ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σύστημα Καταγραφής και Ανάλυσης Δεδομένων Κίνησης και Βιοσημάτων από Φορητές Συσκευές Internet of Things και Υπηρεσίες Cloud

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Χαράλαμπου Ιωάννου

Επιβλέπων : Παναγιώτης Τσανάκας

Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σύστημα Καταγραφής και Ανάλυσης Δεδομένων Κίνησης και Βιοσημάτων από Φορητές Συσκευές Internet of Things και Υπηρεσίες Cloud

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Χαράλαμπου Ιωάννου

Επιβλέπων : Παναγιώτης Τσανάκας

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22η Φεβρουαρίου 2019.

.....	.....	.....
Π. Τσανάκας	Γ. Ματσόπουλος	Δ. Κουτσούρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2019

(Υπογραφή)

.....

Χαράλαμπος Ιωάννου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η υλοποίηση μιας διαδικτυακής πλατφόρμας καταγραφής και αποθήκευσης δεδομένων κίνησης και βιοσημάτων απο “έξυπνες” φορητές συσκευές smartwatch, με στόχο την μετέπειτα ερευνητική εξόρυξη πληροφοριων όπως προτύπων φυσικής δραστηριότητας.

Η ραγδαία αύξηση της χρήσης φορητών έξυπνων συσκευών όπως wearables, έχουν οδηγήσει στην παραγωγή εξαιρετικά προσιτών συσκευών, οι οποίες φέρουν πληθώρα υψηλής ακρίβειας αισθητήρες και ισχυρής επεξεργαστικής ισχύος. Ταυτόχρονα, η αυξανόμενη τάση στον χώρο της υγείας για ποσοτικοποίηση και καταγραφή βιομετρικών δεδομένων με στόχο την ανάλυση και την εξαγωγή συμπερασμάτων, δημιουργεί την ορατή ανάγκη για ομογενοποιημένη διαχείριση τέτοιων δεδομένων. Κατ’ αυτόν τον τρόπο η κατασκευή μιας υπηρεσίας που αναλαμβάνει αυτοματοποιημένα την συλλογή, διαχείριση και επεξεργασία τέτοιων δεδομένων, προσφέροντας ένα επίπεδο αφαίρεσης και υλοποιώντας όλα τα κρίσιμα για επιδόσεις κομμάτια του συστήματος, αποκτα μεγάλη προστιθέμενη αξία σε ερευνητικό και επιχειρησιακό περιβάλλον.

Στα πλαίσια της διπλωματικής εργασίας, αναπτύχθηκαν δύο συμπληρωματικές εφαρμογές οι οποίες συλλογικά αποτελούν την πλατφόρμα καταγραφής και ανάλυσης δεδομένων. Η εφαρμογή συλλογής και προσωρινής αποθήκευσης δεδομένων εκτελείται επί της συσκευής έξυπνου ρολογιού Samsung Gear S3 Frontier που φορά ο χρήστης. Είναι υπεύθυνη για την λήψη μετρήσεων ανα τακτά χρονικά διαστήματα από τους διάφορους αισθητήρες της συσκευής και για την προσωρινή αποθήκευση και προεπεξεργασία των δεδομένων. Η διαδικτυακή εφαρμογή ορίζει την διεπαφή για την ασφαλή μεταφόρτωση των δεδομένων απο την συσκευή του χρήστη καθώς και την αποδοτική αποθήκευση αυτών. Επιπλέον, επιτρέπει την εγγραφή και διαχείριση χρηστών, την παρακολούθηση της εξέλιξης των ληφθέντων δεδομένων τους στον χρόνο καθώς και στατιστικών στοιχείων σχετικά με αυτά. Ταυτόχρονα, παρέχει στους διαχειριστές του συστήματος την δυνατότητα φιλτραρίσματος και ταξινόμησης με διάφορα κριτήρια, ώστε η ανάκτηση ερευνητικής φύσης πληροφοριών από τα σύνολα δεδομένων να γίνει με ασφαλή και εύκολο τρόπο. Κατά την υλοποίηση χρησιμοποιήθηκαν MongoDB και PostgreSQL,τεχνολογίες βάσεων δεδομένων, ενώ ο εξυπηρετητής και η εφαρμογή αναπτύχθηκαν σε JavaScript, σε αρχιτεκτονική Node.js.

Μετά το πέρας της υλοποίησης της, η πλατφόρμα χρησιμοποιήθηκε για εκτενές χρονικό διάστημα σε ερευνητικό περιβάλλον φιλοξενώντας πληθώρα πραγματικών χρηστών και εκατοντάδων χιλιάδων δεδομένων, με απώτερο στόχο την εύρεση κρυμμένων συσχετίσεων μεταξύ προτύπων δραστηριότητας και χρόνιων ψυχολογικών παθήσεων.

Λέξεις κλειδιά: Σύστημα Απομακρυσμένης Παρακολούθησης, Samsung Gear S3, Έξυπνο ρολόι, Tizen OS, Διαδίκτυο των Πραγμάτων, Υπολογιστικό Νέφος, RESTful Αρχιτεκτονική, JavaScript, Node.js, SQL, PostgreSQL, No-SQL, MongoDB

Abstract

The objective of this diploma thesis was the design and implementation of a system that collects and aggregates biosignals and motion data from users via a smartwatch device, in order to provide a uniform technological backbone for building datasets intended for pattern recognition and activity detection research.

The rapid market growth of wearable devices worldwide, has led to the production of very affordable devices with many powerful on-board sensors and advanced processing capacity. At the same time, the continuous growing trend of using wearable devices to quantify wellness and fitness related parameters has caused demand for management and processing such collected data. Especially in the health sector, where biosignals can be used for remote monitoring of an individual's health status or extract meaningful insights about it, the demand for a systematic and holistic information extraction and secure management of these data is crucial. As a result, the development of a system that takes advantage of the advanced hardware possibilities of the modern wearables and couples them with an efficient, secure data collection and management system creates great added value in research and business setting.

In order to accomplish the aforementioned goals we developed a system which is comprised of two separate applications which work in conjunction and collectively make up the data collection and analysis system. At first, a native application running on Samsung Gear S3 Frontier smartwatch, collects biosignals and activity related data regularly from the on-board sensors. The collected data are temporarily stored locally and are uploaded to the cloud asynchronously via the internet connectivity of the device. Secondly, a web-application running on cloud infrastructure implements the communication interface and protocols for secure data upload and their efficient storage. The system provides an easy-to-use user-interface where the users can register, manage their devices and see their collected data in the system. In addition, the system provides the system administrators with filtering and sorting mechanisms in order to make information retrieval from the datasets secure and easy. Lastly, the server was implemented in JavaScript in Node.js framework and the data store is comprised by a hybrid system utilizing both MongoDB and PostgreSQL databases.

After the completion of the development phase of the system, the service has been utilized for an extended period of time in a research program, serving multiple real users and hundreds of thousands of data, with the objective to reveal hidden correlations between user activity patterns and mental disorders.

Keywords: Remote Monitoring System, Samsung Gear S3, Smartwatch, TizenOS, Internet Of Things, Cloud Computing, RESTful Architecture, JavaScript, Node.js, SQL, PostgreSQL, No-SQL, MongoDB

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά την διάρκεια του ακαδημαϊκού έτους 2018-2019 στους Τομείς Τεχνολογίας Πληροφορικής και Υπολογιστών καθώς και Σημάτων, Ελέγχου και Ρομποτικής, της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Με αφορμή λοιπόν την περάτωση των προπτυχιακών σπουδών μου στο Εθνικό Μετσόβιο Πολυτεχνείο, νιώθω την υποχρέωση να ευχαριστήσω:

Τον επιβλέποντα καθηγητή μου, κ. Παναγιώτη Τσανάκα από τον οποίο μου δόθηκε η ευκαιρία και η εμπιστοσύνη να εκπονήσω την παρούσα διπλωματική εργασία και να ασχοληθώ με ένα άκρως σύγχρονο και ενδιαφέρον για εμένα ερευνητικό αντικείμενο. Καθώς επίσης και για την συνεχή καθοδήγηση του καθ'όλη την διάρκεια των σπουδών μου και της ερευνητικής μας δραστηριότητας.

Τους καθηγητές: Ανδρέα-Γεώργιο Σταφυλοπάτη, Ηλία Μαγκλογιάννη και Θεμιστοκλή Ρασσιά καθώς υπήρξαν πρότυπα καθηγητών αλλά και ανθρώπων, τόσο στην ερευνητική διαδικασία, όσο και για την βοήθεια τους στην εύρεση και επίτευξή των προσωπικών μου στόχων, αλλά και την προσωπική μου ολοκλήρωση.

Τους προσωπικούς μέντορες και φίλους, Νίκο Κουμέττη, Δημήτρη Καλαυρό-Γουσίου και Σήλια Κουνούσου που αποτέλεσαν φωτεινό παράδειγμα προς μίμηση για εμένα και μια πηγή διαρκούς έμπνευσης και γνώσης. Όπως επίσης και τον μεταδιδακτορικό ερευνητή του Ε.Π.Μ. κ. Ανδρέα Μενύχτα για την βοήθεια και την άψογη συνεργασία μας σε όλο το διάστημα της φοίτησης μου, που ήταν καθοριστικής σημασίας για την επιστημονική μου ωρίμανση και φυσικά για την εκπόνηση της παρούσας εργασίας.

Τους κ. Δ. Κουτσούρη, Καθηγητή Ε.Μ.Π. και Γ. Ματσόπουλο, Καθηγητή Ε.Μ.Π., για την τιμή που μου έκαναν όντας μέλη της επιτροπής εξέτασης της διπλωματικής μου εργασίας.

Τους φίλους και τις φίλες μου, όντας δίπλα μου σε όλες τις δυσκολίες που αντιμετώπισα αυτά τα χρόνια, προσφέροντας μου ο καθένας κάτι ξεχωριστό, δημιουργώντας αξέχαστες αναμνήσεις.

Τέλος, ολοκληρώνοντας το πρώτο και μεγαλύτερο όνειρο της μέχρι τώρα ζωής μου, θα ήθελα να εκφράσω την απέραντη ευγνωμοσύνη μου και να αφιερώσω εξ΄ολοκλήρου τη διπλωματική αυτή στην οικογένεια μου, για την αδιάλειπτη υποστήριξή της στις επιλογές μου και για την παρότρυνσή τους να ακολουθώ αυτό που επιθυμώ.

Χαράλαμπος Ιωάννου

Πίνακας Περιεχομένων

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

1 Εισαγωγή:

1.1 Σκοπός της διπλωματικής εργασίας

Οι ραγδαίες τεχνολογικές και επιχειρηματικές εξελίξεις των τελευταίων 10 χρόνων έχουν κατακλύσει την αγορά ψηφιακών συσκευών με πληθώρα φθηνών και ταυτόχρονα εξαιρετικά ικανών έξυπνων φορητών συσκευών. Εταιρείες από όλα τα μέρη του κόσμου ανταγωνίζονται, για το ποια θα μπορέσει να κατακτήσει την επόμενη φυσική διεπαφή μετά από αυτή του κινητού τηλεφώνου. Αυτή η ανταγωνιστική διαδικασία έχει να προσφέρει μια νέα οικογένεια "έξυπνων" συσκευών, τα wearables, φορητές δηλαδή συσκευές, που φοράει ο χρήστης καθημερινά, επαυξάνοντας με κάποιον τρόπο την εμπειρία χρήσης τους με μη-παρεμβατικό τρόπο. Σε ένα τέτοιο περιβάλλον ρούχα, ρολόγια, γυαλιά, ακουστικά κ.α. γίνονται "έξυπνα" με την προσάρτηση σε αυτά προηγμένων υπολογιστικών συστημάτων και αισθητήρων. Η δικτύωση αυτών, αποτελεί μέρος του οικοσυστήματος του Διαδικτύου των Πραγμάτων (Internet of Things), που τα καθιστά εξαιρετικά χρήσιμες πλατφόρμες για την ανάπτυξη νέων εφαρμογών και τεχνολογιών, αφού το λογισμικό πλέον επικοινωνεί άμεσα με τον φυσικό χώρο.

Ένα τόσο ταχέως αναπτυσσόμενο τεχνολογικό οικοσύστημα δεν θα μπορούσε παρά να προκαλέσει ακούσια εξελίξεις και σε άλλα συναφή αντικείμενα, όπως αυτό της βιοϊατρικής, ένα πεδίο που εκμεταλλεύεται στο έπακρο τεχνολογικές καινοτομίες. Έτσι λοιπόν, η χρήση τέτοιου είδους έξυπνων συσκευών με υψηλή ποιότητα μετρήσεων και επεξεργαστική δυνατότητα, έχουν καταστήσει εξαιρετικά εύκολη την παρακολούθηση μιας πληθώρας βιομετρικών χαρακτηριστικών από κάθε χρήστη. Αποτελεί πλέον σύγχρονο κοινωνικό φαινόμενο το "Quantified self", η αυτόνομη ποσοτικοποίηση δηλαδή μετρήσεων και δεδομένων άμεσα σχετιζόμενων με την ζωή και την καθημερινότητα του ίδιου το χρήστη.

Ταυτόχρονα, η επιστημονική ανάλυση τέτοιων δεδομένων αποκαλύπτει τη βαθύτερη γνώση της φυσικής κατάστασης, των φυσικών δυνατοτήτων και της προσωπικότητας του χρήστη, προσδίδοντας πραγματική προστιθέμενη αξία σε εφαρμογές τηλεϊατρικής. Επιπλέον, τέτοιου είδους ολοκληρωμένα συστήματα για απομακρυσμένη και συστηματική παρακολούθηση χρηστών έχουν τεράστια δυναμική. Ειδικότερα, χρησιμοποιούμενα απο ευάλωτες ομάδες του πληθυσμού, μπορούν να εντοπίσουν επείγοντα περιστατικά ή να παρέχουν εύκολη εξ-αποστάσεως πρόσβαση σε υπηρεσίες υγείας.

Ωστόσο, μέχρι τώρα το πεδίο ήταν διάσπαρτα δομημένο για τους εξής λόγους: Πρώτον, κάθε επιστημονικός ή επιχειρηματικός φορέας που ήθελε να δημιουργήσει μια τέτοιου είδους υπηρεσία κατέφευγε σε μια ad-hoc λύση που στερούνταν ενιαίας προτυποποίησης και προκαλούσε ασυνέπεια στις μορφές των δεδομένων, μειώνοντας την ερευνητική αξία και δυσχεραίνοντας την ηλεκτρονική προστασία τους. Δεύτερον, ερευνητικές ομάδες που ήθελαν να ασχοληθούν με την χρήση και εξόρυξη πληροφορίας και συμπερασμάτων από τέτοιου είδους δεδομένα, έπρεπε να κατασκευάσουν εξ' αρχης ένα σύστημα, λόγω της απουσίας δεδομένων στο διαδίκτυο, αυξάνοντας κατ' αυτον τον τρόπο το κόστος, τον χρόνο ολοκλήρωσης και την έκταση

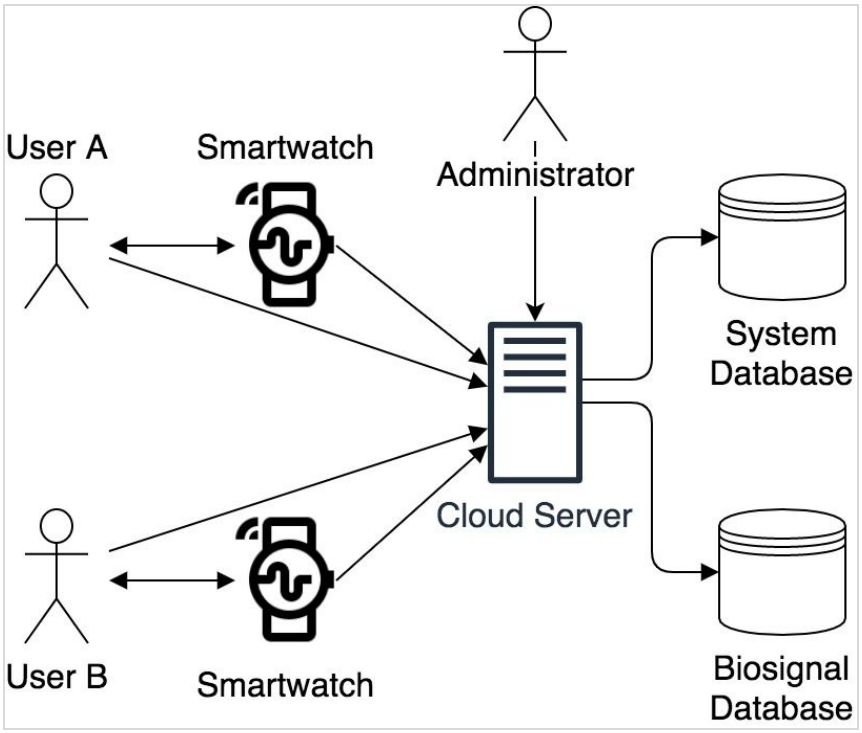
του έργου. Τρίτον, η ανομοιομορφία των διαφορετικών διεπαφών και δυνατοτήτων που προσφέρει κάθε σύστημα, τα καθιστά συνολικώς δύσχρηστα, αφού οι χρήστες πρέπει κάθε φορά να προσαρμοστούν στον τρόπο χρήσης κάθε διαφορετικού συστήματος.

Σκοπός της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός ενός τεχνικού υποβάθρου και η υλοποίηση μιας πρωτότυπης πλατφόρμας για την αποδοτική συλλογή, αποθήκευση και επεξεργασία δεδομένων από ετερογενείς πηγές. Η πλατφόρμα, θα προσφέρει μια ολοκληρωμένη εμπειρία στον χρήστη αποτελούμενη από δύο εφαρμογές λειτουργώντας συμπληρωματικά μεταξύ τους· την εφαρμογή που εκτελείται σε wearable και την εφαρμογή διαδικτύου που εκτελείται στο cloud, προσβάσιμη από τον περιηγητή ιστού του χρήστη.

Για την πρώτη εφαρμογή, ως ενδεικτική συσκευή συλλογής δεδομένων κίνησης και προσωρινής αποθήκευσης τους, επιλέχθηκε το έξυπνο ρολόι της Samsung Gear S3 Frontier. Σημαντικά πλεονεκτήματα της συσκευής είναι· η πλήρης συλλογή από εξαιρετικά ακριβείς αισθητήρες, όπως αισθητήρες επιτάχυνσης, γυροσκόπιο, αισθητήρες καρδιακών παλμών και G.P. S, η προσιτή τιμή, η αυτονομία τουλάχιστον μιας ημέρας συνεχούς χρήσης. Συγχρόνως, η συνδεσιμότητα με wi-fi και bluetooth καθιστά δυνατή την μεταφορά δεδομένων στο κεντρικό σύστημα. Επιπλέον, η συσκευή χάρη στον σχεδιασμό της, φοριέται απλά ως "έξυπνο" ρολόι χωρίς να περιορίζει κινητικά τον χρήστη, συμβάλοντας στην καλύτερη ποιότητα των συλλεγόμενων δεδομένων.

Η διαδικτυακή εφαρμογή, αποτελεί τον πυρήνα του συστήματος αφού είναι υπεύθυνη για την συγκεντρωτική αποθήκευση των συλλεγόμενων δεδομένων από τις συσκευές των χρηστών. Αξιοποιώντας τις δυνατότητες τελευταίων τεχνολογιών του παγκόσμιου ιστού, δίνει την δυνατότητα σε κάθε χρήστη να επιλέξει τι είδους δεδομένα θέλει να καταγράφονται, από ποιές πηγές, να παρακολουθεί το ιστορικό και να διαχειρίζεται τις μετρήσεις του. Παράλληλα προσφέρεται η δυνατότητα σε ερευνητές και διαχειριστές του συστήματος, η συνάθροιση, επισήμανση και ανωνυμοποίηση των δεδομένων για ερευνητικούς σκοπούς επι του συστήματος.

Στο Σχήμα 1, παρουσιάζεται μια αφαιρετική γενική άποψη του συστήματος που υλοποιήθηκε, στο οποίο φαίνονται τόσο τα μέρη που απαρτίζουν το σύστημα συνολικά, αλλά και ποιες κατηγορίες χρηστών έχουν πρόσβαση σε αυτά.



Σχήμα 1: Γενική Αρχιτεκτονική του Συστήματος

1.2 Συνεισφορά της διπλωματικής εργασίας

Η βασικότερη συνεισφορά της εργασίας αυτής είναι πως θέτει τις βάσεις τόσο σε θεωρητικό επίπεδο όσο και σε επίπεδο υλοποίησης για ένα σύστημα γενικού. Μπορεί να φιλοξενήσει πολλές διαφορετικές συσκευές συλλογής ετερογενών βιοσημάτων και να επιτρέψει την εύκολη διαχείριση τους τόσο σε επιχειρηματικό και σε ερευνητικό περιβάλλον, αφού αφαιρεί εξ'ολοκλήρου την ανάγκη για εκ νέου ανάπτυξη κάποιου συστήματος.

Πιλοτικά, αποτελεί το πρώτο βασικό κομμάτι ενός ερευνητικού προγράμματος με στόχο την ανάδειξη των ευκαιριών που παρουσιάζονται στα οικοσυστήματα των φορητών συσκευών (wearables), του Διαδικτύου των Πραγμάτων (Internet of Things) καθώς και του Quantified Self. Μέσα από την ανάπτυξη και υλοποίηση της παρούσας δουλειάς, έχουν τεθεί βάσεις για μελλοντικές εργασίες και επίσημες δημοσιεύσεις των αποτελεσμάτων, αναδεικνύοντας έτσι την επικαιρότητα και τη χρησιμότητα της εφαρμογής.

1.3 Οργάνωση κειμένου

Η διπλωματική εργασία είναι οργανωμένη σε έξι κεφάλαια ως εξής:

Στο Κεφάλαιο 2 διαμορφώνεται το βασικό θεωρητικό και τεχνικό υπόβαθρο, άμεσα σχετιζόμενο με την ανάπτυξη του συστήματος. Επιπλέον πραγματοποιείται μια επισκόπηση της τρέχουσας έρευνας και των τάσεων στο συγκεκριμένο ερευνητικό πεδίο.

Στο Κεφάλαιο 3 περιγράφεται σε υψηλό επίπεδο η αρχιτεκτονική του συστήματος, ενώ παρουσιάζονται: η ανάλυση προδιαγραφών, οι απαιτήσεις και τα σημαντικότερα σενάρια χρήσης και λειτουργιών του συστήματος.

Στο Κεφάλαιο 4 παρουσιάζονται οι λεπτομέρειες υλοποίησης του συστήματος συνολικά/καθολικά, αναλυόντας την λειτουργία και διασύνδεση όλων των επιμέρους υποσυστημάτων του εξυπηρετητή(server), της συσκευής συλλογής δεδομένων κίνησης(smartwatch) και των διεπαφών χρήστη(user interfaces). Παράλληλα, εξηγούνται λεπτομερώς οι τεχνικές που χρησιμοποιήθηκαν για την αποδοτική συλλογή, επικοινωνία και ασφαλή αποθήκευση όλων των δεδομένων των χρηστών με αναφορές στα σημαντικότερα σημεία του κώδικα.

Στο Κεφάλαιο 5 παρουσιάζεται το αποτέλεσμα της υλοποίησης, δηλαδή οι διεπαφές με τις οποίες επιδρά ο χρήστης και πως αυτές εξυπηρετούν τις προδιαγραφές του συστήματος. Επιπλέον αποτιμάται συνολικά η λειτουργικότητα και η προστιθέμενη αξία της διπλωματικής, ενώ καταγράφονται μελλοντικές επεκτάσεις και βελτιώσεις της λειτουργικότητας του συστήματος.

Στο Κεφάλαιο 6 συγκεντρώνεται η βιβλιογραφία που χρησιμοποιήθηκε κατα την εκπόνηση της διπλωματικής εργασίας.

2 Θεωρητικό και Τεχνολογικό Υπόβαθρο - Βασικές έννοιες

Σκοπός του παρόντος κεφαλαίου είναι η συνοπτική εισαγωγή του αναγνώστη στο απαιτούμενο θεωρητικό και τεχνολογικό υπόβαθρο, για την ευκολότερη κατανόηση των εννοιών και μεθόδων πραγματεύονται στα επόμενα κεφάλαια. Καθότι η παρούσα διπλωματική εργασία, συνδυάζει δύο διαφορετικούς επιστημονικούς κλάδους, αυτόν της Πληροφορικής και αυτόν της Ιατρικής, το κεφάλαιο χωρίζεται σε δύο ενότητες. Στην πρώτη ενότητα, θα εξετάσουμε ιατρικά ζητήματα σχετικά με την σωματική δραστηριότητα, τα βιοσήματα και την ποσοτικοποίηση τους. Στην δεύτερη ενότητα, θα αναλύσουμε τις βασικότερες τεχνολογίες που χρησιμοποιήθηκαν στην πρακτική υλοποίηση της εργασίας. Τελος θα αναφέρουμε τα προγραμματιστικά εργαλεία και περιβάλλοντα που χρησιμοποιουμε κατα την διάρκεια ανάπτυξης του συστήματος.

2.1 Θεωρητικό Υπόβαθρο

Στην παρούσα εργασία, το σύστημα είναι σε θέση να καταγράψει και να αναλύσει πληθώρα δεδομένων κίνησης και βιοσημάτων άμεσα συνδεδεμένων με την ανάλυση της σωματικής δραστηριότητας. Συγκεκριμένα με την χρήση της συσκευής Samsung Gear S3 Frontier έχουμε πρόσβαση στους αισθητήρες επιταχυνσιόμετρο, γυροσκοπίου, G.P.S. και καρδιακού ρυθμού. Με βάση λοιπόν αυτές τις μετρίσιμες συνιστώσες θα παράξουμε μέσω της εφαρμογής datasets ώστε μελλοντικά να μελετηθεί λεπτομερώς η σωματική δραστηριότητα.

2.1.1 Σωματική Δραστηριότητα

Στην ενότητα αυτή θα γίνει μια εν’συντομία αναφορά, στους τύπους των δεδομένων που θα καταγράφει και θα αναλύει το σύστημα. Συγκεκριμένα τα δεδομένα αυτά εμπίπτουν σε δύο κατηγορίες: δεδομένα που αφορούν την σωματική δραστηριότητα και δεδομένα βιοσημάτων χρήστη.

2.1.1.1 Ορισμός Φυσικής Δραστηριότητας

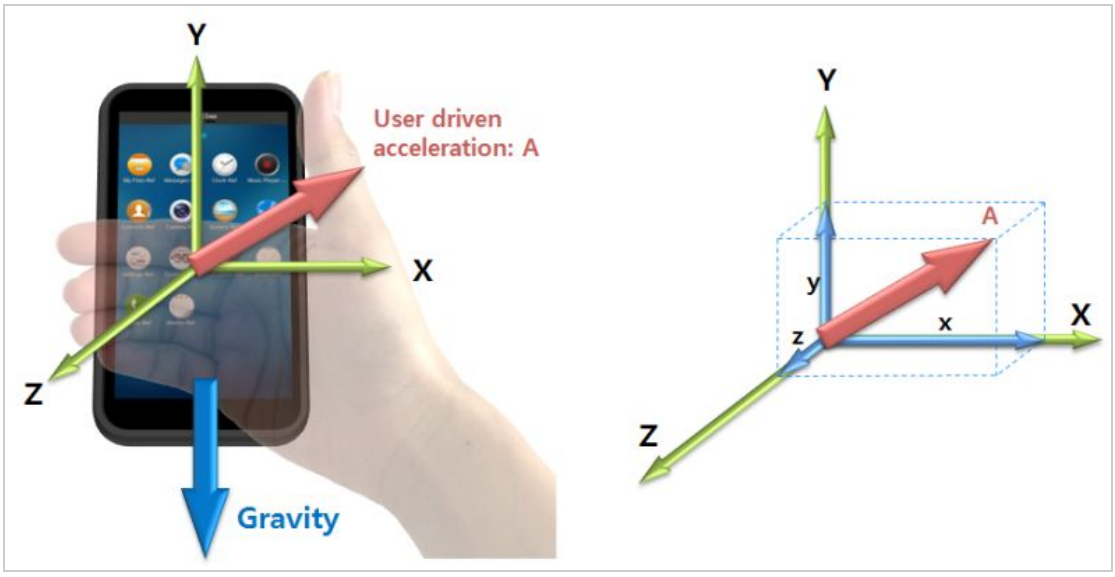
Activities of Daily Living(ADL)[1] ορίζονται οι βασικές δραστηριότητες στις οποίες επιδίδεται ο άνθρωπος στην καθημερινή του ζωή. Αποτελούν δείκτη της λειτουργικής ικανότητας του ανθρώπινου σώματος, ιδιαίτερα σε άτομα που έχουν υποστεί τραυματισμούς ή είναι ηλικιωμένα με ενδεχομένως κινητικές δυσκολίες. Η σωματική δραστηριότητα είναι αδύνατον να ποσοτικοποιηθεί μόνο με την αυστηρή μέτρηση ενός υπολογισμού μεγέθους. Αντ’ αυτού, χρησιμοποιούμε πληθώρα αισθητήρων που μετρούν συγκεκριμένες παραμέτρους και στην συνέχεια συνθετικά αποφαινόμαστε για τις βασικές σωματικές δραστηριότητες στις οποίες επιδίδεται ο χρήστης στην καθημερινότητα. Τέλος στις ADL μπορούν να περιλαμβάνονται διαφορετικές δραστηριότητες ανάλογα με τη φύση της εφαρμογής και τα προβλήματα που αντιμετωπίζουν τα άτομα τα οποία αφορούν[11]. Η κατηγοριοποίηση και η ανάλυση των δραστηριοτήτων όμως δεν βρίσκονται εντός του πεδίου της παρούσας εργασίας, παρα μόνο η πρωτογενής συλλογή δεδομένων για την επιτυχή αναγνώριση τους.

2.1.1.2 Τρόποι Μέτρησης

Στο παρόν σύστημα οι συνισταμένες που μετριούνται για να την μελλοντική μελέτη των ADL από τους αισθητήρες είναι: οι χωρικές επιταχύνσεις του καρπού του χρήστη απο το επιταχυνσιόμετρο και το γυροσκόπιο, η γεωγραφική θέση του χρήστη απο το G.P.S. και ο καρδιακός του ρυθμός του χρήστη από τον αντίστοιχο αισθητήρα.

2.1.1.2.1 Επιταχυνσιόμετρο

Ο αισθητήρας του επιταχυνσιομέτρου[2][3], αποτελεί έναν Μικροηλεκτρονικόμηχανικό (Micro-ElectroMechanical System - MEMS) αδρανειακό αισθητήρα ο οποίος μετρα γραμμικές επιταχύνσεις στον χώρο, κατά μήκος των τριών διαστάσεων της συσκευής. Η αρχή λειτουργίας του βασίζεται στην μέτρηση αλλαγών στην χωρητικότητα ενός πυκνωτή, του οποίου η χωρητικότητα μεταβάλλεται καθώς αποτελεί μέρος αναρτημένης μάζας και η τιμή του μεταβάλλεται με την κίνηση αυτής της. Με αυτόν τον τρόπο, μπορεί και μετρά στατικές δυνάμεις, όπως η επιτάχυνση της βαρύτητας αλλά και δυναμικές όταν προέρχονται από αλλαγές στην ταχύτητα και την διεύθυνση της κίνησης, πράγμα που το κάνει χρήσιμο για ανίχνευση του προσανατολισμού της συσκευής.

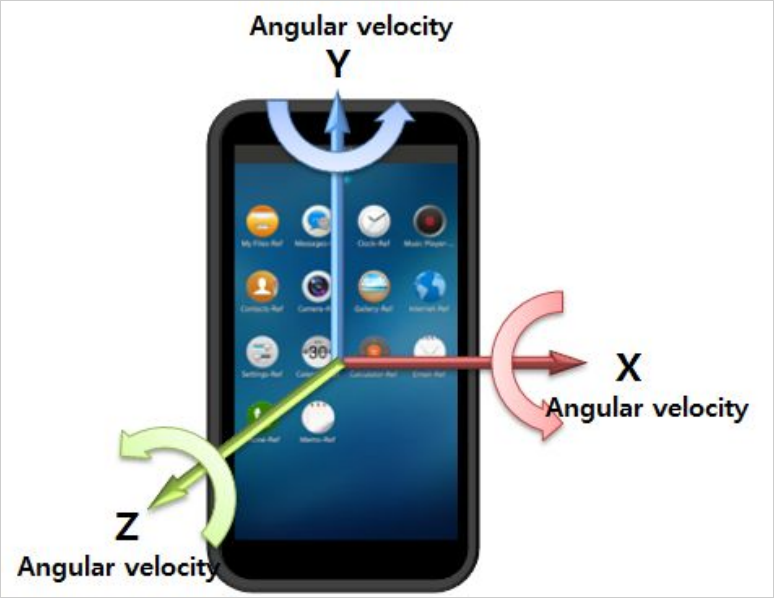


Σχήμα 2: Προσανατολισμός των Αξόνων Μέτρησης Επιταχύνσεων απο το Επιταχυνσιόμετρο στις Συσκευές Tizen

2.1.1.2.2 Γυροσκόπιο

Ο αισθητήρας του γυροσκοπίου[3][4], αποτελεί έναν Μικροηλεκτρονικόμηχανικό (Micro-ElectroMechanical System - MEMS) αδρανειακό αισθητήρα και σε αντιστοιχία με τον αισθητήρα του επιταχυνσιόμετρου, μετρά στροφικές επιταχύνσεις στον χώρο, γύρω από τους άξονες των τριών διαστάσεων της συσκευής. Η αρχή λειτουργίας του βασίζεται στην ανίχνευση της Coriolis

Acceleration του ενός δονούμενου στοιχείου όπως μιας χορδής καθώς το γυροσκόπιο περιστρέφεται. Με αυτόν τον τρόπο, μπορεί και μετρά αλλαγές στην περιστροφή της συσκευής.



Σχήμα 3: Προσανατολισμός των Αξόνων Μέτρησης Περιστροφικών Επιταχύνσεων απο το Γυροσκόπιο στις Συσκευές Tizen

2.1.1.2.3 Παγκόσμιο Σύστημα Στιγματοθέτησης (G.P.S)

Ο αισθητήρας του G.P.S.(Global Positioning System ή Παγκόσμιο Σύστημα Στιγματοθέτησης)[3][5] αποτελεί έναν πομποδέκτη οποίος ο προσδιοριζει την γεωγραφική του θέση σε πραγματικό χρόνο. Η αρχή λειτουργίας του βασίζεται στην χρήση τεσσάρων απο το πλέγμα των εικοσιτεσσάρων διαθέσιμων δορυφόρων της Γης στους οποίους υπάρχουν πομποδέκτες GPS και παρέχουν συνδιαστηκά στον δέκτη της συσκευής τις απαραίτητες πληροφορίες όπως τη θέση, το υψόμετρο, την ταχύτητα και την κατεύθυνση κίνησης του.

