς είναι κρίσιμη, εφόσον εμφανιστεί η ένδειξη κίνδυνος δίπλα στο όνομα του ασθενούς του.

Στη συνέχεια παρουσιάζεται ο σχεδιασμός του συστήματος με τα εξής αρχιτεκτονικά διαγράμματα UML :

- **Διάγραμμα Κλάσεων (Class diagram):** Στατικό διάγραμμα που περιγράφει τη δομή του συστήματος δείχνοντας τις κλάσεις του, τα χαρακτηριστικά του, τις μεθόδους του (υπηρεσίες) και τη συσχέτιση μεταξύ τους [13].

- **Διάγραμμα Δραστηριοτήτων (Activity diagram):** Γραφική αναπαράσταση ροής εργασιών (workflow) των σταδιακών δραστηριοτήτων και δράσεων για μια επιλογή, μια επανάληψη και ένα συγχρονισμό [14].

- **Διάγραμμα Αλληλεπίδρασης (Interaction UML diagram):** Διάγραμμα αλληλεπίδρασης που δείχνει πως λειτουργούν τα αντικείμενα το ένα με το άλλο και με ποιά σειρά (είναι σε ακολουθία χρόνου) [15] [16].

Στο Σχήμα 9, βλέπουμε το διάγραμμα κλάσεων από τις οποίες αποτελείται το σύστημά μας. Στόχος του είναι η περιγραφή των διαφόρων κλάσεων καθώς και των μεταξύ τους σχέσεων. Οι βασικές κλάσεις του συστήματός μας είναι:

- Κλάση «Άνθρωπος» (Person class): Αποτελεί μια υπερκλάση που περιέχει το σύνολο των χρηστών. Η κλάση διαθέτει τρεις υποκλάσεις ανάλογα με την κατηγορία στην οποία ανήκει ο χρήστης (ασθενής-ιατρός-διαχειριστής). Το χαρακτηριστικό της υπερκλάσης είναι ότι «κληροδοτεί» τα χαρατκηριστικά της στις υποκλάσεις της. Κάθε υποκλάση περιλαμβάνει τα χαρατκηριστικά αυτά και μπορεί να διαθέτει και επιπλεόν δικά της χαρακτηριστικά. Συγκεκριμένα κάθε υποκλάση «κληρονομεί» το όνομα, το επίθετο, το email και τον κωδικό από την υπερκλάση «Άνθρωπος».

- Κλάση Ιατρός (doctor class): Περιέχει τα στοιχεία όνομα, το επίθετο, το email και τον κωδικό του, που τα κληρονομεί από την υπερκλάση «Άνθρωπος» (Person) και επιπλέον, του δίνεται ένα μοναδικό αναγνωριστικό νούμερο που αποτελεί το id_doctor. Το κάθε αντικείμενο αναγνωρίζεται από αυτόν τον μοναδικό αναγνωριστικό αριθμό καθώς και από το email του, το οποίο είναι για κάθε χρήστη-ιατρό μοναδικό. Η κλάση αυτή επιτρέπει την προβολή όλων των ασθενών του ή μεμονομένων και του ιστορικού των μετρήσεων των ασθενών του.

- Ασθενής (patient): Ομοίως, κλάση αυτή κληρονομεί τα στοιχεία της κλάσης «Άνθρωπος» με τα επιπλέον χαρακτηριστικά, ημερομηνία γέννησης, κατοικία, τύπος ασθένειας, νοσηλεία που στο σχήμα αναγράφονται ως «πληροφορίες ασθενούς» (patient's info). Διαθέτει, ομοίως ένα μοναδικό αναγνωριστικό νούμερο που του δίνεται αυτόματα, το id_patient. Το κάθε αντικείμενο αναγνωρίζεται από τον μοναδικό αναγανωριστικό αυτό αριθμό καθώ και από το email του που είναι μοναδικό. Η κλάση αυτή επιτρέπει τη λήψη μηνημάτων, την προβολή του ιστορικού καθώς και τη λήψη κανόνων που αφορούν τις μετρήσεις τους. Σχετίζεται επομένως με την κλάση «Ιστορικό ασθενούς» με σχέση «σύνθεσης» (composition) που δηλώνει την αναγκαιότητα της ύπαρξης της κλάσης «Ασθενής» για να υπάρξη η κλάση «Ιστορικό ασθενούς». Επίσης σχετίζεται με την κλάση «Ιατρός» με απλή συσχέτιση ενω με την κλάση «Συσκευές-Αισθητήρες» με σχέση «σύνθεσης», εφόσον αν δεν υπάρχει ασθενής δε θα υπάρχει και συσκευή για αυτόν.

- Συσκευές-Αισθητήρες (SensorDevices): Κρατά την απαραίτητη πληροφορία που φέρει το xml σχήμα, του αισθητήρα. Συγκεκριμένα, από όλα τα στοιχεία που περιέχονται στο αρχείο του αισθητήρα, κρατάμε το όνομα του αισθητήρα (πχ. NoninOnix[20], Polar H7[21]), τα είδη των μετρήσεων που διαθέτει (πχ. καρδιακός παλμός-pulse_Rate, ποσοστό

[20] http://www.nonin.com/Manuals-IFUs

[21] https://www.polar.com/en/products/accessories/H7_heart_rate_sensor

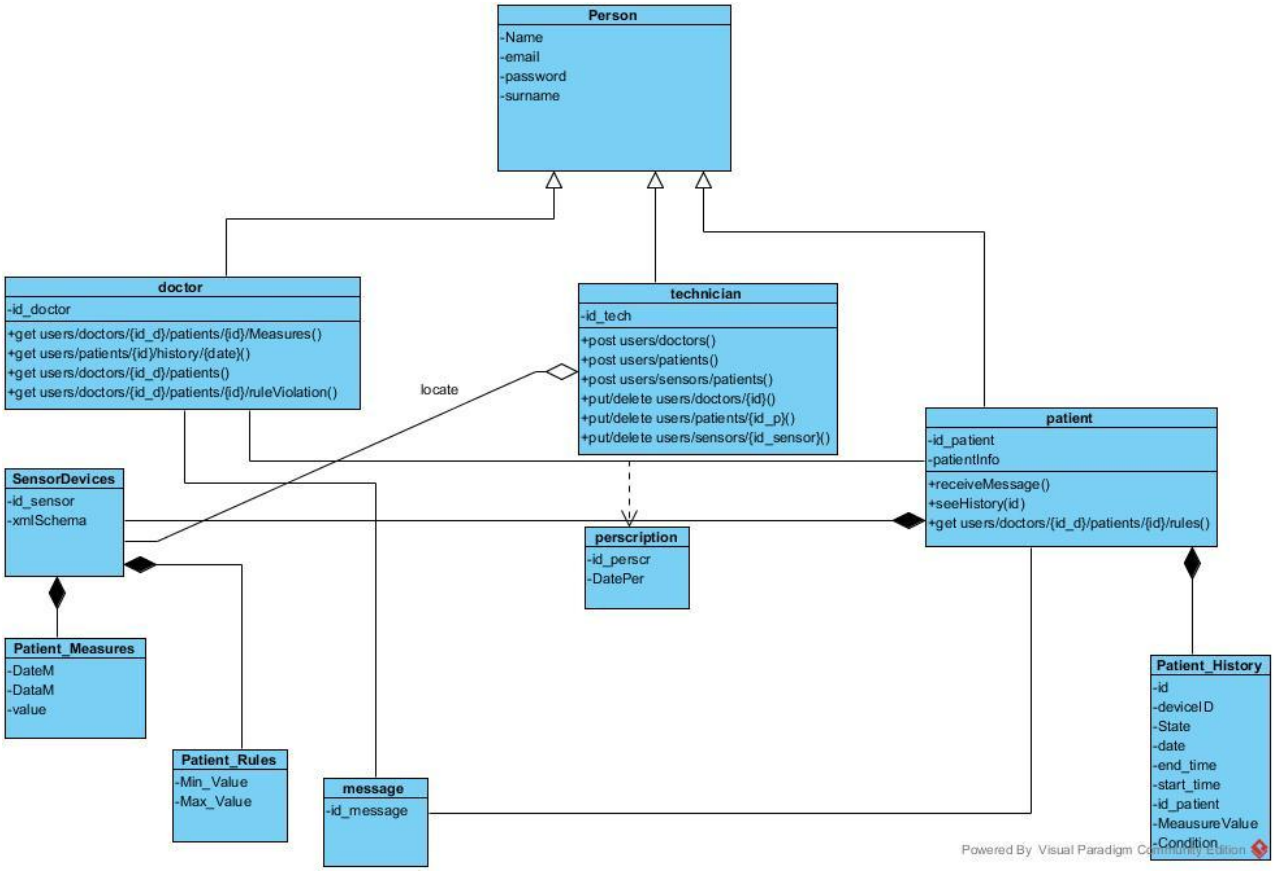
οξυγόνου στο αίμα-SPO2), τους τύπους μέτρησής του, δηλαδή σε τι μετράται κάθε είδος μέτηρησης (πχ. bpm,zbmp) και το κωδικό όνομα που διαθέτει κάθε αισθητήρας. Επίσης, δίνεται αυτόματα και ένα μοναδικό αναγνωριστικό νούμερο, id_sensor. Η κλάση συνδέεται με σχέση «σύνθεσης» με την κλάση «Μετρήσες ασθενούς», εφόσν εαν δεν υπάρχει συσκευή-αισθητήρα δε θα υπάρχουν και μετρήσεις για τους ασθενείς καθώς και με την κλάση «Κανόνες ασθενών» με τον ίδιο τρόπο εξάρτησης.

- Συνταγή (prescription): Κρατά το αναγνωριστικό id του ασθενούς, το αναγνωριστικό id του ιατρού και το αναγνωριστικό id του αισθητήρα. Με βάση αυτής της κλάσης γίνεται ο συσχετισμός των τριών αυτών κλάσεων, προκειμένου να ανατεθεί ο σωστός αισθητήρας στον σωστό ασθενή κάποιου ιατρού. Επομένως η κλάση αυτή συνδέεται με τις κλάσεις «Ιατρός» και την κλάση «Ασθενής».

- Μετρήσεις Ασθενούς (Patient_Measures): Εδώ βρίσκονται αποθηκευμένες οι μετρήσεις των αισθητήρων. Περιέχουν την τιμή της μέτρησης του ασθενούς, το είδος της μέτρησης και τον τύπο, το όνομα της συσκευής-αισθητήρα αλλά περιέχει και την πληροφορία του αναγνωριστικού id του ασθενούς που είναι αποθηκευμένη στην κλάση «SensorDevices», ώστε να ξέρουμε ποιου ασθενούς είναι η μέτρηση, στο σχήμα αναφέρονται ως DataM. Εαν , δεν υπάρχει συσκευή-αισθητήρα, δεν υπάρχουν και μετρήσεις.

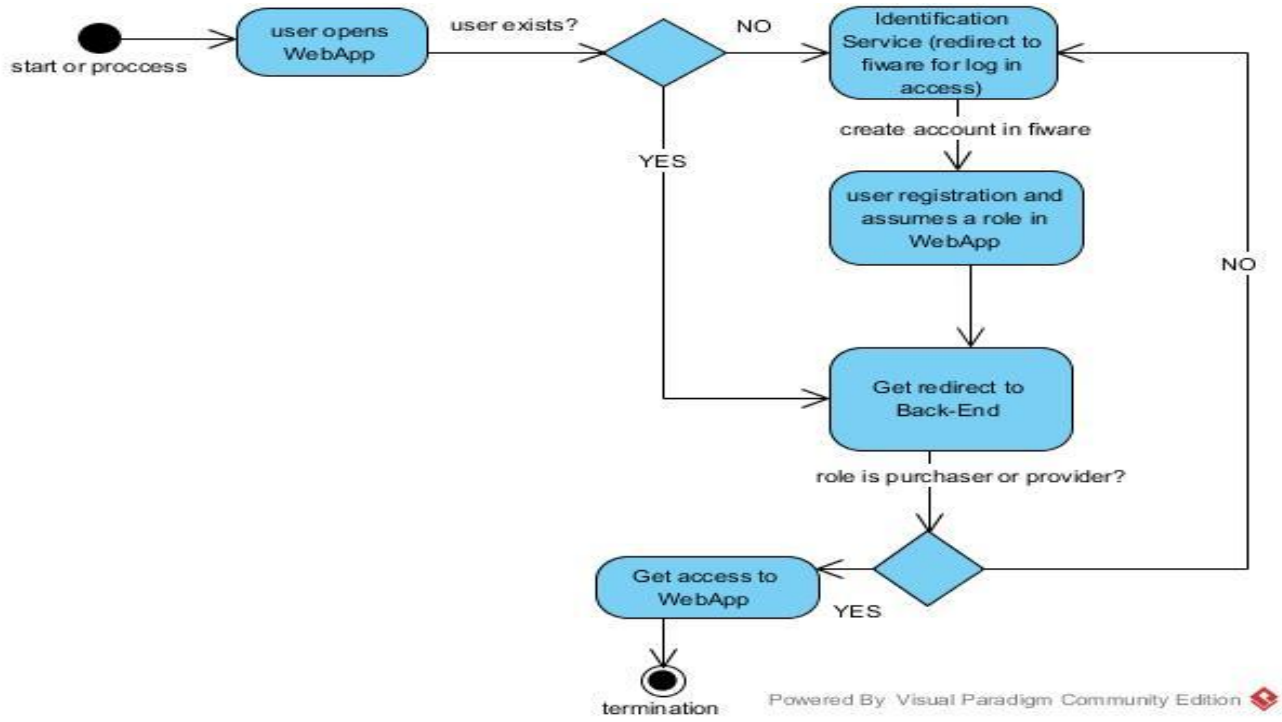
- Κανόνες Ασθενούς (Patient_Rules): Εδώ βρίσκονται οι κανόνες που ορίζονται από τον ιατρό και αφορούν τις μετρήσεις για κάποιον συγκεκριμένο αισθητήρα ενός ασθενούς. Η κλάση αυτή περιέχει τη μέγιστη επιτρεπτή τιμή (Max_Value) και την ελάχιστη επιτρεπτή τιμή (Min_Value) για τις μετρήσεις του ασθενούς με id που προέρχεται από την κλάση «SensorDevices». Εαν δεν υπάρχει ασθενής δεν υπάρχουν και κανόνες.

- Ιστορικό Ασθενούς (Patient_History): Πρόκειται για το ιστορικό του ασθενούς. Περιέχει ημερομηνία μέτρησης, ώρα έναρξης, ώρα λήξης μέτρησης, το αναγνωριστικό id του ασθενούς, την τιμή της μέτρησης, το αναγνωριστικό id του αισθητήρα και την κατάσταση του ασθενούς. Εαν δεν υπάρχει ασθενής δεν υπάρχει και ιστορικό ασθενούς.