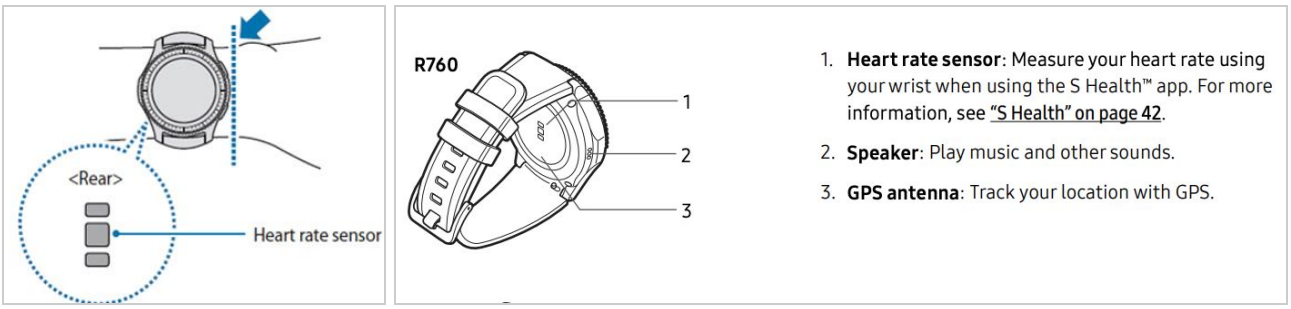
2.1.2 Βιοσήματα

Στα πλαίσια της Βιοϊατρικής, ένα βιοσήμα αποτελεί την περιγραφή ενός φυσιολογικού φαινομένου, ανεξάρτητα της φύσης της περιγραφής. Βιοσήμα μπορεί να θεωρηθεί η οπτική επιθεώρηση ενός ασθενή, ενώ επίσης μπορεί να θεωρηθεί και η παρακολούθηση του ανθρώπινου σώματος μέσω αισθητήρων, όπως για παράδειγμα το ηλεκτροκαρδιογράφημα. Έτσι τα βιοσήματα, εξ'ορισμού μπορούν να χρησιμοποιηθούν τόσο σε διάγνωση όσο και σε θεραπεία των ασθενών, αφού μπορούν να προβλέψουν παθήσεις, ή να βρουν την πηγή παθήσεων, μελετώντας τις τιμές του ασθενή με μοντέλα που έχουν δημιουργηθεί από προηγούμενες αναλύσεις.

2.1.2.1 Ο Καρδιακός Ρυθμός

Καρδιακός ρυθμός(Heart Rate) ονομάζεται η μετάδοση του κύματος αίματος, που προκαλείται από την καρδιακή συστολή, στο τοίχωμα των αγγείων[6][7][8]. Ο σφυγμός(ή παλμός) οφείλεται στην μεταβολή της πίεσης του αίματος που προκαλείται από τις κοιλιακές συστολές της καρδιάς και της ελαστικότητας των αρτηριών. Μονάδα μέτρησης του καρδιακού ρυθμού είναι οι χτύποι ανά λεπτό (Beats Per Minute-BPM).

Το βιοσήμα του καρδιακού ρυθμού στις smartwatch συσκευές, όπως και στο Samsung Gear S3 Frontier μετριέται μέσω του αισθητήρα που ονομάζεται φωτοπληθυσμιογράφος (Photoplethysmiograph - PPG)[3][6]. Ο αισθητήρας αποτελείται από ένα λαμπάκι led και έναν οπτικό αισθητήρα, ο οποίος μετρά την ένταση του φωτός που δέχεται από το led. Ο φωτοπληθυσμιογράφος βρίσκεται σε επαφή με δέρμα του χρήστη στο ύψος του καρπού, αφού βρίσκεται στην πλευρά του smartwatch που έχει φυσική επαφή με τον χρήστη. Σε κάθε χτύπο της καρδιάς, το αίμα που περνάει από το συγκεκριμένο σημείο, αλλάζει σε όγκο, αλλάζοντας και το ποσό του φωτός που αντανακλάται και δέχεται ο οπτικός αισθητήρας του φωτοπληθυσμιογράφου. Περισσότερος όγκος αίματος σημαίνει πως το ποσό του φωτός το οποίο θα ληφθεί από τον οπτικό αισθητήρα είναι λιγότερο και αντιστροφα. Η αυξομείωση αυτή του φωτός που δέχεται ο οπτικός αισθητήρας, σε κάθε χτύπο της καρδιάς, παράγει ένα αναλογικό σήμα που είναι ανάλογο του χτύπου της καρδιάς. Με αυτό το τρόπο, είναι δυνατή η μέτρηση του καρδιακού παλμού μέσω της χρήσης φωτοπληθυσιογράφου. Η μέθοδος αυτή εφαρμόζεται μαζικά στο εμπόριο τα τελευταία χρόνια, λόγω του μικρού μεγέθους του αισθητήρα και του πολύ μικρού του κόστους, με αποτέλεσμα να συναντάται και σε έξυπνα ρολόγια, όπως αυτό που χρησιμοποιήθηκε για τη συλλογή των δεδομένων.



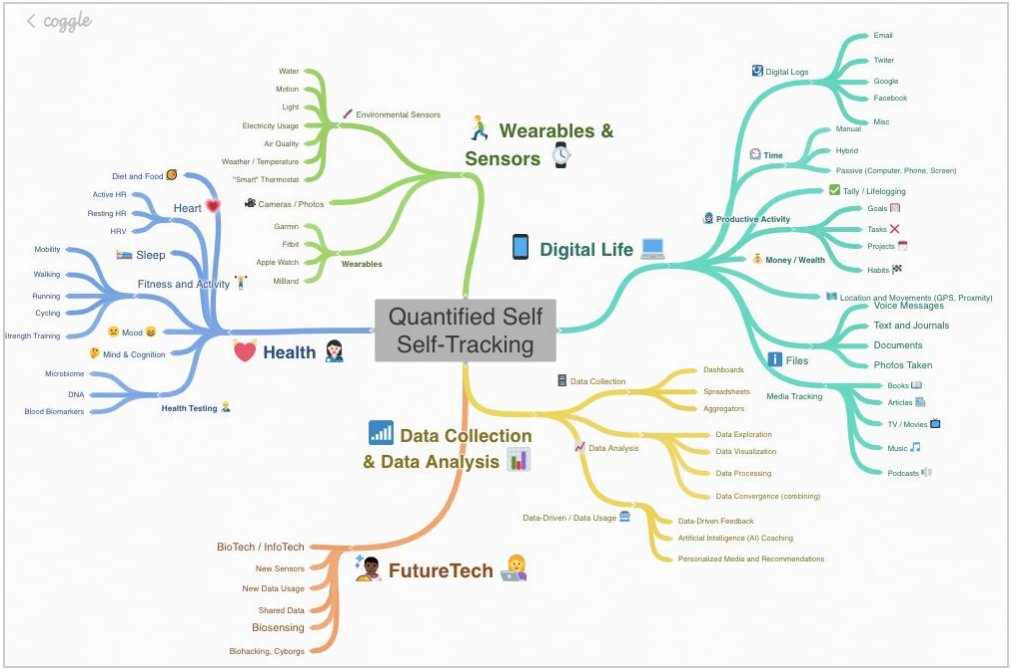
Σχήμα 4: Γραφική Αναπαράσταση Αισθητήρα Μέτρησης Καρδιακού Ρυθμού στο Samsung Gear S3 Frontier Smartwatch

2.2 Τεχνολογικό Υπόβαθρο

Σε αυτή την ενότητα, θα παρουσιάσουμε την απαιτούμενη τεχνική ορολογία, καθώς και τις τεχνολογίες, τα εργαλεια και τα περιβάλλοντα εκτέλεσης που συνθέτουν το σύστημα.

2.2.1 Quantified Self

Η ευρεία ύπαρξη πληθώρας συσκευών για την παρακολούθηση φυσικής δραστηριότητας χρηστών, έχει οδηγήσει στην ανάπτυξη του κοινωνικού φαινομένου του Quantified Self[9][10]. Όλο και περισσότεροι είναι οι χρήστες που τα τελευταία χρόνια χρησιμοποιούν τέτοιες συσκευές, που μέσω εφαρμογών καταγράφουν, ποσοτικοποιούν και αναλύουν τα δεδομένα της φυσικής τους δραστηριότητας. Μεσα απο την ανάλυση των δεδομένων αυτών, παρέχονται στους χρήστες πληροφορίες και συμβουλές για το πως μπορούν να βελτιώσουν τα αποτελέσματα και τις επιδόσεις τους σε διάφορες καθημερινές δραστηριότητες. Υπάρχουν κατάλληλα τεχνολογικά εργαλεία που επιτρέπουν την αυτόματη καταγραφή δεδομένων όπως: βιοσήματα (βάρος, καρδιακός παλμός κλπ.), σωματική δραστηριότητα (περπάτημα, τρέξιμο, αριθμός βημάτων κλπ.), διατροφικές συνήθειες (κατανάλωση θερμίδων, καφέ, αλκοόλ, κλπ.), ή ακόμα και την ψυχική διάθεση, τον βαθμό ευτυχίας και την προσωπική οικονομική διαχείριση. Μάλιστα, μια σημαντική και ενδιαφέρουσα πτυχή του Quantified Self αποτελεί η κοινωνική του διάσταση, καθώς, συχνά, ο χρήστης μοιράζεται τα δεδομένα του με άλλους χρήστες στο δίκτυο του, ενώ ταυτόχρονα, συγκρίνει τα δικά του αποτελέσματα με αυτά του συνόλου.



Σχήμα 5: Ιεραρχική Αναπαράσταση του Οικοσυστήματος του Quantified Self

2.2.2 Internet of Things

Το Internet of Things[11][12], αποτελεί ένα οικοσύστημα όπου διαδικτυωμένα φυσικά αντικείμενα ενσωματώνουν κάποιου είδους υπολογιστικό σύστημα, ώστε να μπορούν να επικοινωνούν, να ανταλλάσσουν και να επεξεργάζονται δεδομένα και πληροφορίες. Τέτοια φυσικά διαδικτυωμένα αντικείμενα δεν αποτελούν μόνο οι παραδοσιακοί υπολογιστές, όπως laptop, PC, smartphones και

tablets, αλλά κάθε είδους “έξυπνα” αντικείμενα του φυσικού κόσμου, από αισθητήρες και οικιακές συσκευές μέχρι αυτοκίνητα και ολόκληρα κτίρια. Η έκρηξη του πλήθου, της χρησιμότητας αλλά και της ποικιλίας των συσκευών που πλέον συνδέονται μεταξύ τους και στο Internet έχει οδηγήσει στην ανάπτυξη του οικοσυστήματος που ονομάζεται Internet of Things (Διαδίκτυο Πραγμάτων). Ιδιαίτερη σημασία, στα πλαίσια της διπλωματικής εργασίας, έχουν οι “έξυπνες” ιατρικές συσκευές, όπως ζυγαριές, πιεσόμετρα, οξύμετρα. Οι συσκευές αυτές, αποτελούν παραλλαγές των αντίστοιχων συμβατικών ιατρικών οργάνων που, αφενός, λαμβάνουν μετρήσεις των απαραίτητων βιοσημάτων και αφετέρου, έχουν την δυνατότητα να συνδεθούν ασύρματα σε κάποιου είδους δίκτυο ώστε να μεταδώσουν αυτές τις μετρήσεις. Με την σύνθεση ροών δεδομένων από τέτοιου είδους αισθητήρες και με την χρήση προηγμένων αλγορίθμων για την αναλυση τους προκύπτουν τα συστήματα καταγραφής και ανάλυσης φυσικών δραστηριοτήτων ή Activity Tracking Systems[45][46]. Στη πληθώρα τους αποτελούν εφαρμογές έξυπνων κινητών που καταγράφουν την καθημερινότητα του χρήστη, τις διατροφικές του συνήθειες, τις αθλητικές του δραστηριότητες, τα έξοδα του ή οποιαδήποτε άλλη ομάδα δεδομένων μπορεί να ποσοτικοποιηθεί μέσα από ροές δεδομένων.

Σχήμα 6: Αναπαράσταση του Οικοσυστήματος του Internet of Things

2.2.3 Cloud Computing

Το Cloud Computing(υπολογιστικό νέφος)[14] παρέχει υπηρεσίες υπολογισμού, λογισμικού, διαμοιρασμού πόρων, πρόσβασης σε πληροφορίες και αποθήκευσης δεδομένων χωρίς να απαιτείται η εξ αρχής διαμόρφωση του συστήματος από τον χρήστη. Αποτελεί ένα νέο επιχειρηματικό μοντέλο παροχής υπηρεσιών και συνιστά μια μεγάλη αλλαγή στον τρόπο με τον οποίο παρέχονται οι υπολογιστικοί πόροι. Αυτό συμβαίνει καθώς επιτρέπει την ανάκτηση πληροφοριών, την επεξεργασία

και την αποθήκευση δεδομένων και τη χρήση εφαρμογών όπου και αν βρισκόμαστε χωρίς την απαίτηση για αγορά και εγκατάσταση των φυσικών μηχανημάτων.

2.2.4 Τεχνολογίες Προηγμένων Φορητών Συσκευών και Samsung Gear S3

Τα έξυπνα ρολόγια είναι ψηφιακές συσκευές των οποίων οι δυνατότητες δεν περιορίζονται μόνο στην εμφάνιση ώρας. Παρότι τα πρώτα μοντέλα είχαν πολύ περιορισμένες δυνατότητες, τα σύγχρονα αποτελούν ουσιαστικά υπολογιστές χειρός αφού ενσωματώνουν λειτουργικά συστήματα φορητών συσκευών και μπορούν να εκτελούν πληθώρα εφαρμογών. Τα γνωστότερα λειτουργικά συστήματα που τρέχουν σε έξυπνα ρολόγια είναι το Android Wear, WatchOS, Pebble OS, WebOs και TizenOs. Τα περισσότερα είναι σχεδιασμένα με τέτοιο τρόπο ώστε να επιτρέπουν τη σύνδεση μέσω bluetooth με κάποιο smartphone ή tablet ή/και Internet. Αυτό επιτρέπει την ανταλλαγή μηνυμάτων μεταξύ των δύο συσκευών και του δικτύου αυξάνοντας κατα πολυ την χρηστικότητα τους. Τα περισσότερα διαθέτουν επαναφορτιζόμενη μπαταρία η οποία μπορεί να διαρκέσει από μερικές ώρες έως και μια ολόκληρη εβδομάδα με μια φόρτιση. Η οθόνη μπορεί να είναι είτε τετράγωνη είτε κυκλική, έγχρωμη, ασπρόμαυρη ενώ κάποια διαθέτουν ακόμα και οθόνη αφής. Τα έξυπνα ρολόγια επίσης ενσωματώνουν διαφόρων ειδών αισθητήρες όπως θερμόμετρο, επιταχυνσιόμετρο, αλτίμετρο, βαρόμετρο, πυξίδα, γυροσκόπιο, GPS και μετρητή καρδιακών παλμών. Κάποια διαθέτουν επίσης ενσωματωμένη κάμερα και μικρόφωνο. Τέλος, ιδιαίτερα διαδεδομένες στα έξυπνα ρολόγια είναι οι εφαρμογές παρακολούθησης φυσικής κατάστασης (fitness tracking) που είναι ευρέως διαθέσιμες.

Στην παρούσα διπλωματική, όπως έχει αναφερθεί ήδη, επιλέχθηκε το έξυπνο ρολόι Samsung Gear S3 Frontier smartwatch[15] για την ανάπτυξη της εφαρμογής συλλογής και αποθήκευσης δεδομένων. Το Samsung Gear S3 τρέχει το λειτουργικό σύστημα TizenOS[50] το οποίο έχει αναπτυχθεί από την Samsung και την Intel και βασίζεται στο MeeGo που με την σειρά του χρησιμοποιεί στοιχεία από το Linux. Το Gear S3 διαθέτει 2 κουμπιά (Back και Home) καθώς και περιστρεφόμενο κυκλικό δακτύλιο. Το κουμπί Back χρησιμοποιείται για να μεταφέρει το χρήστη ένα παράθυρο προς τα πίσω ενώ το κουμπί Home τον επιστρέφει στην αρχική οθόνη πλοήγησης της συσκευής. Ο περιστρεφόμενος κυκλικός δακτύλιος χρησιμοποιούνται για περιήγηση του χρήστη στα μενού και τις εφαρμογές της συσκευής. Ο επεξεργαστής είναι της σειράς Exynos 7 Dual 7270. Συγκεκριμένα χρησιμοποιεί τον διπύρηνο Cortex-A53 με μέγιστη συχνότητα ρολογιού τα 1.0 GHz. Διαθέτει επίσης 768MB μνήμης RAM και 4GB κύριας μνήμης. Η οθόνη είναι έγχρωμη AMOLED, χαμηλής κατανάλωσης ενέργειας, με ανάλυση 360 x 360(~278 ppi) και είναι πάντα αναμμένη κατα την λειτουργία της συσκευής. Το ρολόι επίσης διαθέτει αισθητήρα φωτός του περιβάλλοντος (ambient light sensor), βαρόμετρο επιταχυνσιόμετρο 3 αξόνων, γυροσκόπιο 3, G.P.S. και αισθητήρα μέτρησης καρδιακού παλμού. Η σύνδεση με συσκευές smartphone μέσω Bluetooth 4.2, A2DP, LE. Ενώ παρέχεται και η δυνατότητα για σύνδεση με το διαδίκτυο μέσω Wi-Fi 802.11 b/g/n. Η μπαταρία είναι λιθίου, 380 mAh και διαρκεί περίπου 72 ώρες, με μια φόρτιση, υπό φυσιολογική χρήση. Η φόρτιση γίνεται ασύρματα μέσω μαγνητικής βάσης που εφάπτεται το ρολόι. Τέλος η συσκευή παρουσιάζει αδιαβροχοποίηση πιστοποιημένη με IP68.

Σχήμα 7: Η συσκευή smartwatch Samsung Gear S3 Frontier

2.2.5 Τεχνολογίες Διαδικτύου

Σε αυτή την ενότητα, θα παρουσιάσουμε τις διαδικτυακές τεχνολογίες και εργαλεία που χρησιμοποιεί το σύστημα. Παράλληλα, θα αναλύσουμε τις αρχές και λεπτομέρειες λειτουργίας τους.

2.2.5.1 Περιηγητές Ιστού (Web Browsers)

Οι περιηγητές ιστού ή αλλιώς web browsers, είναι προγράμματα που επιτρέπουν στους χρήστες να επισκέπτονται και να αλληλεπιδρούν με πληροφορίες συνήθως αναρτημένες σε κάποιον ιστότοπο στον Παγκόσμιο Ιστό ή σε ένα τοπικό δίκτυο[16]. Πλέον η χρήση τους είναι τόσο διαδεδομένη -έως και απαραίτητη- που εκτελούνται σε οποιοδήποτε λειτουργικό σύστημα που παρουσιάζει γραφική διεπαφή και έχει συνδεσιμότητα με το διαδίκτυο. Οι Web browsers επιτρέπουν στον χρήστη την γρήγορη και εύκολη πρόσβαση σε πληροφορίες που βρίσκονται σε διάφορες ιστοσελίδες εκτελώντας HTTP αιτήματα προς διάφορες διαδικτυακές υπηρεσίες και χρησιμοποιούν τη γλώσσα μορφοποίησης HTML για την προβολή τους.

2.2.5.2 Web Applications ως Ολοκληρωμένες Εφαρμογές

Μια Εφαρμογή Ιστού (Web Application) έχει ως σκοπό την αλληλεπίδραση του χρήστη όχι μόνο με απλή προβολή κειμένου και εικόνας αλλά με πιο πλούσια και δυναμική λειτουργικότητα. Η γλώσσα HTML(Hypertext Markup Language) αναλαμβάνει την ιεραρχική δόμηση της πληροφορίας της εφαρμογής. Ο κώδικας HTML εκτελείται από τον browser του χρήστη και εμφανίζει το επιθυμητό περιεχόμενο στην οθόνη με την μορφή γραφικής διεπαφής. Συνάμα η χρήση της γλώσσα CSS (Cascading Style Sheet) παραμετροποιεί την εμφάνιση του HTML περιεχομένου, προσθέτοντας γνωρίσματα όπως στυλ γραμματοσειράς, χρώμα, διαστάσεις κτλ. Τέλος με την χρήση της γλώσσας προγραμματισμού JavaScript οι ιστοσελίδες είναι δυναμικές αφού υπάρχει απευθείας αλληλεπίδραση

της με την ιεραρχική δομή της HTML. Έτσι εκτελούνται δυναμικά υπολογισμοί, αλγόριθμοι και επικοινωνία με τον εξυπηρετητή.

2.2.5.2.1 Κεντρικός Διακομιστής - Node.js & Express.js

Για την ανάπτυξη του διαδικτυακού εξυπηρετητή, χρησιμοποιήσαμε το Node.js[18] που αποτελεί μια πλατφόρμα ανάπτυξης λογισμικού σε περιβάλλον JavaScript, βασισμένο πάνω στη μηχανή JavaScript V8 του Chrome (Chrome V8 JavaScript Engine)[17]. Σε αντίθεση με τους παραδοσιακούς εξυπηρετητές που στηρίζονται στην πολυνηματικότητα, το Node.js ακολουθεί την προσέγγιση της ασύγχρονης επικοινωνίας εισόδου/εξόδου. Όπως ο κωδικας JavaScript που τρέχει σε έναν browser βασίζεται στα callbacks και στην ασύγχρονη εκτέλεσή τους, έτσι όταν ο επεξεργαστής περιμένει ένα γεγονός, ορίζεται μια συνάρτηση επιστροφής η οποία θα κληθεί όταν ολοκληρωθεί η διεργασία. Μέχρι να ολοκληρωθεί η διαδικασία ο επεξεργαστής μένει ανενεργός και ελεύθερος για την εκτέλεση άλλων διεργασιών, ελαχιστοποιώντας την αναμονή και την απαίτηση σε μνήμη. Για παράδειγμα, όταν ο Server λάβει ένα HTTP αίτημα και πρέπει να διαβάσει ένα αρχείο ώστε να εξυπηρετήσει το αίτημα, τότε αναθέτει στο λειτουργικό σύστημα το άνοιγμα του αρχείου και μέχρι να ολοκληρωθεί η διαδικασία, ξεκινά να εξυπηρετεί το επόμενο αίτημα. Όταν ολοκληρωθεί επιτυχώς το άνοιγμα και η ανάγνωση του αρχείου, επιστρέφει το επιθυμητό αποτέλεσμα.

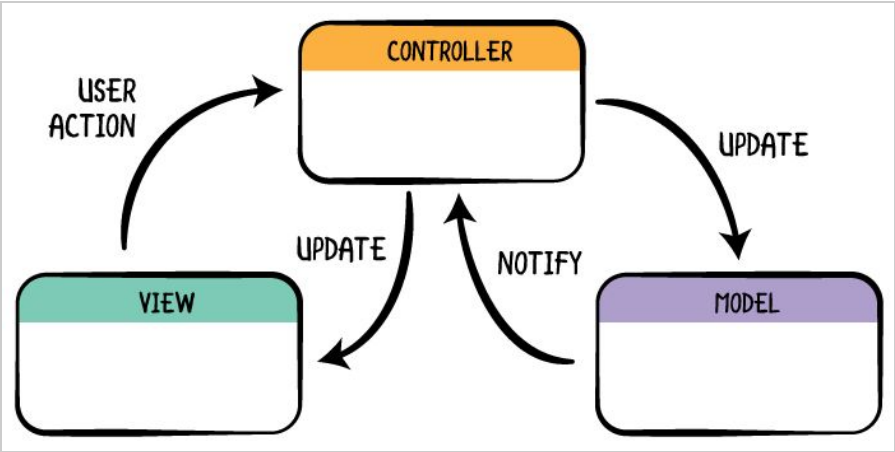
Στα πλαίσια της ανάπτυξης λογισμικού με την χρήση του Node.js, χρησιμοποιείται και το Express.js[19] που αποτελεί ένα Web Application Framework (WAF). Είναι κατάλληλο για την ανάπτυξη εφαρμογών ιστού και APIs, καθώς υλοποιεί πάρα πολλές βοηθητικές μεθόδους HTTP και Middleware, δηλαδή λογισμικό που μεσολαβεί μεταξύ του λειτουργικού συστήματος ή της βάσης δεδομένων και της εφαρμογής.

2.2.5.2.2 Μοντέλο MVC

Για την σχεδίαση της αρχιτεκτονικής και της λειτουργίας του συστήματος ακολουθήθηκε το σχεδιαστικό πρότυπο του Model–View–Controller (MVC Design Pattern)[20]. Η σχεδίαση αυτή, αποτελείται από τρία μέρη:

- Τα Μοντέλα(Models), που αναπαριστούν τις σχετικές με την εφαρμογή πληροφορίες και δεδομένα και μπορούν να ενημερώνουν τυχόν παρατηρητές όταν η κατάστασή τους αλλάζει.
- Οι Όψεις(Views), που θεωρούνται οι διεπαφές του χρήστη με την εφαρμογή (π.χ. ο κώδικας HTML και CSS). Πρέπει να γνωρίζουν για την ύπαρξη των μοντέλων, έτσι ώστε να τα παρατηρούν, αλλά δεν επικοινωνούν κατευθείαν μαζί τους.
- Οι Ελεγκτές(Controllers), που υλοποιούν τη λογική παρουσίασης (presentation logic) της εφαρμογής. Αυτοί είναι που παίρνουν τις αποφάσεις και είναι ο συνδετικός κρίκος μεταξύ Μοντέλων και Όψεων. Η πληθώρα των MVC frameworks που υπάρχουν διαθέσιμα σήμερα, ανάλογα με τους στόχους τους, δεν υλοποιούν με τον ίδιο τρόπο το παραπάνω προγραμματιστικό μοτίβο αλλά συνήθως κάποια παραλλαγή του. Συνήθως, παρατηρείται η συγχώνευση του ρόλου του ελεγκτή και της όψης σε μία οντότητα με αυξημένες

αρμοδιότητες και λειτουργικότητα και συναντώνται στη βιβλιογραφία και ως MV(Model-View Design Pattern.)

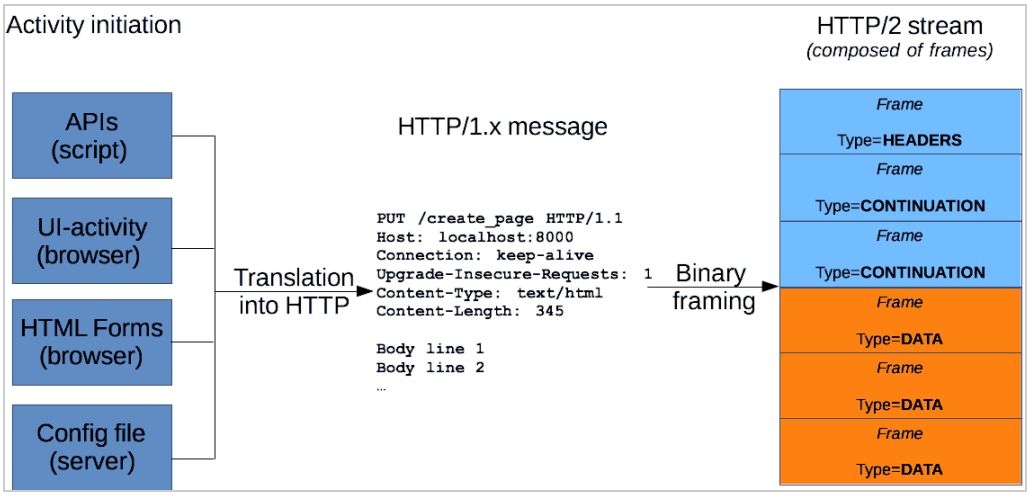


Σχήμα 8: Οντολογική Αναπαράσταση του Σχεδιαστικού Προτύπου Model–View–Controller

2.2.5.2.3 HTTP Αιτήματα

Για την αποστολή μηνυμάτων μέσω του διαδικτύου, είναι ανεξάρτητο πλατφόρμας και γλώσσας προγραμματισμού και χρησιμοποιεί το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol, HTTP)[21]. Συνιστά θεμελιώδες πρωτόκολλο επικοινωνίας που χρησιμοποιείται στους browsers την μεταφορά δεδομένων ανάμεσα σε έναν διακομιστή (server) και έναν πελάτη (client). Κάθε πόρος εντοπίζεται μέσω μιας μοναδικής διεύθυνσης ιστού(web address) η οποία ονομάζεται URL(Universal Resource Locator). Αποτελεί lightweight εναλλακτική σε πιο σύνθετους μηχανισμούς όπως RPC(Remote Procedure Calls) και Web Services(SOAP, WSDL κλπ) ενώ παρά την απλότητά του είναι πλήρες, παρέχοντας τις ίδιες δυνατότητες. Για κάθε εντοπιζόμενο πόρο, το πρωτόκολλο ορίζει ένα σύνολο μεθόδων που μπορουν να εκτελεστούν σε αυτόν. Οι σημαντικότερες εκ των οποίων είναι οι εξής:

- GET: Αίτημα του πελάτη, με το οποίο ζητάει από τον εξυπηρετητή να του στείλει τον πόρο, χωρίς κάποια άλλη παρενέργεια στα δεδομένα δεδομένα του.
- POST: Αίτημα του πελάτη, με το οποίο ζητάει από τον εξυπηρετητή να δεχτεί τα δεδομένα που περιέχονται στο αίτημα σαν νέα υφιστάμενα ενός πόρου ή την δημιουργία νέου. Έπειτα από κάποια ενδεχόμενη επεξεργασία, ο πόρος αποθηκεύετε ώστε να είναι δυνατή η ανάκτηση του στη συνέχεια.
- PUT: Αίτημα του πελάτη, με το οποίο ζητάει από τον εξυπηρετητή την επεξεργασία και την αλλαγή των δεδομένων ενός ήδη υπάρχοντα πόρου.
- DELETE: Αίτημα του πελάτη, με το οποίο ζητάει από τον εξυπηρετητή την διαγραφή ενός πόρου του συστήματος.



Σχήμα 9: Οντολογική Αναπαράσταση ενός HTTP Αιτήματος

2.2.5.2.4 Υπηρεσίες REST & Application Programming Interfaces (API)

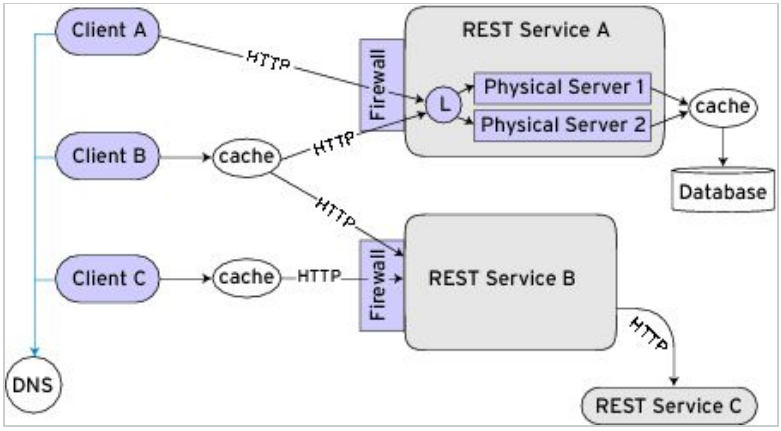
Στο διαδίκτυο όπως έχουμε εξηγήσει η βασική τοπολογία επικοινωνίας είναι αυτή του εξυπηρετητή(server) που παρέχει τα δεδομένα και τους πόρους, και του πελάτη(client) που τροποποιεί και εμφανίζει τα δεδομένα αυτά. Το REST(Representational State Transfer)[22] είναι μια αρχιτεκτονική διασύνδεσης δικτυακών εφαρμογών η οποία χρησιμοποιεί το πρωτόκολλο HTTP για την ανταλλαγή δεδομένων. Αποτελεί έναν τρόπο με τον οποίο μπορούμε να μεταφέρουμε μέσω δικτύου πόρους και δεδομένα από τον εξυπηρετητή δημιουργώντας την ψευδαίσθηση ότι server και client εκτελούνται στο ίδιο περιβάλλον εκτέλεσης. Οι μεταφερόμενοι πόροι αναπαρίστανται με διάφορες μορφές εκ των οποίων οι πιο διαδεδομένες είναι JSON Objects XML και απλό κείμενο.

Στο περιβάλλον μιας RESTful εφαρμογής τα πάντα θεωρούνται πόροι (resources), η πρόσβαση και η επικοινωνία με τους οποίους ακολουθεί τις εξής βασικές αρχές:

- Αρχιτεκτονική Πελάτη – Εξυπηρετητή (Client – Server Architecture): Οι βασικοί δρώντες σε ένα REST σύστημα είναι ο REST Server και ο REST Client. Συγκεκριμένα, ο πελάτης ζητάει πρόσβαση σε συγκεκριμένους πόρους τους οποίους διαθέτει ο εξυπηρετητής, χρησιμοποιώντας ένα μοναδικό αναγνωριστικό. Η αρχή πίσω από το μοντέλο Πελάτη–Εξυπηρετητή είναι ο Διαχωρισμός των Ανησυχιών (Separation of Concerns), κάτι το οποίο απλοποιεί τον σχεδιασμό της διαπροσωπείας χρήστη (User Interface) και υποστηρίζει την κλιμακωσιμότητα (scalability) του συστήματος.
- Ομοιόμορφη Διεπαφή (Uniform Interface): Αποτελεί θεμελιώδη αρχή οποιασδήποτε εφαρμογής του REST. Η ομοιόμορφη διεπαφή απλοποιεί και αποσυνδέει την αρχιτεκτονική πελάτη – διακομιστή, επιτρέποντας σε κάθε πλευρά να εξελιχθεί ανεξάρτητα.
- Χωρίς Κατάσταση (Stateless): Κάθε αίτημα κάποιου πελάτη αντιμετωπίζεται ως ανεξάρτητο και περιέχει όλες τις απαραίτητες πληροφορίες που απαιτούνται για το χειρισμό του. Η κατάσταση της συνεδρίας(Session State) διατηρείται στον πελάτη και όχι στον εξυπηρετητή. Με αυτόν τον τρόπο απλοποιείται ο σχεδιασμός των εξυπηρετητών.

- Χρήση της Cache Μνήμης (Cacheable): Παλαιότερες απαντήσεις του server, αποθηκεύονται στα ενδιάμεσα επίπεδα σε μνήμες Cache, ώστε στην ύπαρξη ίδιου ερωτήματος, να επαναχρησιμοποιηθεί η ίδια απάντηση. βελτιώνεται με αυτόν τον τρόπο η επεκτασιμότητα και η απόδοση του συστήματος.
- Διαστρωματωμένο Σύστημα (Layered System): Ο πελάτης είναι σε θέση να γνωρίζει εάν συνδέεται άμεσα με τον εξυπηρετητή ή με ένα ενδιάμεσο επίπεδο. Οι ενδιάμεσοι εξυπηρετητές μπορούν να βελτιώσουν την επεκτασιμότητα του συστήματος αλλά και την ασφάλειά του.
- Κώδικας κατά Απαίτηση (Code on Demand): Οι διακομιστές έχουν τη δυνατότητα να παρατείνουν προσωρινά ή να προσαρμόζουν τη λειτουργία ενός πελάτη με τη μεταφορά εκτελέσιμου κώδικα.

Τα API(Application Programming Interfaces)[23] ή αλλιώς Διεπαφές Προγραμματισμού Εφαρμογών αποτελούν την διεπαφή των προγραμματιστικών διαδικασιών που χρησιμοποιεί μια υπηρεσία. Μια RESTful εφαρμογή επιτρέπει προς τα APIs τις αιτήσεις από άλλα προγράμματα για ανταλλαγές και επεξεργασία δεδομένων.



Σχήμα 10: Οντολογική Αναπαράσταση μια RESTful Αρχιτεκτονικής

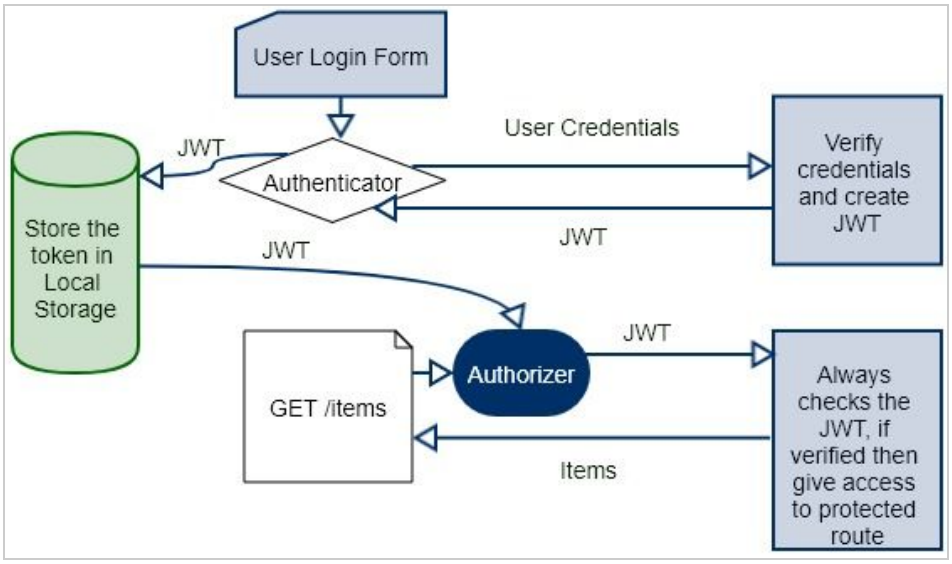
2.2.5.2.5 AJAX(Asynchronous JavaScript and XML)

Για την ασύγχρονη επικοινωνία του client και του server χρησιμοποιούμε την τεχνική AJAX (Asynchronous JavaScript and XML)[24]. Με αυτόν τον τρόπο μπορούμε να ζητήσουμε δεδομένα ή/και να ανανεώσουμε την διεπαφή χρήστη χωρίς την εκ νέου φόρτωσης της σελίδας. Έτσι επιτυγχάνουμε διαμοιρασμό της λογικής εκτέλεσης μεταξύ πελάτη – εξυπηρετητή, αφού μετατοπίζεται ο έλεγχος της ροής στον client.

2.2.5.2.6 Επαλήθευση Ταυτότητας και Εξουσιοδότηση με JSON Web Tokens

Για την επαλήθευση και την ταυτοποίηση του χρήστη την και εξουσιοδότηση των αιτημάτων του προς τον εξυπηρετητή χρησιμοποιούμε cookies. Όμως αντί να αποθηκεύουμε αποκρυπτογράφιτα το

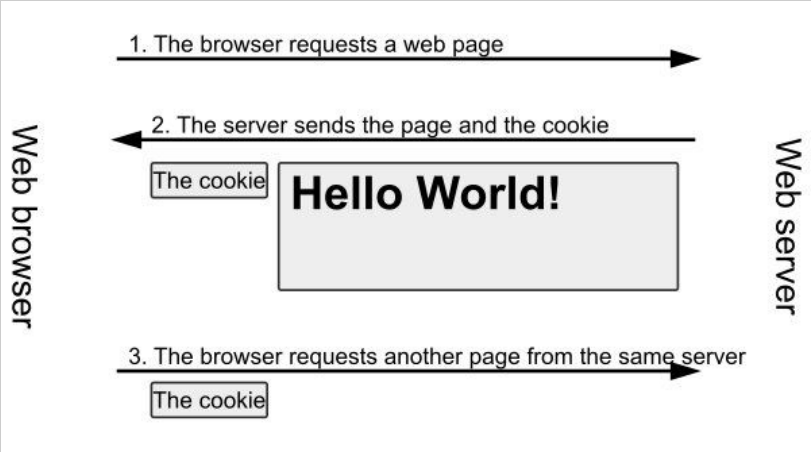
email και των κωδικό πρόσβασης του χρήστη στον browser του, στο σύστημα που αναπτύξαμε, χρησιμοποιούμε JSON Web Tokens(JWT)[25]. Ο χρήστης μετα την αρχική πιστοποίηση του στο σύστημα με email και κωδικό πρόσβασης, ο εξυπηρετητής δημιουργεί ένα μοναδικό αναγνωριστικό για τον χρήστη το οποίο και του αποστέλλει με την μορφή cookie την καταργώντας την ανάγκη για χρήση στοιχείων του χρήστη από αυτό το σημείο και μετά. Το αναγνωριστικό αυτό αποστέλλεται αυτοματοποιημένα κάθε φορά απο τον browser του χρήστη και πιστοποιεί την ταυτότητα του. Για την δημιουργία του πιστοποιητικού, ο server δημιουργεί ένα αναγνωριστικό του χρήστη με βάση κάποιο μοναδικό του χαρακτηριστικό, στην υλοποίηση μας χρησιμοποιούμε ένα id μοναδικό για κάθε χρήστη. Στην συνέχεια προσθέτονται λεπτομέρειες για την χρήση του JWT, όπως διάρκεια ζωης του, συνάρτηση κατακερματισμού και άλλα. Τέλος ο εξυπηρετητής υπογράφει με το ιδιωτικό του κλειδί το συνολικό αναγνωριστικό και το αποστέλλει στον χρήστη για την χρήση του.



Σχήμα 11: Γραφική Αναπαράσταση Επαλήθευσης και την Ταυτοποίησης Αιτημάτων με JSON Web Tokens

2.2.5.2.7 Cookies

Τα cookies αποτελούν μικρά αρχεία κειμένου σε κωδικοποίηση JSON τα οποία αποθηκεύονται στον browser του χρήστη κατά την πλοήγησή του στο διαδίκτυο. Ο browser του χρήστη είναι υπεύθυνος για την αναλλοίωτη αποστολή τους πίσω στον server κάθε φορά που τον επισκέπτεται[26]. Τα cookies μπορούν να βελτιώσουν την εμπειρία περιήγησης επιτρέποντας στις τοποθεσίες να απομνημονεύουν τις προτιμήσεις του χρήστη. Συνήθως, όπως και στην περίπτωση μας, αποθηκεύουν στοιχεία και πληροφορίες για την ταυτοποίηση και εξουσιοδότηση χρήστη. Συγκεκριμένα στην περίπτωση του συστήματος μας, ο server αποθηκεύει ένα cookie στον browser του χρήστη, που περιέχει το JWT για την ταυτοποίηση του χρήστη την και εξουσιοδότηση των αιτημάτων του προς τον εξυπηρετητή.



Σχήμα 12: Γραφική Αναπαράσταση Αλληλεπίδρασης με Cookie

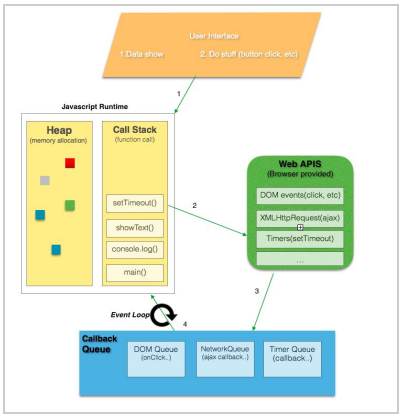
2.2.5.3 JavaScript

Η JavaScript[27] αποτελεί μια ελαφριά αντικειμενοστρεφή και interpreted γλώσσα προγραμματισμού. Βασίζεται πάνω σε πρωτότυπα (prototype-based), είναι πολυ-παραδειγματική και υποστηρίζει αντικειμενοστραφή, προστακτικό και συναρτησιακό προγραμματισμό. Επιπροσθέτως, είναι μονονηματική (single-threaded) και εκτελεί δυναμικό συμπερασμό τύπου μεταβλητών. Στα πλαίσια της παρούσας εργασίας, χρησιμοποιήσαμε την JavaScript, στον διαδικτυακό εξυπηρετητή, στον web-Client αλλά και στην εφαρμογή συλλογής και προσωρινής αποθήκευσης δεδομένων. Η εκτέλεση σε διαφορετικά επεξεργαστικά περιβάλλοντα είναι δυνατή με την χρήση μηχανών JavaScript που βρίσκονται στον πυρήνα τους.

2.2.4.3.1 Ο Βρόχος Συμβάντων της JavaScript(JavaScript Event Loop)

Όπως αναφέραμε, η JavaScript έχει μονονηματική(single-threaded) εκτέλεση, έτσι σχεδόν όλες οι επεξεργαστικά απαιτητικές εντολές εκτελούνται ασύγχρονα, χωρίς να μπλοκάρουν την εκτέλεση του κεντρικού προγράμματος(non-blocking execution). Σε αυτές περιλαμβάνονται οι εγγραφές και αναγνώσεις από το σκληρό δίσκο, οι ενέργειες πάνω σε δεδομένα βάσεων δεδομένων καθώς και οι HTTP αιτήσεις. Όταν το κεντρικό και μοναδικό νήμα εκτέλεσης ζητά από το περιβάλλον εκτέλεσης να πραγματοποιήσει μια απαιτητική ενέργεια, του παρέχει μια συνάρτηση callback και συνεχίζει με την εκτέλεση άλλων εργασιών. Όταν η απαιτητικές ενέργεια που ζητήθηκε ολοκληρωθεί, ένα μήνυμα εισάγεται σε μια ουρά προτεραιότητας μαζί με το παρεχόμενο callback. Όταν αυτό το μήνυμα αφαιρεθεί από την ουρά από τον επεξεργαστή και θα εκτελεστεί η callback συνάρτηση. Αυτό το διαδραστικό[29], πλήρως ασύγχρονο[28], μοντέλο είναι όπως γίνεται αντιληπτό ένα από τα σημαντικότερα πλεονεκτήματα της γλώσσας αλλά διαφέρει σημαντικά από το σύγχρονο μοντέλο αίτησης-απόκρισης που συναντάται σε τυπικές υλοποιήσεις εφαρμογών εξυπηρετητή. Η ουρά προτεραιότητας στην οποία τα μηνύματα αποθηκεύονται προσωρινά μαζί με τα αντίστοιχα callbacks, ονομάζεται βρόχος συμβάντων. Η απεμπλοκή αυτή του καλούντος από την απάντηση που αυτός

αναμένει, επιτρέπει στο περιβάλλον εκτέλεσης της JavaScript να ασχοληθεί με άλλες διεργασίες ενώ «περιμένει» την ασύγχρονη ενέργεια να διεκπεραιωθεί.



Σχήμα 13: Οντολογική Αναπαράσταση του Βρόχου Συμβάντων της JavaScript

2.2.4.3.2 Αντικειμενα JSON

Για την μεταφορά δεδομένων από και προς τον εξυπηρετητή αλλά και δια-διεργασιακά σε περιβάλλον JavaScript, χρησιμοποιούμε την κωδικοποίηση JSON(JavaScript Object Notation) [30][31]. Αποτελεί μια απλή μορφή αναπαράστασης δεδομένων βασισμένη στην αναπαράσταση δεδομένων ως ζεύγη ιδοτήτων-τιμών και λόγω της ευρείας χρήσης και απλότητας του, τείνει να αντικαταστήσει τη γλώσσα σήμανσης XML. Σε κάθε τέτοιο ζεύγος, η ιδιότητα είναι πάντα τύπου String, ενώ η τιμή ανήκει σε έναν από τους υποστηριζόμενους τύπους δεδομένων που υποστηρίζει το JSON, οι οποίοι είναι Integers, Strings, Boolean, Arrays, Objects, Null.

2.2.5.4 Αποθήκευση Δεδομένων

Σε αυτή την ενότητα, παρουσιάσουμε τα αποθηκευτικά εργαλεία και περιβάλλοντα που χρησιμοποιήθηκαν στο σύστημα.

2.2.5.4.1 Open mHealth Schema

Το Open mHealth[32], αποτελεί έναν οργανισμό ο οποίος έχει αναπτύξει εργαλεία[33] για την συλλογή, αναπαράσταση, επεξεργασία και οπτικοποίηση ιατρικών δεδομένων και δεδομένων υγείας χρηστών. Στα πλαίσια του συστήματος μας έγινε προσπάθεια ώστε η αναπαράσταση των συλλεχθέντων δεδομένων στο σύστημα να γινει με ομοιόμορφο και συνεπή τρόπο ως πρός το ορισμένο πρότυπο του οργανισμού. Αυτό έγινε ώστε η ανταλλαγή δεδομένων με άλλες αντίστοιχες εφαρμογές να είναι απλή αλλά και όποια μελλοντική επέκταση του συστήματος να είναι εύκολη.

2.2.5.4.2 PostgreSQL

Η ανοιχτού κώδικα βάση δεδομένων PostgreSQL[34] υλοποιεί το σχεσιακό μοντέλο αποθήκευσης. Αποτελεί ένα ισχυρό σύστημα το οποίο χρησιμοποιείται ευρέως στην βιομηχανία, παρέχοντας αξιοπιστία, ακεραιότητα και ορθότητα στα δεδομένα. Η αποθήκευση των δεδομένων γίνεται σε πίνακες όπως επιτάσει η σχεσιακή μοντελοποίηση. Στα κελιά των πινάκων δομούνται και αποθηκεύονται τα δεδομένα, τα οποία μπορούν να έχουν τύπους: INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, TIMESTAMP ακόμη και δυαδικών αντικειμένων όπως εικόνας, ήχου ή βίντεο.

2.2.5.4.3 MongoDB

Η βάση δεδομένων MongoDB[35] υλοποιεί ένα μη-σχεσιακό μοντέλο αποθήκευσης. Αποτελεί μια εγγραφο-στρεφή ή document-oriented βάση δεδομένων η οποία αποθηκεύει τα δεδομένα της σε έγγραφα με δομές JSON και όχι σε μορφή πινάκων όπως η SQL. Η δομή JSON περιγράφεται από τα σχήματα (schemas) που ορίζει ο χρήστης και δεδομένα αυτής της μορφής καλούνται ημι-δομημένα (semi-structured) δεδομένα. Στην υλοποίηση μας, επιλέχθηκε η μορφή αποθήκευσης των αντικειμένων να είναι BSON(Binary JSON) η οποία υποστηρίζει επιπλέον τύπους δεδομένων και είναι σχεδιασμένη για αποδοτικότερες λειτουργίες σάρωσης και αποθήκευσης.

2.2.5.4.4 IndexedDB

Η βάση δεδομένων IndexedDB[36] υλοποιεί ένα μη-σχεσιακό μοντέλο αποθήκευσης στο web-περιβάλλον, όπως ενός browser. Αποτελεί μια αντικειμενο-στρεφή βάση δεδομένων η οποία αποθηκεύει τα δεδομένα σε δομές JSON. Είναι μέρος της HTML5 του Web Storage API, ενώ παρέχει σύγχρονες και ασύγχρονες λειτουργίες στην JavaScript.

2.2.5.4.5 Αντικειμενο-σχεσιακή Απεικόνιση - Object-Relational Mapping

Η αντικειμενο-σχεσιακή απεικόνιση (Object-Relational Mapping ή ORM)[37] αποτελεί έναν αυτοματοποιημένο τρόπο διασύνδεσης του μοντέλου αντικειμένων (object model) της αντικειμενοστραφούς εφαρμογής μας με την βάση δεδομένων, χρησιμοποιώντας μετά-δεδομένα (metadata) για την περιγραφή του τρόπου της διασύνδεσης τους. Στα πλαίσια του συστηματος μας οι database object-relational mappers προσφέρουν εύκολη αποθήκευση ολόκληρων συνδεδεμένων αντικειμένων στις βάσεις δεδομένων της εφαρμογής, διαμορφώνοντας αυτόματα τα κατάλληλα ερωτήματα προς τη βάση για την εκτέλεση της κάθε λειτουργίας. Με αυτόν τον τρόπο, παρέχεται εκτεταμένη συνδεσιμότητα, ασφάλεια και βελτιστοποίηση του αποθηκευτικού συστήματος μας.

2.2.5.5 Πρωτόκολλο HTTPS

Οποιαδήποτε απομακρυσμένη διαδικτυακή επικοινωνία επιτελείται στο σύστημα χρησιμοποιεί το πρωτόκολλο HTTPS(Hypertext Transfer Protocol Secure)[38]. Παρότι δεν αποτελεί ξεχωριστό πρωτόκολλο, η ασφαλής δικτυακής σύνδεση που παρέχει, οφείλεται στον συνδυασμό του απλού

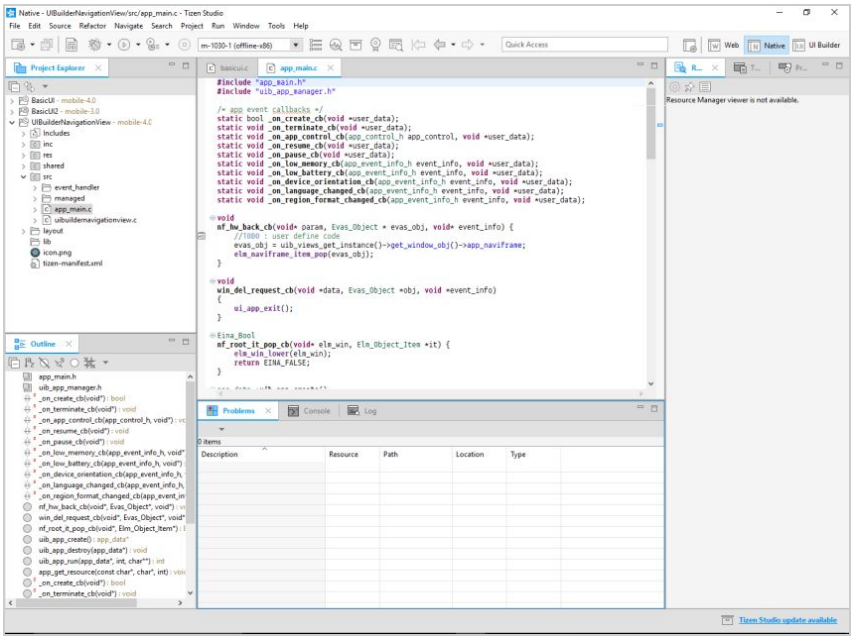
HTTP πρωτοκόλλου και των δυνατοτήτων κρυπτογράφησης που παρέχει το πρωτόκολλο SSL(Secure Sockets Layer). Με την χρήση λοιπόν κρυπτογραφίας δημοσίου κλειδιού, τα δεδομένα ανταλλάσσονται κρυπτογραφημένα και δεν μπορούν να υποκλαπούν από άλλους κακόβουλους χρήστες ή από επιθέσεις man-in-the-middle(MitM).

2.2.6 Περιβάλλον Ανάπτυξης

Τέλος, θα κάνουμε παρουσιάσουμε τα εργαλεία και τα προγραμματιστικά περιβάλλοντα που χρησιμοποιήθηκαν κατά την ανάπτυξη του συστήματος.

2.2.6.1 Tizen Studio

Για την ανάπτυξη της εφαρμογής συλλογής και αποθήκευσης δεδομένων στην συσκευή smartwatch Samsung Gear S3 Frontier, χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) Tizen Studio[39]. Αποτελεί προϊόν της εταιρείας Samsung και αποτελεί το κύριο περιβάλλον ανάπτυξης native και web applications για το λειτουργικό σύστημα TizenOS. Περιλαμβάνει λειτουργίες επεξεργαστή κειμένου, αποσφαλατωτή, προσομοιωτή συσκευών καθώς και εργαλεία για την εκτέλεση εφαρμογών σε φυσικές συσκευές.



Σχήμα 14: Γραφική Διεπαφή του Tizen Studio

2.2.6.2 WebStorm

Για την ανάπτυξη της διαδικτυακής υπηρεσίας και την υλοποίηση των REST APIs, χρησιμοποιήσαμε το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) WebStorm[40]. Αποτελεί

προϊόν της εταιρείας JetBrains και μας παρέχει ένα εύχρηστο περιβάλλον που βελτιώνει την οργάνωση της εργασίας και επιταχύνει την ανάπτυξή. Προσφέρει αυξημένων δυνατοτήτων επεξεργαστή κώδικα(code editor) και αποσφαλματωτή.

2.2.6.3 Postman

Για την δοκιμή και αξιολόγηση της λειτουργικότητας των REST APIs που αναπτύχθηκαν, χρησιμοποιήσαμε το εργαλείο Postman[41]. Το πρόγραμμα υλοποιεί έναν (REST Client), ο οποίος έχει την δυνατότητα να στέλνει αιτήματα από το γραφικό του περιβάλλον και να λαμβάνει απαντήσεις από έναν REST Server. Με αυτόν τον τρόπο επαληθεύουμε αποτελεσματικά την ορθή λειτουργία APIs και να εντοπίζουμε τυχόν σφάλματα. Στα επόμενα κεφάλαια θα παρουσιάσουμε πολλά αιτήματα και τις απαντήσεις που λάβαμε μέσω του εργαλείου Postman απο τον server.

2.2.6.4 Σύστημα ελέγχου εκδόσεων Git και Bitbucket

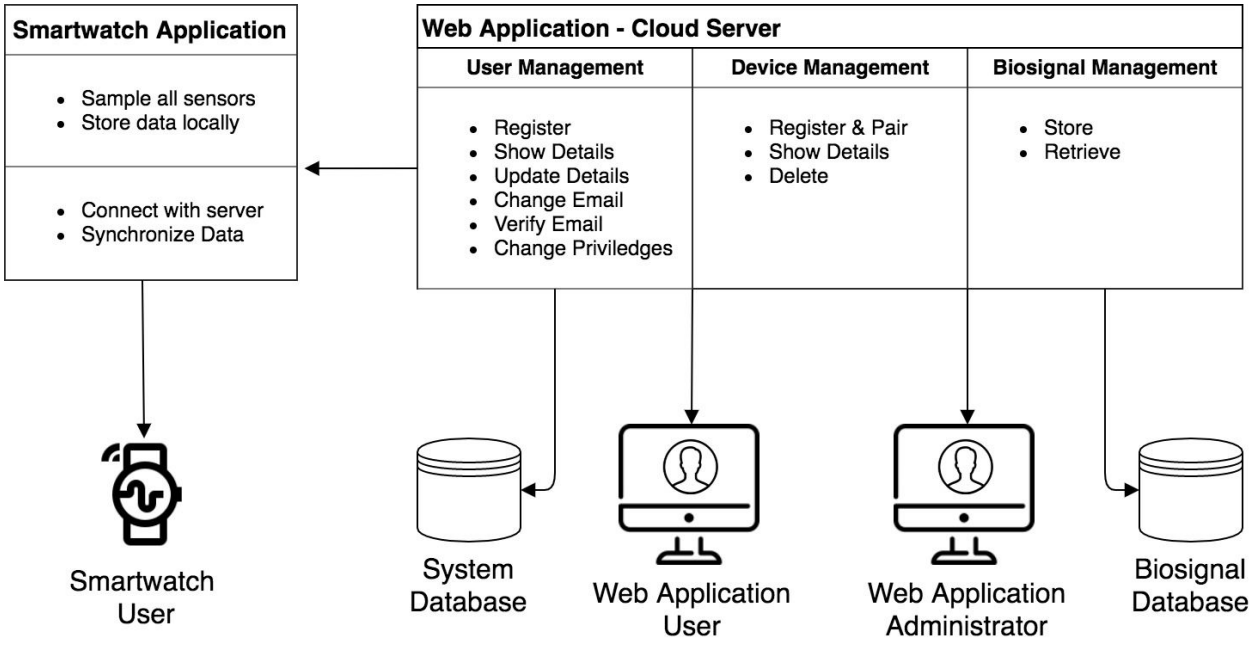
Η υλοποίηση του πηγαίου κώδικα των δύο εφαρμογών της υπηρεσίας μας πραγματοποιήθηκε στα περιβάλλοντα ανάπτυξης Tizen Studio και WebStorm αντίστοιχα όπως παρουσιάσαμε. Η διαχείριση όμως των εκδόσεων και η δημιουργία αντιγράφων ασφαλείας του παραγόμενου κώδικα έγινε χρησιμοποιώντας το ελεύθερο λογισμικό Git[42]. Χρησιμοποιήσαμε την τοπική εκτέλεση του Git ώστε να αποθηκεύουμε κάθε φορά την τρέχουσα κατάσταση του κώδικα, να δημιουργούμε νέες εκδόσεις και λειτουργίες με ιεραρχικό τρόπο διαχείρισης. Κάθε φορά, που αποθηκεύεται ένα στιγμιότυπο του κώδικα, το λογισμικό Git καταγράφει την κατάσταση των αρχείων εκείνη τη στιγμή, και αποθηκεύει μια αναφορά του στιγμιότυπου στην μνήμη τοπικά. Για την ασφαλή όμως δημιουργία αντιγράφων ασφαλείας και την προώθηση του κώδικα στον διαδικτυακό εξυπηρετητή χρησιμοποιήσαμε την υπηρεσία Bitbucket[43]. Μια διαδικτυακή υπηρεσία η οποία φιλοξενεί τα αποθετήρια κώδικα τύπου Git που δημιουργούμε, κάνοντας έτσι την διαχείριση κώδικα αποκεντρωμένη, δομημένη και ασφαλή.

2.2.6.5 pgAdmin

Για την διαχείριση των εγγράφων της σχεσιακής βάσης δεδομένων σε γραφικό περιβάλλον χρησιμοποιήσαμε την εφαρμογή ανοικτού κώδικα pgAdmin[44]. Η γραφική διεπαφή που μας παρέχεται από την υπηρεσία μας επιτρέπει: να συνδεθούμε απομακρυσμένα με τον εξυπηρετητή της βάσης δεδομένων, να εκτελέσουμε αιτήματα σε γλώσσα SQL και να προβάλουμε τα αποτελέσματα τους σε μορφή πινάκων. Τέλος, μας παρέχεται η δυνατότητα για εξαγωγή των αποτελεσμάτων των ερωτημάτων σε μορφές SQL,XML,XHTML,CSV και pg_dump για περαιτέρω επεξεργασία τοπικά.

3 Ανάλυση Προδιαγραφών και Σχεδίαση Συστήματος

Σε αυτό το κεφάλαιο, θα παρουσιάσουμε αναλυτικά τις προδιαγραφές λειτουργίας και τις απαιτήσεις συστήματος. Παράλληλα, θα μοντελοποιηθούν οι ανάγκες, τα χαρακτηριστικά των χρηστών και τα σενάρια χρήσης της εφαρμογης. Στην συνέχεια θα γίνει ανάλυση της σχεδίασης και της αρχιτεκτονικής του συστήματος. Τέλος, θα δοθεί βαρύτητα στην περιγραφή των επιμέρους συστημάτων καθώς επίσης στην σύνδεση και την μεταξύ τους επικοινωνία.



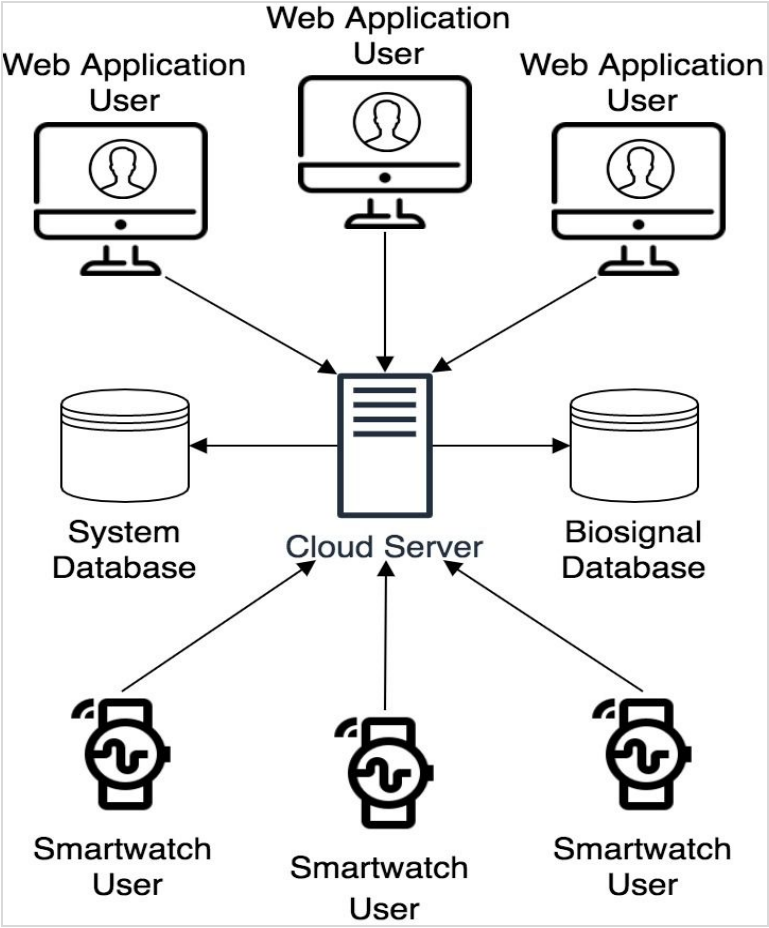
Σχήμα 15: Το Προτεινόμενο Σύστημα

3.0 Γενική Περιγραφή Συστήματος

Σκοπός του συστήματος είναι να συλλέξουμε τοπικά στην smartwatch συσκευή τα δεδομένα ενδιαφέροντος και στην συνέχεια να τα μεταφέρουμε αυτοματοποιημένα στο κεντρικό σύστημα ώστε να γίνει η μόνιμη αποθήκευση και επεξεργασία τους.

Για τον λόγο αυτό, το σύστημα αποτελείται από δύο εφαρμογές που λειτουργούν και συνεργάζονται συμπληρωματικά. Συγκεκριμένα στην smartwatch συσκευή του χρήστη, συλλέγουμε αυτοματοποιημένα τα βιοσήματα απο τους αισθητήρες και αφού τα αποθηκεύσουμε προσωρινά μέχρι να βρεθεί σύνδεση δικτύου, τα αποστέλλουμε στον κεντρικό εξυπηρετητή της εφαρμογής στο cloud. Συμπληρωματικά, η κεντρική διαδικτυακή εφαρμογή συλλέγει τα δεδομένα ασύγχρονα από όλες τις συσκευές των χρηστών και είναι υπεύθυνη για την ασφαλή διαχείριση των χρηστών, των συσκευών και των δεδομένων τους. Οι χρήστες μπορούν να δημιουργήσουν τον προσωπικό τους λογαριασμό, να δηλώσουν τις συσκευές τους αλλά και να

προβάλουν το ιστορικό των μετρήσεων τους στο περιβάλλον του browser, που εξυπηρετείται από τον κεντρικό server.



Σχήμα 16: Η Τοπολογία του Συστήματος

Η δομή του συστήματος είναι στηριγμένη γύρω από την τοπολογία ενός κεντρικού εξυπηρετητή (Central Server Based Application), ο οποίος αναλαμβάνει τον κεντρικό ρόλο για την επιτυχή λειτουργία συνολικά της πλατφόρμας. Παρ'όλα αυτά η αδιάκοπη συλλογή δεδομένων και η ασφάλεια αυτών είναι ο πυρήνας λειτουργίας του συστήματος. Κατ'αυτόν τον τρόπο η λειτουργικότητα της εφαρμογής που εκτελείται επι της smartwatch συσκευής, έχει σχεδιαστεί ώστε να είναι ανεξάρτητη από την λειτουργία του κεντρικού server για δύο λόγους:

1. Η συλλογή, προ-επεξεργασία και προσωρινή αποθήκευση δεδομένων πρέπει να είναι δυνατή ακόμη και αν η συσκευή δεν βρίσκεται εντός κάλυψης δικτύου. Ενώ, πρέπει να εκτελείται ακόμη και σε περίπτωση που το κεντρικό σύστημα δεν είναι διαθέσιμο προσωρινά ή μη αποκρίσιμο.
2. Μεταφέροντας ολόκληρο το επεξεργαστικό φορτίο που σχετίζεται με την συλλογή δεδομένων επί της smartwatch συσκευής, μειώνεται δραστικά το ποσοστό των

διεργασιών που εκτελούνται επί του κεντρικού server, επιτυγχάνοντας έτσι πιο γρήγορη απόκριση του συστήματος και μείωση της επικοινωνιακής επιβάρυνσης στο ελάχιστο δυνατό. Τέλος, η web εφαρμογή που εκτελείται στον σελιδομετρητή δικτύου(web browser) των χρηστών ως single-page-application, μεταφέρει το επεξεργαστικό φορτίο στο τοπικό μηχάνημα του χρήστη επιτυγχάνοντας αντίστοιχη βελτίωση στον server.

3.1 Ανάλυση Απαιτήσεων και Προδιαγραφών

Για την ομαλή χρήση του συστήματος θα πρέπει να εκπληρώνονται μια σειρά από λειτουργικές και μη απαιτήσεις, ώστε να διασφαλίζεται η ορθή λειτουργία και κάλυψη των αναγκών για τις οποίες σχεδιάστηκε εξαρχής. Επιπλέον, θα αναφερθούμε στις πιο σημαντικές απαιτήσεις και σενάρια χρήσης οι οποίες θα καλυφθούν από την υλοποίηση στο επόμενο κεφάλαιο.

3.1.1 Οι Χρήστες του συστήματος και τα χαρακτηριστικά τους

Το σύστημα σχεδιάστηκε, έχοντας τις ακόλουθες παραδοχές για τους τελικούς χρήστες:

- Οι χρήστες ενδιαφέρονται για την καταγραφή των δεδομένων τους που προκύπτουν απο τους αισθητήρες της smartwatch συσκευής.
- Οι χρήστες πιθανόν να κατέχουν περισσότερες από μία smartwatch συσκευές.
- Οι χρήστες ενδιαφέρονται να δουν ενοποιημένα τα δεδομένα που προκύπτουν από αυτές τις συσκευές.
- Ερευνητές και αναλυτές ενδιαφέρονται να αναλύσουν περαιτέρω αυτά τα δεδομένα, με στόχο την εξαγωγή ευρύτερων συμπερασμάτων σχετικά με τους χρήστες της πλατφόρμας.

Τα άνωθι, μοντελοποίηθηκαν στο σύστημα σε δύο διαφορετικά είδη χρηστών. Η πρώτη κατηγορία είναι οι απλοί χρήστες, οι οποίοι χρησιμοποιούν την πλατφόρμα με σκοπό να συλλέγουν αυτοματοποιημένα και να διαχειρίζονται τα δεδομένα τους. Η δεύτερη κατηγορία είναι οι διαχειριστές της πλατφόρμας, οι οποίοι χρησιμοποιούν το σύστημα ώστε να διασφαλίσουν την ομαλή του λειτουργία. Παράλληλα, οι διαχειριστές έχουν πρόσβαση στα συνολικά δεδομένα των χρηστών με στόχο την περαιτέρω ερευνητική επεξεργασία τους, με ασφαλή και ανώνυμο τρόπο.

Οι απλοί χρήστες: Αποτελούν τον κορμό των απαιτήσεων που καλείται να καλύψει το σύστημα και είναι τα φυσικά πρόσωπα στα οποία παρέχονται δύο είδη λειτουργιών. Πρώτον, εγγράφονται στο διαδικτυακό σύστημα της εφαρμογής, προκειμένου να είναι σε θέση να προβάλουν και να διαχειριστούν το ιστορικό των μετρήσεων τους. Δεύτερον είναι σε θέση μετά την εγγραφή τους να δηλώσουν τις συσκευές που θα συλλέγουν τα δεδομένα τους, αλλά και να τις χρησιμοποιήσουν αυτοτελώς.