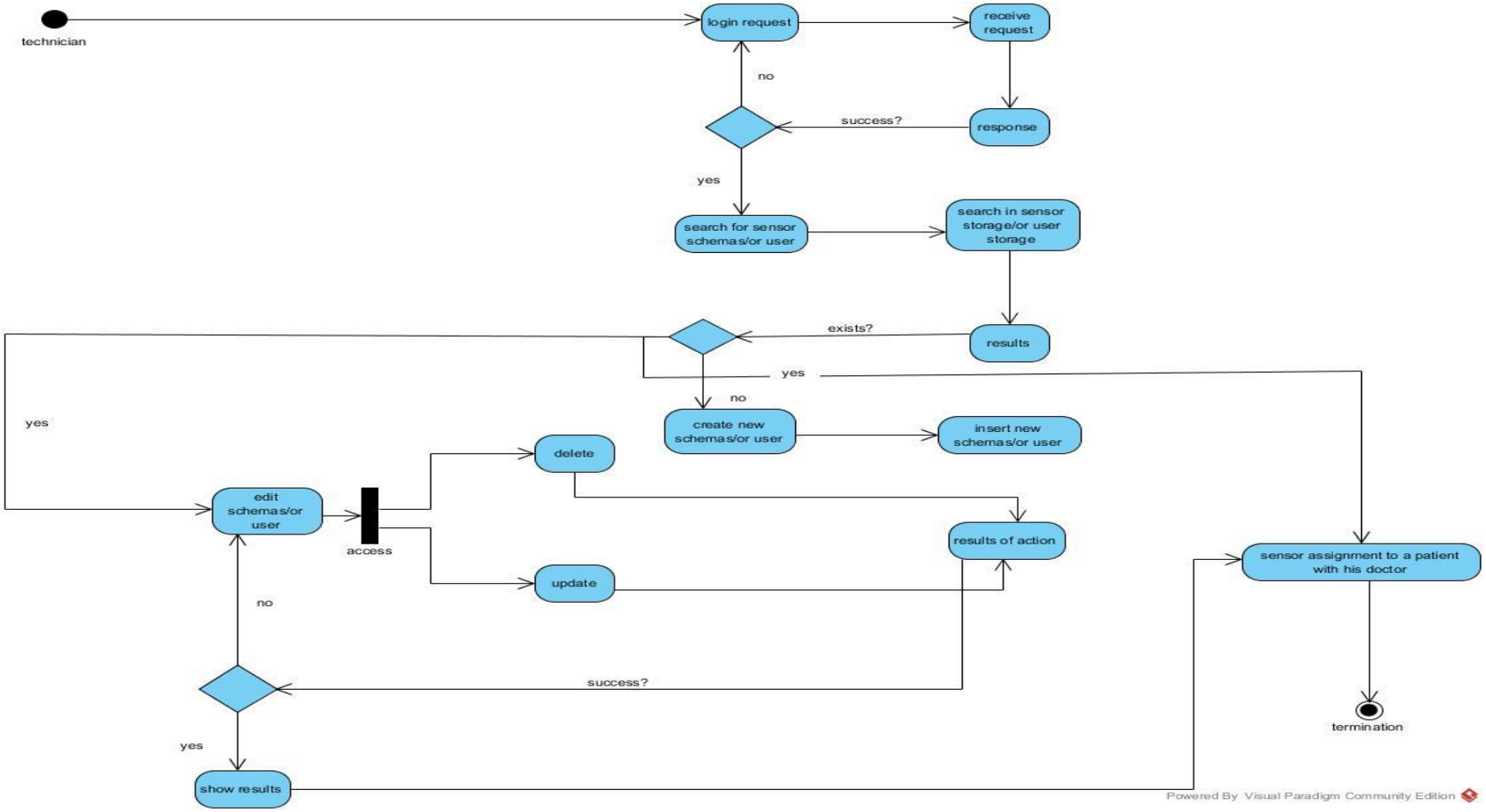
Σχήμα 9- Διάγραμμα Κλάσεων (Class Diagram - Information Viewpoint)

Στο Σχήμα 10, βλέπουμε την πορεία που ακολουθεί ο χρήστης-ιατρός για να συνδεθεί στην εφαρμογή μας, είτε είναι η πρώτη φορά που κάνει είσοδο στην εφαρμογή είτε όχι. Αναλυτικά τα βήματα είναι: 1. Άνοιγμα εφαρμογής από τον υπολογιστής του 2. Εαν πρόκειται για χρήστη που έχει πραγματοποιήσει σύνδεση ξανά, τότε τα στοιχεία της συνδεσής του υπάρχουν αποθηκευμένα. Αποστέλλονται στο Νέφος (Back-end) για επιβεβαιώση και εξακρίβωση και αν είναι σωστά τότε ο χρήστης-ιατρός αποκτά δικαίωμα πρόσβασης στις υπηρεσίες της εφαρμογής 3. Εαν πρόκειται για νέο χρήστη, ή τα στοιχεία του χρήστη δεν είναι έγκυρα, τότε ο χρήστης στέλνεται μέσω της εφαρμογής στο Fiware για να συνδεθεί 4. Δημιουργεί λογαριασμό στο Fiware, εαν δε διαθέτει, και περιμένει από τον διαχειριστή να τον προσθέσει στην εφαρμογή δίνοντάς του και τον κατάλληλο ρόλο (ιατρός-ασθενής-διαχειριστής) 5. Αποκτά πρόσβαση στην εφαρμογή.

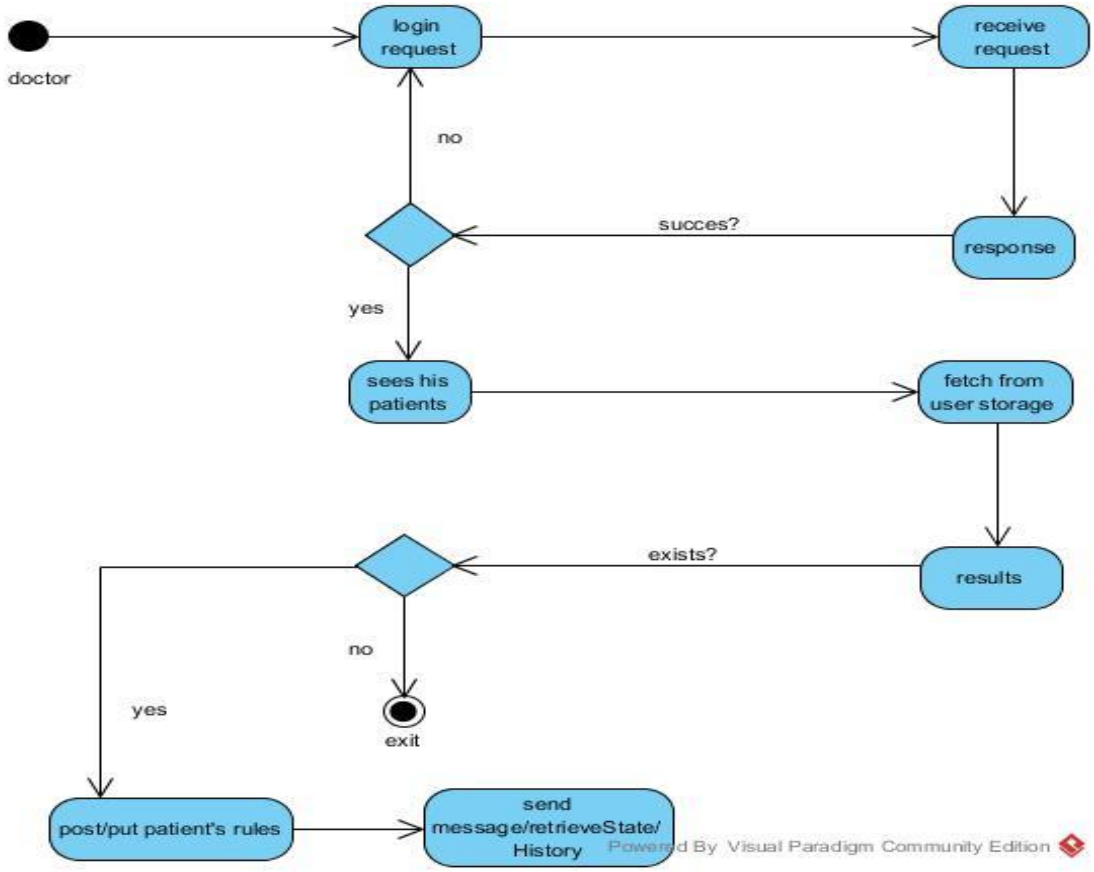


Σχήμα 10 – Διάγραμμα Δραστηριοτήτων (Activity Diagram) - Σύνδεση στην εφαρμογή

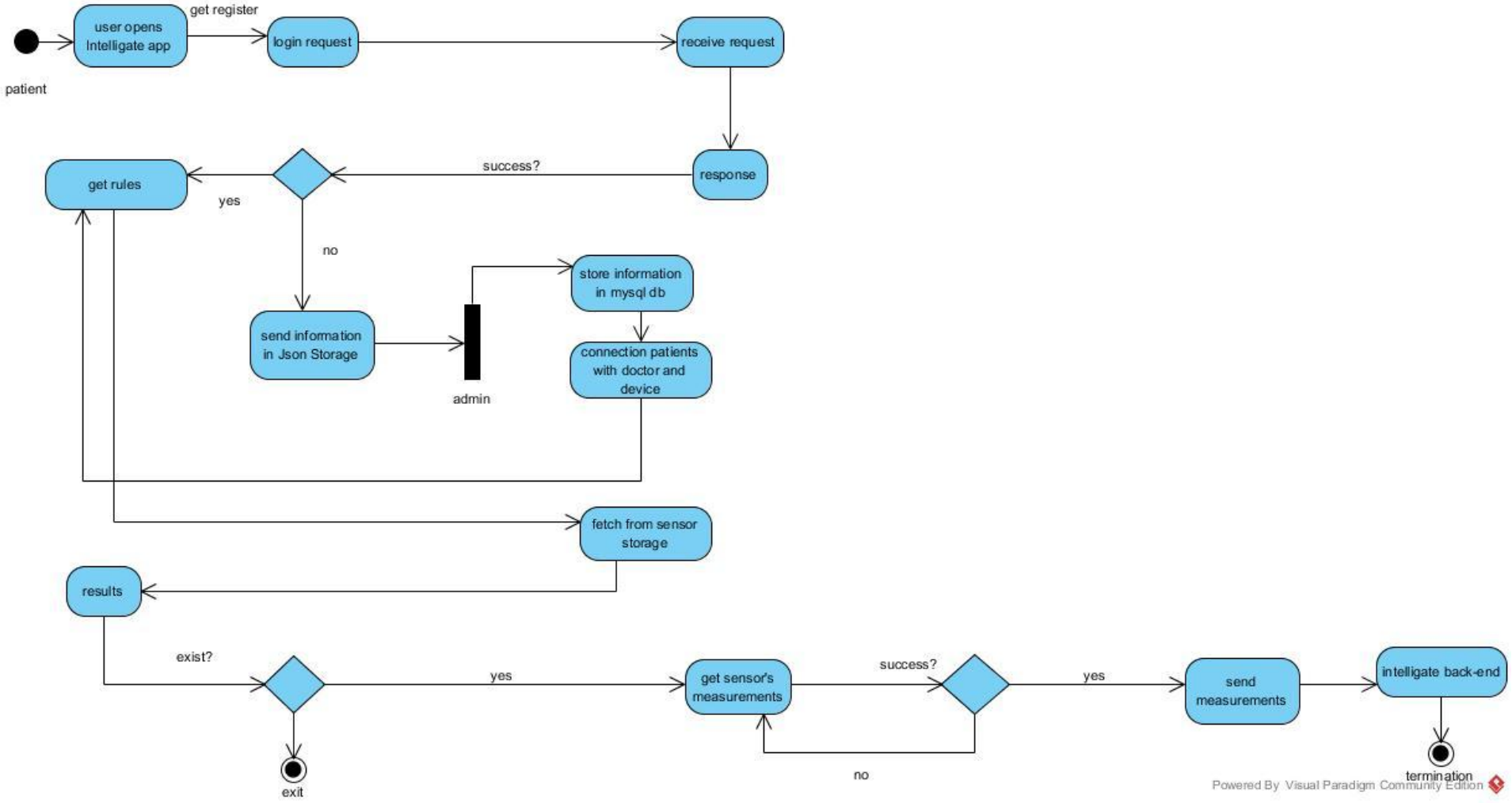
Παρακάτω, στα Σχήματα 11, 12, 13 βλέπουμε τις ενέργειες που μπορεί να πραγματοποιήσει ο διαχειριστής, ο ιατρός και ο ασθενής αντίστοιχα, κάνοντας χρήση ο καθένας της δικιάς του διεπαφής. Αρχικά, ο χρήστης-ιατρός αλλά και ο διαχειριστής πραγματοποιούν είσοδο στην εφαρμογή με τη διαδικασία που περιγράφηκε στο Διάγραμμα 10. Κατόπιν αυτής, είναι έτοιμοι να χρησιμοποιήσουν την εφαρμογή με τις λειτουργίες που προσφέρονται για τον κάθε χρήστη ξεχωριστά όπως έχουν περιγραφεί στο Κεφάλαιο 3.1 Λειτουργικές Απαιτήσεις Χρηστών.



Σχήμα 11 - Διάγραμμα Δραστηριοτήτων (Activity Diagram) - Δυνατότητες Διαχειριστή στο WebApp

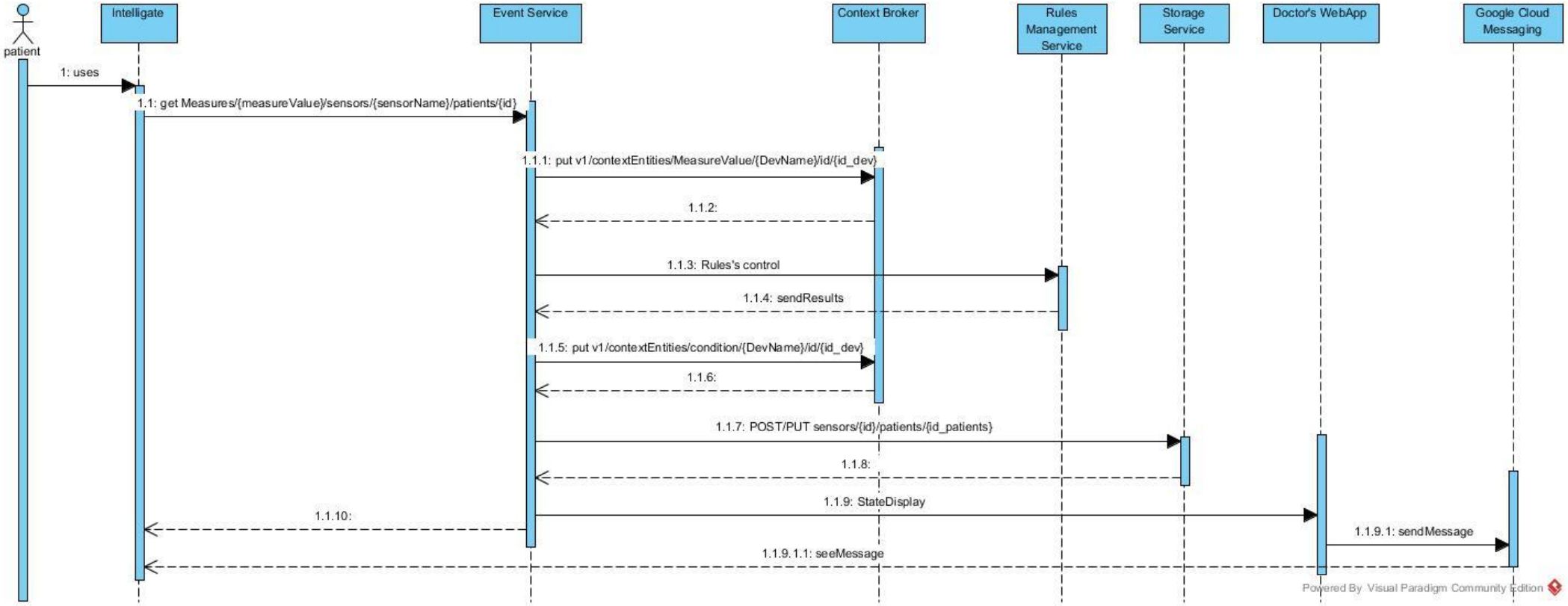


Σχήμα 12 – Διάγραμμα Δραστηριοτήτων (Activity Diagram) - Δυνατότητες ιατρού μέσω της εφαρμογής