Οι διαχειριστές συστήματος: Αποτελούν τα φυσικά πρόσωπα τα οποία απαιτείται να εγγραφούν στο διαδικτυακό σύστημα της εφαρμογής και να εγκριθεί ο ρόλος τους απο τους ήδη υπάρχοντες διαχειριστές. Έχουν πρόσβαση στα δεδομένα του συστήματος με σκοπό την επεξεργασία τους για ερευνητικούς σκοπούς.

3.1.2 Λειτουργικές Απαιτήσεις Συστήματος

Το μέρος της εφαρμογής που αφορά τους απλούς χρήστες, θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:

- Την εγγραφή νέου χρήστη στο σύστημα και τη δημιουργία προσωπικού λογαριασμού.
- Την σύνδεση υπάρχοντος χρήστη στο σύστημα.
- Την διασφάλιση του απορρήτου του λογαριασμού, ώστε να έχει μόνο αυτός πρόσβαση.
- Την επιτυχή σύνδεση του σε παραπάνω από ενά μηχανήματα καθώς και την ομαλή διαχείριση των συνδέσεων αυτών.
- Την αποσύνδεση υπάρχοντος προφίλ από το σύστημα.
- Την οριστική διαγραφή του προσωπικού προφίλ και των σχετιζόμενων με αυτό δεδομένων, σε περίπτωση που παύσει η χρήση του μόνιμα.
- Την πρόσβαση του χρήστη σε μια εύχρηστη και αισθητικά ευχάριστη διεπαφή, μέσω της οποίας να υποστηρίζονται όλα τα λειτουργικά σενάρια.
- Την ενημέρωση του χρήστη για την κατάσταση λειτουργίας των χρησιμοποιούμενων υπηρεσιών.

Το μέρος της εφαρμογής που αφορά την διαχείριση των δεδομένων κίνησης και βιοσημάτων των χρηστών της εφαρμογής θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:

- Την αυτοματοποιημένη συλλογή δεδομένων κίνησης και βιοσημάτων απο την επιλεγμένη συσκευή του χρήστη.
- Την παρακολούθηση στατιστικών σχετικά με τα ληφθέντα δεδομένα, επί της συσκευής συλλογής.
- Την αποστολή με αυτοματοποιημένο τρόπο των μετρήσεων όταν βρεθεί η συσκευή εντός δικτύου.
- Την αποστολή με μη-αυτοματοποιημένο τρόπο των μετρήσεων όταν ο χρήστης το επιθυμεί και βρίσκεται εντός δικτύου.

Το μέρος της εφαρμογής που αφορά την διαχείριση των συσκευών των χρηστών της εφαρμογής θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:

- Την σύνδεση και διαχείριση συσκευών συλλογής δεδομένων κίνησης και βιοσημάτων, όπως του Samsung Gear S3 Frontier smartwatch με την διαδικτυακή διεπαφή.
- Την παρακολούθηση του ιστορικού των ληφθέντων δεδομένων από αυτές τις συνδεδεμένες συσκευές του χρήστη σε γραφικό περιβάλλον. Η προβολή αυτή περιλαμβάνει όλες τις μετρήσεις ή μηχανισμούς αναζήτησης και φιλτραρίσματος

βασισμένο στον τύπο του αισθητήρα που χρησιμοποιηθηκε ή σε συγκεκριμένο χρονικό παράθυρο.
- Την ζευγοποίηση μιας συσκευής smartwatch με το προφίλ του χρήστη, προτού μεταδοθεί οποιοδήποτε δεδομένο στο κεντρικό σύστημα.
- Την αποζευγοποίηση μιας συσκευής smartwatch από το προφίλ του χρήστη, χωρίς την απώλεια δεδομένων και περιορισμό της λειτουργικότητας της εφαρμογής επι της συσκευής.

Το μέρος της εφαρμογής που αφορά τους διαχειριστές συστήματος, θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:

- Την παρακολούθηση του ιστορικού των ληφθέντων δεδομένων που αφορούν το σύνολο των χρηστών σε γραφικό περιβάλλον. Η διεπαφή αυτή περιλαμβάνει όλες τις αποθηκευμένες μετρήσεις και μηχανισμούς αναζήτησης και φιλτραρίσματος με κριτήριο τον τύπο του αισθητήρα που χρησιμοποιηθηκε ή συγκεκριμένο χρονικό παράθυρο.
- Την εποπτεία όλων των συνδεδεμένων συσκευών στο σύστημα που αφορούν το σύνολο των χρηστών σε γραφικό περιβάλλον.
- Την προβολή και αλλαγή δικαιωμάτων των χρηστών επι της εφαρμογής, μέσω κατάλληλων γραφικού περιβάλλοντος. Η αλλαγή δικαιωμάτων περιλαμβάνει, επιβεβαίωση, μπλοκάρισμα, διαγραφή και αναβάθμιση απλού χρήστη σε διαχειριστή συστήματος και αντίστροφα υποβάθμιση του.

3.1.3 Μη Λειτουργικές Απαιτήσεις Συστήματος

Ευχρηστία Συστήματος: Το σύστημα έχει αναπτυχθεί δίνοντας έμφαση σε εύχρηστες γραφικές διεπαφές και με απλή σχεδίαση, ώστε να μπορούν να χρησιμοποιηθούν από πολλές κοινωνικές ομάδες με ευκολία. Η ιεραρχική οργάνωση και λίγα κουμπιά στα γραφικά περιβάλλοντα συντελούν ώστε το σύστημα να μπορεί να χρησιμοποιηθεί από χρήστες δίχως προηγούμενη εμπειρία και να ελαχιστοποιήσει τυχόν λάθη τους. Ακολουθώντας τα σχετικά σχεδιαστικά πρότυπα, αποτρέπονται πολλαπλές αιτήσεις πρίν ληφθεί απάντηση από τον εξυπηρετητή, ενώ προβάλλεται ειδική ειδοποίηση πριν επιβληθεί κάποια απόφαση στο σύστημα κ.α.

Διαλειτουργικότητα Συστήματος: Όπως έχει ήδη αναφερθεί, στην παρούσα διπλωματική εργασία, χρησιμοποιούμε ως proof-of-concept το περιβάλλον του smartwatch Samsung Gear S3 Frontier για την εκτέλεση της εφαρμογής συλλογής και πρόχειρης αποθήκευσης δεδομένων. Εν τούτοις, μέσω της σχεδίασης και υλοποίησης της υπηρεσίας του κεντρικού εξυπηρετητή, ορίστηκαν πλήρως όλες οι διεπαφές επικοινωνίας ακολουθώντας όλα τα σχετικά τεχνολογικά πρότυπα. Ως αποτέλεσμα, οποιαδήποτε εφαρμογή ‘‘έξυπνης¨ συσκευής που υλοποιεί τα πρωτόκολλα και διεπαφές επικοινωνίας που ορίσαμε, μπορεί να χρησιμοποιηθεί επιτυχώς από τον κεντρικό εξυπηρετητή.

Διαθεσιμότητα και Ταχύτητα Συστήματος: Ο κεντρικός διακομιστής της πλατφόρμας φιλοξενείται σε υποδομή cloud, η οποία μας δίνει την δυνατότητα να δημιουργήσουμε κατ' απαίτηση νέα στιγμιότυπα των εκτελούμενων διεργασιών, σε περίπτωση αυξημένου φόρτου ή σφάλματος στην υπάρχουσα υποδομή. Με αυτόν τον τρόπο διασφαλίζουμε την απρόσκοπτη λειτουργία του συστηματος και συνεχή διαθεσιμότητα των δεδομένων του. Επιπλέον, οι δύο εφαρμογές που υλοποιήθηκαν ακολουθώντας το μοντέλο της ασύγχρονης ενημέρωσης των πόρων τους, εξαλείφουν την ανάγκη για προσεγγίσεις long-polling(ερωτήματα ανα τακτά χρονικά διαστήματα προς κάποιον πόρο) και καθιστούν το σύστημα αποδοτικότερο.

Στιβαρότητα και Αξιοπιστία Συστήματος: Η ιεραρχική δομή της αρχιτεκτονικής του συστήματος, μαζί με δόμες που αποτρέπουν την αποτυχία κάποιας λειτουργίας του συστήματος διασφαλίζουν σε πολύ μεγάλο βαθμό την αδιάλειπτη λειτουργία του. Για παράδειγμα, όπως έχει ήδη αναφερθεί, η εφαρμογή συλλογής και πρόχειρης αποθήκευσης δεδομένων λειτουργεί αυτοτελώς έως ότου βρεθεί διαθέσιμη διαδικτυακή σύνδεση με την κεντρική υπηρεσία. Σε περίπτωση που κάτι τέτοιο δεν συμβεί τα δεδομένα του χρήστη δεν βρίσκονται σε κίνδυνο. Παράλληλα, όπως θα αναλύσουμε στην επόμενη ενότητα, οι βάσεις δεδομένων που χρησιμοποιεί ο κεντρικός διακομιστής βρίσκονται κάτω από εντελώς διαφορετική υποδομή.

Ασφάλεια Συστήματος και Προστασία Προσωπικών Δεδομένων: Κατα την υλοποίηση του συστήματος συνολικά, επιλέχθηκαν όλα τα ερωτήματα που θα κατευθυνόντουσαν απο και προς το διαδίκτυο να χρησιμοποιούν το πρωτόκολλο επικοινωνίας TLS(Transport Layer Security). Έτσι,όλα τα δεδομένα χρησιμοποιούν κρυπτογραφία δημόσιου κλειδιού προτού προωθηθούν στο διαδίκτυο. Παράλληλα, οι κωδικοί και τα ευαίσθητα δεδομένα των χρηστών κρυπτογραφούνται προτού αποθηκευτούν στις βάσεις δεδομένων του κεντρικού server.

Συντηρησιμότητα και Δυνατότητα Επαναχρησιμοποίησης Συστήματος: Εξαιρετικά μεγάλο βάρος δόθηκε κατα την σχεδίαση και υλοποίηση του συστήματος, έτσι ώστε να μπορεί να επιτευχθεί μέγιστη επαναχρησιμοποίηση της υπηρεσίας από άλλα μέλη της ακαδημαικής ή επιχειρηματικής κοινότητας. Χάρη στην ιεραρχική δομή της αρχιτεκτονικής του συστήματος, των σύγχρονων τεχνολογιών, πρωτοκόλλων και διεπαφών επικοινωνίας που υλοποιήθηκαν, κάθε διακριτό κομμάτι της τοπολογίας μπορεί να επαναχρησιμοποιηθεί ή μεταποιηθεί σε κάτι άλλο εξυπηρετώντας περισσότερες και νέες ανάγκες.

3.2 Αρχιτεκτονική Συστήματος

Όπως έχει ήδη αναφερθεί, η πλατφόρμα αποτελείται από δύο βασικά επιμέρους συστήματα. Τον κεντρικό διακομιστή(Cloud Server) και την εφαρμογή συλλογής και αποθήκευσης δεδομένων χρήστη. Ο κεντρικός διακομιστής εκτελείται σε περιβάλλον υπολογιστικού νέφους(Cloud) και είναι υπεύθυνος για την παροχή τις διαδικτυακής διεπαφής των χρηστών καθώς και την

επικοινωνία με την εφαρμογή συλλογής δεδομένων κάθε χρήστη. Η εφαρμογή συλλογής δεδομένων εκτελείται επι της "έξυπνης" συσκευής-smartwatch Samsung Gear S3 Frontier.

3.2.1 Επικοινωνία και Διασύνδεση Μεταξύ των Εφαρμογών

Στην περίπτωση μας αποφασίστηκε η ίδια αρχιτεκτονική διακομιστή να υλοποιεί την επικοινωνία με τα δύο διαφορετικά μέρη. Το πρώτο, είναι η επικοινωνία μεταξύ της εφαρμογής συλλογής και αποθήκευσης δεδομένων που εκτελείται στο smartwatch του χρήστη προς τον διακομιστή και αντίστροφα. Το δεύτερο, είναι η επικοινωνία μεταξύ της διαδικτυακής διεπαφής που οι χρήστες βλέπουν τον browser με τον διακομιστή και αντίστροφα.

Για τον λόγο αυτό, επιλέξαμε να υλοποιηθεί μια ομοιόμορφη διεπαφή στην πλευρά του διακομιστή και οι δύο διαφορετικοί "πελάτες" διαφοροποιώντας τα ερωτήματα τους να έχουν πρόσβαση και να επιτελούν όλες τις λειτουργίες που απαιτούνται. Έτσι, συγκεντρώνουμε την υλοποίηση σε μία κεντρική μονάδα, παρά σε περισσότερες οι οποίες αφενός θα αύξαναν το χρόνο και το κόστος υλοποίησης του συστήματος, αφετέρου θα καθιστούσαν την υλοποίηση δυσκολότερη λόγω της εγγενούς προστιθέμενης πολυπλοκότητας.

3.2.1.1 REST Αρχιτεκτονική

Για την ασύγχρονη επικοινωνία της εφαρμογής smartwatch αλλά και της διαδικτυακής εφαρμογής με τον κεντρικό διακομιστή, επιλέχθηκε η χρήση της αρχιτεκτονικής REST(REpresentational State Transfer). Όπως έχουμε αναλύσει στην ενότητα **2.2.5.2.4** η αρχιτεκτονική του REST μας επιτρέπει την επικοινωνία μεταξύ πελάτη-διακομιστή να γίνεται με ξεχωριστά και αυτόνομα ερωτήματα, ταυτόχρονα χωρίς να επιβάλλει συγκεκριμένη τεχνολογία υλοποίησης. Αυτό είναι πολύ σημαντικό χαρακτηριστικό αφού οι δύο εφαρμογές εκτελούνται σε εντελώς διαφορετικά περιβάλλοντα εκ σχεδιασμού. Η εφαρμογή του smartwatch εκτελείται στο περιβάλλον της συσκευής, φιλοξενούμενη από το λειτουργικό σύστημα και τις λειτουργίες που προσφέρει. Η διαδικτυακή εφαρμογή εκτελείται στο περιβάλλον του browser σε κάποιον προσωπικό υπολογιστή και παρότι και αυτή φιλοξενείται απο λειτουργικό σύστημα οι δυνατότητες είναι πολύ περισσότερες.

Πρέπει να γίνει σαφές όμως ότι στο μοντέλο επικοινωνίας πελάτη-διακομιστή που χρησιμοποιεί η REST αρχιτεκτονική, όταν ο χρήστης χρησιμοποιεί την διαδικτυακή εφαρμογή τότε αυτή είναι ο πελάτης και ο διακομιστής είναι ο κεντρικός server. Αντίστοιχα, όταν η εφαρμογή συλλογής και αποθήκευσης δεδομένων που εκτελείται στο smartwatch του χρήστη μεταφέρει δεδομένα στον διακομιστή, τότε αυτή η εφαρμογή είναι ο πελάτης, παρότι ο χρήστης παραμένει ο ίδιος και στα δύο σενάρια.

Οι δύο επιμέρους εφαρμογές είναι υπεύθυνες για την τοποθέτηση αναγνωριστικών εντός των αιτημάτων τους προς τον διακομιστή ώστε να τα επεξεργαστεί και να απαντήσει σωστά σ' αυτά. Ο διακομιστής είναι υπεύθυνος για την ακολούθηση των προτύπων επικοινωνίας,

επεξεργασίας που έχουν οριστεί και για την μορφοποίηση των απαντήσεων και δεδομένων ώστε να είναι επεξεργάσιμα από τους παραλήπτες σε μορφή JSON ή html.

3.2.1.2 HTTP Αιτήματα & AJAX

Χρησιμοποιώντας την αρχιτεκτονική REST, μπορούμε να μεταφέρουμε δεδομένα και να κάνουμε ερωτήσεις στον διακομιστή από δύο διαφορετικές αρχιτεκτονικές επιμέρους εφαρμογών, μέσα από διαδικτυακά HTTP αιτήματα όπως αναλύσαμε στην ενότητα **2.2.5.2.3**. Αυτό οφείλεται στο γεγονός ότι, τα αιτήματα αυτά εκτελούνται από το λειτουργικό σύστημα της εκάστοτε συσκευής και είναι όπως αναφέραμε αυτοτελή.

Σε περίπτωση λοιπόν, ενός τυπικού σεναρίου όπου κάποια δεδομένα που ζητήσαμε από τον κεντρικό διακομιστή στο παρελθόν έχουν αλλάξει, δεν έχουμε τρόπο να ειδοποιηθούμε από το πρωτόκολλο γι' αυτήν την αλλαγή. Για τον λόγο αυτό χρησιμοποιούμε ασύγχρονα αιτήματα (Asynchronous JavaScript and Xml -AJAX) εκτελώντας XMLHttpRequests σε διάφορα σημεία της εκτέλεσης των δύο εφαρμογών χρήστη για δύο λόγους. Πρώτον, να βεβαιωθούμε για την εγκυρότητα των δεδομένων προτού εκτελέσουμε κάποια διεργασία επι αυτών. Δεύτερον, για να παρέχουμε μια καλύτερη εμπειρία χρήστη, αυτοματοποιώντας πληθώρα σεναρίων αντί να εκτελούνται χειροκίνητα.

```
POST /biosignals/upload HTTP/1.1
Host: localhost:3000
x-access-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9
    .eyJpZCI6IjVjMDkxZjUzZTE0NGVkYWQwODBkOWI3MiIsImlhdCI6MTU0NDEwMTcxNX0
    .BMqkpyQsItQjngV8x1OBX4dBHu8tMi_nM1kcp-GGp6YBQb11c1Qd4Fzu6R-U0yrJ_m4GDLYK4G1xjmDtCBlZvg
Content-Type: application/json
Cache-Control: no-cache


{
    "type":"accelerometer",
    "sensor_id":"",
    "sensor_body_location":"",
    "annotated": true,
    "format": "array",
    "data":[
        [3045, 3145, 3245, "2037-11-13T00:21:01.173Z","swipe_up"],
        [5045, 5145, 5245, "2037-11-13T00:21:01.174Z","swipe_around"]
    ]
}
```

Σχήμα 17: Παράδειγμα HTTP REST Αιτήματος στον Διακομιστή για Αποθήκευση Βιοσημάτων

3.2.2 Η Εφαρμογή Συλλογής Δεδομένων και οι Υπηρεσίες της

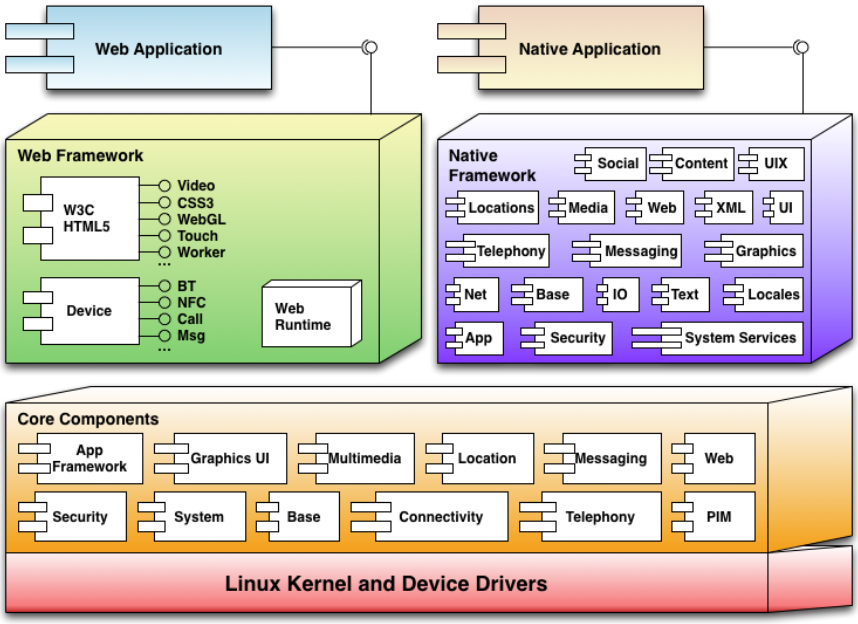
Ο σκοπός της εφαρμογής που εκτελείται στην "έξυπνη" συσκευή-smartwatch που φορά ο χρήστης είναι τριπλός. Πρώτον, να συλλέξει τα δεδομένα κίνησης και βιοσημάτων απο τους αισθητήρες της συσκευής αυτοματοποιημένα και σε προκαθορισμένα χρονικά διαστήματα. Δεύτερον, να αποθηκεύει σωστά και δομημένα όλη την δυνατή πληροφορία με την μεγαλύτερη δυνατή πιστότητα ώστε τα δεδομένα να είναι πλούσια σε πληροφορία. Τρίτον, να συγχρονίζει και να μεταφέρει τα συλλεγόμενα δεδομένα στον κεντρικό διακομιστή της πλατφόρμας αυτοματοποιημένα μέσω δικτύου. Παράλληλα, οι στόχοι αυτοί πρέπει να επιτυγχάνονται με

μέγιστη χρήση των πόρων της συσκευής, ώστε ο χρήστης να συλλέγει δεδομένα για όσο τον δυνατόν μεγαλύτερο χρονικό διάστημα, χωρίς να απαιτείται επαναφόρτιση της συσκευής.

Για να πετύχουμε τους άνωθι στόχους κρίνεται απαραίτητη η κατασκευή μιας εφαρμογής που θα εκτελείται στο native περιβάλλον της "έξυπνης" συσκευής-smartwatch που φορά ο χρήστης. Έχουμε ήδη εξηγήσει, ότι η πλατφόρμα μπορεί να λειτουργήσει με οποιαδήποτε εφαρμογή και συσκευή smartwatch που υλοποιεί την διεπαφή επικοινωνίας με τον κεντρικό εξυπηρετητή. Στη παρούσα διπλωματική εργαία όμως έχουμε επιλέξει πιλοτικά να χρησιμοποιήσουμε την "έξυπνη" συσκευή-smartwatch, Samsung Gear S3 Frontier η οποία θα υλοποιήσει στο περιβάλλον την συσκευής την διεπαφή επικοινωνίας.

Το περιβάλλον εκτέλεσης της συσκευής είναι το λειτουργικό σύστημα TizenOS όπως έχουμε ήδη αναφέρει στην ενότητα **2.2.4**, το οποίο αποτελεί μια ελαφριά και εξειδικευμένη διανομή του λειτουργικού Linux. Στο περιβάλλον αυτό μας δίνεται η δυνατότητα ανάπτυξης με δύο τρόπους:

1. Δημιουργία native εφαρμογής με χρήση του Native API του TizenOS και υλοποίηση σε γλώσσα προγραμματισμού C. Με αυτόν τον τρόπο έχουμε πληρέστερο έλεγχο επι της υλοποίησης αφού ο προγραμματισμός γίνεται στο χαμηλότερο επίπεδο αφαίρεσης που επιτρέπει το λειτουργικό σύστημα. Αυτό συνεπάγεται πέρα από μεγαλύτερο έλεγχο των λειτουργιών, χαμηλότερη απαίτηση ενέργειας και πληρέστερη πρόσβαση στον χώρο μνήμης της συσκευής.
2. Δημιουργία web εφαρμογής με χρήση του Web API του TizenOS και υλοποίηση σε γλώσσα προγραμματισμού JavaScript και χρήση τεχνολογιών web. Επι της ουσίας αυτή η μέθοδος μας επιτρέπει την κατασκευή ιστοσελίδας που εκτελείται επί της συσκευής και με χρήση εξειδικευμένων μεθόδων έχουμε έλεγχο των πόρων της συσκευής. Αυτή η επιλογή συνεπάγεται ευκολότερη ανάπτυξη καθ'ότι ο προγραμματισμός γίνεται σε υψηλότερο επίπεδο αφαίρεσης αλλά και ελαφρώς αυξημένη απαίτηση σε ενέργεια λόγω της μεγαλύτερης πολυπλοκότητας των εντολών υψηλότερου επιπέδου.



Σχήμα 18: Αρχιτεκτονική Λειτουργικού TizenOS

Επιλέξαμε τον 2ο τρόπο ανάπτυξης, καθώς η υλοποίηση σε περιβάλλον JavaScript και web εν γένει τεχνολογιών προσφέρει βασικά πλεονεκτήματα έναντι της ανάπτυξης native εφαρμογής:

1. Η εκτέλεση κώδικα JavaScript, όπως έχουμε αναφέρει στην ενότητα **2.2.4.3.1** στηρίζεται στο event-driven μοντέλο, όπου κομμάτια κώδικα δηλώνουν το ενδιαφέρον τους για διάφορα γεγονότα τα οποία εκτελούνται σύγχρονα και όχι σειριακά όπως σε μια απλή υλοποίηση σε γλώσσα προγραμματισμού C. Έτσι, αποφεύγονται ατέρμονοι βρόχοι οι οποίοι ελέγχουν μια συνθήκη και μεταβαίνουμε σε εκτέλεση κώδικα μόνο όταν αυτή η συνθήκη πληρείται, εξοικονομώντας ενέργεια στην συσκευή.
2. Η υλοποίηση με web τεχνολογιών μας δίνει την δυνατότητα για ομοιογενέστερη εμπειρία χρήστη σε επίπεδο γραφικής διεπαφής αφού μπορούμε πολύ πιο εύκολα να υλοποιήσουμε ένα ενοποιημένο γραφικό περιβάλλον. Το πλεονέκτημα αυτό ενισχύεται καθώς σε περιβάλλον native εφαρμογής η υλοποίηση με χρήση της γλώσσας προγραμματισμού C, αφαιρεί πολλά από τα διαδραστικά κομμάτια που υλοποιεί η λογική αφαίρεση με τον προγραμματισμό σε web περιβαλλον.
3. Το σημαντικότερο όμως πλεονέκτημα αυτής της επιλογής είναι ότι χρησιμοποιούμε συνολικά στο σύστημα κοινό τύπο δεδομένων και μεταβλητών. Συγκεκριμένα, όπως έχουμε αναφέρει στην ενότητα **2.2.5.2.4** με την χρήση web τεχνολογιών και την υλοποίηση του κεντρικού εξυπηρετητή με αρχιτεκτονική REST, όλες οι πληροφορίες μεταφέρονται εντός του συστήματος με δομή δεδομένων τύπου JSON. Το γεγονός αυτό δεν προκαλεί μεγάλη διαφορά σε περιβάλλοντα με πληθώρα επεξεργατηκών πόρων ή/και μη περιορισμένης κατανάλωσης ενέργειας. Όμως στο περιορισμένο περιβάλλον του smartwatch η χρήση επεξεργαστικού χρόνου για την μετατροπή των τύπων δεδομένων

απο την native μορφή στην web εκατοντάδες φορές το δευτερόλεπτο έχει σημαντικό αντίκρισμα στην κατανάλωση ενέργειας.

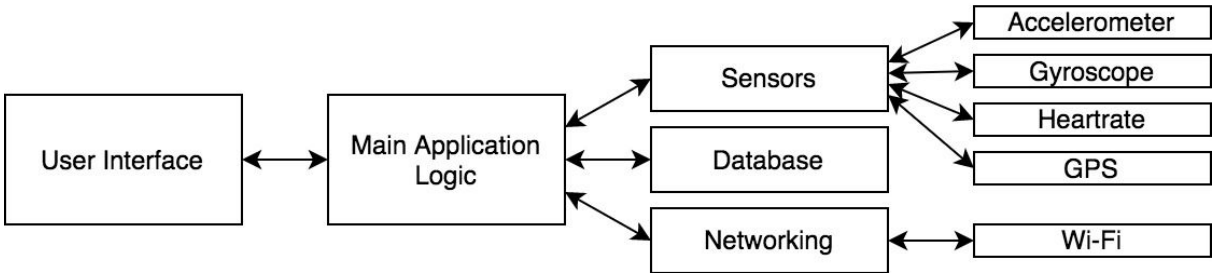
Παρότι χάνουμε λοιπόν ένα μικρό πλεονέκτημα στο να έχουμε απόλυτο έλεγχο επι του υλικού της συσκευής κερδίζουμε μακροπρόθεσμα σε κατανάλωση ενέργειας, ευκολία αναπαραγωγής ομοιόμορφης εμπειρίας χρήστη και μείωση χρόνου ανάπτυξης.

3.2.2.1 Οντολογική Σχεδίαση

Η εφαρμογή που εκτελείται επι της συσκευής smartwatch, Samsung Gear S3 Frontier, αποτελείται από την οντολογική ιεραρχία που παρουσιάζεται στο κάτωθι Σχήμα 19. Συγκεκριμένα ο πυρήνας της εφαρμογής είναι η συλλογή και η αποθήκευση δεδομένων κίνησης και βιοσημάτων του χρήστη. Υιοθετήθηκε η σχεδίαση του συστήματος σε ξεχωριστές μονάδες-στοιχεία σε ιεραρχική δομή ώστε να διευκολυνθεί η ανάπτυξη αλλά και η μελλοντική επέκταση και συντήρηση του κώδικα. Βασικό στοιχείο αυτών των μονάδων είναι ο διαχωρισμός των “ανησυχιών” κάθε μίας, αφού είναι διαχωρισμένες με λειτουργίες χωρίς επικαλύψεις. Ταυτόχρονα όπου χρειάζεται ο διαμοιρασμός δεδομένων μεταξύ των μονάδων, υιοθετήσαμε την σχεδίαση publisher-subscriber, ώστε να μην εμφανιστούν συνθήκες συναγωνισμού(race conditions).

Η φύση της εφαρμογής μπορεί να χωριστεί σε τρία μέρη με την κεντρική μονάδα-στοιχείο της εφαρμογής να είναι υπεύθυνη για:

- Την χρονοδρομολόγηση της δειγματοληψίας απο τους αισθητήρες της συσκευής και την αποθήκευση των ληφθέντων δεδομένων.
- Την εγγραφή σε γεγονότα σύνδεσης της συσκευής στο διαδίκτυο και εκτέλεσης αυτοματοποιημένου συγχρονισμού με τον κεντρικό διακομιστή.
- Την παροχή και ενημέρωση της γραφικής διεπαφής ώστε να είναι διαδραστική και λειτουργική για τον χρήστη.



Σχήμα 19: Η Τοπολογία της Οντολογικής Προσέγγισης που Αποτελεί την Εφαρμογή

3.2.2.2 Συλλογή Δεδομένων Δραστηριότητας και Βιοσήματος

Η συλλογή δεδομένων δραστηριότητας και βιοσημάτων είναι οι βασικές παράμετροι που παρακολουθεί το σύστημα άλλα και ο λόγος ύπαρξης της πλατφόρμας συνολικά. Στην εφαρμογή που εκτελείται τοπικά στην smartwatch συσκευή, τα δεδομένα δραστηριότητας αποτελούνται από την λήψη μετρήσεων από το επιταχυνσιόμετρο, γυροσκόπιο και το GPS. Αντίστοιχα το βιοσήμα του χρήστη που παρακολουθείται από την εφαρμογή είναι αυτό του καρδιακού ρυθμού, μέσα από παλμικό οπτικό αισθητήρα όπως έχουμε αναφέρει στην ενότητα **2.1.2.1**. Η δειγματοληψία των αισθητήρων συνολικά γίνεται εντελώς αυτοματοποιημένα και σε προκαθορισμένα διαστήματα από τους διαχειριστές της πλατφόρμας, χωρίς να απαιτείται καμία ενέργεια από τον χρήστη. Με αυτόν τον τρόπο μπορούμε και επιδιώκουμε να έχουμε φυσιολογικά δεδομένα από τους χρήστες με τιμές που εμφανίζονται εντός της καθημερινότητας τους και όχι σε κάποιο εργαστηριακό πείραμα με προσομοίωση.

3.2.2.2.1 Συλλογή Δεδομένων Κίνησης

Για τους σκοπούς της παρούσας διπλωματικής, ως δεδομένα κίνησης ορίζουμε την λήψη μετρήσεων από το επιταχυνσιόμετρο και το γυροσκόπιο. Όπως έχει δείξει η βιβλιογραφία [45][46] μπορούμε με μεγάλη ακρίβεια να αποφανθούμε για την δραστηριότητα του χρήστη βασιζόμενοι μόνο σε αυτά τα δεδομένα με ικανή δειγματοληψία.

Η συσκευή smartwatch Samsung Gear S3 Frontier περιλαμβάνει επιταχυνσιόμετρο και γυροσκόπιο με χρόνο δειγματοληψίας από 10 έως 1000 millisecond για κάθε μέτρηση τους. Στο περιβάλλον web και JavaScript που υλοποιείται η εφαρμογή μας το αποτέλεσμα κάθε δειγματοληψίας είναι μια δομή δεδομένων τύπου JSON όπως φαίνονται στα ακόλουθα σχήματα.

```
[NoInterfaceObject] interface SensorAccelerationData : SensorData {
  readonly attribute double x;
  readonly attribute double y;
  readonly attribute double z;
};
```

Σχήμα 20: Η JSON Δομή Δεδομένων που Επιστρέφει η κάθε Δειγματοληψία απο το Επιταχυνσιόμετρο

```
[NoInterfaceObject] interface SensorGyroscopeData : SensorData {
  readonly attribute double x;
  readonly attribute double y;
  readonly attribute double z;
};
```

Σχήμα 21: Η JSON Δομή Δεδομένων που Επιστρέφει η κάθε Δειγματοληψία απο το Γυροσκόπιο

3.2.2.2.2 Συλλογή Δεδομένων Τοποθεσίας

Παράλληλα με τα δεδομένα κίνησης απο τους προαναφερθείς αισθητήρες, η συσκευή smartwatch Samsung Gear S3 Frontier περιλαμβάνει και αισθητήρα GPS. Με την βοήθεια αυτού, μπορούμε να γνωρίζουμε με μεγάλη ακρίβεια την γεωγραφική θέση του χρήστη. Συμπληρωματικά λοιπόν γίνεται λήψη και της γεωγραφικής τοποθεσίας του χρήστη σε αραιά χρονικά διαστήματα όπως θα δούμε στο κεφάλαιο 4. Αυτό συνέβει, καθώς δεν κρίθηκε σκόπιμο, στην παρούσα εργασία να δημιουργηθεί λεπτομερές σύνολο δεδομένων χρηστών με τις γεωγραφικές τους τοποθεσίες. Αναγνωρίζοντας όμως ότι η υλοποίηση ενός τέτοιου χαρακτηριστικού θα αύξανε μελλοντικά τις δυνατότητες της πλατφόρμας, επιλέγει να υλοποιηθεί σε αυτό το σημείο. Έτσι λοιπόν, στο περιβάλλον web και JavaScript που υλοποιείται η εφαρμογή μας το αποτέλεσμα κάθε δειγματοληψίας είναι μια δομή δεδομένων τύπου JSON όπως φαίνεται στο ακόλουθο σχήμα.

```
[NoInterfaceObject] interface HumanActivityGPSInfo {
  readonly attribute double latitude;
  readonly attribute double longitude;
  readonly attribute double altitude;
  readonly attribute double speed;
  readonly attribute long errorRange;
  readonly attribute long timestamp;
};
```

Σχήμα 22: Η JSON Δομή Δεδομένων που Επιστρέφει η κάθε Δειγματοληψία από το GPS

3.2.2.2.3 Συλλογή Βιοσήματος Καρδιακού Ρυθμού

Η επιτυχής δειγματοληψία του βιοσήματος του καρδιακού ρυθμού είναι καίριας σημασίας για την παρούσα εργασία. Αποτελεί μια από τις βασικότερες παραμέτρους σε σενάρια αναγνώρισης φυσικής δραστηριότητας ακόμη και συναισθημάτων όπως του άγχους[47][48].

Για την λήψη και συλλογή του καρδιακού ρυθμού του χρήστη η συσκευή smartwatch Samsung Gear S3 Frontier περιλαμβάνει παλμικό οπτικό αισθητήρα από τον οποίο δειγματοληπτούμε μετρήσεις αρκετά συχνά προσπαθώντας παράλληλα να περιορίσουμε το ενεργειακό αντίκτυπο στην συσκευή. Στο περιβάλλον web και JavaScript που υλοποιείται η εφαρμογή μας το αποτέλεσμα κάθε δειγματοληψίας είναι μια δομή δεδομένων τύπου JSON όπως φαίνεται στο ακόλουθο σχήμα.

```
[NoInterfaceObject] interface HumanActivityHRMData : HumanActivityData {
 readonly attribute long heartRate;
 readonly attribute long rRInterval;
};
```

Σχήμα 23: Η JSON Δομή Δεδομένων που Επιστρέφει η κάθε Δειγματοληψία από τον Αισθητήρα Καρδιακού Ρυθμού

3.2.2.3 Προσωρινή Αποθήκευση Δεδομένων

Η αποδοτική προσωρινή αποθήκευση των ληφθέντων δεδομένων από τους αισθητήρες της συσκευής γίνεται με τοπική αποθήκευση τους σε περιβάλλον web με την χρήση κατάλληλων APIs. Συγκεκριμένα λόγω της υψηλης δομημένης φύσης των δεδομένων που συλλέγονται, επιλέχθηκε η αποθήκευση σε μοντέλο με ζεύγη κλειδιού-τιμής. Όπως αναφέραμε στην ενότητα **3.2.2.1**, η κεντρική μονάδα-στοιχείο είναι υπεύθυνη για την εγγραφή στο αποθηκευτικό μέσο των ληφθέντων δεδομένων καθώς και για την αποστολή αυτών στον κεντρικό διακομιστή. Κατ' αυτόν τον τρόπο και σε συνδυασμό με την υποστήριξη συναλλαγών απο το API, διασφαλίζεται η ακεραιότητα των δεδομένων όπως συμβαίνει σε συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων. Τέλος, σε περίπτωση που ο χρήστης δεν συνδεθεί σε δίκτυο ικανό να μεταφέρει τα αποθηκευμένα δεδομένα στον κεντρικό διακομιστή και ο διαθέσιμος αποθηκευτικός χώρος είναι πλήρης στην εφαρμογή, τότε η στρατηγική που ακολουθούμε είναι ότι προτιμάτε να γράφουμε επί των παλιών δεδομένων και όχι επί των νέων.

3.2.2.4 Συνδεσιμότητα και Συχρονισμός με τον Κεντρικό Server

Ακολουθώντας την οντολογική σχεδίαση της ενότητας **3.2.2.1** που αναφέραμε παραπάνω, έχουμε το υποσύστημα υπεύθυνο για την δικτύωση και την ασύγχρονη μεταφορά δεδομένων της συσκευής.

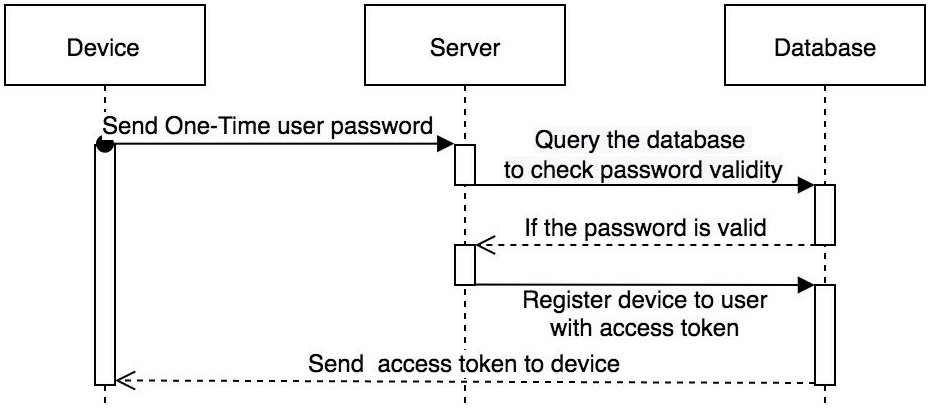
3.2.2.4.1 Συνδεσιμότητα Συσκευής με το Δίκτυο

Ακολουθώντας το προαναφερθέν, event-driven μοντέλο της γλώσσας προγραμματισμού JavaScript, το υποσύστημα ειδοποιείται ασύγχρονα και εκτελείται από το λειτουργικό σύστημα της συσκευής, όταν η συσκευή βρεθεί ενός δικτύου internet. Έτσι η εφαρμογή εκτελεί HTTP αυτοματοποιημένα αιτήματα προς τον κεντρικό διακομιστή ώστε να συγχρονίσει ή/και να ανανεώσει τους πόρους της.

3.2.2.4.2 Επαλήθευση Ταυτότητας και Εξουσιοδότηση Συσκευής

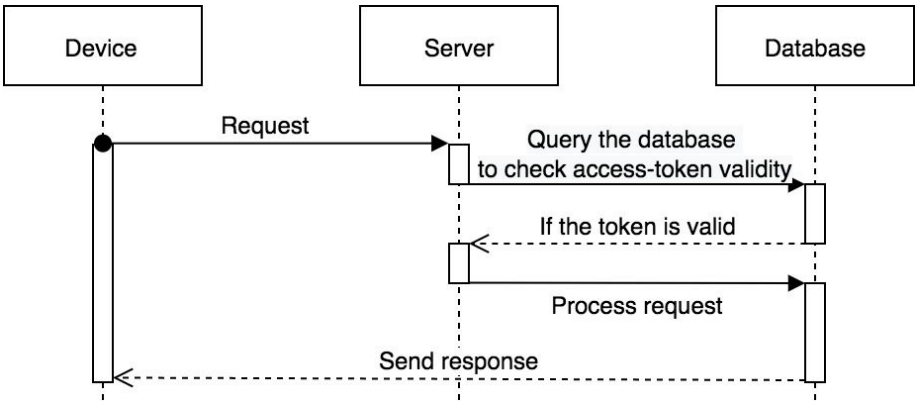
Είναι σαφές ότι όταν ο χρήστης εγκαταστήσει για πρώτη φορά την εφαρμογή του στην συσκευή smartwatch Samsung Gear S3 Frontier η εφαρμογή δεν είναι σε θέση να γνωρίζει τα στοιχεία του χρήστη της. Αυτό έχει ως αποτέλεσμα να απαιτείται η επαλήθευση της ταυτότητας της συσκευής και κατ' επέκταση και του χρήστη ώστε να είναι δυνατή η χρήση της από την διαδικτυακή πλατφόρμα. Καθότι όμως η γραφική διεπαφή των smartwatches είναι περιορισμένη, αποφασίστηκε ο χρήστης να εισάγει έναν μοναδικό 4-ψήφιο κωδικό ο οποίος να λήγει ανά τακτά χρονικά διαστήματα και να δημιουργείται νέος. Με αυτόν τον τρόπο ο χρήστης δεν χρειάζεται να εισάγει μακροσκελή δεδομένα για την ταυτοποίησή του με τον κίνδυνο να κάνει λάθος αλλά και να χρειαστεί επιπλέον χρόνο. Αντ' αυτού, με έναν απλό κωδικό έχουμε τα ίδια οφέλη επαλήθευσης και εξουσιοδότησης.

Συγκεκριμένα, στο ακόλουθο σχήμα περιγράφεται το πρωτόκολλο επαλήθευσης της ταυτότητας του χρήστη και η εγγραφή της νέας του συσκευής στο σύστημα. Υποθέτοντας ότι ο χρήστης είναι ήδη εγγεγραμμένος στην διαδικτυακή πλατφόρμα, αρχικά εισάγει τον προσωρινό κωδικό του εντός της εφαρμογής που εκτελείται στην smartwatch συσκευή του. Ο κεντρικός διακομιστής τότε με αίτημα στην βάση δεδομένων ταυτοποιεί τον χρήστη μέσω του 8-ψήφιου κωδικού. Σε περίπτωση επιτυχούς ταυτοποίησης ο εξυπηρετητής δημιουργεί ένα αναγνωριστικό πρόσβασης(access token) το οποίο αποθηκεύει στην βάση δεδομένων και αποστέλλεται στην συσκευή με το οποίο εξουσιοδοτείται η χρήση της.



Σχήμα 24: Πρωτόκολλο Επαλήθευσης Ταυτότητας Χρήστη από την Συσκευή

Για την επιτυχή εξουσιοδότηση της smartwatch συσκευής πρέπει να έχει προηγηθεί η εγγραφή και η ταυτοποίηση της. Το αναγνωριστικό πρόσβασης που αποστάλθηκε απο τον κεντρικό διακομιστή την πρώτη φορά που εξουσιοδοτήθηκε η συσκευή, αποθηκεύεται μόνιμα τοπικά στην συσκευή έως ότου λήξει και ανανεωθεί αυτόματα. Όπως φαίνεται στο ακόλουθο σχήμα, σε κάθε αίτημα HTTP που απευθύνεται προς τον κεντρικό διακομιστή το αναγνωριστικό πρόσβασης ανακαλείται από την τοπική αποθήκευση και αποστέλλεται στην επικεφαλίδα του αιτήματος. Αυτό το αναγνωριστικό πρόσβασης χρησιμοποιεί κάθε φορά ο διακομιστής ώστε να πιστοποιήσει την συσκευή αλλά και να εξουσιοδοτήσει την πρόσβαση της στους πόρους του.

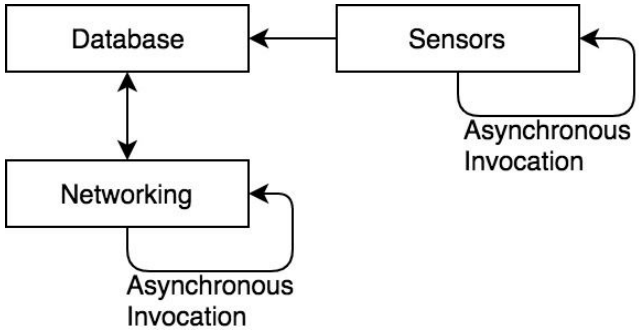


Σχήμα 25: Πρωτόκολλο Εξουσιοδότησης Συσκευής

3.2.2.4.3 Συγχρονισμός Δεδομένων με τον Κεντρικό Server

Ολοκληρώνοντας τα σχεδιαστικά χαρακτηριστικά της εφαρμογής συλλογής δεδομένων, καταλήγουμε σ' ένα σύστημα το οποίο ασύγχρονα συλλέγει δεδομένα και ασύγχρονα τα αποστέλλει στον κεντρικό διακομιστή. Η έννοια της ασύγχρονης εκτέλεσης κώδικα είναι καίριας σημασίας για την χαμηλή κατανάλωση ενέργειας της συσκευής. Σε περίπτωση που η συσκευή βρεθεί εντός δικτύου τότε άμεσα το λειτουργικό επιτρέπει την εκτέλεση της ώστε να συγχρονίσει τα συλλεχθέντα δεδομένα με τον κεντρικό εξυπηρετητή. Παράλληλα, χρησιμοποιώντας κατάλληλες εντολές απο το API της συσκευής επιτυγχάνουμε βελτίωση σε επίπεδο υλικού καθώς σε χρονικές περιόδους που ο επεξεργαστής δεν χρειάζεται, τότε παύει προσωρινά την εκτέλεση του, μειώνοντας δραστικά τις ενεργειακές του ανάγκες.

Επίσης για τους σκοπούς της παρούσας διπλωματικής εργασίας παρέχεται και η δυνατότητα εξαναγκασμένης αποστολής δεδομένων με παρουσία δικτύου για πειραματικές δοκιμές.



Σχήμα 26: Τοπολογία Ασύγχρονης Λειτουργίας των Οντοτήτων της Συσκευής

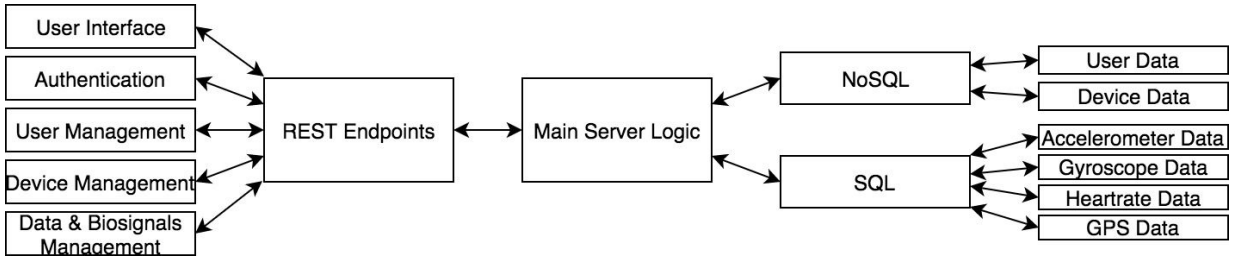
3.2.3 Η Διαδικτυακή Εφαρμογή Συλλογής Δεδομένων και οι Υπηρεσίες της

Ο σκοπός της διαδικτυακής εφαρμογής, που εκτελείται στο cloud και εξυπηρετείται απο τον Server, είναι τριπλός. Πρώτον, να παρέχει την γραφική διαδικτυακή επαφή(front-end) με την οποία αλληλεπιδρά ο χρήστης μέσω του browser. Δεύτερον, να παρέχει τις καθορισμένες διεπαφές επικοινωνίας ώστε να είναι δυνατή η ανταλλαγή και ο συγχρονισμός δεδομένων με τις γραφικές διαδικτυακές επαφές (front-end) αλλά και τις εφαρμογές που εκτελούνται στις συσκευές των χρηστών. Τρίτον, να αποθηκεύει σωστά και δομημένα όλη την απαιτούμενη πληροφορία ώστε η πλατφόρμα συνολικά να καλύπτει τις λειτουργικές απαιτήσεις.
Η εφαρμογή λοιπόν διαμερίζεται σε τρεις ξεχωριστές λογικές μονάδες-στοιχεία για την επίτευξη των επιμέρους στόχων, οι οποίες και αναλύονται στην συνέχεια.

3.2.3.1 Σχεδιασμός του Κεντρικού REST Εξυπηρετητή

Ο κεντρικός REST εξυπηρετητής αποτελείται από την μονάδα που υλοποιεί τις διεπαφές προγραμματισμού εφαρμογών APIs τύπου REST και την μονάδα που εκτελεί την κυρίως επεξεργασία της εφαρμογής. Οι δύο λειτουργικές μονάδες, παρότι ξεχωριστές εκτελούνται στο ίδιο περιβάλλον εκτέλεσης. Η αποθήκευση δεδομένων υλοποιείται από την τρίτη λογική μονάδα, την βάση δεδομένων η οποία δεν αποτελεί λειτουργικό κομμάτι της υλοποίησης των διεπαφών του εξυπηρετητή. Εν γένει, η βάση δεδομένων δεν εκτελείται στο ίδιο περιβάλλον εκτέλεσης με τον εξυπηρετητή όπως θα δείξουμε παρακάτω. Όμως αποτελεί λειτουργικό κομμάτι στην επεξεργασία δεδομένων και εξυπηρέτησης αιτημάτων προς τον εξυπηρετητή, αφού είναι η μονάδα που αποθηκεύει όλες τις πληροφορίες του συστήματος.

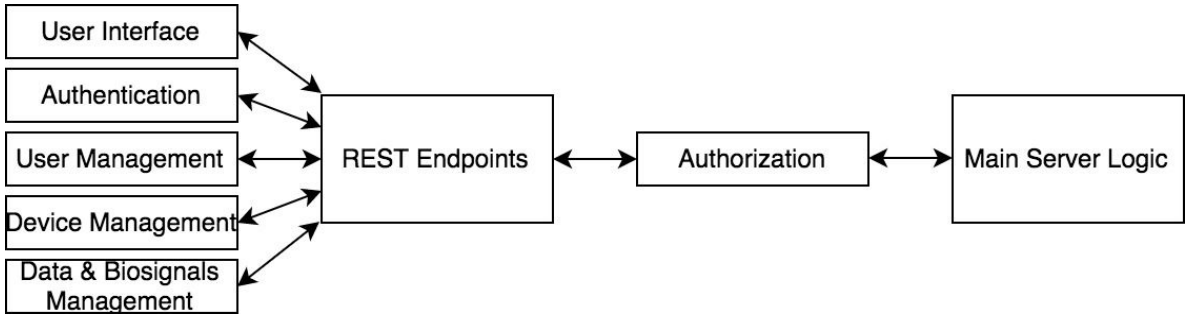
Στο κάτωθι σχημα, ο REST εξυπηρετητής καταλαμβάνει την αριστερή πλευρά στην οποία φαίνονται οι διαφορετικές κατηγορίες των διεπαφών προγραμματισμού εφαρμογών APIs που υλοποιεί. Στο κέντρο φαίνεται η κεντρική λογική μονάδα επεξεργασίας του εξυπηρετητή, ενώ το δεξί μέρος καταλαμβάνει η αποθήκευση δεδομένων στο οποίο φαίνεται ο διαχωρισμός στα μοντέλα αποθήκευσης που θα ακολουθήσουμε στην συνέχεια.



Σχήμα 27: Η Τοπολογία Διασύνδεσης των Διεπαφών Υποσυστημάτων σε Υψηλό Επίπεδο στον Κεντρικό Διακομιστή

3.2.3.1.1 Οντολογική Σχεδίαση

Η αρχιτεκτονική που ακολουθεί η διαδικτυακή εφαρμογή είναι το μοντέλο Client-Server, με την τοπολογία του REST εξυπηρετητή και τις αρμοδιότητες του να εμφανίζεται στο ακόλουθο σχήμα επιγραμματικά και σε υψηλό επίπεδο αφαίρεσης. Η πρώτη αρμοδιότητα του είναι η παροχή όλων των απαιτούμενων πόρων για την προβολή της γραφικής διεπαφής χρήστη στον browser. Η δεύτερη αρμοδιότητα είναι η παροχή των διεπαφών για την πιστοποίηση και εξουσιοδότηση χρηστών και συσκευών. Η τρίτη, τέταρτη και πέμπτη αρμοδιότητα είναι η παροχή διεπαφών για την διαχείριση χρηστών, συσκευών, βιοσημάτων και δεδομένων κίνησης αντίστοιχα. Πρέπει επίσης να τονίσουμε ότι η λειτουργική μονάδα εξουσιοδότησης αιτημάτων επιβλέπει όλα τα αιτήματα αφού εισέλθουν στον REST εξυπηρετητή και όχι πριν. Στην συνέχεια θα περιγράψουμε εποπτικά τις απαιτούμενες διεπαφές για την επίτευξή των προδιαγεγραμμένων λειτουργιών καθώς επίσης και ποιο σύνολο χρηστών εξυπηρετεί η κάθε μια.



Σχήμα 28: Η Τοπολογία της Οντολογικής Προσεγγισης του REST Εξυπηρετητή

3.2.3.1.2 Τρόπος Επικοινωνίας Client και Κεντρικού Εξυπηρετητή

Στον ακόλουθο πίνακα(Πίνακας 1), περιγράφονται τα APIs που απαιτούνται για την ορθή λειτουργία και επικοινωνία της της γραφικής διεπαφής χρήστη(front-end) με τον κεντρικό εξυπηρετητή. Συγκεκριμένα κάθε API αντιστοιχεί σε ξεχωριστό αίτημα HTTP που εκτελείται χρησιμοποιώντας μεθόδους HTTP GET ή POST. Συνήθως οι μέθοδοι GET επιστρέφουν αρχεία html και άλλους πόρους για προβολή τους στον browser του χρήστη. Οι μέθοδοι POST,

χρησιμοποιούνται για την μεταφορά δεδομένων, συνήθως με την μορφή JSON μεταξύ του client και του server.

Τοποθεσία	Μέθοδος HTTP	Λειτουργία
Απλοί Χρήστες		
baseURL/users/register	GET/POST	Προβολή φόρμας για την δημιουργία λογαριασμού χρήστη προφίλ χρήστη
baseURL/users/panel	GET	Προβολή προσωπικού προφίλ χρήστη
baseURL/users/login	GET/POST	Προβολή φόρμας για την είσοδο του χρήστη στο σύστημα
baseURL/users/update	POST	Προβολή φόρμας για την ενημέρωση των στοιχείων λογαριασμού του χρήστη
baseURL/users/delete	POST	Διαγραφή χρήστη
baseURL/users/show	POST	Προβολή στοιχείων χρήστη
baseURL/users/changeEmail	POST	Αλλαγή email χρήστη
baseURL/users/requestChangePasswordPanel	GET	Προβολή για αίτηση αλλαγής κωδικού πρόσβασης χρήστη
baseURL/users/requestChangePassword	POST	Αίτηση αλλαγής κωδικού πρόσβασης χρήστη
baseURL/users/changePassword	POST	Αλλαγή κωδικού πρόσβασης χρήστη
baseURL/users/verifyEmail	GET	Επιβεβαίωση email χρήστη
Διαχειριστές Συστήματος		
baseURL/admin/index	GET	Προβολή πάνελ διαχείρισης συστήματος
baseURL/users/changeStatus	POST	Αλλαγή δικαιωμάτων χρήστη
baseURL/users/getAll	POST	Προβολή όλων των χρηστών του συστήματος
Συσκευές		
baseURL/devices/show	POST	Προβολή στοιχείων συσκευής

baseURL/devices/delete	POST	Διαγραφή συσκευής
Βιοσήματα και Δεδομένα Κίνησης		
baseURL/biosignals/retrieve	POST	Προβολή βιοσημάτων και δεδομένων κίνησης χρήστη

Πίνακας 1: Απαιτούμενα APIs για την Λειτουργία και Επικοινωνία της Γραφικής Διεπαφής Χρήστη με τον Εξυπηρετητή

3.2.3.1.3 Τρόπος Επικοινωνίας Εφαρμογής Συλλογής Δεδομένων και Κεντρικού του Εξυπηρετητή

Στον ακόλουθο πίνακα(Πίνακας 2), περιγράφονται τα δύο APIs που απαιτούνται για την ορθή λειτουργία και επικοινωνία της συσκευής smartwatch του χρήστη με τον κεντρικό εξυπηρετητή. Ομοίως με την προηγούμενη ενότητα κάθε API αντιστοιχεί σε ξεχωριστό αίτημα HTTP που εκτελείται χρησιμοποιώντας μεθόδους HTTP GET ή POST.

Τοποθεσία	Μέθοδος HTTP	Λειτουργία
baseURL/devices/register	POST	Καταχώρηση και εξουσιοδότηση συσκευής smartwatch
baseURL/biosignals/upload	POST	Αποθήκευση δεδομένων κίνησης και βιοσημάτων στον κεντρικό εξυπηρετητή

Πίνακας 2: Απαιτούμενα APIs για την Λειτουργία και Επικοινωνία της Συσκευής Smartwatch του Χρήστη με τον Εξυπηρετητή

3.2.3.2 Σχεδίαση Βάσης Δεδομένων

Απαραίτητο δομικό στοιχείο της διαδικτυακής εφαρμογής αποτελεί η βάση δεδομένων του συστήματος που είναι υπεύθυνη για τη μόνιμη και ασφαλή αποθήκευση των πληροφοριών και δεδομένων, αναγκαία για τη λειτουργία της. Η βάση δεδομένων αποτελεί ξεχωριστή οντότητα(entity) του συστήματος καθώς δε συνεπάγεται ότι αποτελεί μια μονάδα, ούτε ότι βρίσκεται εντός του ίδιου εξυπηρετητή στο περιβάλλον εκτέλεσης. Στην περίπτωση του συστήματος μας, η οντότητα της βάσης δεδομένων αποτελείται από δύο διαφορετικούς εξυπηρετητές βάσης δεδομένων που εκτελούν διαφορετικά μοντέλα αποθήκευσης ένα σχεσιακό και ένα μη-σχεσιακό όπως θα αναλύσουμε στη συνέχεια. Εν γένει όμως αναφερόμαστε στην οντότητα της αποθήκευσης δεδομένων ως βάση δεδομένων, αγνοώντας σε αυτό το επίπεδο αφαίρεσης, την υλοποίηση και το περιβάλλον εκτέλεσης.

Η δομή της βάσης δεδομένων προσπελαύνεται εξ'ολοκλήρου από τον κεντρικό εξυπηρετητή ώστε: να ανακτήσει και να επεξεργαστεί υπάρχοντα δεδομένα που ζητήθηκαν ή να αποθηκεύσει νέα δεδομένα που καταχωρήθηκαν από τον χρήστη. Επιπλέον, διατηρεί τα απαραίτητα διαπιστευτήρια και πιστοποιητικά για την ασφαλή πιστοποίηση και εξουσιοδότηση χρηστών, εφαρμογών και συσκευών. Σημαντική λεπτομέρεια της σχεδίασης και υλοποίησης αποτελεί ότι η βάση δεδομένων δεν είναι απευθείας προσπελάσιμη από τους πελάτες (Clients) της εφαρμογής παρα μόνο μέσω APIs που παρέχει ο κεντρικός διακομιστής. Αυτό οφείλεται στο γεγονός ότι η βάση δεδομένων είναι υπεύθυνη μόνο για την ασφαλή αποθήκευση των δεδομένων και όχι για τη διασφάλιση και επιβολή δικαιωμάτων προσβασιμότητας τους.

3.2.3.2.1 Η μορφή των δεδομένων

Όπως αναφέραμε σε υψηλό επίπεδο σχεδίασης αντιμετωπίζουμε την βάση δεδομένων σαν ενιαία μοναδική οντότητα. Η φύση όμως των προς-αποθήκευση δεδομένων καθώς και το προφίλ εγγραφών-ανάγνωσεων που απαιτεί το σύστημα στη βάση, επιβάλλει την περαιτέρω μοντελοποίηση της. Σε ένα πρώτο επίπεδο, τα προς-αποθήκευση δεδομένα του συστήματος χωρίζονται σε δύο κατηγορίες ανάλογα με την ποσότητα, την μορφή τους και το προφίλ εγγραφών-ανάγνωσεων του συστήματος: τα βιοσήματα και δεδομένα κίνησης και τα δεδομένα χρηστών και συσκευών. Τα χαρακτηριστικά τους αυτά αναλύονται στην συνέχεια καθώς επίσης και οι αρχιτεκτονικές αποφάσεις που επιβάλλουν στην σχεδίαση του συστήματος.

3.2.3.2.1.1 Δεδομένα Χρηστών και Δεδομένα Συσκευών

Το σύστημα είναι απαραίτητο να κρατά στοιχεία σχετικά με τους χρήστες και τις συσκευές τους. Συγκεκριμένα το σύστημα πρέπει να μπορεί να αποθηκεύει πρωτογενή αναγνωριστικά σχετικά των χρηστών του όπως email, κωδικό πρόσβασης, ηλικία, φύλο, στοιχεία επικοινωνίας όπως τηλέφωνο και διεύθυνση κ.α. για την αποκλειστική ταυτοποίηση των χρηστών. Ταυτόχρονα, απαιτούνται επιπρόσθετα αναγνωριστικά, όπως η ημερομηνία που γράφτηκε ο χρήστης στο σύστημα, η ημερομηνία που υπήρξε τελευταία φορά ενεργός ή αναγνωριστικά των συσκευών που του ανήκουν κ.α. Συνολικά τα προαναφερθέντα, διαμορφώνουν τα απαραίτητα στοιχεία για την πλήρη λειτουργία του συστήματος και την παροχή εμπειρίας χρήσης όπως σκιαγραφήσαμε.

Όμοιες παραδοχές εμφανίζονται και στα δεδομένα που πρέπει το σύστημα να βρίσκεται σε θέση να αποθηκεύει για της συσκευές των χρηστών της. Αναγνωριστικά όπως, το όνομα της συσκευής, σε ποιόν χρήστη ανήκει, πότε ήταν τελευταία φορά ενεργή, πότε εισήχθη πρώτη φορά στο σύστημα κ.α είναι απαραίτητα για την ορθή χρήση της συσκευής από το σύστημα.

Σημαντική ιδιότητα όλων αυτών των αναγνωριστικών είναι η ευμεταβλητότητα και ποικιλομορφία τους, και ως προς το πλήθος τους και ως προς τον τύπο των δεδομένων τους. Είναι λοιπόν αναγκαίο να αυτοματοποιήσουμε αυτή την διαδικασία επιτρέποντας τα δεδομένα μας να αλλάζουν μορφή ανάλογα με την χρήση. Για παράδειγμα, αν στο μέλλον θέλουμε να

συλλέγουμε και κάποιες περαιτέρω πληροφορίες για τους χρήστες ή τις συσκευές τους θα πρέπει να είμαστε σε θέση να κάνουμε αυτή την προσθήκη αποδοτικά.

Όσον αφορά το προφίλ χρήσης των δεδομένων αυτών στο σύστημα μας, ορίζεται ως read-heavy. Αυτό συμβαίνει καθότι τα δεδομένα που σχετίζονται με τους χρήστες και τις συσκευές τους παράγονται και αποθηκεύονται μία φορά στην αρχή. Η ενημέρωση τους είναι πολύ σπάνια, αφού δεν τείνουν να αλλάζουν συχνά. Αντίθετα όμως από την στιγμή που τα δεδομένα αυτά αποθηκευτούν, ανακαλούνται και επεξεργάζονται διαρκώς από τον εξυπηρετητή για την παροχή των προκαθορισμένων υπηρεσιών του. Για παράδειγμα, σ' ένα τυπικό σενάριο που ο χρήστης επιθυμεί να προβάλει τα βήματα του στο προσωπικό του προφίλ, απαιτείται μια σειρά απο ταχύτατα ερωτήματα ανάκλησης και προβολής αποθηκευμένων δεδομένων και ο διακομιστής και κατα συνέπεια η βάση δεδομένων είναι υπεύθυνη για την επιτυχή εξυπηρέτηση των αιτημάτων.

Οι αναγνώσεις που εκτελεί το σύστημα στα δεδομένα του: απαιτούν εκτεταμένη επεξεργασία, πραγματοποιούνται σε μεγάλο όγκο και σε πολύ μεγαλύτερη συχνότητα από αυτή των εγγραφών.

3.2.3.2.1.2 Βιοσήματα και Δεδομένα Κίνησης

Όπως αναλύσαμε στην ενότητα **3.2.2.2**, τα δεδομένα που συλλέγονται από τους αισθητήρες της συσκευής είναι συγκεκριμένα ως προς το πλήθος τους και έχουν συγκεκριμένο τύπο. Υποθέτουμε χωρίς βλάβη της γενικότητας, ότι σε περίπτωση που υλοποιηθεί η διεπαφή επικοινωνίας με άλλη συσκευή smartwatch, τα ίδια δεδομένα θα απαιτηθούν απο τους αισθητήρες της νέας συσκευής. Στα πλαίσια του συστήματος μας δηλαδή ο τύπος και η πληθώρα των δεδομένων από τους αισθητήρες δεν θα μεταβληθεί. Αν όμως συμβεί κάτι τέτοιο, κάνουμε την παραδοχή ότι δεν θα συμβεί αρκετά συχνά έτσι ώστε να αυτοματοποιηθεί στο σύστημα μας.

Όσον αφορά το προφίλ χρήσης των δεδομένων αυτών στο σύστημα μας, ορίζεται ως write-heavy. Αυτό συμβαίνει καθότι τα δεδομένα συλλέγονται από την συσκευή smartwatch του χρήστη και μεταφέρονται πολύ συχνά στην κεντρική βάση δεδομένων μας. Κάθε φορά που ανεβάζονται δεδομένα από την συσκευή του χρήστη το πλήθος τους θα κυμαίνεται από μερικές χιλιάδες αν η συσκευή είχε συνδεθεί στο πρόσφατο παρελθόν, μέχρι εκατοντάδες χιλιάδες αν η συσκευή λειτούργησε για εκτενές διάστημα σε απουσία δικτύου. Ο διακομιστής και κατα συνέπεια η βάση δεδομένων είναι υπεύθυνη για την ομαδική και ταχύτατη αποθήκευση αυτών των δεδομένων ώστε να εξυπηρετηθούν τα αιτήματα με επιτυχία. Οι εγγραφές δηλαδή στο σύστημα είναι πολύ συχνές και σε μεγάλο όγκο, χωρίς να απαιτείται επεξεργασία προτού εισαχθούν. Αντίθετα, από την στιγμή που τα δεδομένα αποθηκευτούν, η ανάγκη προς το σύστημα είναι η ομαδική τους χρήση ώστε να εξαχθούν χρήσιμα συμπεράσματα. Οι αναγνώσεις δηλαδή που εκτελεί το σύστημα πραγματοποιούνται σε πολύ μικρότερη συχνότητα από αυτή των εγγραφών.

3.2.3.2.2 Βάση Δεδομένων για τα Δεδομένα Χρηστών - NoSQL

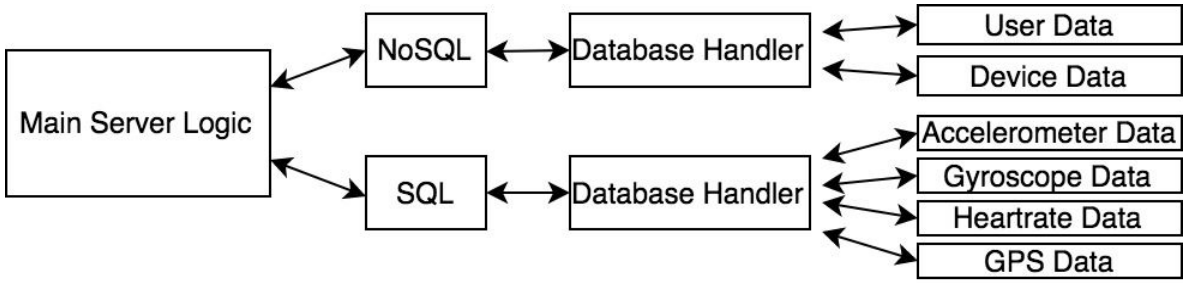
Οι προδιαγραφές που θέσαμε για τα δεδομένα χρηστών και συσκευών που κρίνονται απαραίτητες για την ορθή λειτουργία της εφαρμογής κάνουν επικρατέστερο το μη-σχεσιακό μοντέλο βάσης δεδομένων. Όπως αναλύσαμε στην ενότητα **2.2.5.4.3** στο μη-σχεσιακό μοντέλο οι πληροφορίες αποθηκεύονται σε έγγραφα. Το κάθε έγγραφο αντιπροσωπεύει μια ξεχωριστή καταχώρηση και αποθηκεύει την πληροφορία της ακολουθώντας την δική του διάρθρωση η οποία δεν απαιτείται να είναι κοινή για όλη την συλλογή εγγράφων στην βάση δεδομένων. Η αναπαράσταση αυτή ενδείκνυται για την αποθήκευση δεδομένων που εμφανίζουν ποικιλομορφία τύπων και δομών όπως μπορεί εύκολα να συμβεί στο σύστημά μας. Τέλος, μιας και τα έγγραφα είναι ανεξάρτητα μεταξύ τους, η διαδικασία ανάγνωσης είναι εξαιρετικά γρήγορη ιδίως σε κατανεμημένες υλοποιήσεις.