Σχήμα 13 – Διάγραμμα Δραστηριοτήτων (Activity Diagram) - Δυνατότητες ασθενούς στο Intelligate

Στο Σχήμα 14 παρακάτω, βλέπουμε τη σειρά των γεγονότων από την εφαρμογή του ασθενούς (intelligate) έως το Υπολογιστικό Νέφος, ώστε να ενημερωθεί ο ιατρός για την «παραβίαση» των κανόνων που έχει ορίσει για κάποιον ασθενή του και να μπορέσει να αποστείλει κάποια ειδοποίηση-μήνυμα στον ασθενή του. Η λειτουργικότητα αυτή αφορά στην υπηρεσία “Event Service” (που αναλύεται στο Κεφάλαιο 4.3), η οποία λαμβάνει το μήνυμα από την εφαρμογή του ασθενή και επικοινωνεί με τις υπόλοιπες υπηρεσίες του Υπολογιστικού Νέφους. Όπως μπορούμε να δούμε και στο Σχήμα 14, μια νέα μέτρηση έρχεται από τη διεπαφή του ασθενούς (1:1 REST κλήση getMeasures με βάση κωδικό αισθητήρα και ασθενή) όταν ο ασθενής ενεργοποιήσει την επιλογή On Time ή On Demand. Η «Υπηρεσία Διαχείρισης Γεγονότων - Event Service» του Υπολογιστικού Νέφους αναλαμβάνει να ενημερώσει (update) την αλλαγή στον Context Broker, στον οποίον έχει γίνει η δημιουργία της οντότητας του αισθητήρα, από την Υπηρεσία «Διαχείρισης Συμβάντων και Συνδρομών – Publish Subscribe Service» (Κεφάλαιο 4.3) σε προηγούμενο βήμα. Στη συνέχεια (το Event Service) ενεργοποιεί τον έλεγχο της τιμής στην οντότητα του Context Broker στέλνοντας μήνυμα στην «Υπηρεσία Διαχείρισης Κανόνων – Rules Management Service» (αναλυτική περιγραφή της στην Ενότητα 4.3.1). Ο έλεγχος της τιμής γίνεται με βάση τους κανόνες που έχει ορίσει ο ιατρός για το συγκεκριμένο ασθενή και οι οποίοι έχουν καταχωρηθεί στη βάση δεδομένων του Υπολογιστικού Νέφους. Εάν η τιμή της μέτρησης είναι εκτός των επιτρεπτών ορίων που έχουν τεθεί, τότε έχουμε «παραβίαση κανόνα». Η υπηρεσία Event Service λαμβάνει μήνυμα από την Υπηρεσία Διαχείρισης κανόνων ώστε να ενημερώσει την οντότητα του αισθητήρα που υπάρχει στον Context Broker, αλλάζοντας το στοιχείο «Κατάσταση». Για την διασφάλιση της διαλειτουργικότητας του Event Service, η ενημέρωση της κατάστασης στην οντότητα του αισθητήρα στον Context Broker ενεργοποιεί αυτόματα μία REST κλήση ενημέρωσης της Υπηρεσίας Διαχείρισης Δεδομένων (Storage Service) του Υπολογιστικού Νέφους. Εν συνεχεία εμφανίζεται στην Web εφαρμογή του ιατρού η ένδειξη «Κίνδυνος» για το συγκεκριμένο ασθενή μέσω αυτόματης ανανέωσης της πληροφορίας από τη βάση δεδομένων. Ο ιατρός στη συνέχεια μπορεί να αποστείλει μήνυμα στον ασθενή του δίνοντάς του τις απαραίτητες οδηγίες.

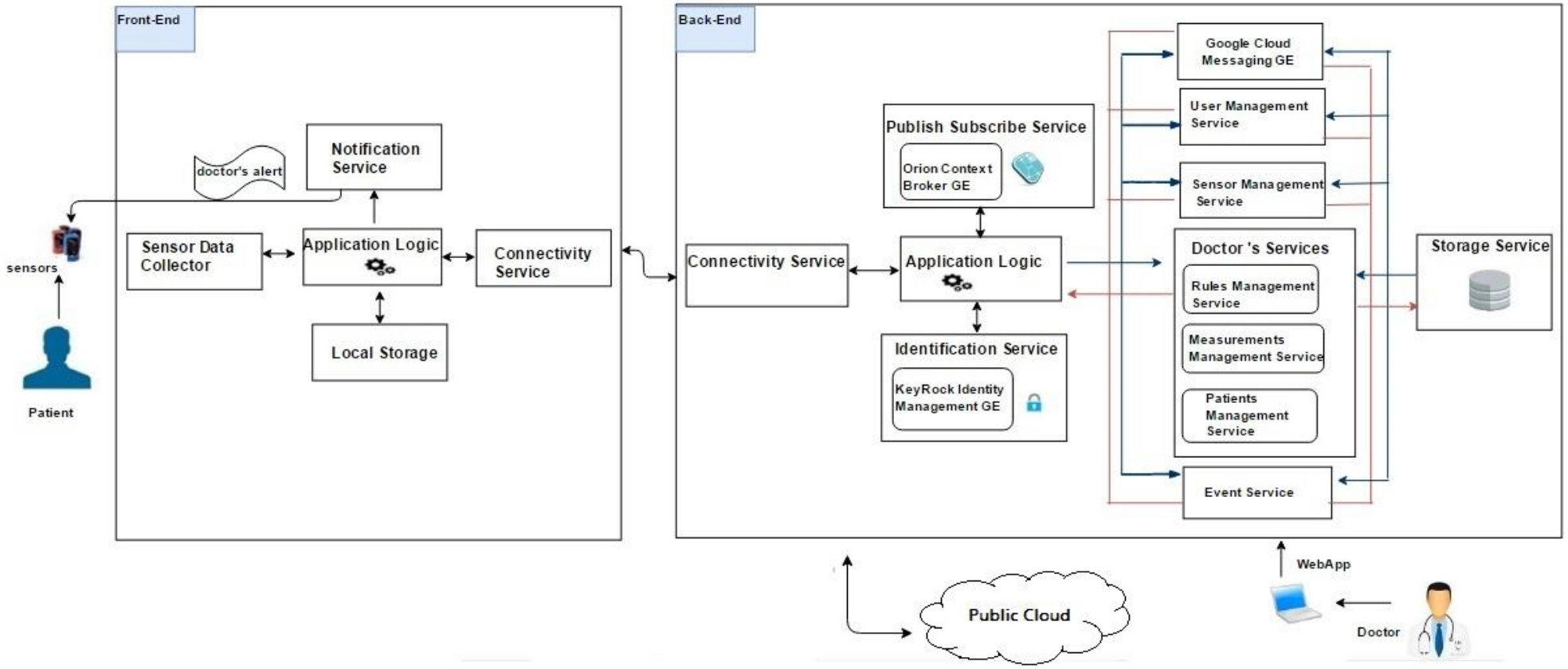


Σχήμα 14 – Διάγραμμα Αλληλεπίδρασης (Interaction - Sequence Diagram) για την Υπηρεσία Event Service

4 Υλοποίηση Συστήματος

4.1 Η αρχιτεκτονική του συστήματος

Σε αυτό το κεφάλαιο, αρχικά παρουσιάζουμε το αρχιτεκτονικό διάγραμμα του συστήματός μας, Σχήμα 15, δείχνοντας κυρίως τις υπηρεσίες από τις οποίες αποτελείται. Στη συνέχεια, αναλύουμε τα επιμέρους στοιχεία του/υπηρεσίες για κάθε κομμάτι του συστήματος, Διεπαφής (Front-End) και Υπολογιστικού Νέφους (Back-End).



Σχήμα 15 – Αρχιτεκτονική Συστήματος Παροχής Υπηρεσιών στο Υπολογιστικό Νέφος

4.2 Έξυπνη Πύλη διαδικτύου Ασθενών (Front-End Intelligate)

Το εν λόγω λειτουργικό κομμάτι αποτελείται από το σύστημα διεπαφής (User Interface), του ασθενούς από την εργασία «Monitoring and Processing of vital data through mobile platform in the "cloud"» (Intelligate)[22] .

Διεπαφή Ασθενούς: Ουσιαστικά αποτελείται από μια έξυπνη εφαρμογή η οποία υλοποιεί υπηρεσίες συλλογής πληροφορίας απο αισθητήρες διαχείρισης δεδομένων (προσωρινή αποθήκευση, κωδικοποίηση), μετάδοσης και επικοινωνίας με το Υπολογιστικό Νέφος. Η εφαρμογή επιπλέον διαθέτει μια διεπαφή χρήστη-ασθενούς με το σύστημα, που δίνει την δυνατότητα στον χρήστη-ασθενή να μπορεί να ελέγχει ανα πάσα στιγμή τις μετρήσεις των BLE αισθητήρων καθώς και να τροποποιήσει τον τρόπο αποστολής των δεδομένων, αλλα και να μπορεί να επικοινωνεί με τον ιατρό του μέσω μηνυμάτων.

Αποτελείται από 5 συγκεκριμένες ενότητες, όπως φαίνεται και στο σχήμα πάνω:

- **Υπηρεσία συλλογής δεδεομένων απο αισθητήρες (Sensor Data Collector):** Η συγκεκριμένη υπο-ενότητα έχει σαν στόχο την συλλογή των δεδομένων από τους αισθητήρες και την προώθηση αυτών των δεδομένων προς τις άλλες υπο-ενότητες του Intelligate.

- **Υπηρεσία τοπικής Αποθήκευσης (Local Storage Service):** Σε αυτήν την ενότητα γίνεται η τοπική αποθήκευση συγκεκριμένων τύπων δεδομένων. Χαρακτηριστικά το Local Storage Service περιλαμβάνει το XML σχήμα όλων των αισθητήρων που υποστηρίζονται από το συγκεκριμένο app,καθώς και αναγνωριστικά στοιχεία του ίδιου του ασθενούς(email) με τους κανόνες που τον διέπουν κωδικοποιημένα κατάλληλα. Τέλος περιλαμβάνει μέσους όρους μετρήσεων που παράγουν οι αισθητήρες σε συγκεκριμένα χρονικά διαστήματα σε περίπτωση όπου η κινητή συσκευή βρεθεί εκτός δικτύου. Με αυτόν τον τρόπο υπάρχει η διασφάλιση δεν υπάρχει χαμένη πληροφορία.

- **Υπηρεσία ειδοποιήσεων (Notification Service):** Το Notification Service έχει τον έλεγχο της εφαρμογής για πιθανές αλλαγές καταστάσεων, με στόχο την άμεση ενημέρωση του χρήστη. Παραδειγματικά μιλώντας όταν υπάρχουν δεδομένα που δεν έχουν αποσταλεί στο νέφος με το που συνδεθεί ο αισθητήρας με την συσκευή εμφανίζεται ένα σχετικό μήνυμα

[22] http://www.intelligence.tuc.gr/publications.php?pub_author=All&pub_type=8&pub_subject=All

«Υπάρχει διαθέσιμη πληροφορία που δεν έχει αποσταλεί στο νέφος, θέλετε να συνεχίσετε με την αποστολή των δεδομένων;» , αφήνοντας έτσι τον χρήστη να αποφασίσει αν θέλει η όχι να μεταφέρει τα δεδομένα του στο νέφος. Ένα επιπλέον παράδειγμα είναι η ενημέρωση του ασθενούς με ηχητική σήμανση(alert) όταν ο γιατρός στείλει ένα μήνυμα στην συσκευή του χρήστη-ασθενή.

- **Υπηρεσία διαχείρισης συνδέσεων (Connectivity Service - Gateway):** Εδώ γίνεται η επικοινωνία μεταξύ Gateway και Cloud (Back end).Τα δεδομένα αφού κωδικοποιηθούν καταλλήλως αποστέλλονται στο νέφος με βάση το πρωτόκολλο HTTP. Η Διαχείριση των απάντησεων από τον διακομιστή server είναι ευθύνη της συγκεκριμένης υπο-ενότητας (Πχ Σφάλμα 400 λανθασμένη αίτηση).

- **Λογική συστήματος (Application Logic - Gateway):** Η Λογική Συστήματος αποτελεί το σύνολο εκείνων των διεργασιών που έχουμε αναπτύξει προκειμένου οι υπηρεσίες που έχουμε χρησιμοποιήσει να μπορούν να επικοινωνούν μεταξύ τους αποτελεσματικά. Ορίζει τον τρόπο με τον οποίο επικοινωνούν οι υπηρεσίες ώστε η πληροφορία που παράγεται από το τμήμα των Παραγωγών να αξιοποιείται και να φθάνει στο τμήμα των Καταναλωτών. Θα μπορούσαμε να πούμε ότι η Λογική Συστήματος (Application Logic) είναι ο πυρήνας της εφαρμογής που συντονίζει όλες τις υπηρεσίες και τις λειτουργίες που μεσολαβούν μεταξύ της λήψης των δεδομένων και του γραφικού περιβάλλοντος της εφαρμογής στην οποία έχει πρόσβαση ο χρήστης [12].

4.3 Υπηρεσίες Στο Υπολογιστικό Νέφος (Back-End Intelligate)

Ουσιαστικά το Back-End Intelligate αποτελεί το βασικό κομμάτι της εργασίας μας, με στόχο τη διαχείριση, την επεξεργασία και την αποθήκευση των δεδομένων που έρχονται από το Intelligate, που αποτελεί τη διεπαφή του ασθενούς (κεφάλαιο 4.2). Ότι δεδομένα παράγουν οι αισθητήρες ανα χρονική στιγμή , στέλνονται στο Νέφος με βάση το πρότυπο σχήμα που έχει ορίσει ο γιατρός σε κάθε ασθενή του από το σικό του σύστημα διεπαφής.

Περιγράφονται στη συνέχεια, τα επιμέρους κομμάτια-υπηρεσίες που απαρτίζουν το Back-End Intelligate:

- **Λογική Συστήματος (Application Logic):** Εδώ γίνεται η ενορχήστρωση όλων των υπηρεσιών. Για να πραγματοποιηθεί οποιαδήποτε λειτουργία στο Υπολογιστικό Νέφος (Back-End), τα αιτήματα περνούν από αυτό το τμήμα για να σταλούν σε άλλη υπηρεσία η οποία θα παρέχει την «απάντηση» (response). Οι υπηρεσίες που ενορχηστρώνει το Application Logic είναι: η Υπηρεσία Διαχείρισης Χρηστών, η Υπηρεσία Διαχείρισης Συμβάντων και Συνδρομών, η Υπηρεσία Διαχείρισης Γεγονότων, η Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης χρηστών, η Υπηρεσία Διαχείρισης Κανόνων, η Υπηρεσία Διαχείρισης Μετρήσεων , η πηρεσίας Διαχείρισης Ασθενών, η Υπηρεσίας Διαχείρισης Συνδέσεων, η υπηρεσία Διαχείρισης αισθητήρων και η Υπηρεσίας Διαχείρισης Μηνυμάτων, οι οποίες περιγράφονται με λεπτομέρεια παρακάτω. Κάποιες από αυτές τις υπηρεσίες παρέχονται από τρίτους και συγκεκριμένα μέρος της Υπηρεσίας Διαχείρισης Συνδρομών και η Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών παρέχονται από την πλατφόρμα του Fiware, ενώ η Υπηρεσία Αποστολής Μηνυμάτων είναι υπηρεσία του υπολογιστικού νέφους της Google.

- **Υπηρεσία Διαχείρισης Δημοσιεύσεων και Συνδρομών (Publish/Subscribe Service):** Η Υπηρεσία Publish/Subscribe είναι υπεύθυνη για τη διαχείριση δημοσιεύσεων και συνδρομών που αφορούν στις μετρήσεις των αισθητήρων και στην περίπτωσή μας χρησιμοποιείται για την υποστήριξη της Υπηρεσίας Διαχείρισης Γεγονότων με τον τρόπο που αναφέρεται παρακάτω. Οι αισθητήρες καταχωρούνται ως οντότητες (Published Entities) στη συγκεκριμένη υπηρεσία και οι ιατροί αποτελούν τους συνδρομητές (Subscribers) σε αυτές τις οντότητες. Η υπηρεσία αυτή χρησιμοποιεί τη δημόσια υπηρεσία Orion Context Broker του περιβάλλοντος του Fiware για την αποθήκευση των οντοτήτων των αισθητήρων και για την αυτοματοποίηση της αποστολής ενημερώσεων προς την Υπηρεσία Αποθήκευσης του Back-End. Όταν εισάγουμε έναν καινούριο αισθητήρα για συγκεκριμένο ασθενή ενεργοποιείται η δημιουργία μίας οντότητας στον Context Broker, από την Υπηρεσία Διαχείρισης Δημοσιεύσεων και Συνδρομών. Η οντότητα αυτή περιέχει όλα τα στοιχεία για το όνομα της μέτρησης ή των μετρήσεων που διαθέτει ο αισθητήρας, την ημερομηνία της μέτρησης, την ώρα έναρξης και την ώρα λήξης της μέτρησης, την κατάσταση του ασθενούς, καθώς και το όνομα του ασθενούς με τον οποίο συνδέεται. Επιπλέον, δημιουργείται και η συνδρομή (Subscription) του ιατρού που παρακολουθεί τον ασθενή για τη συγκεκριμένη οντότητα-αισθητήρα.

- **Υπηρεσία Διαχείρισης Γεγονότων (Event Service):** Είναι η Υπηρεσία που χρησιμοποιούμε για τη διαχείριση της μέτρησης που έρχεται από τη διεπαφή του ασθενούς. Η υπηρεσία αυτή στηρίζεται στην Υπηρεσία Διαχείρισης Δημοσιεύσεων και Συνδρομών. Όταν δημιουργείται μια νέα μέτρηση, από την Υπηρεσία Συλλογής Δεδομένων Αισθητήρων (Ενότητα 4.2) της διεπαφής του ασθενούς, η Υπηρεσία Διαχείρισης Γεγονότων ενημερώνει την αλλαγή της τιμής της μέτρησης στην αντίστοιχη οντότητα του αισθητήρα στον Context Broker και ενεργοποιεί τον έλεγχο των κανόνων που έχουν οριστεί από τον ιατρό. Η μέτρηση ελέγχεται με βάση τους κανόνες που έχουν οριστεί από τον ιατρό στην Υπηρεσία Διαχείρισης Κανόνων - Rules Management (αναλυτική περιγραφή της στο κεφάλαιο 4.3.1). Γίνεται έλεγχος εαν η τιμή είναι μέσα στο όρια των κανόνων αυτών και αν ξεφεύγει από τα όρια έχουμε «Παραβίαση Κανόνα». Στην περίπτωση παραβίασης, η Υπηρεσία Διαχείρισης Κανόνων καλεί την Υπηρεσία Διαχείρισης Γεγονότων με σκοπό την ενημέρωση της κατάστασης του ασθενούς στην οντότητα του Context Broker. Για την εξασφάλιση της διαλειτουργικότητας του Event Service, η ενημέρωση της κατάστασης στην οντότητα του αισθητήρα στον Context Broker ενεργοποιεί την REST κλήση ενημέρωσης της βάσης δεδομένων του Υπολογιστικού Νέφους. Στη συνέχεια έχουμε την αποστολή της ένδειξης «Κίνδυνος» για τον ασθενή, στη διεπαφή του ιατρού. Ο ιατρός τότε έχει την δυνατότητα αποστολής μηνύματος προς την συσκευή του συγκεκριμένου χρήστη με την βοήθεια του Google Cloud messaging GE, ο οποίος θα μεριμνήσει να στείλει (ακόμα και σε περίπτωση απουσίας δικτύου) μήνυμα στην συσκευή με την μορφή ενημέρωσης. Ο χρήστης ενημερώνεται με ηχητικό και παλμικό(δόνηση) μήνυμα και μπορεί να διαβάσει το μήνυμα του ιατρού και να προβεί στις κατάλληλες ενέργειες που τον συμβουλεύει ο ιατρός του. Η ένδειξη «Κίνδυνος», θα φύγει μόλις οι τιμές των μετρήσεων επανέλθουν στις κανονικές για τον ασθενή τιμές.

- **Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών – KeyRock Identity Manager (Identification service):** Εδώ γίνεται η ταυτοποίηση των χρηστών. Πραγματοποιείται σε 2 φάσεις. Αρχικά μέσω του KeyRock Identity Management GE και έπειτα μέσω της υπηρεσίας. Στην ουσία ο «διαχειριστής» έχει τη δυνατότητα να ταυτοποιήσει τον χρήστη ώστε να έχει πρόσβαση στις λειτουργίες του. Ο χρήστης πρέπει να είναι εγγεγραμμένος στο FIWARE LAB, και εν συνεχεία παίρνει πρόσβαση και στις λειτουργίες της εφαρμογής.

- **Υπηρεσία Αποθήκευσης Δεδομένων (Storage service):** Αποθηκεύει τις πληροφορίες των χρηστών, των αισθητήρων καθώς και το ιστορικό των δεδομένων που έχει σταλεί από κάποιον αισθητήρα σε συνδυασμό με την ημερομηνία μέτρησης και την κατάσταση του ασθενούς. Οι κανόνες που έχει ορίσει ο ιατρός και οι μετρήσεις που έρχονται από τον ασθενή αποθηκεύονται και αυτά σε αυτή την υπηρεσία. Για την υλοποίηση της υπηρεσίας αποθήκευσης (Storage Service), χρησιμοποίησαμε τη MySql DB

λόγω της εύκολης δημιουργίας πιο σύνθετων queries και την παροχή της δημιουργίας περιορισμών που προσδίδουν καλύτερο έλεγχο της βάσης.

- **Υπηρεσίες Διαχείρισης Ιατρού (Doctor's Services):** Εδώ περιέχονται οι υπηρεσίες που διαθέτει ο ιατρός μέσω της διεπαφής του. Αυτές είναι: Η Διαχείριση των Κανόνων, η Διαχείριση των Μετρήσεων και η Διαχείριση των Ασθενών του και περιγράφονται αναλυτικά στο επόμενο υποκεφάλαιο 4.3.1.

- **Υπηρεσία Διαχείρισης Συνδέσεων (Connectivity Service - Back end):** Αποτελεί τον δίαυλο επικοινωνίας μεταξύ Διεπαφής (Front-End) και Νέφους (Back-end). Τα δεδομένα που εισέρχονται από την ενότητα Λογικής Συστήματος, Application Logic, του front-end μετατρέπονται σε JSON μορφή. Αυτά τα δεδομένα μεταφέρονται από την συσκευή στο Υπολογιστικό Νέφος (και αντίστροφα) μέσω HTTP αιτημάτων προς τον διακομιστή. Πιο συγκεκριμένα, χρησιμοποιείται για τη μετατροπή των μετρήσεων που έρχονται από τη διεπαφή του χρήστη –ασθενούς οι οποίες είναι σε JSON μορφή ώστε να αποσταλλούν στο Υπολογιστικό Νέφος και στην κατάλληλη Υπηρεσία που είναι αυτή της Διαχείρισης Γεγονότων. Επίσης χρησιμοποιείται για τους ασθενείς που πρέπει να μεταφερθούν στη βάση MySQL του Υπολογιστικού Νέφους. Οι χρήστες-ασθενείς έχουν τη δυνατότητα να εγγράφονται μόνο μέσω της εφαρμογής που είναι εγκατεστημένη στη συσκευή-tablet, όταν πραγματοποιήσουν την εγγραφή τους αποθηκεύονται στον JSON Storage και συγκεκριμένα στη συλλογή "USERS" με το email τους, που είναι μοναδικό. Από τον JSON Storage οι χρήστες-ασθενείς μεταφέρονται στην ενιαία βάση MySQL και συμπληρώνονται τα υπόλοιπα στοιχεία, πέραν του email τους, από το διαχειριστή του συστήματος ώστε να διατηρείται όλη η πληροφορία συγκεντρωμένη στη βάση του Back-end. Επίσης, η επόμενη πληροφορία που ανταλλάσσεται μεταξύ Front-end και υπολογιστικού νέφους (Back-end) είναι αυτή των κανόνων. Δηλαδή, ο ασθενής για να μπορέσει να πάρει μία μέτρηση από κάποιον αισθητήρα, θα πρέπει πρώτα μέσω τις διεπαφής του να παραλάβει τους κανόνες που έχει ορίσει ο ιατρός για εκείνον. Αυτούς τους κανόνες η «Λογική Συστήματος» του Front-end τους αναζητά στη συλλογή "SensorsRules" του JSON Storage. Επομένως, η «Υπηρεσία Διαχείρισης Κανόνων», που αναλύεται στην ενότητα 4.3.1, αποστέλλει/ενημερώνει τους κανόνες που ορίζει ο ιατρός στη συγκεκριμένη συλλογή του JSON Storage, σε JSON μορφή, με τη βοήθεια της Υπηρεσία Συνδέσεων, ώστε να είναι αναγνωρίσιμοι οι κανόνες από την εφαρμογή.

- **Υπηρεσία Διαχείρισης Χρηστών (User Management Service):** Υπηρεσία για την επεξεργασία, διαγραφή, και προβολή όλων των στοιχείων και των χαρακτηριστικών των χρηστών, όπως είναι η προβολή των αισθητήρων που με τους οποίους είναι συνδεδεμένος ένας χρήστης-ασθενής, ή τον ιατρό που τον παρακολουθεί.

- **Υπηρεσία Διαχείρισης Αισθητήρων (Sensor Management service):** Είναι υπηρεσία υπεύθυνη για την εισαγωγή, την επεξεργασία, την προβολή και τη διαγραφή των διαθέσιμων αισθητήρων-συσκευών. Συγκεκριμένα, ο διαχειριστής αναλαμβάνει να εισάγει ένα XML αρχείο. Ουσιαστικά το XML αυτό αρχείο διαθέτει όλη την χρήσιμη πληροφορία για τους αισθητήρες ούτως ώστε να μπορούν να αναγνωριστούν από την εφαρμογή Intelligate, που διαθέτει ο ασθενής. Με την εισαγωγή ενός XML σχήματος δίνεται η δυνατότητα δυναμικής εισαγωγής νέων σχημάτων αισθητήρων για την αναγνώριση των τιμών των χαρακτηριστικών τους όπως η μέτρηση του καρδιακού παλμού. Η περιγραφή του XML σχήματος περιγράφεται παρακάτω, σε δεντρική μορφή:
  - **<sensor> :** Εδώ περιλαμβάνονται όλες οι BLE συσκευές-αισθητήρες που υποστηρίζει η συσκευή.
  - **<device> :** Ένας συγκεκριμένος αισθητήρας όπου χαρακτηρίζεται από την οικογένεια του , device family (πχ Polar H7).
  - **<values>:** Αναφέρονται στο όνομα της μέτρησης καθώς και στις μονάδες μέτρησης της. Για παράδειγμα η συσκευή Polar H7 παράγει μέτρηση καρδιακού παλμού με μονάδα μέτρησης τα zbpm. Ενώ η συσκευή NoninOnix παράγει μέτρηση καρδιακού παλμού με μονάδα μέτρησης τα bpm.
  - **<format>:** Αποτελεί το τύπο (format) που είναι ορισμένες οι τιμές που παράγουν οι αισθητήρες.(Πχ FORMAT_UINT8- ακέραιος 8 bit).
  - **<position> & <multi>**: Πολλές φορές οι κατασκευαστές των BLE αισθητήρων ορίζουν μια πιο περίπλοκη συνάρτηση ώστε να μπορεί να αποκωδικοποιηθεί το χαρακτηριστικό το οποίο μας ενδιαφέρει. Πιο συγκεκριμένα η τιμή μπορεί να αναγνωρίζεται από έναν συγκεκριμένο λογικό συνδυασμό διάφορων τιμών ενός χαρακτηριστικού. Στη συγκεκριμένη περίπτωση βοηθούν στην δυναμική διαχείριση του συνδυασμού τιμών χαρακτηριστικών. προσδιορίζοντας την θέση της τιμής εντός του χαρακτηριστικού (<position>) και πολλαπλασιάζοντας το κατάλληλα με μια ακέραια τιμή (<multi>).

Για να γίνει πιο κατανοητή η χρήση του XML σχήματος παραθέτουμε το ήδη υπάρχον XML σχήμα υποστήριξης των αισθητήρων Polar H7 και NoninOnix 3250.

```
<? xml version="1.0" encoding="UTF-8" ?>
  <sensors>
        <device name="NoninOnix">
            <uuid id="0aad7ea0-0d60-11e2-8e3c-0002a5d5c51b">
                <values name="SPO2" type="%">
                    <format>FORMAT_UINT8</format>
                    <position>7</position>
                    <multi>1</multi>
                </values>
                <values name="pulse_Rate" type="zbpm" picture="1">
                    <format>FORMAT_UINT8</format>
                    <position>[8,9]</position>
                    <multi>[256,1]</multi>
                </values>
            </uuid>
        </device>

              <device name="Polar H7">

            <uuid id="00002a37-0000-1000-8000-00805f9b34fb">
                <values name="pulse_Rate" type="bpm">
                    <format>FORMAT_UINT8</format>
                    <position>1</position>
                    <multi>1</multi>
                </values>
            </uuid>

        </device>
 </sensors>
```

Έχουμε 2 συσκευές, η πρώτη είναι η NoninOnix με ένα UUID που αποτελεί το αναγνωριστικό της, η οποία παράγει το εύρος ποσοστού οξυγόνου στο αίμα (Spo2) καθώς και τους καρδιακούς παλμούς(pulse_Rate) τις μονάδες μέτρησής τους καθώς και το format. Αντίστοιχα η δεύτερη συσκευή είναι η Polar H7 ή οποία παράγει μόνο τον καρδιακό ρυθμό με ένα διαφορετικό αναγνωριστικό UUID. Το XML σχήμα κάθε αισθητήρα με την παραπάνω μορφή μπορεί να εισαχθεί από τον διαχειριστή ανα πάσα στιγμή και να συνδεθεί η συσκευή με κάποιο ασθενή. Μια συσκευή συνδέεται με έναν ασθενή. Ωστόσο εαν αποσυνδεθεί από τον ασθενή μπορεί στη συνέχεια να συνδεθεί με άλλον. Δηλαδή αισθητήρες που χρησιμοποιήσαμε στο παρελθόν και το σύστημά μας ξέρει να διαβάζει την πληροφορία τους μπορούν να ξαναχρησιμοποιηθούν.