3.2.3.2.3 Βάση Δεδομένων για τα Βιοσήματα και Δεδομένα Κίνησης Χρηστών - SQL

Αντίστοιχα, η στατικότητα της μορφής των δεδομένων βιοσημάτων και των δεδομένων κινήσεων σε συνδυασμό με το write-heavy προφίλ χρήσης τους, κάνει την εφαρμογή ενός σχεσιακού μοντέλου ιδανική. Όπως αναλύσαμε στην ενότητα **2.2.5.4.2** το σχεσιακό μοντέλο αναπαριστά την πληροφορία των δεδομένων σε μορφή πινάκων όπου κάθε σειρά αναπαριστά μια πλειάδα ενώ κάθε στήλη αναπαριστά κάποιο πεδίο που περιέχουν όλες οι πλειάδες. Η αναπαράσταση αυτή ενδείκνυται για την αποθήκευση δεδομένων σε σειρές που είναι πλήρης και δεν αφήνουν άδεια κελιά ώστε να υπάρχει σπατάλη του χώρου αποθήκευσης, όπως στην περίπτωση μας. Επιπλέον, αφού η αποθήκευση νέων δεδομένων-πλειάδων γίνεται στο τέλος του κάθε πίνακα η διαδικασία είναι σύντομη.

3.2.3.2.4 Σχεσιακή Αντιστοίχιση Αντικειμένων - ORM

Η σχεσιακή αντιστοίχιση αντικειμένων, όπως αναλύσαμε στην ενότητα **2.2.5.4.5** είναι ένας μηχανισμός που μας επιτρέπει την εκτέλεση ερωτημάτων προς την εκάστοτε βάση δεδομένων σε υψηλό επίπεδο αφαίρεσης αντικειμενοστραφούς προγραμματισμού. Κατ' αυτόν τον τρόπο πετυχαίνουμε ασφαλέστερη επικοινωνία με την βάση, ταχύτερη εξυπηρέτηση των ερωτημάτων μας σε πληθώρα περιπτώσεων και εύκολη επέκταση της υλοποίησης. Όπως παρουσιάσαμε, η διαδικτυακή εφαρμογή χρησιμοποιεί και σχεσιακό και μη-σχεσιακό μοντέλο βάσης δεδομένων για την αποδοτική αποθήκευση των δεδομένων της. Για τον λόγο αυτό, στο Σχήμα 29 βλέπουμε σχηματικά ότι χρησιμοποιούμε εννοιολογικά -και πρακτικά όπως θα δούμε στην συνέχεια- διαφορετικό ORM για το σχεσιακό μοντέλο και διαφορετικό ORM για το μη-σχεσιακό μοντέλο.



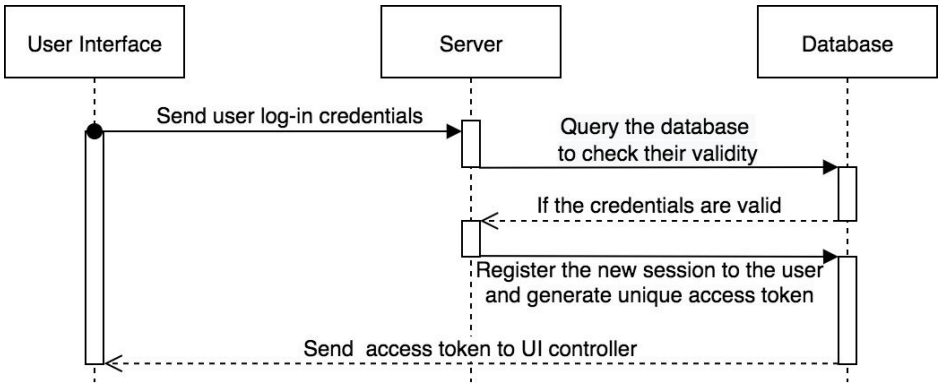
Σχήμα 29: Η Τοπολογία της Οντολογικής Προσεγγισης της Βάσης Δεδομένων

3.2.3.3 Συνδεσιμότητα και Συγχρονισμός Client με τον Κεντρικό Εξυπηρετητή

Σε πλήρη αναλογία της ενότητας **3.2.2.4**, η συνδεσιμότητα και συγχρονισμός με τον κεντρικό εξυπηρετητή ακολουθεί την ίδια αλληλουχία διαδικασιών πλην κάποιων εγγενών διαφορών που εμφανίζονται στις δύο εφαρμογές. Ο χρήστης έχει πρόσβαση στην διαδικτυακή εφαρμογή μέσω του browser την οποία παρέχει ο κεντρικός εξυπηρετητής όπως περιγράψαμε. Για την χρήση της εφαρμογής απαιτείται εκ των πραγμάτων σύνδεση με το δίκτυο καθώς οποιοσδήποτε συγχρονισμός επιτελείται με τον εξυπηρετητή πραγματοποιείται σύγχρονα μέσω δικτύου.

3.2.3.3.1 Επαλήθευση Ταυτότητας και Εξουσιοδότηση Χρήστη

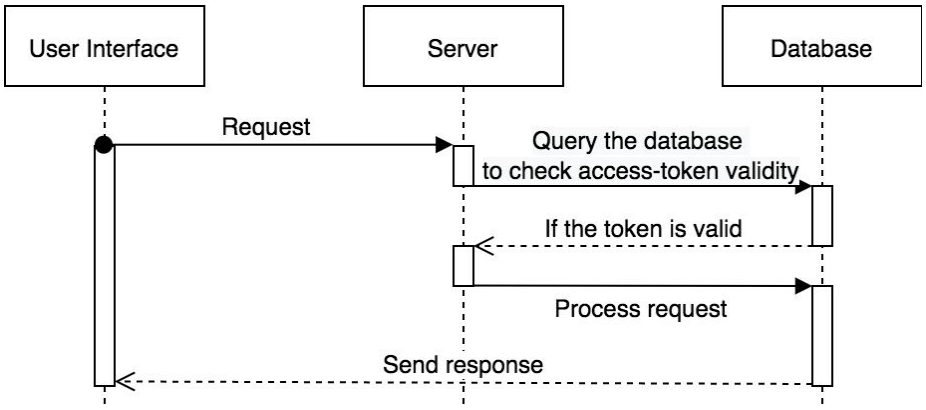
Για την χρήση της πλατφόρμας συνολικά, ο χρήστης υποχρεούται να έχει δημιουργήσει τον προσωπικό του λογαριασμό, έχοντας εισάγει όλα τα στοιχεία που απαιτούνται από την εφαρμογή. Εφόσον έχει ολοκληρωθεί με επιτυχία αυτό το βήμα, ο χρήστης για να χρησιμοποιήσει την εφαρμογή θα πρέπει να επαληθεύσει την ταυτότητα του εισάγοντας το προσωπικό email και τον κωδικό πρόσβασης του. Ο κεντρικός διακομιστής τότε, -σε αναλογία με την διαδικασία της ενότητας **3.2.2.4.2**- με αίτημα στην βάση δεδομένων ταυτοποιεί τον χρήστη και δημιουργεί ένα αναγνωριστικό πρόσβασης(access token) το οποίο και στελνει στην διεπαφή(front-end).



Σχήμα 30: Πρωτόκολλο Επαλήθευσης Ταυτότητας Χρήστη από την Διαδικτυακή Εφαρμογή

Για την επιτυχή εξουσιοδότηση των αιτημάτων που εκτελεί η γραφική διεπαφή στον κεντρικό εξυπηρετητή πρέπει να έχει προηγηθεί η εγγραφή και η ταυτοποίηση του χρήστη στην εφαρμογή. Το αναγνωριστικό πρόσβασης που αποστάλθηκε απο τον κεντρικό διακομιστή την πρώτη φορά που επαληθεύτηκε η ταυτότητα του χρήστη, αποθηκεύεται στον browser με την μορφή cookie εώς ότου λήξει.

Όπως αναφέρεται στην ενότητα **2.2.5.2.7**, το cookie αποστέλλετε από τον browser αυτοματοποιημένα σε κάθε αίτημα προς τον εξυπηρετητή, που χρησιμοποιεί την τιμή του για την εξουσιοδότηση των αιτημάτων και την πρόσβαση της στους πόρους του.



Σχήμα 31: Πρωτόκολλο Εξουσιοδότησης Χρήστη από το Γραφικό Περιβάλλον

4 Υλοποίηση Συστήματος

Σε αυτό το κεφάλαιο, θα παρουσιάσουμε αναλυτικά την υλοποίηση της πλατφόρμας ολιστικά. Θα δοθεί βαρύτητα στην υλοποίηση των επιμέρους συστημάτων καθώς επίσης, στην σύνδεση και την μεταξύ τους επικοινωνία. Συγκεκριμένα, θα αναλύσουμε πως το υπόβαθρο που παρουσιάσαμε στο κεφάλαιο 2 και, οι προδιαγραφές και αρχιτεκτονική του συστήματος που παρουσιάσαμε στο κεφάλαιο 3 υλοποιήθηκαν πρακτικά. Ταυτόχρονα, θα αναφερθούμε εκτενώς στις τεχνολογίες και μεθόδους που χρησιμοποιήσαμε στην υλοποίηση με κομμάτια κώδικα. Τέλος, θα παρουσιάσουμε όλες τις γραφικές διεπαφές και πως συνολικά καλύπτουν τα σενάρια χρήσης που θέσαμε.

4.1 Περιβάλλον εκτέλεσης

Η πλατφόρμα συλλογής και διαχείρισης δεδομένων κίνησης και βιοσημάτων όπως έχουμε περιγράψει δεν εκτελείται μόνο σε ένα περιβάλλον εκτέλεσης, αφού αποτελείται από λειτουργικώς ξεχωριστα υποσυστήματα. Συλλογικά επικοινωνώντας μεταξύ τους αυτά όμως και ανταλλάσσοντας πληροφορίες, συνθετουν το σύστημα συνολικά. Στην συνέχεια θα παρουσιάσουμε τα περιβάλλοντα εκτέλεσης και πως αυτά συνιστούν την απαιτούμενη λειτουργική υποδομή της πλατφόρμας.

4.1.1 NodeJS

Η υλοποίηση του διαδικτυακού εξυπηρετητή έγινε σε γλώσσα προγραμματισμού JavaScript στο περιβάλλον εκτέλεσης Node.js. Η διεργασία εκτελείτε επι λειτουργικού συστήματος, χρησιμοποιώντας το Chrome V8 για την εκτέλεση του κώδικα JavaScript και ακολουθεί το single-thread και το event-driven μοντελο της όπως έχουμε περιγράψει στην ενότητα **2.2.4.3.1**. Η υλοποίηση σε JavaScript ακολουθώντας το μοντέλο προγραμματισμου της, απλοποιεί την εξυπηρέτηση ταυτόχρονων αιτημάτων προς τον εξυπηρετητή καθώς επίσης και την διαχείριση πολλαπλών αιτημάτων εισόδου/εξόδου όπως αιτήματα στην βάση δεδομένων. Ταυτόχρονα, η χρήση περιορισμένων πόρων από το Node.js, διευκολύνει κατα πολύ την κλιμακωσιμότητα, αφού σε συνθήκες αυξημένου φόρτου καθιστά την αρχικοποίηση και εκτέλεση διεργασιών-κλώνων εξαιρετικά σύντομη. Τέλος έγινε χρήση της βιβλιοθήκης (framework) ExpressJS, λόγω της ευκολίας που παρέχει για την αποτελεσματική οργάνωση και δρομολόγηση αιτημάτων, τις μεθόδους API που υποστηρίζει.

4.1.2 Heroku

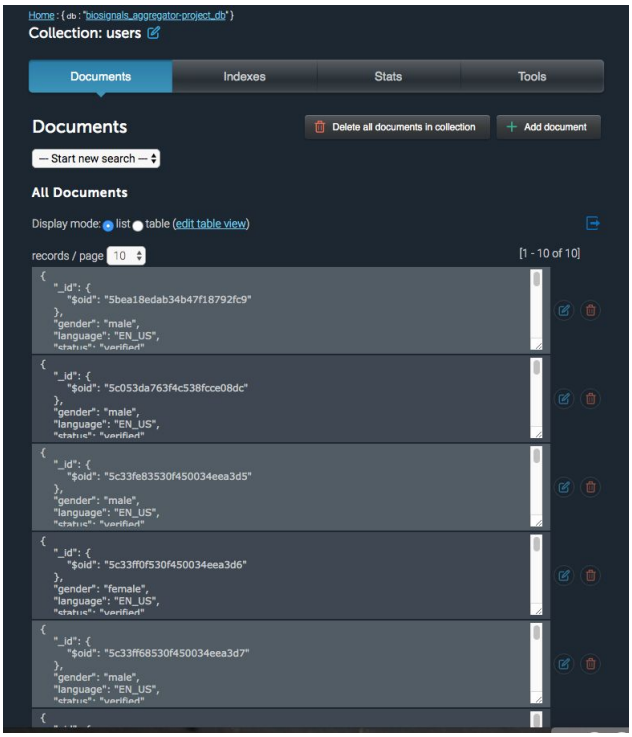
Ο διαδικτυακος εξυπηρετητης εκτελείται σε linux cloud περιβάλλον, το οποίο παρέχεται από τον πάροχο Heroku. Η εφαρμογή μας με αυτόν τον τρόπο είναι προσβάσιμη και εκτός του τοπικού δικτύου, στην δημόσια διεύθυνση:

https://gearapp-server.herokuapp.com

Η διεύθυνση αυτή αποτελεί και το baseURL ή αλλιώς την διεύθυνση που θα έχουν σαν ρίζα όλα τα αιτήματα μας προς τον εξυπηρετητη. Σημαντική υπηρεσία που μας παρέχετε επίσεις απο τον πάροχο, είναι η υποστήριξη του πρωτοκόλλου HTTPS που όπως έχουμε αναλύσει στην ενότητα **2.2.5.5**, πιστοποιεί και κρυπτογραφεί τα δεδομένα μας απο και προς το εξυπηρετητή. Έτσι παρέχουμε ασφάλεια και εμπιστευτικότητα κατα την χρήση των υπηρεσιών μας, στοιχεία πολύ σημαντικά για την λειτουργία του συστήματος συνολικά.

4.1.3 mLab

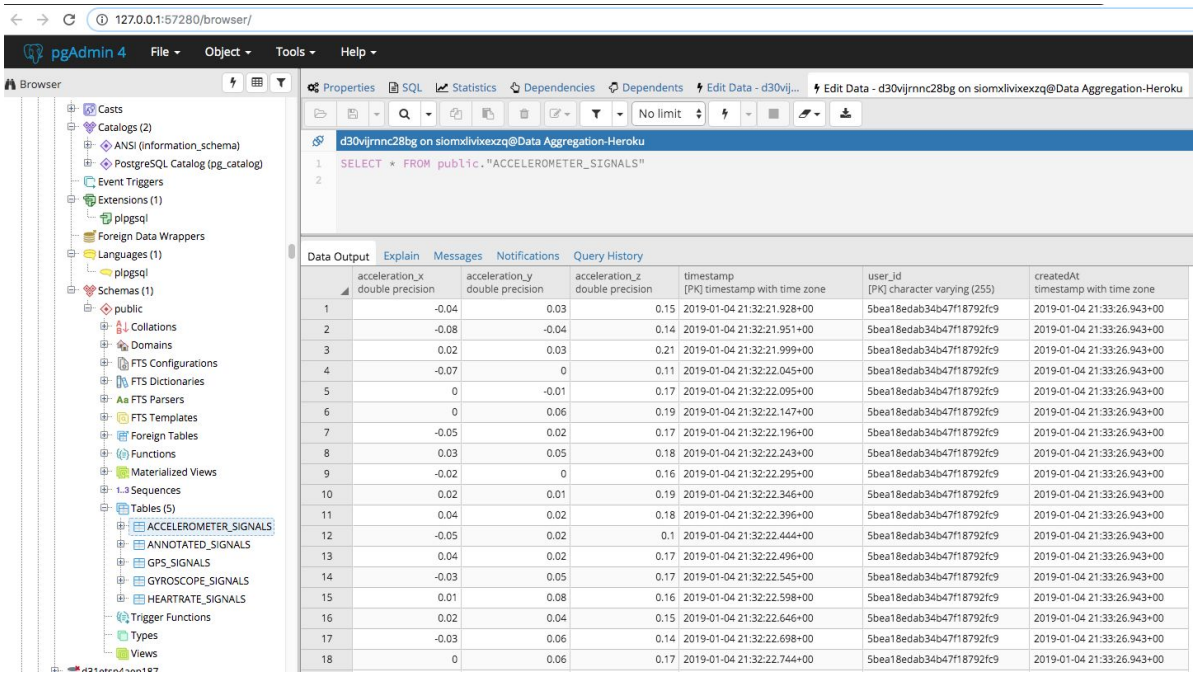
Η φιλοξενία του εξυπηρετητη της μη-σχεσιακής βάσης δεδομένων του συστήματος πραγματοποιείται σε διαφορετικό περιβάλλον εκτέλεσης από αυτό του κεντρικού εξυπηρετητή. Χρησιμοποιούμε τον πάροχο mLab για την φιλοξενία του MongoDB εξυπηρετητη μας, καθώς μας παρέχετε δωρεαν 500mb αποθηκευτικός χώρος, αρκετός για τις απαιτήσεις του συστήματος μας. Σημαντικότερο χαρακτηριστικό είναι ότι μας παρέχετε γραφική διεπαφή για την διαχείριση όλων των εγγραφών και των συλλογων του συστήματος μας καθώς και metrics για την απόδοση του συστήματος. Έτσι πέρα από την υλοποίηση, το διαχειριστικό μέρος της λειτουργικότητας του συστήματος απλοποιειται μακροπρόθεσμα.



Σχήμα 32: Γραφική Διεπαφή Διαχείρισης εγγραφών Χρηστών στον Πάροχο mLab

4.1.4 Okeanos

Η φιλοξενία του εξυπηρετητη της σχεσιακής βάσης δεδομένων του συστήματος πραγματοποιείται επίσης σε διαφορετικό περιβάλλον εκτέλεσης από αυτό του κεντρικού εξυπηρετητή. Χρησιμοποιούμε την υποδομή του Okeanos[51] για την φιλοξενία του PostgreSQL εξυπηρετητη μας, καθώς μας παρέχετε δωρεάν αποθηκευτικός χώρος, αρκετός για τις απαιτήσεις του συστήματος μας. Όμως το γεγονός ότι η υποδομή του στερείτε γραφικου περιβάλλοντος για την διαχείριση των εγγραφών του συστήματος, χρησιμοποιήσαμε το πρόγραμμα pgAdmin, όπως αναφέραμε στην ενότητα **2.2.6.5**,το οποίο μας παρέχει και την ζητούμενη γραφική διεπαφή με εξ αποστασεως σύνδεση στον PostgreSQL εξυπηρετητη.



Σχήμα 33: Στιγμιότυπο από την γραφική διεπαφή χρήσης της εφαρμογής pgAdmin

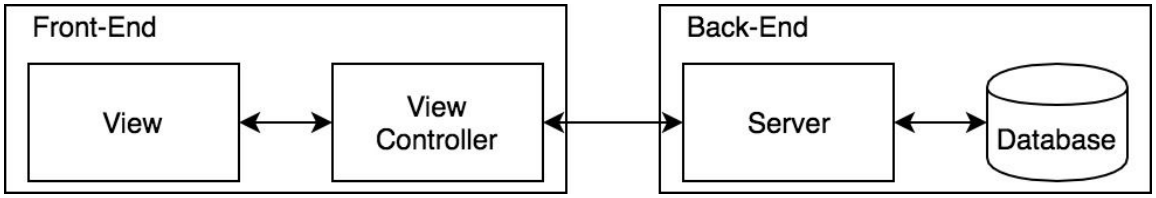
4.1.5 TizenOS

Η υλοποίηση της εφαρμογής συλλογής και προσωρινής αποθήκευσης βιοσημάτων και δεδομένων κίνησης υλοποιείται ως native-web εφαρμογή στο λειτουργικό σύστημα TizenOS. Το λειτουργικό αυτό αποτελεί μια κοινή προσπάθεια της Samsung, Intel και του Linux Foundation[50] για την δημιουργία ενός ενιαίου λειτουργικού συστήματος για εκτέλεση σε συσκευές IoT, όπως smartphones, tablets, έξυπνες τηλεοράσεις (Smart TV) ακόμη και συσκευές αυτοκινήτων. Πρόκειται για ένα σύστημα εν γένει ανοικτού κώδικα -με το SDK του να είναι κλειστού κώδικα-, με τον πυρήνα(kernel) του TizenOS να έχει δανειστεί από το Linux, ενώ επιπλέον APIs και λειτουργικότητα αναπτύσσονται ad-hoc σε κάθε νέα κατηγορία συσκευών. Η διανομή και εγκατάσταση νέων εφαρμογών επι της συσκευής Samsung Gear S3 Frontier, πραγματοποιείται μόνο μέσα από το Gear App Store, που βρίσκεται κάτω από την ιδιοκτησία

της Samsung. Καθε εφαρμογή για να είναι διαθέσιμη στο ηλεκτρονικό κατάστημα, περνά από χρονοβόρα διαδικασία εγκρισης, διάρκειας τουλάχιστον μιας εβδομάδας. Η εφαρμογη που αναπτυξαμε στα πλαίσια αυτής της διπλωματικής, βρίσκεται επιτυχώς διαθέσιμη στο Gear App Store, ώστε παρόντες και μελλοντικοι χρήστες να έχουν πρόσβαση στην τελευταία έκδοση της εφαρμογής. Τέλος, όπως έχουμε αναφέρει στην ενότητα **2.2.4** η ανάπτυξη εφαρμογής στο περιβάλλον TizenOS, μπορεί να γίνει με δύο τρόπους. Είτε ως native-εφαρμογή σε χαμηλό επίπεδο χρησιμοποιώντας γλώσσα προγραμματισμού C, είτε ως web-εφαρμογή σε υψηλό επίπεδο χρησιμοποιώντας τεχνολογίες web και γλώσσα προγραμματισμού JavaScript. Αποφασίσαμε να υλοποιήσουμε την εφαρμογή μας ακολουθώντας το δευτερο σενάριο, χρησιμοποιώντας όλες τις διαθέσιμες και σχετικες λειτουργίες της συσκευής, όπως θα αναλύσουμε στην συνεχεια.

4.2 Υλοποίηση Διαδικτυακής Εφαρμογής Διαχείρισης Δεδομένων

Η υλοποίηση της διαδικτυακής εφαρμογής διαχείρισης δεδομένων και βιοσημάτων όπως έχουμε περιγράψει από δύο λειτουργικά εξαρτώμενα κομμάτια. Την γραφική διεπαφή(front-end) που εκτελείται στον browser του χρήστη και τον δικτυακό εξυπηρετητή(back-end) που εκτελείται στην υποδομή υπολογιστικού νέφους. Το πρώτο είναι υπεύθυνο για την γραφική αναπαράσταση της πληροφορίας με καλαίσθητο και λειτουργικό τρόπο καθώς επίσης και για την εκτέλεση αιτήματος προς τον εξυπηρετητή. Το δεύτερο είναι υπεύθυνο για την εξυπηρέτηση αυτών των αιτηματων και για την εκτέλεση υπολογισμών επι των δεδομένων των χρηστών.
Το front-end, αποτελείται κυρίως από την γραφική διεπαφή υλοποιημένη σε HTML η οποία παράγεται δυναμικά απο τον server ανάλογα με το αίτημα που εξυπηρετεί. Παράλληλα χρησιμοποιεί στατικούς πόρους όπως αρχεία CSS και JavaScript. Συγκεκριμένα ακολουθήσαμε την σχεδιαστική επιλογή MVC όπως ορίσαμε στην ενότητα **2.2.5.2.2**, ώστε κάθε αρχείο γραφικής διεπαφής να έχει το δικό του αρχείο JavaScript(view-controller), υπεύθυνο για την διαδραστική φύση της διεπαφής. Ο view-controller είναι υπεύθυνος για τον έλεγχο δεδομένων που εισαγονται απο τον χρήστη, την δυναμική παραγωγή γραφικού περιεχομένου και την εκτέλεση ασύγχρονων αιτημάτων στον εξυπηρετητή μέσω AJAX, όπως περιγράφηκε στην ενότητα **2.2.5.2.5**. Τέλος πρέπει να γίνει σαφές ότι ο JavaScript view-controller εκτελείται μόνο στον χώρο του χρήστη στον browser και η μόνη διασύνδεση με τα δεδομένα του, τις πληροφορίες αλλά και την κατάσταση του συστήματος, γίνεται μόνο με χρήση ασύγχρονων HTTP αιτημάτων.



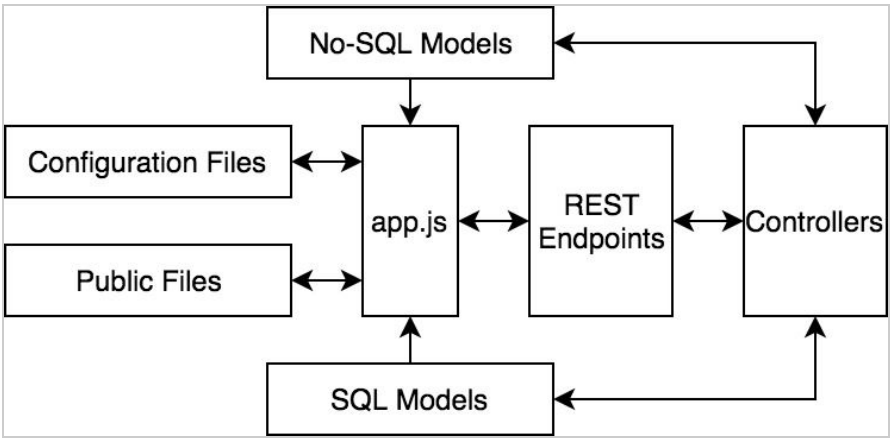
Σχήμα 34: Η Τοπολογία Επικοινωνίας Front-End και Back-End

4.2.1 Υλοποίηση του Κεντρικού Εξυπηρετητή(Cloud Server)

Συγκεντρώνοντας τα προαναφερθέντα, η εκτέλεση του κώδικα του εξυπηρετητη πραγματοποιείται στον πάροχο Heroku, ενώ όπως προαναφέραμε, η υλοποίηση του εξυπηρετητη πραγματοποιείται σε τεχνολογία Node.js, με την χρήση της βιβλιοθήκης(framework) ExpressJS. Για την μοντελοποίηση των σχεσιακών και μη μοντέλων χρησιμοποιούνται οι βιβλιοθήκες Sequelize και Mongoose, ενώ οι εξυπηρετητές των βάσεων δεδομένων εκτελούνται στους παρόχους Okeanos και mLab αντίστοιχα. Ο κώδικάς της εφαρμογής αναπτύχθηκε στο περιβάλλον Webstorm όπως παρουσιάσαμε στην ενότητα **2.2.6.2** και χρησιμοποιήθηκε το σύστημα ελέγχου έκδοσης git. Στην ακόλουθες ενότητες αναλύουμε τις λεπτομέρειες υλοποίησης του διαδικτυακού εξυπηρετητή.

4.2.1.1 Δομή Κώδικα Εφαρμογής

Ο κώδικας του διαδικτυακού εξυπηρετητή διαχωρίστηκε σε διαφορετικές λειτουργικές μονάδες με κριτήριο την αποφυγή επικάλυψης λειτουργιών και των διαχωρισμό των ανησυχιών(separation of concerns), όπως φαίνεται στο Σχήμα 35.



Σχήμα 35: Η Τοπολογία Δρομολόγησης Αιτημάτων στον Κεντρικό Εξυπηρετητή

Το αρχείο app.js αποτελεί την προταρχικότερη λειτουργία του συστήματος για τον λόγο αυτό αποτελεί και ξεχωριστή οντολογία στο σχήμα μας. Ο Node.js server εκτελεί αυτό το αρχείο και γνωρίζει ποια αιτήματα επιτρέπεται να δρομολογηθούν, ποιες βάσεις δεδομένων είναι διαθέσιμες και με ποιά σχεσιακά μοντέλα. Έχοντας πρόσβαση στις μεταβλητές περιβάλλοντος και τα αρχεία configuration, πιστοποιεί την συνδεσιμότητα με τις βάσεις δεδομένων και ενεργοποιεί τις διαθέσιμες θύρες για τα πρωτόκολλα επικοινωνιας απο και προς αυτόν. Τέλος εξυπηρετεί δημόσια ορατούς φακέλους, όπως φωτογραφίες αρχεία html και css, που βρίσκονται στην κάτω απο το “Public Files”.

```
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
app.use(logger('dev'));
app.use(express.json({limit: '50mb'}));
app.use(bodyParser.urlencoded({ limit: '50mb', extended: true }));
bodyParser.urlencoded({ extended: true });
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

//Connect with the MONGO database and setup Monggose models
require('./main_app/models/mongo_models/mongoose');
//Connect with the SQL database and setup Sequelize models
require('./main_app/models/sql_models/sequelize');

//Setup Application Routers
require('./main_app/routes/users_routes.js')(app);
require('./main_app/routes/biosignals_routes.js')(app);
require('./main_app/routes/devices_routes.js')(app);
require('./main_app/routes/admin_routes.js')(app);
```

Σχήμα 36: Μέρος του Αρχείου app.js

Η οντολογία REST Endpoints, αποτελείται από ένα σύνολο αρχείων το οποίο δηλώνει στον εξυπηρετητή ποια APIs να δρομολογούνται, απο ποιές HTTP μεθόδους και ποιες λειτουργίες απαιτούνται για την εξυπηρετηση τους. Για παράδειγμα, μπορούμε να δηλώσουμε ένα εισερχόμενο αίτημα POST, στην τοποθεσία: https://gearapp-server.herokuapp.com/users/login να εκτελεί την λειτουργία στον εξυπηρετητη που είναι υπεύθυνη για την πιστοποίηση του χρήστη προτού εισέλθει στο προσωπικό του προφίλ. Επίσης η υλοποίηση της δρομολόγησης στο περιβάλλον Node.js, υποστηρίζει και πολυμορφισμό όπως φαίνεται στο κάτωθι σχήμα. Παρατηρούμε ότι μπορούν να κληθούν διαφορετικές λειτουργίες(*users.loginPanel, users.login*) για αιτήματα που απευθύνονται στην ίδια τοποθεσία(*/users/login*) αλλά μέσω διαφορετικής HTTP μεθόδου (*get* και *post* αντίστοιχα).

```
var users = require('../controllers/users_controller.js');
var authentication = require('../controllers/authentication_controller.js');
```

```
module.exports = function (app) {
   app.route('/users/login').get(users.loginPanel);
   app.route('/users/login').post(users.login);
};
```

Σχήμα 37: Παράδειγμα Δήλωσης REST Endpoint στον Node.js Εξυπηρετητη και Εμφάνισης Πολυμορφισμού

Ο ορισμος των λειτουργιών που επιτελούνται αυτοτελως για κάθε εισερχόμενο αίτημα στα REST Endpoints, υλοποιούνται στην οντοτητα των controllers που παρουσιάζεται στο Σχήμα 38. Οι controllers αποτελούν ένα σύνολο αρχείων στο οποίο υλοποιούνται σε μορφή συναρτήσεων οι λειτουργίες που απαιτούνται από τον εξυπηρετητή για την διεκπεραίωση των αιτημάτων. Ακολουθει παράδειγμα υλοποίησης λειτουργικότητας για την εξυπηρέτηση του HTTP αιτήματός GET στην τοποθεσία: https://gearapp-server.herokuapp.com/users/login που όπως δηλώσαμε στο Σχήμα 37 θα εκτελεσει την συνάρτηση `users.loginPanel`.

```
exports.loginPanel = function (req, res) {
  res.status(200).sendFile(path.join(__dirname,'./../../views/user_login.html'));
};
```

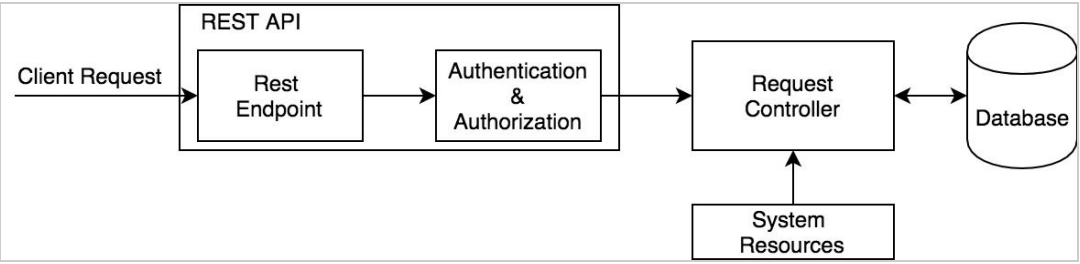
Σχήμα 38: Παράδειγμα Υλοποίησης Εξυπηρέτησης Λειτουργικότητας REST Endpoint στον Node.js

Τέλος οι οντότητες No-SQL και SQL models, υλοποιούν τις συνδέσεις με τους εξυπηρετητές βάσεων δεδομένων καθώς και τα μοντέλα δεδομένων που θα χρησιμοποιηθουν για την αποθήκευση στις βάσεις δεδομένων. Συσκεγκιμένες πληροφορίες και λεπτομερειες υλοποίηση θα παρουσιάσουμε στην ενότητα **4.2.2**.

4.2.1.2 Δρομολόγηση Αιτημάτων

Με την χρήση της βιβλιοθήκης ExpressJS, όπως αναφέραμε, επιτυγχάνουμε την αποτελεσματικη δρομολόγηση αιτημάτων(REST API) με απλό και ομοιόμορφο τρόπο. Η υλοποίηση έγινε ώστε τα αιτήματα που απευθύνει ο πελάτης στον εξυπηρετητή να χωρίζονται σε λογικά επίπεδα, ακολουθώντας την σχεδίαση της ενότητας **3.2.3.1.1**. Κατ αυτόν τον τρόπο σε πρώτο επίπεδο ο διαχωρισμός γίνεται σε τρεις κατηγορίες, αιτήματα διαχείρισης χρηστών(Σχήμα 40), αιτήματα διαχείρισης συσκευών(Σχήμα 42), και αιτήματα συγχρονισμού βιοσημάτων και δεδομένων κίνησης(Σχήμα 41). Σε δεύτερο επίπεδο η δρομολόγηση είτε προχωράει στην κατευθείαν κλήση της μεθόδου εξυπηρέτησης του αιτήματος API μέσω του request controller, είτε παρεμβάλλεται ενδιάμεση επεξεργασία του αιτήματος προτού εξυπηρετηθεί. Η δευτερη αντιμετώπιση στα πλαίσια της υλοποίησης με Node.js και ExpressJS ονομάζεται διαμεσολαβητής (Middleware) και αποτελεί τεχνική κατα την οποία τα αιτηματα επεξεργάζονται από συναρτησεις

εκτελούμενες σειριακα πριν την τελική τους εξυπηρέτηση. Όπως παρουσιάζεται στο κάτωθι Σχήμα 39, στην υλοποίηση μας η δρομολόγηση ξεκινά με το εισερχόμενο αίτημα στα REST Endpoints, στην συνέχεια διέρχεται από το Authentication & Authorization Middleware και στην συνέχεια εξυπηρετείται τελικός από τον request controller.