- **Υπηρεσία Αποστολής Μηνυμάτων (Google Cloud Messaging GE):** Είναι υπηρεσία υπεύθυνη για τη διαχείριση των μηνυμάτων του ιατρού προς τον χρήστη-ασθενή. Αποτελεί μια υπηρεσία γενικού σκοπού όπου μέσω της διεπαφής του ιατρού, δίνεται η δυνατότητα να αποστάλει μηνύματα προς τους ασθενείς του σε πραγματικό χρόνο (real-time). Το Google cloud messaging GE**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** είναι μια δωρεάν υπηρεσία που βοηθά τους προγραμματιστές να στέλνουν μηνύματα σε πολλές πλατφόρμες. Για την προσαρμογή του συγκεκριμένου GE στις δικές μας ανάγκες πραγματοποιήθηκαν, στην

προηγούμενη εργασία Intelligate, που αποτελεί και τη βάση στην οποία στηρίχθηκε η παρούσα εργασία, τα παρακάτω βήματα:

Ενεργοποίηση του GCM από της πλατφόρμα PAAS της GOOGLE και λήψη κωδικού αποστολής (senderID) και κωδικού API (API key).

1. Ανοίγουμε την σελίδα https://console.developers.google.com/project/intelligate-7dd89?pli=1 που ουσιαστικά είναι η Google APIs Console page
2. Αν δεν έχουμε ένα API project , τότε δημιουργούμε ένα καινούργιο.
3. Έπειτα ενεργοποιούμε την υπηρεσία google cloud messaging , και λαμβανουμε το SenderID που μας προσφέρει μέσω του url .
4. Επιλέγουμε την ενότητα API access , και στο κάτω μέρος της συσκευής φαίνεται ο κωδικός API key όπου θα χρησιμοποιηθεί από τον διακομιστή μας για να στείλει μέσω του GCM μηνύματα σε κάθε διαφορετική συσκευή.

Στη συνέχεια έχουμε τη δημιουργία κώδικα από μεριάς διακομιστή (server side code) για αποθήκευση του registration id της συσκευής , και την προώθηση μηνυμάτων στην συσκευή.

Υλοποιείται με την δημιουργία ορισμένων php αρχείων στο Υπολογιστικό Νέφος, που βοηθούν σε αυτήν τη λειτουργία. Έχουμε δύο αρχεία τα οποία είναι:

1. InsertDeviceReg.php : Λαμβάνει ένα JSON String από την συσκευή και το εισάγει αυτούσιο στη συλλογή USERS του JSON STORAGE.Το Json string περιλαμβάνει το όνομα χρήστη καθώς και το αναγνωριστικό της συσκευής (Registration ID).

{
"name": "user1@gmail.com",
"reg_id":"APA91bE1U9LUy7J2mfwvCRUMPOosVpbN2kybRJ_Dk_Upz7eUu6G1g
cSkMI87tQwZz8pzvTNNZCMgsMCfwrL7umcdF-
Y5BNWJHtr_1ARF8rngYOKmC1yI-W0H1QAsyluhltJ5PX-yaEmi"
}

2. GCM/index.php: Περιλαμβάνει ένα κατάλληλο text_area για να μπορέσει ο ιατρός να εισάγει το μήνυμα του, και σε αυτό το php αρχείο είναι επιπλέον αποθηκεμένο το API key με το οποίο γίνεται η ταυτοποίηση.

4.3.1 Υπηρεσίες χρήστη - ιατρού (Doctor's Services)

Πρόκειται για τις υπηρεσίες που υλοποιούν τη λειτουργικότητα των χρηστών-ιατρών. Σε αυτές περιλαμβάνονται:

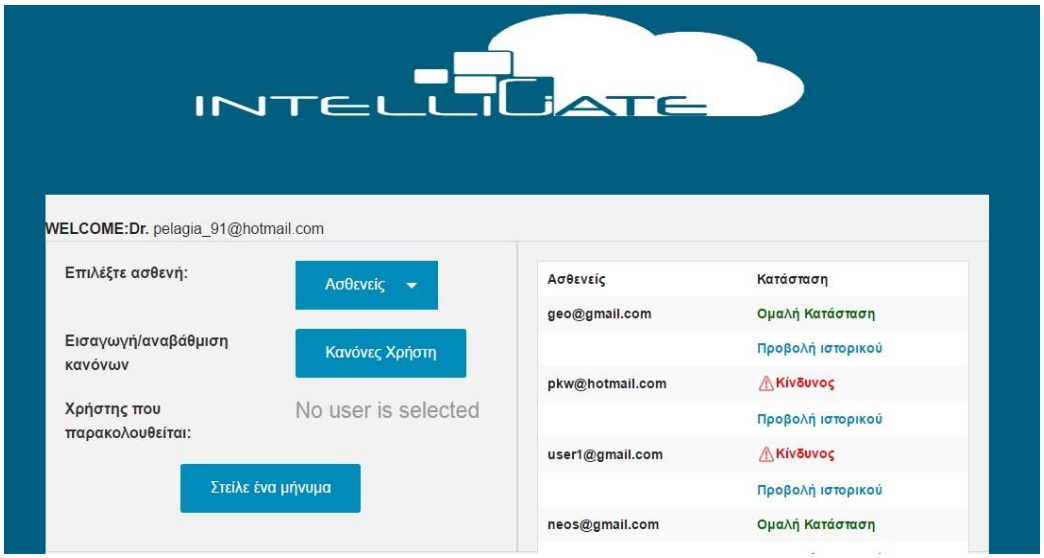
- **Υπηρεσία Διαχειρίσης Κανόνων (Rules Management Service):** Ο ιατρός μέσω της διεπαφής του που φαίνεται στο Σχήμα 16, ορίζει κάποιους κανόνες για κάθε ασθενή που παρακολουθεί. Οι κανόνες ουσιαστικά αποτελούν τα κατώφλια (άνω και κάτω κατώφλι), καθώς προσφέρουν μια μέγιστη επιτρεπτή τιμή και μια ελάχιστη επιτρεπτή τιμή οπου στο ενδιάμεσο αυτόν των τιμών προσδιορίζεται η ομαλή κατάσταση ενός ασθενή. Παραδειγματικά μιλώντας ένας χρήστης μπορεί να έχει ως μέγιστο κατώφλι τα 120 bmp καρδιακού ρυθμού και ως ελάχιστο κατώφλι τα 80 bmp καρδιακού παλμού. Αν η τιμή της μέτρησης του ασθενούς ξεφύγει από αυτά τα όρια τότε ο χρήστης σημαίνει ότι παραβιάζει αυτόν τον κανόνα που ορίστηκε για αυτόν από τον ιατρό του και πλέον η κατάσταση του θεωρείται κρίσιμη. Στο βήμα αυτό αν υπάρχει ήδη για κάποιον χρήστη-ασθενή κανόνας, τότε απλά ανανεώνεται. Επιπλέον, διατίθεται τη δυνατότητα στον ιατρό εαν κρίνει απαραίτητο να αλλάξει τους κανόνες ή να δημιουργήσει νέους για τους διαθέσιμους αισθητήρες, Σχήμα 18. Τέλος, μπορεί να κάνει ανάκτηση των πρόσφατων κανόνων του ασθενούς του.

- **Υπηρεσία Διαχείρισης Μετρήσεων (Measurements Management Service):** Ο ιατρός έχει τη δυνατότητα να βλέπει το ιστορικό των ασθενών που παρακολουθεί, όπως φαίνεται και στο Σχήμα 17, παρακάτω. Δηλαδή τις πιο πρόσφατες αλλά και τις παλαιότερες μετρήσεις του ανα ημερομηνία μέτρησης, με τα στοιχεία όπως το όνομα της μέτρησης, ώρα έναρξης της μέτρησης και ώρα λήξης της μέτρησης.

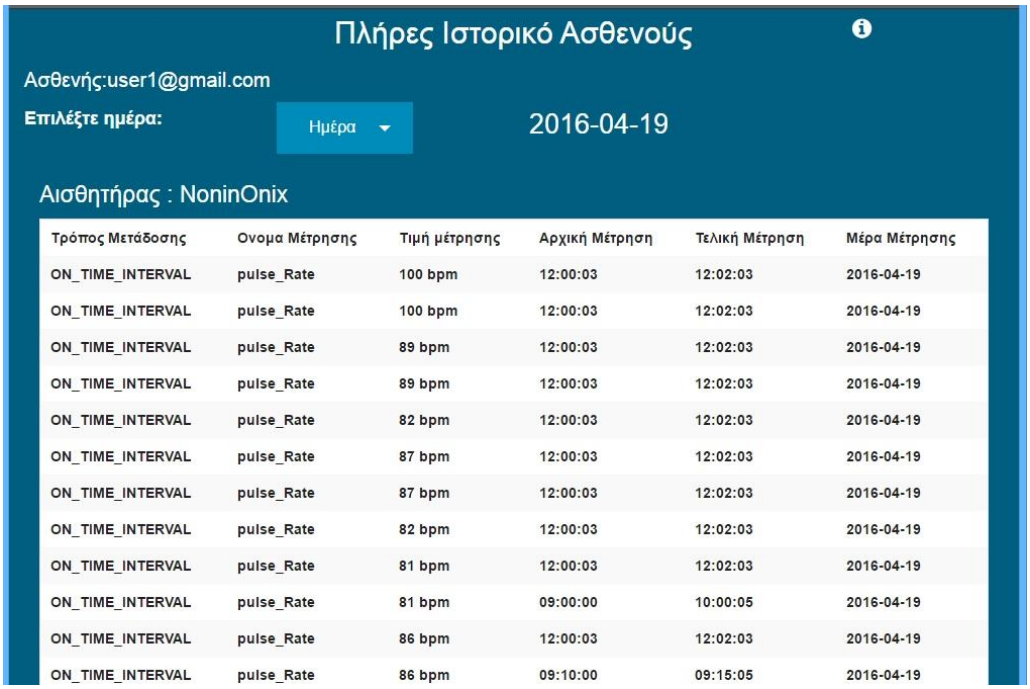
- **Υπηρεσία Διαχείρισης Ασθενών από τον ιατρό τους (Patients Management Service):** Ο ιατρός μέσω της διεπαφής που διαθέτει μπορεί να δει συνολικά πόσους και ποιούς ασθενείς παρακολουθεί μαζί με την κατάσταση στην οποία βρίσκονται (ομαλή ή σε κίνδυνο), αλλά και να επιλέξει να στείλει μήνυμα σε κάποιον ασθενή του. Επιπλεόν μπορεί να δει όλες τις διαθέσιμες πληροφορίες του ασθενούς του , όπως όνομα, επίθετο, ημερομηνία γέννησης, κατοικία, ασθένεια, νοσηλεία ασθενούς (Σχήμα 16).

4.3.2 Διεπαφή χρήστη - ιατρού

Εδώ παρουσιάζονται τα στιγμιότυπα της διεπαφής του χρήστη- ιατρού. Οι ενέργειες και οι υπηρεσίες του έχουν περιγραφεί αναλυτικά στο Κεφάλαιο 4.3.1, με αναφορά στην αντίστοιχη εικόνα της διεπαφής του.

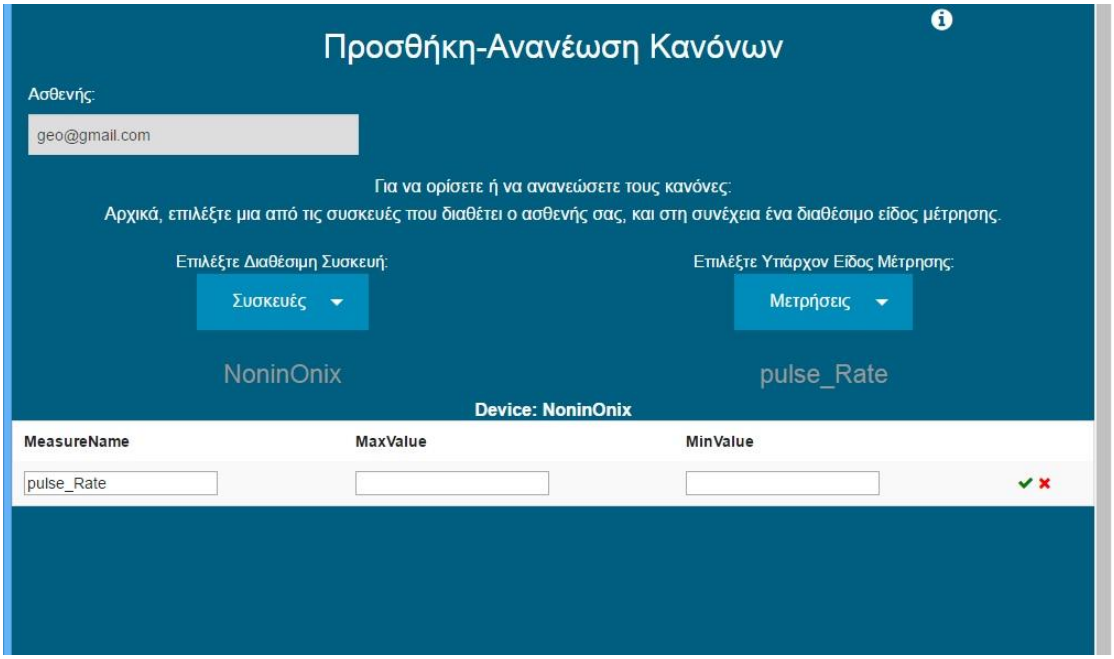


Σχήμα 16- Αρχική σελίδα διεπαφής χρήστη-ιατρού



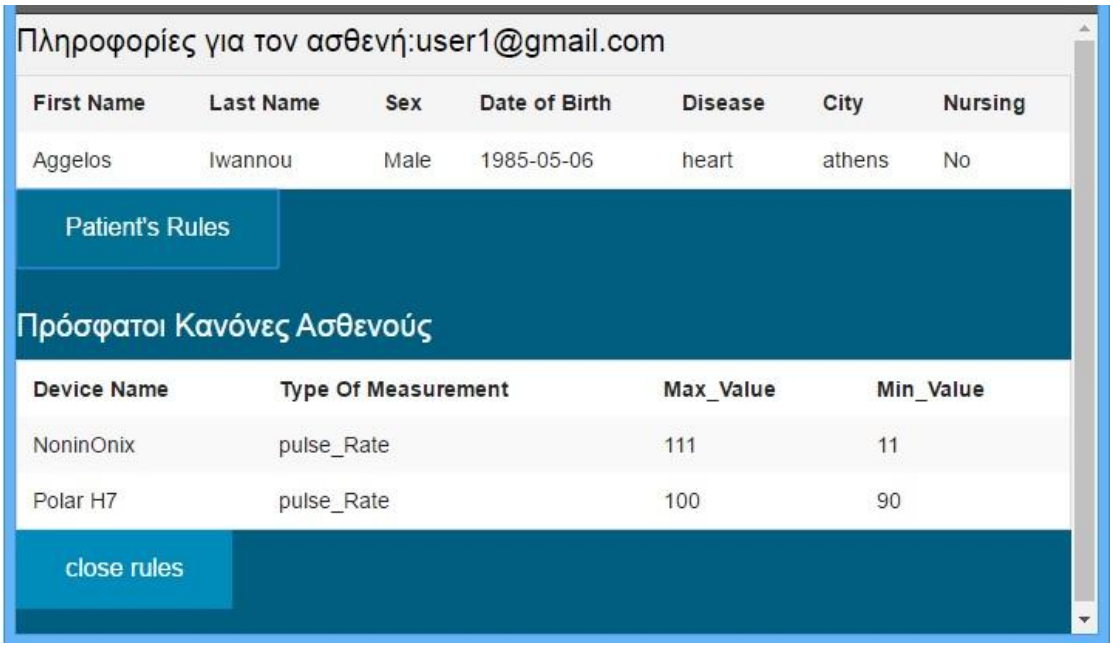
Τρόπος Μετάδοσης	Ονομα Μέτρησης	Τιμή μέτρησης	Αρχική Μέτρηση	Τελική Μέτρηση	Μέρα Μέτρησης
ON_TIME_INTERVAL	pulse_Rate	100 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	100 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	89 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	89 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	82 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	87 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	87 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	82 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	81 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	81 bpm	09:00:00	10:00:05	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	86 bpm	12:00:03	12:02:03	2016-04-19
ON_TIME_INTERVAL	pulse_Rate	86 bpm	09:10:00	09:15:05	2016-04-19

Σχήμα 17- Προβολή ιστορικού ασθενούς



Σχήμα 18- Προσθήκη-Ανανέωση κανόνων ασθενούς

Ο ιατρός, μπορεί να δει ανα πάσα στιγμή όλα τα στοιχεία των ασθενών του: όνομα, επίθετο, κατοικία, ηλικία, ασθένεια, νοσηλεία, καθώς και τους πιο πρόσφατους κανόνες που αφορούν τον ασθενή του για τις συσκευές με τις οποίες συνδέεται, όπως φαίνεται στο Σχήμα 19.

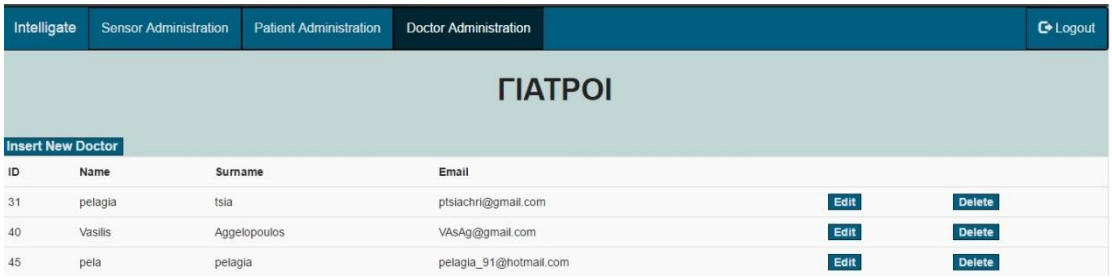


Σχήμα 19- Προβολή Πληροφοριών ασθενούς και Προβολή πρόσφατων κανόνων ασθενούς

4.3.3 Διεπαφή Διαχειριστή

Υπάρχει και η διεπαφή του διαχειριστή που χρησιμεύει στη διευκόλυνση της διαχείρισης του συστήματος. Ο διαχειριστής έχει τον πλήρη έλεγχο της εφαρμογής, μέσω αυτού γίνεται εισαγωγή χρηστών και αισθητήρων, η ανάθεση αισθητήρα σε ασθενή και ο συσχετισμός του με καποιον συγκεκριμένο γιατρό. Μπορεί να ανανεώσει ή να διαγράψει κάποιον χρήστη ή κάποιον αισθητήρα. Παρουσιάζονται και αναλύονται τα στιγμιότυπα της δικής του διεπαφής παρακάτω.

Ο διαχειριστής μπορεί να εισάγει έναν καινούριο ιατρό επιλέγοντας το κουμπί “InsertNew Doctor”, όπως φαίνεται και στο Σχήμα 20, όπου εμφανίζεται η φόρμα συμπλήρωσης (Σχήμα 21) , και συμπληρώνοντας τα στοιχεία του ιατρού τον εισάγει στο σύστημα. Μετά την εισαγωγή του ιατρού, μπορεί να κάνει επεξεργασία των στοιχείων του με την επιλογή “Edit” , Σχήμα 22, ή να διαγράψει εντελώς κάποια εγγραφή ιατρού επιλέγοντας “Delete”.

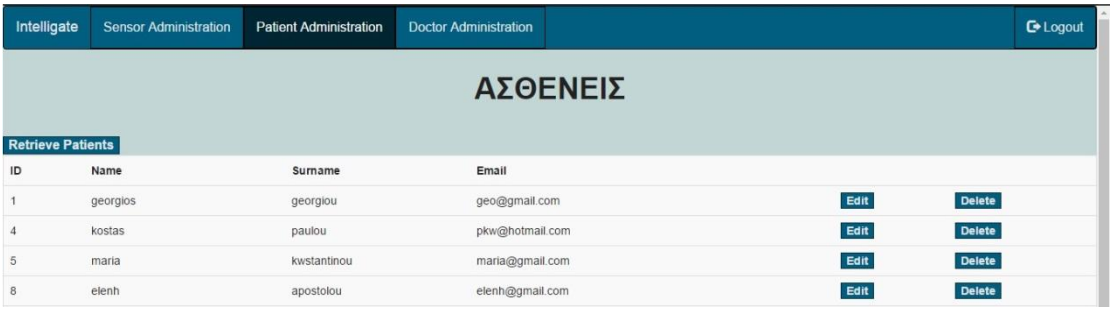


Σχήμα 20- Διαχείριση Ιατρών

Σχήμα 21- Εισαγωγή νέου Ιατρου

Σχήμα 22- Επεξεργασία Ιατρού

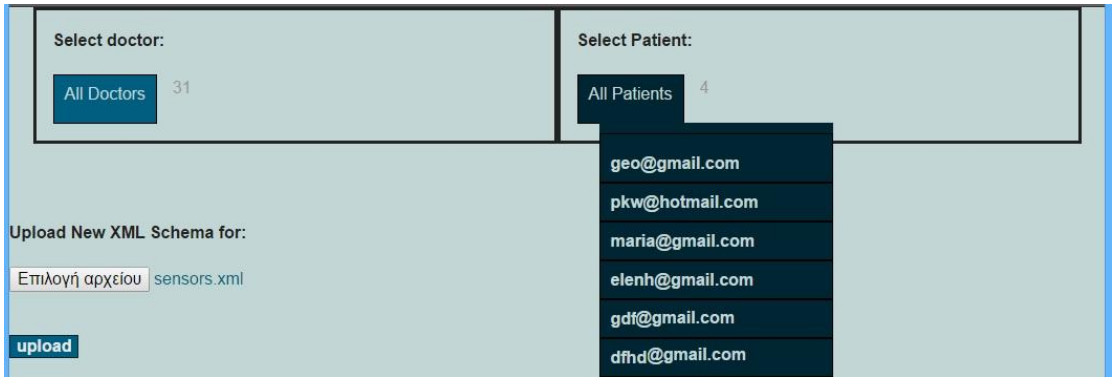
Επιπλέον, όπως βλέπουμε στο Σχήμα 23, ο διαχειριστής μπορεί να εισάγει κάποιον ασθενή. Η εισαγωγή του ασθενούς γίνεται με τον τρόπο που περιγράφτηκε στο Κεφάλαιο 4.3 (Storage service). Ο διαχειριστής εισάγει τους νέους χρήστες-ασθενείς που έχουν εγγραφεί μέσω της εφαρμογής στο tablet. Με την επιλογή του κουμπιού “Retrieve Patients” εισάγεται μόνο το email του ασθενούς στη βάση MySql και στη συνέχεια συμπληρώνει τα στοιχεία τους ο διαχειριστής επιλέγοντας “Edit”. Τέλος, μπορεί να διαγράψει κάποιον ασθενή επιλέγοντας “Delete”.



Σχήμα 23- Διαχείριση Ασθενών

Επίσης ο διαχειριστής μπορεί να προσθέσει έναν νέο αισθητήρα πατώντας στο κουμπί “Insert New Device”, όπως φαίνεται στο Σχήμα 24. Με την επιλογή αυτή μεταφέρεται στη σελίδα του Σχήματος 25. Εκεί έχει τη δυνατότητα να συσχετίσει μια συσκευή με κάποιον ασθενή αλλά και με τον ιατρό του. Συγκεκριμένα, επιλέγει “upload” για να φορτώσει το αρχείο που περιέχει το xml σχήμα του αισθητήρα και στη συνέχεια επιλέγει τον ασθενή για τον οποίο προορίζεται η χρήση του αισθητήρα αλλά και τον ιατρό που τον παρακολουθεί. Τέλος, ο διαχειριστής αφού προσθέσει τον αισθητήρα μπορεί να τον επεξεργαστεί με την επιλογή “Edit” ή μπορεί και να τον διαγράψει τελείως, επιλέγοντας “Delete” (Σχήμα 24).

Σχήμα 24- Διαχείριση Συσκευών-Αισθητήρων



Σχήμα 25- Εισαγωγή Αισθητήρα και Συσχέτιση με ασθενή και ιατρό

4.4 Περιγραφή του REST API του συστήματος

Η υλοποίηση της Λογικής Συστήματος περιλαμβάνει ένα εύχρηστο και ευρύ Restful API, το οποίο μέσα από κλήσεις μεθόδων επιτελεί συγκεκριμένες λειτουργίες. Παρουσιάζονται οι μεθόδοι του API για την : α) Υπηρεσία Διαχείρισης Χρηστών - User Management Service που περιλαμβάνει κλήσεις για τους ασθενείς (patients) και τους γιατρούς (doctors), β)Υπηρεσία Διαχείρισης Αισθητήρων - Sensor Management Service, γ) Υπηρεσία Διαχείρισης Μετρήσεων (Measurements Management Service), δ) Υπηρεσία Διαχείρισης Κανόνων (Rules Management Service), ε) Υπηρεσία Διαχείρισης Ασθενών από τον ιατρό τους (Patients Management Service), στ) Υπηρεσίας Διαχείρισης των Γεγονότων (Event Service), που υποστηρίζονται από τον Context Broker, ζ) Υπηρεσίας Διαχείρισης Συνδέσεων και η) Υπηρεσία Αποστολής Μηνυμάτων (GCM).

Αρχικά, στον πίνακα 1, βλεπουμε τις κλήσεις rest API που πραγματοποιούνται για την Υπηρεσία Διαχείρισης Χρηστών –Ιατρών (User Management Service)

Μέθοδος	URL	Περιγραφή
POST	/users/doctors	Δημιουργία νέου ιατρού
GET	/users/doctors	Ανάκτηση όλων των εγγεγραμμένων ιατρών
GET	/users/doctors/{id}	Προβολή ενός συγκεκριμενου ιατρού με βάση το {id} του
PUT	/users/doctors/{id}	Επεξεργασία ενός ιατρού με βάση το {id
DELETE	/users/doctors/{id}	Διαγραφή ενός ιατρού με βάση το {id}

Πίνακας 1-REST Διαχείρισης χρηστών-ιατρών

Έπειτα, στον πίνακα 2, βλεπουμε τις κλήσεις rest API που πραγματοποιούνται για την Υπηρεσία Διαχείρισης Χρηστών –Ασθενών (User Management Service)

Μέθοδος	URL	Περιγραφή
GET	/users/patients	Ανάκτηση όλων των εγγεγραμμένων ασθενών
GET	/users/patients/{id}	Προβολή ενός συγκεκριμενου ασθενούς με βάση το {id} του
GET	/users/patients/email/{email}	Ανάκτηση όλων των στοιχείων του ασθενούς με email={email}
PUT	/users/patients/{id}	Επεξεργασία ενός ασθενούς με βάση το {id
DELETE	/users/patients/{id}	Διαγραφή ενός ασθενούς με βάση το {id}

Πίνακας 2-REST Διαχείρισης χρηστών-ασθενών

Στον πίνακα 3, βλέπουμε τις κλήσεις rest API που πραγματοποιούνται για την Υπηρεσία Διαχείρισης Αισθητήρων (Sensor Management Service)

Μέθοδος	URL	Περιγραφή
POST	/users/sensors	Δημιουργία νέου σχήματος xml αισθητήρα
GET	/users/sensors	Ανάκτηση όλων των υπάρχουσων αισθητήρων
GET	/users/patients/{id_patient}/sensors/{id_sensor}	Προβολή ενός συγκεκριμενου αισθητήρα με βάση το {id_sensor} του, για κάποιον συγκεκριμένο ασθενή με id={id_patient}
DELETE	/users/sensors/{id_sensor}	Διαγραφή ενός αισθητήρα με βάση το {id_sensor} του
DELETE	/users/patients/{id_patient}/sensors/{id_sensor}	Διαγραφή ενός συγκεκριμένου αισθητήρα με {id_sensor} , κάποιου ασθενούς με id={id_patient}

Πίνακας 3-REST Διαχείρισης αισθητήρων

Στον πίνακα 4, βλέπουμε τις κλήσεις rest API που πραγματοποιούνται για την Υπηρεσία Διαχείρισης Μετρήσεων (Measurements Management Service)

Μέθοδος	URL	Περιγραφή
GET	/users/doctors/{id_doctor}/patients/{id}/sensorsMeasures	Ανάκτηση όλων των μετρήσεων του ασθενούς με id={id} , με γιατρό με id={id_doctor}
GET	/users/doctors/{id_doctor}/patients/{id}/sensorsMeasures/value	Προβολή μόνο των ειδών των μετρήσεων του ασθενούς με id={id_patient}, με γιατρό με id={id_doctor}
GET	/users/doctors/{id_doctor}/patients/{id}/sensors/sensorsMeasures/{dateM}	Προβολή μιας μέτρησης με βάση την ημερομηνία {dateM} για κάποιον ασθενή με {id_patient}

GET	/users/doctors/{id_doctor}/patients/{id}/sensors/{name}/Measures	Προβολή των μετρήσεων, με βάση το όνομα του αισθητήρα ενός ασθενούς με id={id}, με γιατρό με id={id_doctor}

Πίνακας 4-REST Διαχείρισης μετρήσεων

Στον πίνακα 5, βλέπουμε τις κλήσεις rest API που πραγματοποιούνται για την Υπηρεσία Διαχείρισης Κανόνων (Rules Management Service).