Σχήμα 39: Η Τοπολογία Δρομολόγησης Αιτημάτων στον Κεντρικό Εξυπηρετητή

Ο στόχος της λειτουργίας διαμεσολαβησης για την επαλήθευση ταυτότητας και εξουσιοδότησης του αιτήματος εκτελείται ώστε τα εισερχόμενα αιτήματα να έχουν πρόσβαση μόνο στους πόρους που έχουμε προδιαγράψει διασφαλίζοντας τα στοιχεία της εμπιστευτικότητας και ιδιωτικότητας του συστήματος. Παράδειγμα χρήσης του Authentication & Authorization Middleware αποτελεί η εκτέλεση της εντολής `app.route('/users/panel').get(authentication.verifyUser,users.indexPanel);` όπως παρουσιάζεται στο Σχήμα 40. Η εκτέλεση της συνάρτησης `authentication.verifyUser`, ελέγχει αν το εισερχόμενο αίτημα περιέχει τα προδιαγεγραμένα πιστοποιητικά ώστε να προέρχεται από χρήστη του συστήματος. Αν η εκτέλεση της αποφανθεί για την εγκυρότητα του αιτήματος τότε θα εκτελέσει ο request controller, που θα εξυπηρέτηση το αίτημα μέσω της κλήσης της συνάρτησης users.indexPanel. Στα ακόλουθα σχήματα 40, 41, 42, παρουσιάζουμε την υλοποίηση της δρομολόγησης όλων των REST Endpoints που υποστηρίζει το σύστημα μας, βασιζόμενοι στην περιγραφή του Σχήματος 28 του προηγουμένου κεφαλαίου.

```
module.exports = function (app) {
    app.route('/users/panel').get(authentication.verifyUser, users.indexPanel);
    app.route('/users/register').get(users.registrationPanel);
    app.route('/users/register').post(users.register);
    app.route('/users/login').get(users.loginPanel);
    app.route('/users/login').post(users.login);
    app.route('/users/update').post(authentication.verifyUser, users.update);
    app.route('/users/delete').post(authentication.verifyUserOrAdmin, users.delete);
    app.route('/users/verifyEmail').get(users.verifyEmail);
    app.route('/users/changeStatus').post(authentication.verifyAdmin, users.changeStatus);
```

```
    app.route('/users/requestChangePassword').get(users.requestRecoverPassword);
    app.route('/users/requestChangePasswordPanel').get(users.requestRecoverPasswordPanel);
    app.route('/users/changePassword').post(users.recoverPassword);
    app.route('/users/changePasswordPanel').get(users.recoverPasswordPanel);
    app.route('/users/getAll').post(authentication.verifyAdmin, users.getall);
    app.route('/admin/index').get(authentication.verifyAdmin, admin.index);
};
```

Σχήμα 40: Η Δρομολόγηση Αιτημάτων Χρηστών

```
module.exports = function (app) {
    app.route('/biosignals/upload').post(authentication.verifyDevice,
biosignals.extractBiosignalsFromFile, biosignals.toBulk, biosignals.upload);
    app.route('/biosignals/retrieve').post(authentication.verifyAdmin, biosignals.retrieve)
};
```

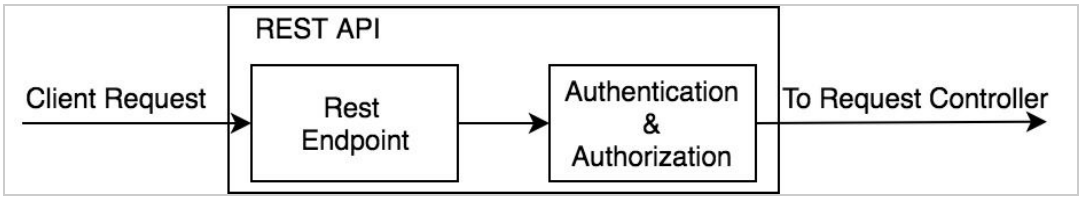
Σχήμα 41: Η Δρομολόγηση Αιτημάτων Βιοσημάτων και Δεδομένων Κίνησης

```
module.exports = function (app) {
    app.route('/devices/register').post(devices.register);
    app.route('/devices/show').post(authentication.verifyUserOrAdmin, devices.show);
    app.route('/devices/delete').post(authentication.verifyUser, devices.delete);
};
```

Σχήμα 42: Η Δρομολόγηση Αιτημάτων Συσκευών

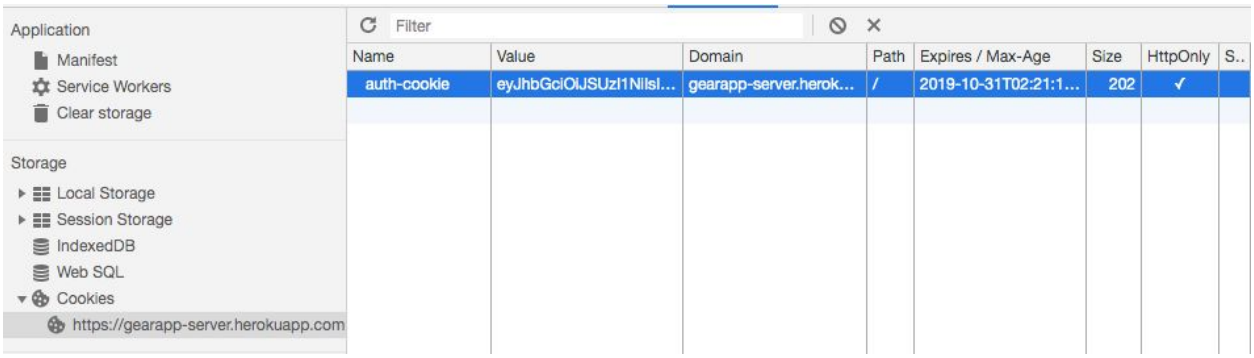
4.1.1.3 Επαλήθευση και Εξουσιοδότηση Αιτημάτων

Η επαλήθευση και εξουσιοδότηση των αιτημάτων, όπως περιγραψαμε υλοποιείται μέσω του Authentication & Authorization Middleware.

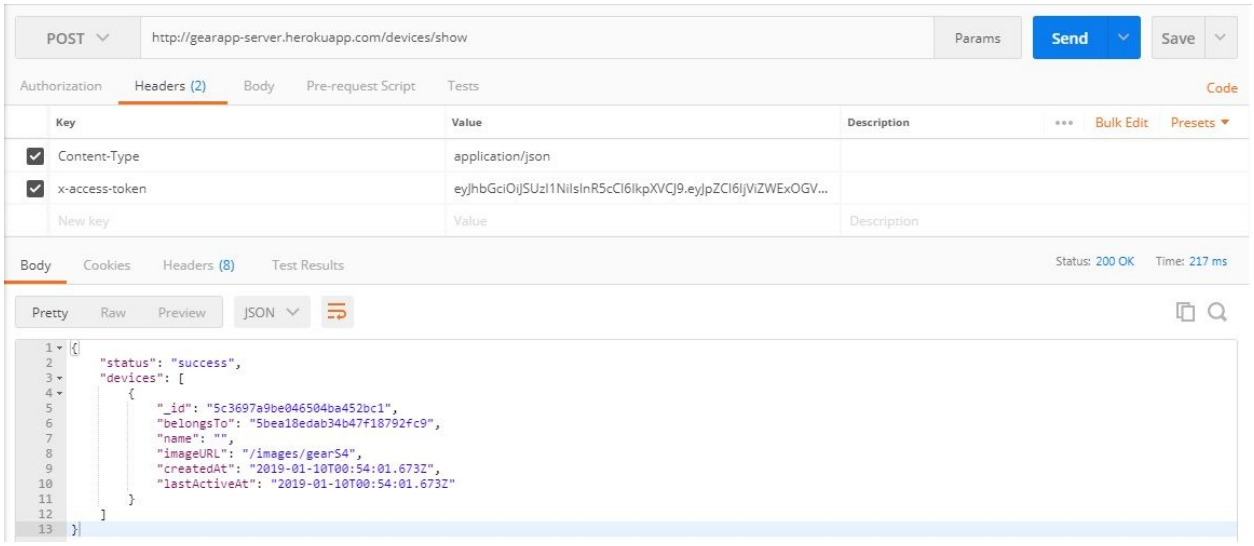


Σχήμα 43: Η Τοπολογία Επαλήθευσης Ταυτότητας και Εξουσιοδότησης Αιτημάτων

Συγκεκριμένα κάθε αίτημα που εκτελεί HTTP request από τον client περιέχει ένα αναγνωριστικό ταυτότητας και εξουσιοδότησης json-web-token το οποίο ακολουθεί την λειτουργία και τους κανόνες δημιουργίας του, που έχουμε περιγράψει στην ενότητα **3.2.2.4.2**. Τα αιτήματα που εκτελούνται από τον browser του χρήστη αποστέλλονται μαζί με το cookie επαλήθευσης και εξουσιοδότησης αιτήματος, που περιέχει το αναγνωριστικό x-access-token, όπως φαίνεται στο Σχήμα 44. Τα αιτήματα που προέρχονται από τις συσκευές των χρηστών περιέχουν το ίδιο αναγνωριστικό στην επικεφαλίδα(header) του αιτήματος όπως φαίνεται στο Σχήμα 45.



Σχήμα 44: Cookie για την Πιστοποίηση Αιτηματος προς τον Εξυπηρετητή

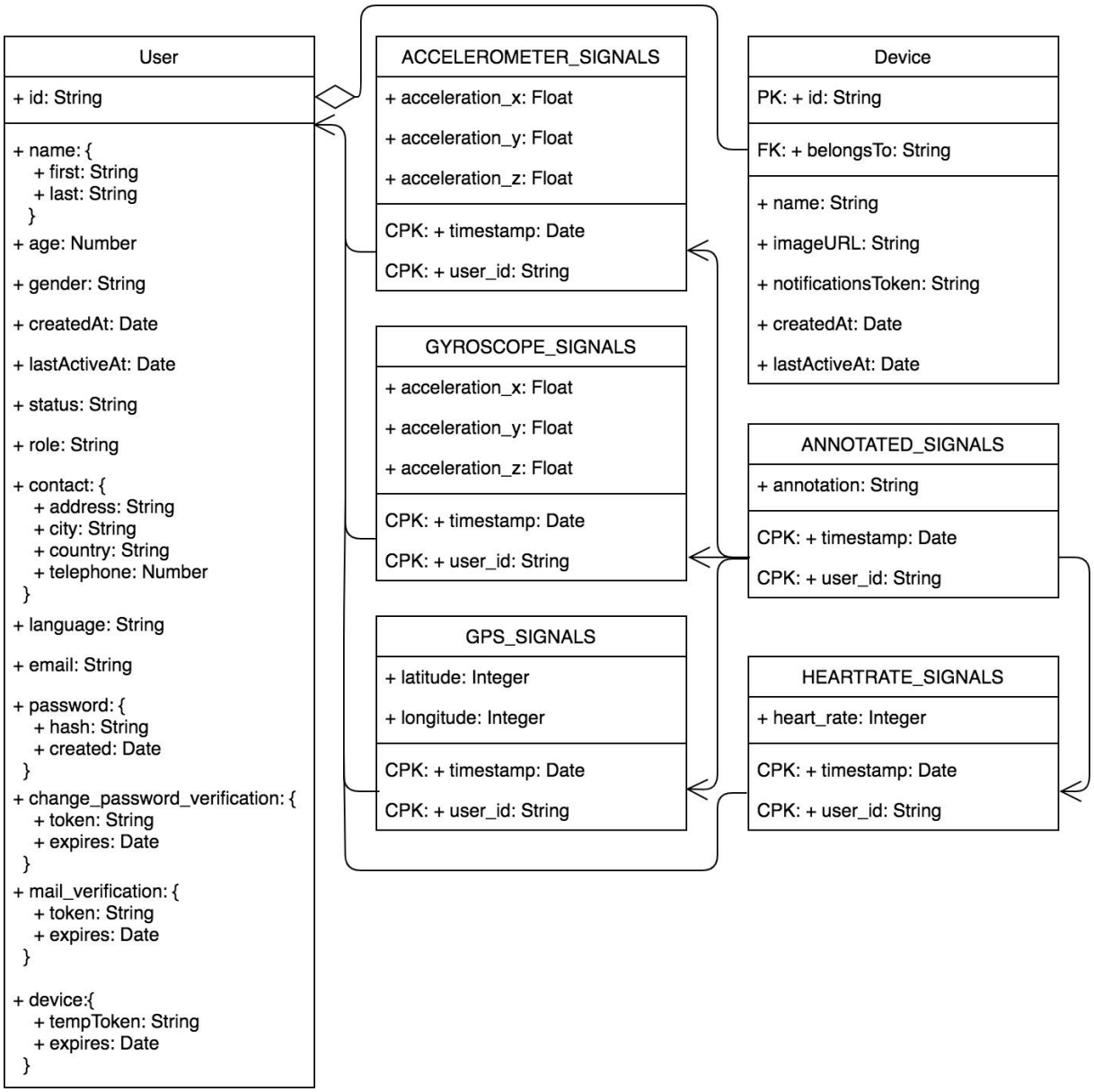


Σχήμα 45: X-access-token για την Πιστοποίηση Αιτηματος προς τον Εξυπηρετητή

4.2.2 Υλοποίηση της Αποθήκευσης Δεδομένων

Όπως αναφέραμε στην ενότητα **3.2.3.2** σε υψηλό επίπεδο σχεδίασης αντιμετωπίζουμε την βάση δεδομένων σαν ενιαία μοναδική οντότητα. Η φύση όμως των προς-αποθήκευση δεδομένων όπως εξετάσαμε επιβάλλει την περαιτέρω μοντελοποίηση με συνδυασμό σχεσιακου και μη-σχεσιακού μοντελου αποθηκευσης.

Στο Σχήμα 46 περιγράφει το σχήμα(schema) που ακολουθεί εξ ολοκλήρου η βάση δεδομένων. Οι δομές που αντιστοιχούν στις οντότητες: Accelerometer_Signals, Gyroscope_Signals, GPS_Signals, Heartrate_Signals και Annotated_Signals υλοποιούνται απο πίνακες SQL, ενώ οι δομές που αντιστοιχούν στις οντότητες: User και Device υλοποιούνται απο συλλογές MongoDB. Για την συσχέτιση των δύο μοντέλων αρχιτεκτονικής, χρησιμοποιούνται primary και foreign keys απο την σχεσιακή σχεδίαση βάσεων δεδομένων, ενώ στο σχήμα φαίνονται οι συσχετίσεις one-to-many, οι εξαρτήσεις που παρουσιάζουν οι δομές καθώς και οι τύποι των προς αποθήκευση δεδομένων.



Σχήμα 46: Το Σχήμα(Schema) της Αποθήκευσης Δεδομένων του συστήματος

Πιο συγκεκριμένα στο σχεσιακό μοντέλο χρησιμοποιούμε τον συνδυασμό timestamp και user_id ως composite primary key(CMP) ώστε να διαχωρίσουμε μοναδικά τα δεδομένα των πινάκων του μοντέλου. Στο μη-σχεσιακό χρησιμοποιουμε μοναδικο user_id και device_id παραγόμενα αυτοματοποιημενα από το σύστημα υλοποίησης της βάσης δεδομένων.
Η υλοποίηση του σχεσιακού μοντέλου αποθήκευσης έγινε με τεχνολογία SQL, ενώ του μη-σχεσιακού μοντέλου με τεχνολογία MongoDB, με τις λεπτομέρειες υλοποίησης να ακολουθούν παρακάτω.

4.2.2.1 Υλοποίηση της Βάσης Δεδομένων SQL

Επιλέξαμε την υλοποίηση του σχεσιακού μοντέλου βάσης δεδομένων να πραγματοποιηθεί με χρήση της τεχνολογίας SQL. Η περιγραφή των δομών εντός της βάσης γίνεται με έγγραφες σε πίνακες, όπως περιγράψαμε στην ενότητα **3.2.3.2.3**. Τα δεδομένα του επιταχυνσιομέτρου, γυροσκοπίου, G.P.S και καρδιακού ρυθμου, αποθηκεύονται σε ξεχωριστούς πίνακες μαζί με την χρονοσήμανση δειγματοληψίας τους καθώς και το αναγνωριστικό χρήστη που τους αντιστοιχεί. Επιπροσθέτως υλοποιήθηκε και πίνακας και η αντίστοιχη υποδομή του στον REST εξυπηρετητή ώστε να διατηρούνται και επισημάνσεις/σχόλια για τα συλλεχθέντα δεδομένα σε περίπτωση που είναι διαθέσιμες. Αυτό κρίθηκε σκόπιμο, σε περίπτωση που χρειαστεί μελλοντικά να παράγουμε μέσω της πλατφορμας σύνολα δεδομένων προοριζόμενα για επιβλεπόμενη εκπαίδευση μοντέλων μηχανικής μάθησης.

Χρησιμοποιηθηκε η βιβλιοθήκη Sequelize[52] ως database object relational mapper η οποία είναι πλήρως συμβατή στο περιβάλλον JavaScript και Node.js. Η υλοποίηση μας χρησιμοποιεί εξυπηρετητή PostgreSQL σε διαφορετικό περιβάλλον εκτέλεσης από αυτό του κεντρικού εξυπηρετητή. Συγχρόνως, η βιβλιοθήκη υποστηρίζει και τις διαλέκτους, MySQL, MariaDB, SQLite και MSSQL πράγμα που γενικοποιεί την χρήση της και σε άλλες υλοποιήσεις σχεσιακής μοντελοποίησης μελλοντικά.

```
//Connection Set-Up
const sequelize = new Sequelize(config.postgres.DB_URI,{
   dialect:  'postgres',
   dialectOptions: {
       ssl: true
   },
   protocol: 'postgres',
   freezeTableName: true,
   operatorsAliases: false,
   logging:  false
});
```

```
//Register models
var AccelerometerModel = AccelerometerSignalModel(sequelize, Sequelize);
var GyroscopeModel = GyroscopeSignalModel(sequelize, Sequelize);
var HeartrateModel = HeartrateSignalModel(sequelize, Sequelize);
var GPSModel = GPSSignalModel(sequelize, Sequelize);
var AnnotationModel = AnnotationSignalModel(sequelize, Sequelize);
//One-To-One relationships
AccelerometerModel.hasOne(AnnotationModel);
AnnotationModel.belongsTo(AccelerometerModel);
GyroscopeModel.hasOne(AnnotationModel);
AnnotationModel.belongsTo(GyroscopeModel);
HeartrateModel.hasOne(AnnotationModel);
AnnotationModel.belongsTo(HeartrateModel);
GPSModel.hasOne(AnnotationModel);
AnnotationModel.belongsTo(GPSModel);
//Debug connection state
sequelize.sync({
   pool: { idle: 10000, acquire: 10000 }
}).then(function() {
   console.log("Connected successfully to project SQL database");
}).catch(function(error) {
   console.error('Error connecting to project database\n\t' + error);
});
module.exports = {
   AccelerometerModel,
   GyroscopeModel,
   HeartrateModel,
   GPSModel,
   AnnotationModel
};
```

Σχήμα 47: Καταχώρηση σχημάτων(schemas) και Διαχείριση Σύνδεσης με τον Εξυπηρετητη Βάσης Δεδομένων SQL - Κομμάτι Αρχείου sequelize.js

Στο άνωθι κομμάτι κώδικα, παρατηρούμε τις λεπτομέρειες υλοποίησης χρησιμοποιώντας την βιβλιοθήκη Sequelize. Συγκεκριμένα, αρχικά συνδεόμαστε στον απομακρυσμένο εξυπηρετητή

PostgreSQL, χρησιμοποιώντας το URI που περιέχει τα διαπιστευτήρια σύνδεσης σε αυτόν απο τις μεταβλητές περιβάλλοντος. Εφόσων καθορίσουμε τις λεπτομέρειες σύνδεσης, στην συνεχεια δηλώνουμε τα μοντέλα σχεσιακής λογικής που θα χρησιμοποιήσουμε καθώς και τις μεταξύ τους συσχετίσεις(hasOne, belongsTo).

Στα ακόλουθα κομμάτια κώδικα, περιγράφονται οι υλοποιήσεις των σχημάτων σχεσιακής λογικής(schemas) που αφορούν την αποθήκευση των δεδομένων κινήσεων και βιοσημάτων όπως αποθηκεύονται στην βάση δεδομένων. Ενδεικτικά σημειώνουμε ότι:

- Στα μοντέλα δεδομένων κινήσεων χρήστη και βιοσημάτων έχουμε ακολουθήσει την περιγραφή των δεδομένων με βάση το πρότυπο mHealth, όπως έχουμε αναφέρει στην ενότητα **2.2.5.4.1**.
- Σε κάθε πεδίο με την βοήθεια της λειτουργικότητας της βιβλιοθηκης Sequelize, επιβάλουμε τον τύπο των δεδομένων και το εύρος τους που θέτουμε ως αποδεκτό καθώς και αν επιτρέπετε να αφεθεί κενό ή όχι κατα την αποθήκευση του.
- Σε κάθε μοντέλο ορίζουμε το όνομα του και το ονομα του πίνακα που ανήκει καθώς η βιβλιοθήκη δημιουργεί αυτοματοποιημένα τους πίνακες που χρειάζονται και τις μεταξύ των αντικειμένων συσχετίσεις στην διάλεκτο SQL που έχουμε ορίσει.

```
module.exports = function(sequelize, DataTypes) {
var AccelerometerSignalModel =
sequelize.define('AccelerometerSignal', {
 //According to Open mHealth standard
    acceleration_x: {
      type: DataTypes.FLOAT,
      allowNull: false
    },
    acceleration_y: {
      type: DataTypes.FLOAT,
      allowNull: false
    },
    acceleration_z: {
      type: DataTypes.FLOAT,
      allowNull: false
    },
    timestamp: {
      type: DataTypes.DATE,
      allowNull: false,
      primaryKey: true
    },
    user_id: {
      type: DataTypes.STRING,
      allowNull: false,
      primaryKey: true,
    }
  }, {
    tableName: 'ACCELEROMETER_SIGNALS',
    id: false,
    updatedAt: false
  });
  return AccelerometerSignalModel;
};
```

Σχήμα 48: Το Σχήμα(schema) Περιγραφής των Δεδομένων Επιταχυνσιομέτρου στην Βάση SQL

```
module.exports = function(sequelize, DataTypes) {
var GyroscopeSignalModel =
sequelize.define('GyroscopeSignal', {
 //According to Open mHealth standard
    acceleration_x: {
      type: DataTypes.FLOAT,
      allowNull: false
    },
    acceleration_y: {
      type: DataTypes.FLOAT,
      allowNull: false
    },
    acceleration_z: {
      type: DataTypes.FLOAT,
      allowNull: false
    },
    timestamp: {
      type: DataTypes.DATE,
      allowNull: false,
      primaryKey: true
    },
    user_id: {
      type: DataTypes.STRING,
      allowNull: false,
      primaryKey: true,
    }
  }, {
    tableName: 'GYROSCOPE_SIGNALS',
    id: false,
    updatedAt: false
  });
  return GyroscopeSignalModel;
};
```

Σχήμα 49: Το Σχήμα(schema) Περιγραφής των Δεδομένων Γυροσκοπίου στην Βάση SQL

```
module.exports = function(sequelize, DataTypes) {
var HeartrateSignalModel =
sequelize.define('HeartrateSignal', {
//According to Open mHealth standard
    heart_rate: {
      type: DataTypes.INTEGER,
      allowNull: false
    },
    timestamp: {
      type: DataTypes.DATE,
      allowNull: false
    },
    user_id: {
      type: DataTypes.STRING,
      allowNull: false
    }
  }, {
    tableName: 'HEARTRATE_SIGNALS',
    id: false,
    updatedAt: false
  });
  return HeartrateSignalModel;
};
```

Σχήμα 50: Το Σχήμα(schema) Περιγραφής των Δεδομένων Καρδιακού Παλμού στην Βάση SQL

```
module.exports = function(sequelize, DataTypes) {
    timestamp: {
```

```
var GPSSignalModel =
sequelize.define('GPSSignal', {
    latitude: {
      type: DataTypes.INTEGER,
      allowNull: false,
      validate: { min: -90, max: 90 }
    },
    longitude: {
      type: DataTypes.INTEGER,
      allowNull: false,
      validate: { min: -90, max: 90 }
    },
      type: DataTypes.DATE,
      allowNull: false
    },
    user_id: {
      type: DataTypes.STRING,
      allowNull: false
    }
  }, {
    tableName: 'GPS_SIGNALS',
    id: false,
    updatedAt: false
  });
  return GPSSignalModel;
};
```

Σχήμα 51: Το Σχήμα(schema) Περιγραφής των Δεδομένων G.P.S στην Βάση SQL

```
module.exports = function(sequelize, DataTypes) {
  var AnnotationSignalModel =
sequelize.define('AnnotationModel', {
    annotation: {
      type: DataTypes.STRING,
      allowNull: false
    },
    timestamp: {
      type: DataTypes.DATE,
      allowNull: false,
      primaryKey: true
    },
    user_id: {
      type: DataTypes.STRING,
      allowNull: false,
      primaryKey: true
    }
  }, {
    tableName: 'ANNOTATED_SIGNALS',
    id: false,
    createdAt: false,
    updatedAt: false
  });
  return AnnotationSignalModel;
};
```

Σχήμα 52: Το Σχήμα(schema) Περιγραφής της Επισήμανσης στην Βάση SQL

4.2.2.2 Υλοποίηση της Βάσης Δεδομένων MongoDB

Επιλέξαμε την υλοποίηση του μη-σχεσιακού μοντέλου βάσης δεδομένων να πραγματοποιηθεί με χρήση της τεχνολογίας MongoDB. Η περιγραφή των δομών εντός της βάσης με έγγραφα json, όπως περιγράψαμε στην ενότητα **3.2.3.2.2** μας δίνει την δυνατότητα να αναπαραστήσουμε

πληρέστερα πολυπαραμετρικά και ετερογενή δεδομένα. Έτσι τα στοιχεία των χρηστών και των συσκευών τους, περιγράφονται στα ακόλουθα σχήματα json, ενώ η υποκείμενη τεχνολογία επιτρέπει ευελιξία στην αποθήκευση και ταχύτητα στην ομαδοποίηση και την επεξεργασία τους.

Δεδομένου ότι η υλοποίηση του κεντρικού εξυπηρετητή έχει γίνει στο περιβάλλον JavaScript και Node.js, χρησιμοποιούμε την βιβλιοθήκη MongooseJS[53] ως database object relational mapper. Παρότι η βιβλιοθήκη απευθύνεται μόνο σε υλοποιήσεις MongoDB και δεν γενικοποιεί την χρήση της και σε άλλες υλοποιήσεις μη-σχεσιακής μοντελοποίησης, εντούτοις χρησιμοποιήθηκε για την μοντελοποίηση αντικειμένων που προσφέρει καθώς και τα υπόλοιπα χαρακτηριστικά συνδεσιμότητας, ασφάλειας και βελτιστοποίησης που προσφέρει η σχεσιακή αντιστοίχιση αντικειμένων όπως ήδη έχουμε συζητήσει, στην ενότητα **3.2.3.2.4**.

```
//Connection Set-Up
mongoose.connect(config.mongo.DB_URI, { useNewUrlParser: true });
mongoose.connection.on('connected', function () {
   console.log("Connected successfully to project MONGODB database");
});
mongoose.connection.on('error',function (error) {
   console.error('Error connecting to project database\n\t' + error);
});

//setup mongoose models
require('./user_model.js');
require('./device_model.js');
```

Σχήμα 53: Καταχώρηση σχημάτων(schemas) και Διαχείριση Σύνδεσης με τον Εξυπηρετητη Βάσης Δεδομένων MongoDB

Ακολουθεί στο Σχήμα 54 η υλοποίηση του σχήματος που περιγραφει τις εγγραφές τύπου User, όπως αποθηκεύονται στην βάση δεδομένων. Ενδεικτικά σημειώνουμε ότι:

- Το πεδίο name, κάνει πλήρη χρήση της εκφραστικότητας που προσφέρει η δομή json, αφού βλέπουμε ότι αποτελείται από first και last εμφωλευμένα. Επίσης παρατηρούμε ότι με την βοήθεια της λειτουργικότητας της βιβλιοθηκης MongooseJS, επιβάλουμε τον τύπο των δεδομένων που θέτουμε ως αποδεκτό.
- Το πεδίο password, είναι απαραίτητο για την επιτυχή καταχώρηση ενός αντικειμένου User στην βάση δεδομένων. Επίσης, η αποθήκευση του πεδίου στην βάση δεδομένων δεν θα πραγματοποιηθεί στο αρχικό μη-κρυπτογραφημένο κείμενο αλλά θα αποθηκεύσουμε την τιμή του μετα απο hashing, όπως θα εξηγήσουμε στην συνέχεια.
- Το πεδίο status, μπορεί να λάβει μόνο συγκεκριμένες τιμές που έχουμε ορίσει εμείς εκ των προτέρων.

var UserSchema = new Schema({	telephone: {	lastActiveAt: {

```
    id: Schema.ObjectId,
    name: {
        first: {
            type: String,
            lowercase: true,
            required: true
        },
        last: {
            type: String,
            lowercase: true,
            required: true
        }
    },
    age: {
        type: Number,
        required: true
    },
    gender: {
        type: String,
        enum: ['male', 'female', 'other',
'n/a'],
        default: 'n/a'
    },
    contact: {
        address:{
            type: String,
            lowercase: true
        },
        city: {
            type: String,
            lowercase: true
        },
        country: {
            type: String,
            lowercase: true
        },
            type: Number
        }
    },
    language: {
        type: String,
        default: 'EN_US'
    },
    email: {
        type: String,
        unique: true,
        required: true
    },
    password: {
        hash: {
            type: String,
            required: true
        },
        created: {
            type: Date,
            default: Date.now
        }
    },
    change_password_verification: {
        token: {
            type: String
        },
        expires: {
            type: Date
        }
    },
    createdAt: {
        type: Date,
        default: Date.now,
        required: true
    },
        type: Date,
        default: Date.now,
    },
    status: {
        type: String,
        enum: ['pending', 'verified',
'blocked', 'deleted','suspicious'],
        default: 'verified'
    },
    role: {
        type: String,
        enum: ['user', 'admin'],
        default: 'user'
    },
    mail_verification: {
        token: {
            type: String
        },
        expires: {
            type: Date
        }
    },
    device:{
        tempToken:{
            type: String,
            required: true
        },
        expires: {
            type: Date
        }
    }
});
UserSchema.plugin(mongooseValid
ator);
mongoose.model('User',
UserSchema);
```

Σχήμα 54: Το Σχήμα(schema) Περιγραφής του Χρήστη της Εφαρμογής στην Βάση MongoDB

Ομοίως ακολουθεί η υλοποίηση του σχήματος που περιγραφει την συσκευή smartwatch του χρήστη. Ενδεικτικά σημειώνουμε ότι:

- Το πεδίο belongsTo, αναφέρεται στο user_id -το χαρακτηριστικό δηλαδη που ταυτοποιεί μοναδικά τον χρήστη στην βάση δεδομένων- και επομένως μεσα απο αυτή την παραπομπή κάθε συσκευή έχει μοναδικό χρήστη και κάθε χρήστης έχει την κυριότητα μίας ή παραπάνω συσκευής.
- Το πεδίο name, επιτρέπει στον χρήστη να ορίσει προσαρμοσμένα το όνομά της αρεσκείας του για την συσκευή, για πληρέστερη εμπειρία χρήσης.

- Το πεδίο imageURL, επιτρέπει στους διαχειριστές του δικτύου να ορίσουν το γραφικό εικονίδιο που θα συνοδευει την συσκευή όταν αυτή προβάλετε στην διαδικτυακή διεπαφή.

```
var DeviceSchema = new Schema({
   id: Schema.ObjectId,

   belongsTo: {
      type: String,
      required: true
   },

   name: {
      type: String,
      required: false
   },

   imageURL: {
      type: String,
      required: false
   },

   notificationsToken: {
      type: String,
      required: false
   },

   createdAt: {
      type: Date,
      default: Date.now,
      required: true
   },

   lastActiveAt: {
      type: Date,
      default: Date.now,
   },
});
DeviceSchema.plugin(mongooseValidator);
mongoose.model('Device', DeviceSchema);
```

Σχήμα 55: Το Σχήμα(schema) Περιγραφής του Συσκευής του Χρήστη της Εφαρμογής στην Βάση MongoDB

4.2.3 Υλοποίηση των Απαιτήσεων Απλών Χρηστών

Στην ενότητα αυτή θα παρουσιάσουμε αναλυτικα την υλοποίηση της προδιαγεγραμμένης λειτουργικότητας των απλών χρηστών του συστήματος, όπως την παρουσιάσαμε στην ενότητα **3.1.2** των λειτουργικών απαιτήσεων του συστήματος. Για κάθε λειτουργική απαίτηση θα παρουσιάσουμε το σενάριο χρήσης που επιβάλλει, κύρια σημεία του κώδικα καθώς και στιγμιότυπα από της γραφικές διεπαφές με τις οποίες αλληλεπιδρούν οι χρήστες.

4.2.3.1 Προβολή Αρχικής Σελίδας

Όπως αναφέραμε στην ενότητα, **4.1.2** ο εξυπηρετητής της διαδικτυακής εφαρμογής φιλοξενείται στον πάροχο υπολογιστικού νέφους Heroku. Έτσι η διεύθυνση στην οποία βρίσκεται η κεντρική σελίδα της υπηρεσίας είναι η: **https://gearapp-server.herokuapp.com**
Εκτελώντας ένα αίτημα HTTP GET από τον browser μας μεταβαίνουμε στην κεντρική σελίδα όπως φαίνεται στο ακόλουθο Σχήμα 56.

Παρατηρούμε ότι οι μόνες πληροφορίες που παρουσιάζονται είναι κυρίως ενημερωτικές και μη-φιλικές προς τον χρήστη. Αυτό συμβαίνει καθότι αποφασίστηκε το σύστημα μας να μην είναι φιλικό σε εξωτερικούς χρήστες, χρήστες δηλαδη που μπορεί τυχαία να βρεθούν στην ιστοσελίδα μας και να δοκιμάσουν το σύστημα από περιέργεια. Ο στόχος της πλατφόρμας

συνολικά είναι να παρέχει την προδιαγεγραμμένη λειτουργικότητα σε χρήστες που έχουν προμηθευτεί την απαραίτητη συσκευή smartwatch και είναι σύμφωνοι με τους ερευνητικούς και επιχειρησιακούς της στόχους. Τέλος, η σελίδα αυτή προβάλλεται και όταν εκτελεστούν αιτήματα προς τον εξυπηρετητή τα οποία δεν δρομολογούνται από την παρούσα υλοποίηση. Επιλέγοντας να μην εμφανίζεται μήνυμα λάθους σε τέτοιες περιπτώσεις, αποφεύγουμε την αποκάλυψη πληροφοριών σε κακόβουλους χρήστες σχετικά με την γλώσσα προγραμματισμού και αρχιτεκτονικής υλοποίησης του server.