Μέθοδος	URL	Περιγραφή
GET	/users/doctors/{id_doctor}/patients/{id}/sensorsRules	Προβολή κανόνων ασθενούς με id={id} , με γιατρό με id={id_doctor}
GET	/users/doctors/{id_doctor}/patients/{id}/sensorRules/{value}	Προβολή των κανόνων για ένα συγκεκριμένο ειδος μέτρησης με {value}, του ασθενούς με id={id_patient}, με γιατρό με id={id_doctor}
GET	/users/doctors/{id_doctor}/patients/{id}/sensors/sensorsMeasures/{dateM}	Προβολή του πιο πρόσφατου κανόνα ασθενούς με {id_patient} και γιατρό με id={id_doctor}
POST	/users/doctors/{id_doctor}/patients/{id}/sensors/{name}/AddMeasures	Εισαγωγή κανόνων για αισθητήρα με nameDevice={name} , ασθενούς με {id} και γιατρό με id={id_doctor}
PUT	/users/doctors/{id_doctor}/patients/{id}/sensors/{name}/Measures/{value}	Ενημέρωση κανόνων αισθητήρα με nameDevice={name} , και για μέτρηση με measureValue={value} του ασθενούς με {id} και γιατρό με id={id_doctor}

Πίνακας 5-REST Διαχείρισης κανόνων

Στον πίνακα 6, βλέπουμε τις κλήσεις rest API που πραγματοποιούνται για την Υπηρεσία Διαχείρισης Ασθενών από τον ιατρό τους (Patients Management Service)

Μέθοδος	URL	Περιγραφή
GET	/users/doctors/{id_doctor}/patients/	Ανάκτηση όλων των ασθενών του συγκεκριμένου ιατρού με id={id_doctor}
GET	/users/doctors/{id_doctor}/patients/{id}	Προβολή ασθενούς με id={id_patient}, από τον ιατρό του με id={id_doctor}
GET	/users/doctors/{id_doctor}/patients/{id}/{patient's info}	Προβολή όλων των στοιχείων κάποιου ασθενούς με {id_patient}, από τον ιατρό του με id={id_doctor}

Πίνακας 6-REST Διαχείρισης ασθενών από τον ιατρό τους

Στον πίνακα 7, βλέπουμε τις κλήσεις rest API του Context Broker και έχουν να κάνουν με τη διαχείριση των γεγονότων. Δηλαδή, αποτελούν μέρος της υπηρεσίας διαχείρισης των γεγονότων (Event Service), υποστηρίζονται από τον Context Broker και επικοινωνούν με το υπόλοιπο Event Service, όπως περιγράφηκε στο κεφάλαιο 4.3.

Μέθοδος	URL	Περιγραφή
GET	/sensors_subscriptions	Επιστρέφει όλες τις οντότητες με τα στοιχεία των αισθητήρων
POST	/sensors-subscriptions	Δημιουργία καινούριας οντότητας
PUT	/sensors_subscriptions/sensor/{id_sensor}/device_name/{name}/patients/{id_patient}/date_ofMeasure/{date}	Ανανέωση της ημερομηνίας {date }, για την τιμή της μέτρησης {name}, ενός συγκεκριμένου ασθενούς με id={id_patient}, για τον αισθητήρα με id={id_sensor}
PUT	/sensors_subscriptions/sensor/{id_sensor}/device_name/{name}/patients/{id_patient}/time_ofMeasure/{time}	Ανανέωση της ώρας {time }, για την τιμή της μέτρησης {name}, ενός συγκεκριμένου ασθενούς με id={id_patient}, για τον αισθητήρα με

		id={id_sensor}
PUT	/sensors_subscriptions/sensor/{id_sensor}/device_name/{name}/patients/{id_patient}/condition_state/{condition	Ανανέωση της κατάστασης {condition},ενός συγκεκριμένου ασθενούς με id={id_patient}, για την τιμή της μέτρησης {name}, του αισθητήρα με id={id_sensor}
PUT	/sensors_subscriptions/sensor/{is_sensor}/patient/{id_patient}/device_name/{name}/value/{MeasureValue}/	Ανανέωση της τιμής της μέτρησης {MeasureValue} του ασθενούς με id={id_patient} για τον αισθητήρα με id={id_sensor} και όνομα μέτρησης {name}
DELETE	/sensors_subscriptions/id/{id}	Διαγραφή μιας οντότητας με βάση το id της

Πίνακας 7-REST Διαχείρισης οντοτήτων και γεγονότων υποστηριζόμενα από τον Context Broker

Στον πίνακα 8, βλέπουμε τις κλήσεις rest API που πραγματοποιούνται από την Υπηρεσία Διαχείρισης Συνδέσεων μεταξύ front-end και Back-end. Όπως αυτές περιγράφονται και στην ενότητα 4.3.

Μέθοδος	URL	Περιγραφή
GET	users/../collections/USERS/records	Παίρνουμε τους χρήστες που έχουν αποθηκευτεί στη συλλογή USERS στον JSON Storage
POST	/users/patients	Δημιουργία του νέου ασθενούς στην ενιαία βάση τπυ συστήματος
POST	users/../collections/SensorRules/records	Δημιουργία-αποστολή των κανόνων στη συλλογή SensorRules του JSON
PUT	users/…/collections/SensorRules/records/{id}	Ανανέωση ενός κανόνα σύμφωνα με το αναγνωριστικό id του
DELETE	users/…/collections/SensorRules/records/{id}	Διαγραφή ενός κανόνα με βάση το id του

Πίνακας 8-REST Διαχείρισης Σενδέσεων

Για την αποστολή των μηνυμάτων έχουμε το μήνυμα-κείμενο του ιατρού που «φεύγει» (tag “m”) από τον διακομιστή προς την συσκευή μας με βάση το Registration ID της συσκευής (tag “to”). Δηλαδή το σώμα (JSON) είναι:

{
"data": {
"m": "Hello User I am your doctor"
},
"to" :
"APA91bGRsygK3GGgFgSIFnIFE4X8AgZcOg9aQdjLSyYRFamkpYfSGl4RoybseFPg2FuOKU4XcJgj21VEzUUEeqeWoF5VfNZE9BWwKqcv-9i1Os3U9R9uVmJQHz2DisK1SjwpczegBZRc"
}

Έτσι με την παρακάτω κλήση που φάινεται στον Πίνακα 9, επιτυγχάνεται η αποστολή ενός μηνύματος από το server μας στην συσκευή.

Μέθοδος	URL	Περιγραφή
POST	https://android.googleapis.com/gcm/send	Αποστολή μηνύματος από Server σε κινητή συσκευή μέσω GCM GE

Πίνακας 9-Κλήση Αποστολής Μηνυμάτων (GCM)

4.5 Περιβάλλον Υλοποίησης – Γλώσσες Προγραμματισμού

Χρησιμοποιήσαμε εικονική μηχανή (VM) [23] από το περιβάλλον του Intellicloud[24], «μικρού» μεγέθους , για να «στεγάσουμε» όλες τις υπηρεσίες που υλοποιήσαμε: Υπηρεσία Διαχείρισης Αισθητήρων, Υπηρεσία Διαχείρισης Χρηστών, Υπηρεσία Διαχείρισης Κανόνων, Υπηρεσία Διαχείρισης Μετρήσεων, Υπηρεσία Διαχείρισης Δεδομένων, Υπηρεσία Διαχείρισης Ασθενών από τους ιατρούς τους, Υπηρεσία Διαχείρισης Γεγονότων και Υπηρεσία Μηνυμάτων (Google Cloud Messaging GE). Στην Υπηρεσία Ταυτοποίησης και Εξουσιοδότησης Χρηστών που χρησιμοποιούμε τον KeyRock Identity Manager και στην Υπηρεσία Διαχείρισης Δημοσιεύσεων και Συνδρομών (Publish Subscribe Service) που χρησιμοποιούμε τον Orion Context Broker αξιοποιήσαμε τις διαθέσιμες κοινόχρηστες εικονικές μηχανές που βρίσκονται στο περιβάλλον του Fiware[25] και για τις δύο υπηρεσίες .

[19] https://el.wikipedia.org/wiki/Global_Positioning_System
[20] http://cloud.intellicloud.tuc.gr/horizon
[21] https://account.lab.fiware.org/oauth2/authorize/?response_type=code&client_id=21&state=xyz&redirect_uri=http://cloud.lab.fiware.org/login

Για την αποθήκευση των δεδομένων που παράγονται από το Back-End, επιλέξαμε την εγκατάσταση σχεσιακής βάσης δεδομένων MySQL και τη χρήση της γλώσσας PHP[26] για τη διαχείρισή της στο Υπολογιστικό Νέφος. Λόγω περισσότερης εμπειρίας στη χρήση της και της δυνατότητας που μας παρέχει για την υποβολή πιο σύνθετων ερωτήσεων. Τη μονάδα αποθήκευσης JSON Storage τη χρησιμοποιήσαμε για το λόγο που αναφέρεται στην Υπηρεσία Διαχείρισης Συνδέσεων (Connectivity Service) στο κεφάλαιο 4.3. Κάναμε χρήση της PHP (Hypertext Preprocessor) γλώσσας προγραμματισμού για την υλοποίηση των ιστοσελίδων, διότι συνεργάζεται άψογα με την MySQL, είναι ανοιχτού κώδικα γλώσσα προγραμματισμού και διαθέτει πολλές έτοιμες συναρτήσεις που κάνουν τη ζωή μας πιο εύκολη. Είναι server side γλώσσα προγραμματισμού, δηλαδή την τρέχει ο διακομιστής (server) και εμείς βλέπουμε το αποτέλεσμα, και όχι client side (όπως είναι για παράδειγμα η HTML). Εκτελείται από ένα ειδικό πρόγραμμα που είναι εγκατεστημένο στον διακομιστή (server) και ονομάζεται apache. Ο Apache HTTP γνωστός και απλά σαν Apache είναι ένας εξυπηρετητής του παγκόσμιου ιστού (web). Όποτε ένας χρήστης επισκέπτεται ένα ιστότοπο το πρόγραμμα πλοήγησης (browser) επικοινωνεί με έναν διακομιστή (server) μέσω του πρωτοκόλλου HTTP, ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει στο πρόγραμμα πλοήγησης. Επίσης κάναμε χρήση του Framework Slim-PHP[27] το οποίο μας βοηθά στη δημιουργία κώδικα για χειρισμό αιτημάτων που βασίζονται σε αρχιτεκτονική REST.

Ο σχεδιασμός των διεπαφών χρήστη-ιατρού και διαχειριστή που αποτελούν δύο Web εφαρμογές προσβάσιμες μέσω φυλλομετρητή, έγινε με τη χρήση των γλωσσών προγραμματισμού HTML[28], CSS[29], Javascript[30] , AJAX[31] και της βιβλιοθήκης jQuery[32].

Αναλυτικότερα, η HTML(Hypertext Markup Language), αποτελεί την κύρια γλώσσα σύμανσης (markup language[33]) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών και τα βασικά της στοιχεία είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Η CSS(Cascading Style Sheet) ή (αλληλουχία φύλλων στυλ) είναι μια γλώσσα που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σύμανσης (HTML).

Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

---

[26] https://en.wikipedia.org/wiki/PHP

[27] https://www.slimframework.com

[28] https://en.wikipedia.org/wiki/HTML

[29] https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[30] https://en.wikipedia.org/wiki/JavaScript

[31] https://en.wikipedia.org/wiki/Ajax_(programming)

[32] https://en.wikipedia.org/wiki/JQuery

[33] https://en.wikipedia.org/wiki/Markup_language

Η Javascript (JS) είναι μια γλώσσα προγραμματισμού που παρέχει τη δυνατότητα δημιουργίας συναρτήσεων, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η τεχνολογία AJAX (Asynchronous Javascript and XML) αυτή τη στιγμή αποτελεί τη πιο σύγχρονη τεχνολογία στον προγραμματισμό. Είναι μια ισχυρή τεχνολογία που επιτρέπει τη δημιουργία δυναμικών web εφαρμογών. Επιπλέον παρέχει δυναμική αλληλεπίδραση στο διαδίκτυο, ενημερώνει σε πραγματικό χρόνο και με απλές αλλαγές στο σχεδιασμό της εφαρμογής μπορεί να χρησιμοποιηθεί και από άλλους. Η κύρια γλώσσα με την οποία εφαρμόζεται η AJAX είναι η JavaScript.

5 Ανάλυση Απόδοσης

5.1 Επιλογή Πειράματος

Σε αυτό το κεφάλαιο θα περιγράψουμε το πειραματικό περιβάλον και τα πειράματα που πραγματοποιήσαμε με στόχο να μετρήσουμε την απόδοση του συστήματός μας.

Οι εικονικές μηχανές που χρησιμοποιούμε είναι τρεις. Μία κοινόχρηστη εικονική μηχανή για τον Orion Context Broker που βρίσκεται και αυτή στο περιβάλλον του Fiware, άλλη μία κοινόχρηστη εικονική μηχανή για τον Json Storage που βρίσκεται στο Intellicloud[34] και μία τέταρτη εικονική μηχανή για τις υπηρεσίες του Υπολογιστικού Νέφους που έχουμε υλοποιήσει η οποία βρίσκεται στο περιβάλλον του Intellicloud. Συνοπτικά οι τρεις εικονικές μηχανές φαίνονται στον Πίνακα 10:

[20] http://cloud.intellicloud.tuc.gr/horizon

VM (διεύθυνση)	Όνομα	Περιβάλλον που φιλοξενείται
1ο 147.27.60.107:1026	Orion Context Broker (κοινόχρηστο)	Fiware
2ο 147.27.50.33:3000	JSON Storage (κοινόχρηστο)	Intellicloud
3ο 147.27.50.154	Intelligate (η δική μας εφαρμογή, με τις υπηρεσίες που έχουμε υλοποιήσει)	Intellicloud

Πίνακας 10- Εικονικές Μηχανές

Τα χαρακτηριστικά και των τριών εικονικών μηχανών φαίνονται στον Πίνακα 11 και αναλύονται από κάτω:

Architecture	VCPUs	Disk	RAM
CPU x86_64	1	20GB	2048MB

Πίνακας 11- Χαρακτηριστικά εικονικής μηχανής

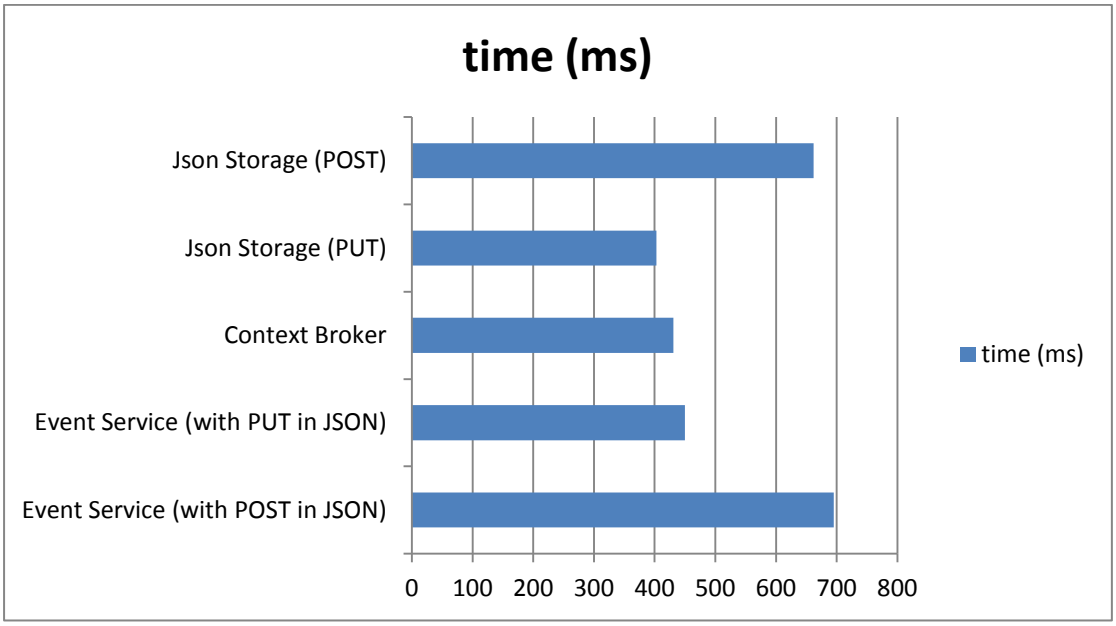
Δηλαδή διαθέτουν:

- 1 εικονικός επεξεργαστής - Αρχιτεκτονική επεξεργαστή x86_64
- Μέγεθος μνήμης τυχαίας πρόσβασης 2048MB
- Μέγεθος χωρητικότητας σκληρού δίσκου 20GB
- Χρονισμός επεξεργαστή 2800Mhz
- Μέγεθος κρυφής μνήμης πρώτου επιπέδου 32K
- Μέγεθος κρυφής μνήμης δευτέρου επιπέδου 4096K
- Εξυπηρετητής Apache HTTP
- Λειτουργικό Σύστημα Ubuntu 14.04

Για να διαπιστώσουμε πόσο χρόνο χρειάστηκε η κάθε εικονική μηχανή να εκτελέσει τους υπολογισμούς της καθώς τις χρησιμοποιούμε στις δικές μας υπηρεσίες (Υπηρεσία Διαχείρισης Γεγονότων, Υπηρεσία Διαχείρισης Δημοσιεύσεων και Συνδρομών, Υπηρεσία Διαχείρισης Κανόνων), δημιουργήσαμε ένα αυτοσχέδιο πείραμα. Στόχος ήταν να μετρήσουμε το χρόνο που χρειάζεται για να εκτελεστούν οι εξής ενέργειες:

- JSON Storage (POST): Δημιουργία κανόνων μέσω της αποστολής αιτημάτων POST στην Υπηρεσία JSON Storage που αποτελεί κοινόχρηστη μηχανή όπως αναφέραμα παραπάνω.
- JSON Storage (PUT): Ανανέωση κανόνων μέσω της αποστολής αιτημάτων PUT στην Υπηρεσία JSON Storage.
- Context Broker: Δημιουργία οντότητας Context (Publish) και Συνδρομή του ιατρού στην αλλαγή της τιμής της μέτρησης (Subscribe).
- Event Service (with POST στον JSON): Δημιουργία κανόνων στον JSON - Έλεγχος κανόνων, αν είναι στα πλαίσια των προκαθορισμένων τιμών, μέσω της Υπηρεσία Διαχείρισης Κανόνων - Ενημέρωση ιατρού για την κατάσταση του ασθενούς μέσω της Υπηρεσίας Γεγονότων.
- Event Service (with PUT in JSON): Ανανέωση κανόνων στον JSON - Έλεγχος κανόνων, αν είναι στα πλαίσια των προκαθορισμένων τιμών μέσω της Υπηρεσίας Διαχείρισης Κανόνων - Ενημέρωση ιατρού για την κατάσταση του ασθενούς μέσω της Υπηρεσίας Γεγονότων.

Τοποθετώντας έναν μετρητή χρόνου και με την προσθήκη διαγνωστικών μηνυμάτων, έπειτα από την εκτέλεση καθε μιας εκ των παραπάνω ενεργειών, βλέπουμε τα αποτελέσματα που προέκυψαν στο Διάγραμμα 12:



Σχήμα 26- Χρόνοι εκτέλεσης ξεχωριστών ενεργειών

Παρατηρούμε πως η αναμονή απάντησης από τον JSON Storage αυξάνει την καθυστέρηση ολοκλήρωσης της Υπηρεσίας Διαχείρισης των γεγονότων, ειδικά όταν πρόκειται για κανόνες που δημιουργούνται πρώτη φορά.

Στη συνέχεια μετρήσαμε την απόδοση της εικονική μηχανής που στεγάζονται όλες οι υπηρεσίες που έχουμε υλοποιήσει (3^{ο} VM Πίνακας 10). Για το σκοπό αυτό κάναμε χρήση του ενσωματωμένου εργαλείου του Apache, το ab-Apache HTTP server benchmarking tool[35], που μας δίνει τη δυνατότητα να στέλνουμε πολλά ταυτόχρονα αιτήματα με σκοπό να δούμε πόσες αιτήσεις ανά δευτερόλεπτο είναι σε θέση να εξυπηρετήσει ο Apache. Αρχικά, ξεκινήσαμε με σειριακή αποστολή 2000 αιτημάτων. Αυτό έγινε θέτοντας την παράμετρο concurrency (c) ίση με 1. Η παράμετρος αυτή δηλώνει τον αριθμό των ταυτόχρονων αιτημάτων που στέλνονται προς εξυπηρέτηση. Κάνοντάς την ίση με 1, στέλνεται ένα αίτημα μόλις ολοκληρωθεί η αποστολή του προηγούμενου.

Για να δούμε τη χρήση των υπολογιστικών πόρων με την εκτέλεση του πειράματος κάναμε χρήση της εντολής Linux top[36], που ενεργοποιεί ένα ενσωματωμένο πρόγραμμα εποπτείας της χρήσεως των υπολογιστικών πόρων της εικονικής μηχανής. Τα αποτελέσματα της εκτέλεσης του πειράματος φαίνονται στον παρακάτω Πίνακα 12:

Ποσοστό των αιτημάτων που εξυπηρετούνται	**Χρόνος Εξυπηρέτησης (σε ms)**
50%	882
66%	930
75%	1086
80%	1158
90%	1454
95%	1544
98%	1682
99%	1844
100%	1844

Πίνακας 12- Απαιτούμενος χρόνος εκτέλεσης 2000 αιτημάτων με concurrency: 1

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

[36] https://linux.die.net/man/1/top

Από τον Πίνακα 12 συμπεραίνουμε πως ο μέσος χρόνος ανα αίτημα ήταν ίσος με 1086 ms και ο ρυθμός μεταφοράς των δεδομένων ήταν ίσος με 0.89 Kbytes/s. Επιπλέον η χρήση των υπολογιστικών πόρων είναι χαμηλή, το οποίο το αναμέναμε μιας και εκτελούνταν ένα αίτημα κάθε χρονική στιγμή.

Στη συνέχεια μετρήσαμε την απόδοση για την περίπτωση που έχουμε ταυτόχρονα αιτήματα. Για το σκοπό αυτό, εκτελέσαμε 2000 αιτήματα, με concurrency ίσο με 40, δηλαδή 40 ταυτόχρονα αιτήματα. Τα αποτελέσματα φαίνονται στον Πίνακα 13, παρακάτω:

Ποσοστό των αιτημάτων που εξυπηρετούνται	Χρόνος Εξυπηρέτησης (σε ms)
50%	1189
66%	1372
75%	1386
80%	1468
90%	1811
95%	1926
98%	1940
99%	1957
100%	2011

Πίνακας 13- Απαιτούμενος χρόνος εκτέλεσης 2000 αιτημάτων με concurrency: 40

Από την εκτέλεση του παραπάνω πειράματος βλέπουμε πως ο μέσος χρόνος ανα αίτημα ήταν ίσος με 1386 ms και ο ρυθμός μεταφοράς των εδομένων ήταν ίσος με 7.82 Kbytes/s. Η χρήση του επεξεργαστή κατά μέσο όρο κυμαίνονταν στο 60% και χρήση της RAM ανέρχεται στα 180 MB. Επομένως, λόγω των ταυτόχρονων αιτημάτων, υπάρχει αύξηση των απαιτούμενων υπολογιστικών πόρων και παρουσιάζεται μια καθυστέρηση κατά την εκτέλεση του συστήματός μας. Γεγονός που είναι αναμενόμενο μιας και ο εξυπηρετητής αναγκάζεται να μεταβεί σε ταυτόχρονη εκτέλεση πολλαπλών ενεργειών (multitasking[37]).

[37] https://en.wikipedia.org/wiki/Computer_multitasking

Στη συνέχεια διπλασιάσαμε τον αριθμό των ταυτόχρονων αιτημάτων, για να δούμε πως θα αυξηθεί η χρήση του επεξεργαστή αλλά και ποιός είναι ο μέγιστος αριθμός αιτημάτων που μπορούμε να εξυπηρετήσουμε. Τα ποτελέσματα αυτής της μέτρησης φαίνονται στον Πίνακα 14:

Ποσοστό των αιτημάτων που εξυπηρετούνται	Χρόνος Εξυπηρέτησης (σε ms)
50%	4862
66%	4912
75%	5106
80%	5269
90%	6011
95%	8850
98%	10350
99%	11869
100%	12350

Πίνακας 14- Απαιτούμενος χρόνος εκτέλεσης 2000 αιτημάτων με concurrency: 80

Από την εκτέλεση του παραπάνω πειράματος, βλέπουμε πως ο μέσος χρόνος ανα αίτημα ήταν ίσος με 6806 ms και ο ρυθμός μεταφοράς των εδομένων ήταν ίσος με 8.14 Kbytes/s. Η χρήση του επεξεργαστή πλέον ήταν σχεδόν η μέγιστη (~95%) και η χρήση της RAM ανήλθε στα 350 MB. Με την αύξηση των ταυτόχρονων αιτημάτων, αυξήθηκε κι άλλο η ανάγκη σε υπολογιστικούς πόρους, όπως ήταν αναμενόμενο.

6 Συμπεράσματα - Μελλοντικές Επεκτάσεις

6.1 Συμπεράσματα

Με την εκμετάλλευση της ιδέας του διαδικτύου των πραγμάτων αλλά και τη βοήθεια των υπηρεσιών που μας προσφέρει το υπολογιστικό νέφος, κατορθώσαμε να κάνουμε εφικτή την απομακρυνσμένη παρακολούθηση ασθενών από γιατρούς οι οποίοι πάσχουν από καρδιακές παθήσεις από όποιο σημείο και αν βρίσκονται με την χρήση απλά ενός φυλλομετρητή(από μεριάς γιατρού) και μιάς έξυπνης κινητής συσκευής(smartphone) από μεριά χρήστη ασθενούς. Ο δυναμικός τρόπος εισαγωγής νέων αισθητήρων δίνει την δυνατότητα στην εφαρμογή να μην υπάρχουν περιορισμοί στην προσθαφέραιση νέων αισθητήρων που παράγουν δεδομένα ζωτικής σημασίας. Ακόμη, η δυνατότητα ενημέρωσης του γιατρού εαν η κατάσταση του ασθενούς είναι κρίσιμη είναι ένα πολύ σημαντικό κομμάτι της υλοποίησής μας. Η λειτουργία υπηρεσιών γενικού σκοπού(Generic Enablers) μας βοήθησαν στην απλούστευση της υλοποίησης του συστήματος καθώς πολλές υπηρεσίες τις οποίες χρειαζόμασταν ήταν ήδη έτοιμες , και μέσω μιας απλής παραμετροποίησης μπορούσαν να εξυπηρετήσουν δικές μας ανάγκες.

Κυριότερα πλεονεκτήματα του Συστήματος Intelligate και Back-End Intelligate:

- **Επεκτασιμότητα:** Το σύστημά μας υπακούει στις βασικές αρχές τις Υπηρεσιοκεντρικής αρχιτεκτονικής, με αυτόν τον τρόπο οποιαδήποτε νέα λειτουργία θέλουμε να ενσωματώσουμε στο σύστημα μας , μπορεί εύκολα να ενσωματωθεί ως κομμάτι μιας ήδη υπάρχουσας υπηρεσίας ή ως ένα νέο ξεχωριστό κομμάτι. Για παράδειγμα, μέσω του ορισμού XML σχήματος αισθητήρων και της υλοποίησης ολόκληρου του συστήματος , επιτυγχάνεται η δυναμική πρόσθεση οποιονδήποτε BLE αισθητήρων που υπακούν στο πρωτόκολλο επικοινωνίας Bluetooth Low energy.

- **Εύκολο στην χρήση:** Προσφέρει ένα απλό και κατανοητό γραφικό περιββάλλον, σε επίπεδο κινητής πατφόρμας αλλά και εφαρμογής web, χωρίς να χρειάζεται ιδιαίτερες γνώσεις υπολογιστών.

6.2 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύξαμε έχει την δυνατότητα με συγκεκριμένες επεκτάσεις, οι οποίες επισημαίνονται συνοπτικά παρακάτω να γίνει περισσότερο λειτουργικό και αποδοτικό ξεπερνώντας τους περιορισμούς της εφαρμογής μας.

- **Σύνδεση πολλών αισθητήρων ταυτόχρονα στην συσκευή:** Ένας περιορισμός της εφαρμογής μας ήταν ότι η κεντρική συσκευή (tablet) μπορούσε να υποστηρίζει και να συνδεθεί μόνο με μια περιφερειακή συσκευή(αισθητήρας). Ως μελλοντική επέκταση της εφαρμογής θα ήταν η ταυτόχρονη υποστήριξη πληθώρας περιφερειακών συσκευών από την κεντρική συσκευή με την συγχρονισμένη ή και ασύγχρονη μεταφορά των δεδομένων τους προς το νέφος.

- **Μείωση του φόρτου του συστήματος:** Συγκεκριμένα**,** η εικονική μηχανή που χρησιμοποιήθηκε για τα παραπάνω πειράματα περιείχε όλες τις υπηρεσίες που υλοποιήθηκαν με αποτέλεσμα να υπάρχει πρόβλημα υπερφόρτωσης. Επομένως, ελαφρύνοντας το φόρτο του VM και μοιράζοντας τις υπηρεσίες σε ξεχωριστά VMs μπορούμε να πετύχουμε καλύτερα αποτελέσματα.

- **Βελτίωση των χρόνων απόκρισης συγκεκριμένων λειτουργιών του συστήματος:** Στο κομμάτι του συστήματος που αναφέρεται στην Υπηρεσία Διαχείρισης των Γεγονότων, που θεωρείται πιο απαιτητικό, θα βοηθούσε η αύξηση των πόρων με την ταυτόχρονη μείωση των πόρων στις άλλες υπηρεσίες.

- **Υποστήριξη της αναγνώρισης θέση του ασθενούς μέσω GPS[38] της κινητής συσκευής:** Ο Γιατρός και μόνος αυτός , να έχει το δικαίωμα να βλέπει την θέση του ασθενή του σε περίπτωση όπου προκύψει κάποιο πρόβλημα και χρειαστεί να παρέμβει.

- **Κρυπτογράφηση στοιχειών ασθενούς:** Σε περιπτώσεις που ζητώνται στοιχεία ασθενών από τοποθεσίες εκτός της νοσοκομειακής μονάδας, να αποστέλλονται κρυπτογραφημένα για να μπορούν να διαφυλλαχτούν τα προσωπικά τους στοιχεία.

[38] https://el.wikipedia.org/wiki/Global_Positioning_System

7 Βιβλιογραφία

[1] http://www.healthcarebusinesstech.com/medical-technology/

[2] https://en.wikipedia.org/wiki/Cloud_computing

[3] http://webapptester.com/ti-einai-cloud-computing/

[4] http://integrationtechnologytrends.blogspot.gr/2014/10/cloudy-cloud-pay-as-you-go-upscale-as.html

[5] https://www.fiware.org/about-us/

[6] http://www.europeanpioneers.eu/en/fiware-technologies.html

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

[8] http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/

[9] https://en.wikipedia.org/wiki/Representational_state_transfer

[10] https://www.tutorialspoint.com/restful/

[11] http://docplayer.gr/7434479-Ypiresiostrefis-arhitektoniki-soa-service-oriented-architecture.html

[12] http://www.intelligence.tuc.gr/publications.php?pub_author=All&pub_type=8&pub_subject=All

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

[14] https://en.wikipedia.org/wiki/Activity_diagram

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

[16] https://en.wikipedia.org/wiki/Unified_Modeling_Language

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