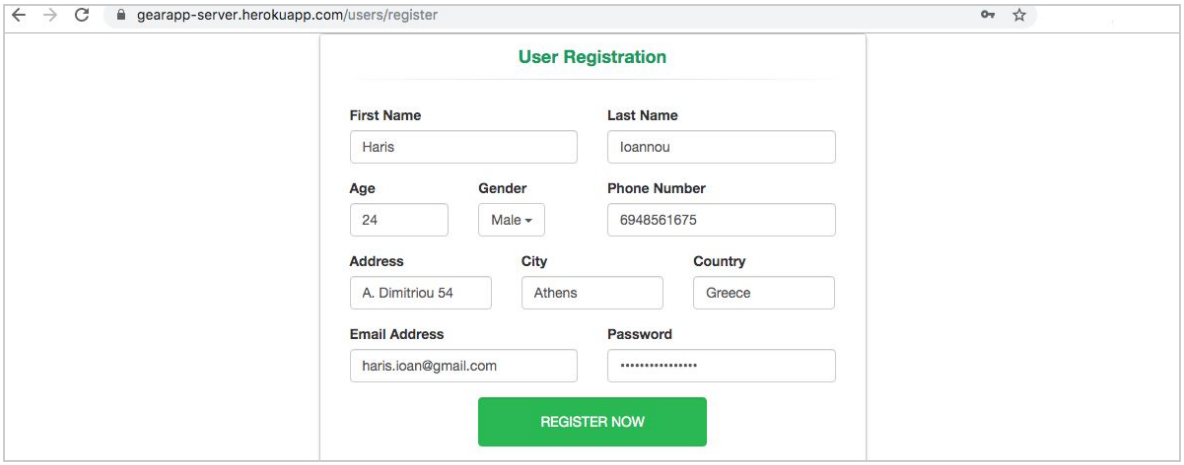
Σχήμα 56: Η Κεντρική Σελίδα της Διαδικτυακής Εφαρμογής

4.2.3.2 Εγγραφή Νέου Χρήστη

Η εγγραφή του χρήστη αποτελεί απαραιτητη προυποθεση για την χρήση της πλατφόρμας συνολικά. Ετσι λοιπόν για την καταχώρηση των στοιχείων και πληροφοριών του, ο χρήστης πραγματοποιεί αίτημα HTTP GET από τον browser του στην διεύθυνση:

https://gearapp-server.herokuapp.com/users/register

Ο χρήστης θα πρέπει εκεί να συμπληρώσει στην γραφική διεπαφή, τις απαιτούμενες πληροφορίες, όπως φαίνεται στο Σχήμα 57.



Σχήμα 57: Η Σελίδα Εγγραφής Νέου Χρήστη στην Διαδικτυακή Εφαρμογή

Μόλις ο χρήστης συμπληρώσει επιτυχώς τα στοιχεία και πατήσει το κουμπί register, ο view-controller που εκτελείται στο front-end θα εκτελέσει ασύγχρονα ένα HTTP POST αίτημα στον εξυπηρετητή στην διεύθυνση https://gearapp-server.herokuapp.com/users/register όπως φένεται στο ακόλουθο Σχήμα 58.

```
$("#register_btn").on("click", function (e) {
    e.preventDefault();
    let userData = getAndValidateUserInput();
    $.post("/users/register", userData).done(function (response) {
        window.location.href = "/users/panel";
    }).fail(function (err) {
        if (err !== undefined && err.description !== undefined &&
          err.description.details !== undefined) {
          alert(err.description.details);
        } else {alert("Something went wrong");
});
```

Σχήμα 58: View-controller για την Εκτέλεση Αιτήματος Register User

Παρατηρούμε ότι ο άνωθι κώδικας είναι εμφωλιασμένος και αυτό γιατί το σώμα του εκτελείται μόνο όταν πατηθεί το κουμπί register με id: `"#register_btn"`, συγκεκριμένα η πρώτη γραμμή του κώδικα, δηλώνει στην γλώσσα JavaScript να περιμένει για να πατηθεί το κουμπί προτού εκτελεστει ο κώδικας. Αυτή η λειτουργία, αποτελεί το πρώτο βασικό πλεονέκτημα της JavaScript, αφού χρησιμοποιώντας το event-driven μοντέλο της, δεν χρειάζεται εμεις να ελέγχουμε διαρκώς την κατάσταση του register κουμπιού ώστε να ανιχνεύσουμε το πάτημα του.

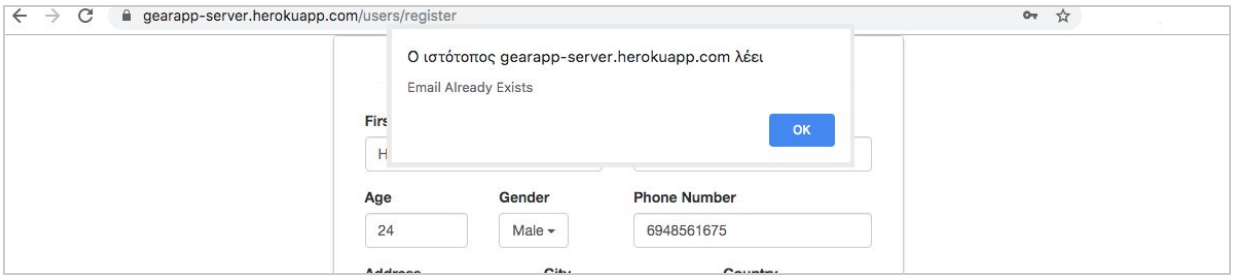
Στην συνέχεια αφού πατηθεί το κουμπί και το ανιχνεύσουμε, το επόμενο σημαντικό σημείο του κώδικα είναι η κλήση της συνάρτησης `getAndValidateUserInput()` η οποία λαμβάνει τα δεδομένα που εισήγαγε ο χρήστης και ελεγχει την πληροτητα, την μορφή και τον τύπο τους. Σε αυτό το στάδιο προλαμβάνουμε πρωτογενώς λάθη εισαγωγής από χρήστη όπως μη-εγκυρη ηλικία, email τηλέφωνο κ.α. Στον server βέβαια, θα ξαναελέγξουμε τα χαρακτηριστικά αυτά καθώς και αρκετά περισσότερα όπως θα περιγράψουμε στην συνέχεια για λόγους ασφαλείας.

Μόλις ο ελεγχος ολοκληρωθεί, τα δεδομένα δομημένα σε μορφή JSON περιεχόμενα στην μεταβλητή `userData,` αποστέλλονται στον server όπως περιγράψαμε μέσω της κλήσης της συνάρτησης `$.post`. Όπως παρατηρούμε η συνάρτηση λαμβάνει ως όρισματα: τα προς-αποστολή εισαχθέντα δεδομένα, την τοποθεσία αποστολής καθώς και 2 callbacks.

Καταρχήν, η τοποθεσία που ορίζουμε την αποστολή των δεδομένων είναι σχετική και όχι απόλυτη, αφού ήδη βρισκόμαστε στην τοποθεσία https://gearapp-server.herokuapp.com,

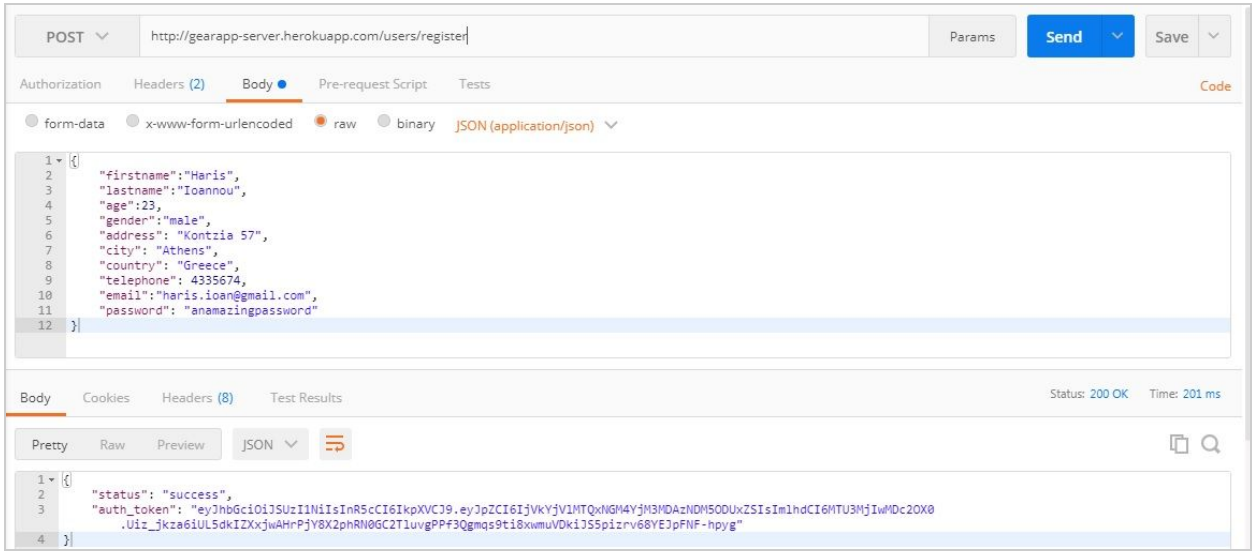
εισάγοντας: *"/users/register"* , υπονούμε ότι το πλήρες μονοπάτι της διεύθυνσης αποστολής είναι το https://gearapp-server.herokuapp.com/users/register, όπως έχουμε ορίσει.

Δευτερευόντως εδώ παρατηρούμε το δεύτερο βασικό πλεονέκτημα που μας προσφέρει η χρήση της JavaScript καθώς, χρησιμοποιούμε callbacks, συναρτήσεις δηλαδή που εκτελούνται υπό προϋποθέσεις. Συγκεκριμένα, αν το αίτημα προς τον εξυπηρετητή επιτύχει και αποσταλεί κωδικός HTTP 200(success) τότε θα εκτελεστει η πρώτη συνάρτηση(*done*) που θα ανακατευθύνει τον νέο χρήστη στο προσωπικό του προφίλ. Σε αντίθετη περίπτωση, θα εκτελεστεί η δεύτερη εν σειρα συνάρτηση(*fail*) η οποία και θα εμφανίσει το αντίστοιχο μήνυμα σφάλματος στην γραφική διεπαφή όπως φαίνεται στο Σχήμα 59.



Σχήμα 59: Η Σελίδα Εγγραφής Νέου Χρήστη στην Διαδικτυακή Εφαρμογή με Σφάλμα Εισαγωγής

Για την δοκιμή του αιτήματος αυτοτελώς το εκτελούμε, στο περιβάλλον της εφαρμογής Postman όπως φαίνεται στο ακόλουθο Σχήμα 60. Επίσης, εμφανίζεται και η επιτυχής απάντηση από τον εξυπηρετητή, η οποία περιέχει και το αναγνωριστικό πιστοποίησης που θα χρησιμοποιεί ο χρήστης στα αιτήματα του με την μορφή cookie.



Σχήμα 60: Εκτέλεση Αιτήματος Register User από την Εφαρμογή Postman

Στην συνέχεια το αίτημα κατευθύνεται στον εξυπηρετητη όπου δρομολογείται αξιοποιώντας την δυνατότητα πολυμορφισμού που έχουμε ήδη αναλύσει, όπως παρουσιάζεται στο Σχήμα 61.

```
app.route('/users/register').get(users.registrationPanel);
app.route('/users/register').post(users.register);
```

Σχήμα 61: Δρομολόγηση Εκμεταλευόμενη τον Πολυμορφισμό στην Εκτέλεση Αιτήματος Register User

Ο εξυπηρετητης για την εξυπηρέτηση του αιτήματος με την σειρά του, εκτελει την συνάρτηση `users.register` της οποίας η υλοποίηση ακολούθει παρακάτω.

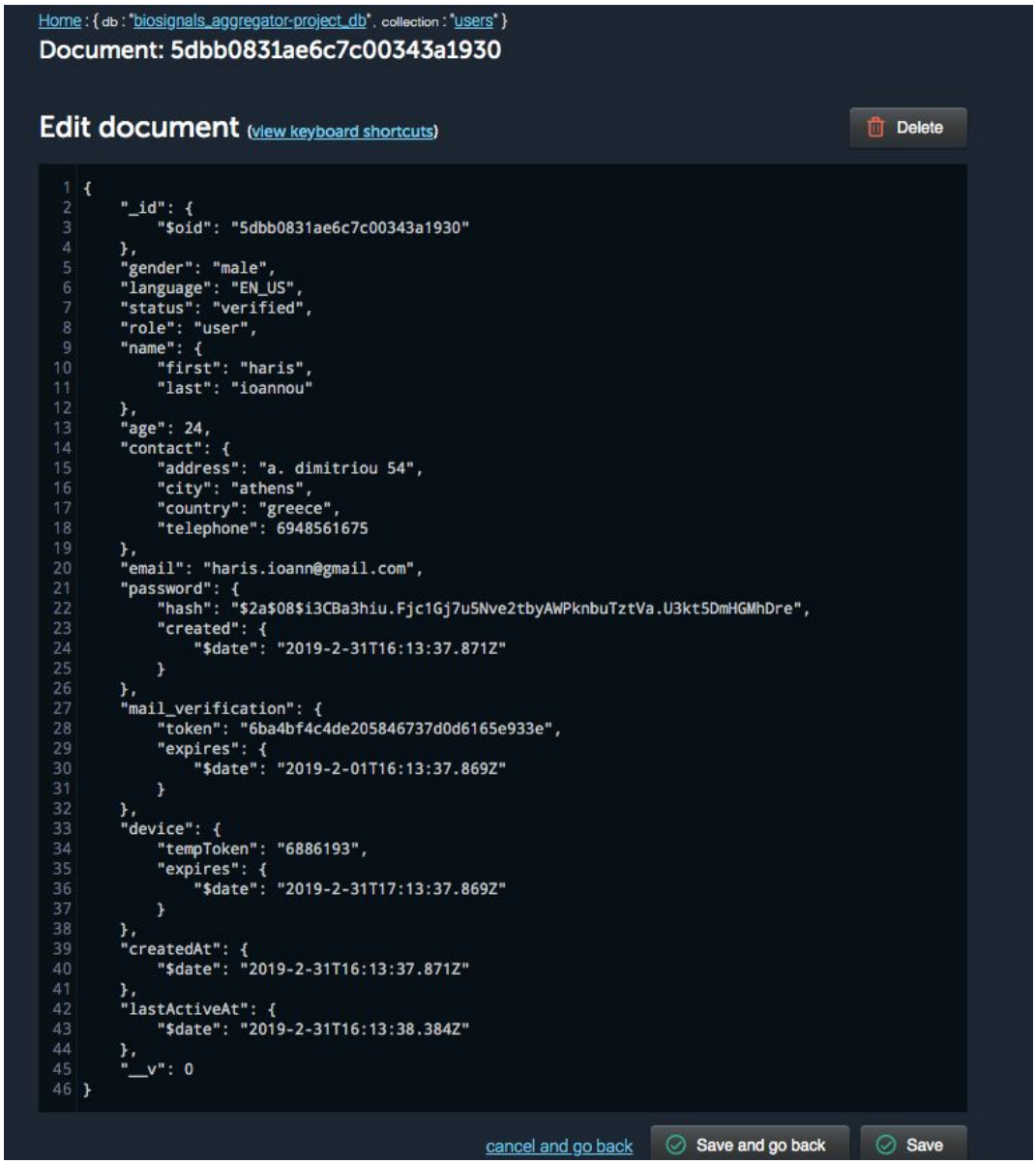
```
exports.register = function (req, res) {
   var user = new createUserFromRequest(req); ({
    user.save(function (err) {
       if (err) return sendError(err, res);
       mailservice.sendRegistrationEmail(user.email, mailtoken, function (){
          return sendResponseAndCookie(res);
   }});
});
```

Σχήμα 62: Καταχώρηση Νέου Χρήστη σε περιβάλλον Node.js και Mongoose

Η συνάρτηση `createUserFromRequest` ελέγχει τον τύπο των δεδομένων που περιέχει το αίτημα και στην συνέχεια δημιουργεί ένα αντικείμενο της βάσης δεδομένων MongoDB τυπου `user` με την βοήθεια της βιβλιοθήκης Mongoose. Σε περίπτωση που κάποιο από τα ορισμένα στοιχεία του αιτήματος είτε λείπουν, είτε δεν έχουν σωστο περιεχόμενο και τύπο, τότε επιστρέφετε μήνυμα λάθους στον χρήστη. Για παράδειγμα, βλέπουμε ένα δομημένο κατα json, μήνυμα λάθους στο ακόλουθο Σχήμα 63.

Σχήμα 63: Απάντηση Εξυπηρετητή με Μήνυμα Λάθους για Διπλότυπου Email

Στην συνέχεια τα στοιχεία του χρήστη αποθηκεύονται ως νέο αντικείμενο χρήστη στην βάση δεδομένων μέσω της συνάρτησης(`user.save`). Αν η αποθήκευση επιτύχει τότε στην βάση δεδομένων MongoDB αποθηκευτηκε η νέα εγγραφή όπως φαίνεται ενδεικτικα στο Σχήμα 64. Παράλληλα, δημιουργουργείται το πιστοποιητικό που θα χρησιμοποιεί ο χρήστης για την ταυτοποίηση του και αποστέλλεται με cookie στον browser του μέσω της συνάρτησης `sendResponseAndCookie`. Ταυτόχρονα αποστέλλεται email επιβεβαίωσης στον χρήστη εκτελώντας την συνάρτηση `mailservice.sendRegistrationEmail`, ώστε να επιβεβαιώσει την κυριότητα του. Επίσης την επιβεβαίωση του email του χρήστη μπορεί να εκτελέσει και ο διαχειριστής συστήματος όπως θα δούμε στην συνέχεια.



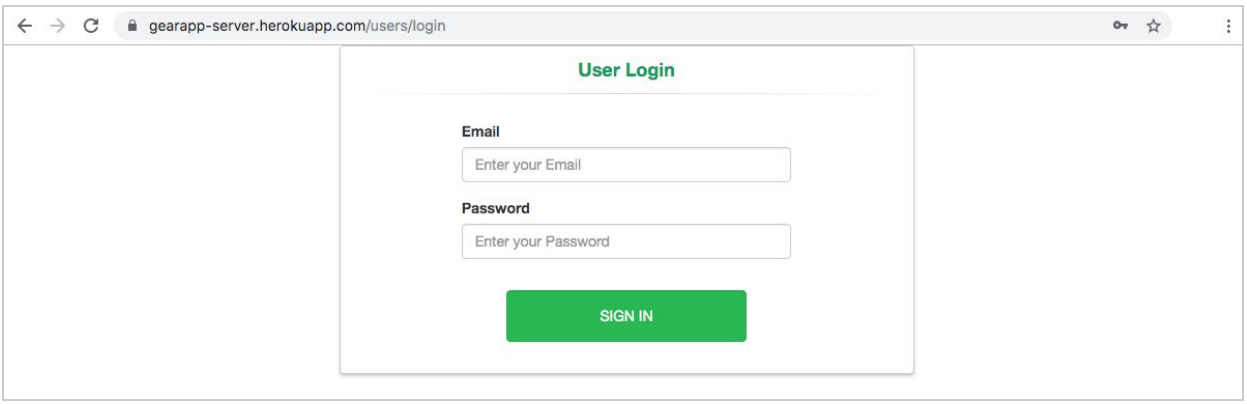
Σχήμα 64: Επιτυχής Εγγραφή Χρήστη στην Βάση Δεδομένων MongoDB

4.2.3.3 Σύνδεση Υπάρχοντος Χρήστη και Επαλήθευση Ταυτότητας

Η είσοδος του χρήστη αποτελεί απαραιτητη προυποθεση για την χρήση της πλατφόρμας συνολικά. Για την σύνδεση του σε αυτό, ο χρήστης πραγματοποιεί ένα αίτημα HTTP GET από τον browser του στην διεύθυνση:

https://gearapp-server.herokuapp.com/users/login

Εντός της γραφικής διεπαφής θα πρέπει να συμπληρώσει, το προσωπικό του email και κωδικό πρόσβασης που είχε παρέχει κατα την εγγραφή του στο σύστημα, όπως φαίνεται στο Σχήμα 65.



Σχήμα 65: Η Σελίδα Εισόδου Υπάρχοντος Χρήστη στην Διαδικτυακή Εφαρμογή

Μόλις ο χρήστης συμπληρώσει επιτυχώς τα στοιχεία και πατήσει το κουμπί Sign-In, ο view-controller που εκτελείται στο front-end θα εκτελέσει ασύγχρονα ένα HTTP POST αίτημα στον εξυπηρετητή στην διεύθυνση: https://gearapp-server.herokuapp.com/users/login όπως φαίνεται στο ακόλουθο Σχήμα 66.

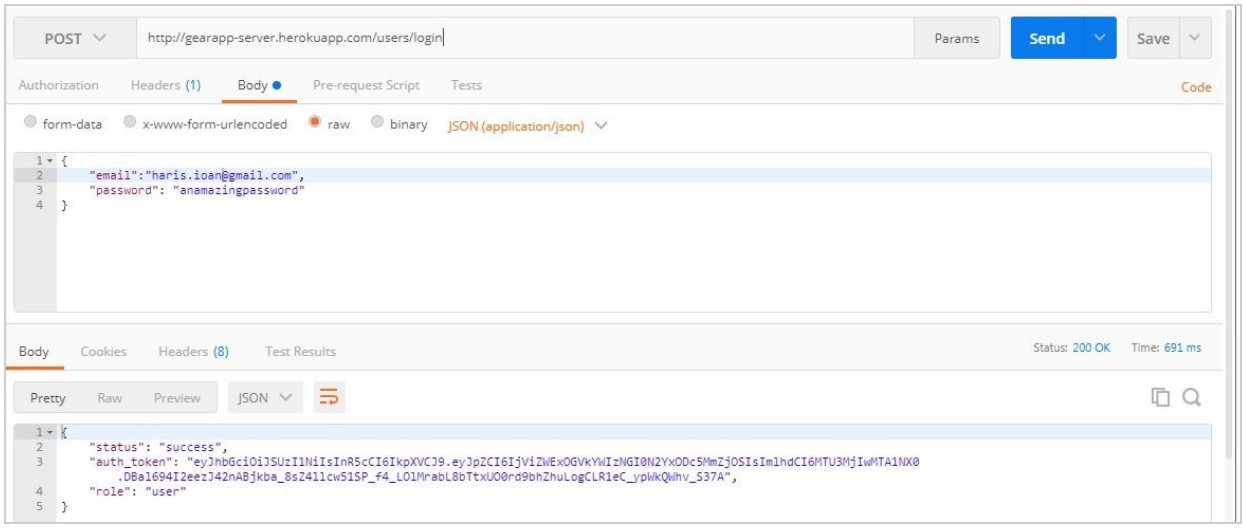
```
$("#login_btn").on("click", function (e) {
    e.preventDefault();
    let userData = getAndValidateUserInput();
    $.post("/users/login", userData).done(function (response) {
        if(response.role === 'admin'){ window.location.href = "/admin/index";
        }else{ window.location.href = "/users/panel" }
    }).fail(function (err) {
        handleError(err)
    });
});
```

Σχήμα 66: View-controller για την Εκτέλεση Αιτήματος Login User

Ο κώδικας εκτελείται εντός του click-listener για το κουμπί “Sign-In”, ακολουθώντας την event-driven μοντελοποίηση που αναλύσαμε στην προηγούμενη ενότητα. Αφού πατηθεί το κουμπί και το ανιχνεύσουμε, το επόμενο σημαντικό σημείο του κώδικα είναι η κλήση της συνάρτησης getAndValidateUserInput() η οποία λαμβάνει τα δεδομένα που εισήγαγε ο χρήστης και ελεγχει την πληροτητα, την μορφή και τον τύπο τους, όπως και κατα την εγγραφή του.

Μόλις ο ελεγχος ολοκληρωθεί, τα δεδομένα δομημένα σε μορφή JSON περιεχόμενα στην μεταβλητή userData, αποστέλλονται στον server όπως περιγράψαμε μέσω της κλήσης της συνάρτησης $.post. Ο εξυπηρετητής απαντά θετικά στο αίτημα σε περίπτωση που τα στοιχεία είναι ορθά, εκτελείται η συνάρτηση *done* ανακατευθύνει τον χρήστη στην γραφική διεπαφή του προσωπικού του προφίλ αν πρόκειται για απλό χρήστη, ενώ τον ανακατευθύνει στην γραφική διεπαφή διαχείρισης του συστήματος αν ο χρήστης είναι ορισμένος ως διαχειριστής συστήματος. Σε περίπτωση που τα στοιχεία είναι λανθασμένα, εκτελείται η συνάρτηση *fail* και εμφανίζεται ανάλογο μήνυμα στην διεπαφή.

Για την δοκιμή του αιτήματος, το εκτελούμε αυτοτελώς στο περιβάλλον της εφαρμογής Postman όπως φαίνεται στο ακόλουθο Σχήμα 67. Επίσης, εμφανίζεται και η επιτυχής απάντηση από τον εξυπηρετητή, η οποία περιέχει και το αναγνωριστικό πιστοποίησης που θα χρησιμοποιεί ο browser για το τρέχον session καθώς και τον ρόλο του χρήστη user ή admin.



Σχήμα 67: Εκτέλεση Αιτήματος Log-In από την Εφαρμογή Postman

Στην συνέχεια το αίτημα κατευθύνεται στον εξυπηρετητη όπου δρομολογείται αξιοποιώντας την δυνατότητα πολυμορφισμού που έχουμε ήδη αναλύσει, όπως παρουσιάζεται στο Σχήμα 68.

```
app.route('/users/login').get(users.loginPanel);
app.route('/users/login').post(users.login);
```

Σχήμα 68: Δρομολόγηση Εκμεταλευόμενη τον Πολυμορφισμό σε Αίτημα User Log-In

Ο εξυπηρετητής, όταν κληθεί να εξυπηρετήσει το μήνυμα GET, απλός μεταφέρει τα αρχεία που απαιτούνται για την αναπαράσταση της γραφικής διεπαφής στον browser, όπως φαίνεται στο κομμάτι κώδικα στο Σχήμα 69. Όταν όμως, ο εξυπηρετητής κληθεί να εξυπηρετήσει το αίτημα POST, εκτελει την συνάρτηση `users.login` της οποίας η υλοποίηση ακολουθεί στο Σχήμα 70.

```
exports.loginPanel = function (req, res){
   res.status(200).sendFile(path.join(__dirname,'./views/user_login.html'));
};
```

Σχήμα 69: Αποστολή Αρχείου Αναπαράστασης Γραφικής Διεπαφής για την Σύνδεση Χρήστη

```
exports.login = function (req, res) {
   let email = req.body.email;
   let password = req.body.password;
   if(!validator.isEmail(email)) invalidInput(res);
   User.findOne({ 'email': email }, function (err, user) {
       if (err || !user || user.status !== 'verified') {
           return sendError(err, res);
       }
       if (bcrypt.compareSync(password, user.password.hash)) {
           let auth_token = jwtservice.sign({id: user._id});
           res.role = user.role;
           res.auth-token = auth_token;
           return sendResponseAndCookie(res);
       } else {
           return sendError("auth-error", res);
       }
   });
};
```

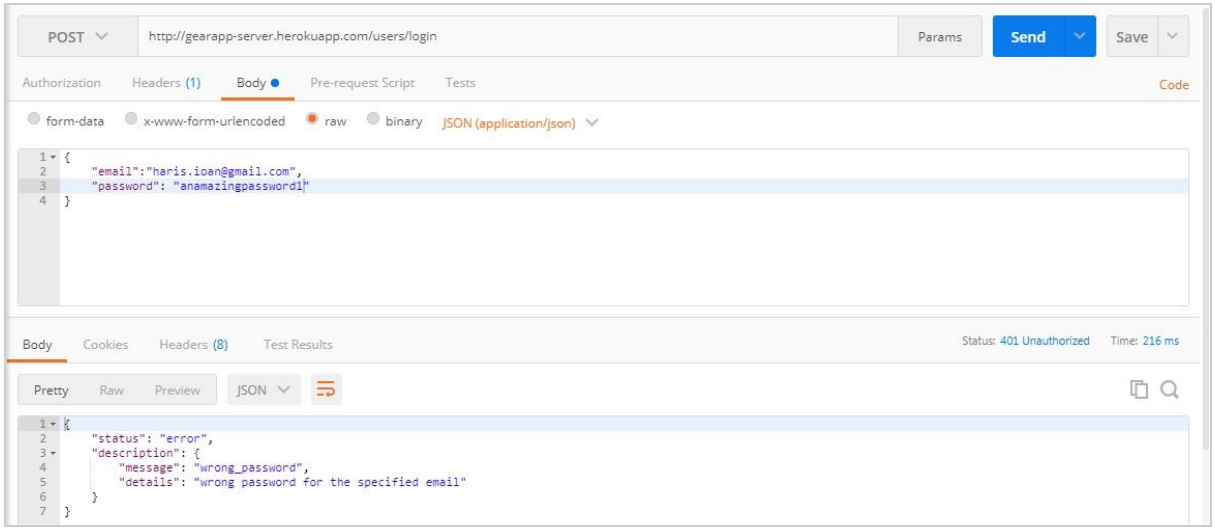
Σχήμα 70: Εξυπηρέτηση Αιτήματος Εισόδου Χρήστη

Αρχικά, εξάγουμε τα στοιχεία του χρήστη από το αίτημα και την δομή JSON κατα την οποία είχαν κωδικοποιηθεί και στην συνέχεια ελέγχουμε των τύπο τους. Αναζητάμε την εγγραφή χρήστη στην βάση δεδομένων με χρήση της βιβλιοθήκης Mongoose και της συνάρτησης

findOne, στην οποία αντιστοιχεί το δοθέν email. Η χρήση της λέξης *User*, αναφέρεται στην συλλογή αντικειμένων τύπου user, της βάσης δεδομένων MongoDB, όπως έχουμε ορίσει.

Σε περίπτωση που προκύψει κάποιο σφάλμα είτε κατα την αναζήτηση ή ο χρήστης δεν έχει επικυρώσει ακόμη το email του, τότε εμφανίζεται αντίστοιχο μήνυμα στην γραφική διεπαφή. Την λειτουργία αυτή υλοποιεί η συνάρτηση *sendError* που αναλαμβανει την αποστολή των αντίστοιχων αναγνωριστικών στον Client.

Σε περίπτωση που βρεθεί επιτυχώς ο χρήστης, τότε συγκρίνουμε τον κωδικό που εισήγαγε με τον κωδικό που έχουμε αποθηκεύσει στην βάση δεδομένων κατά την αρχική του εγγραφή στο σύστημα. Αν οι δύο κωδικοί ταυτίζονται, τότε η είσοδος του χρήστη είναι επιτυχής αφού σε αυτό το σημείο η ταυτότητα του έχει επαληθευτεί πλήρως, ολοκληρώνοντας έτσι το σταδιο του User Authentication. Η είσοδος στην εφαρμογή επιτρέπεται αποστέλλοντας αίτημα απάντησης προς τον Client/browser, μαζί με ρόλο του χρήστη στο σύστημα σε κωδικοποίηση JSON όπως εξηγήσαμε. Αν οι κωδικοι δεν ταυτίζονται, εκτελείται η συνάρτηση *sendError* με την οποία εμφανίζεται αντίστοιχο μήνυμα σφάλματος στον χρήστη.



Σχήμα 71: Απάντηση Εξυπηρετητή με Μήνυμα Λάθους σε Λανθασμένο Κωδικό Πρόσβασης

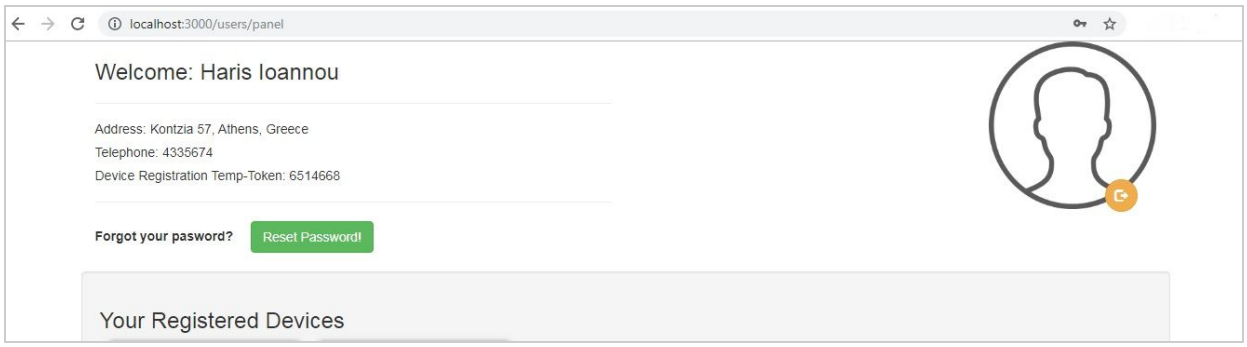
Αξίζει σε αυτό το σημείο να αναφέρουμε, ότι ο κωδικός του χρήστη δεν αποθηκεύεται στην βάση δεδομένων όπως δόθηκε αποκρυπτογραφημένος(plaintext) αλλά σε κατακερματισμένη μορφή(hashed). Συγκεκριμένα, συμβολοσειρά του κωδικού πρόσβασης του χρήστη κατακερματίζεται και αποθηκεύεται κατα την δημιουργία του χρήστη. Στο Σχήμα 72, φαίνεται ο κατακερματισμένος κωδικός ενός χρήστη του συστήματος όπως έχει αποθηκευτεί στην συλλογή *User* στην βάση δεδομένων MongoDB. Αντίστοιχα κατα την σύνδεση υπάρχοντος χρήστη στο σύστημα, για τον έλεγχο του εισαχθέντος κωδικού, εκτελείτε η συνάρτηρη *bcrypt.compareSync* η οποία τον κατακερματίζει και τον ελέγχει με τον ήδη αποθηκευμένο.

```
21      "password": {
22          "hash": "$2a$08$i3CBa3hiu.Fjc1Gj7u5Nve2tbyAWPknbuTztVa.U3kt5DmHGMhDre",
23          "created": {
24              "$date": "2019-2-31T16:13:37.871Z"
25          }
26      },
```

Σχήμα 72: Κατακερματισμένος Κωδικός Χρήστη στην Βάση Δεδομένων MongoDB

4.2.3.4 Αποσύνδεση Χρήστη

Για την αποσύνδεση του χρήστη, προφανώς πρέπει ήδη να έχει συνδεθεί στον προσωπικό του λογαριασμό και να βρίσκεται στο προσωπικό του προφίλ, όπως φαίνεται στο Σχήμα 73.



Σχήμα 73: Προφίλ Απλού Χρήστη

Η γραφική διεπαφή του προσωπικού προφίλ των απλών χρηστών προβάλλεται πραγματοποιόντας αίτημα HTTP GET από τον browser στην διεύθυνση:

https://gearapp-server.herokuapp.com/users/panel

Ο χρήστης για την αποσύνδεση του, πρέπει να πατήσει το κουμπί πορτοκαλί χρώματος, κάτω-δεξιά του εικοδίου χρήστη. Μόλις ο χρήστης εκτελέσει αυτή την ενέργεια, ο view-controller που εκτελείται στο front-end θα εκτελέσει ασύγχρονα ένα HTTP POST αίτημα στον εξυπηρετητή στην διεύθυνση https://gearapp-server.herokuapp.com/users/logout όπως φένεται στο ακόλουθο Σχήμα 74.

```
$("#logout_button").on("click", function (e) {
    $.post("/users/logout", "").done(function (response) {
        window.location.href = "";
    }).fail(function (err) {
        alert("Something went wrong" + err);
});
```

Σχήμα 74: View-controller για την Εκτέλεση Αιτήματος User Log-Out

Θυμίζουμε ότι η πιστοποίηση του χρήστη στο περιβάλλον του browser υλοποιείται με cookie το οποίο βρίσκεται αποθηκευμένο στον Client και αποστέλλετε αυτοματοποιημενα στον server σε κάθε αίτημα. Έτσι λοιπόν η αποσύνδεση του χρήστη στηρίζεται στην ακύρωση της εγκυρότητας του auth-token πιστοποίησης που περιέχει το cookie. Για τον λόγο αυτό εκτελούμε αίτημα προς τον server όποιος για να το εξυπηρετήσει εκτελει τον κώδικα της συνάρτησης `users.login` όπως φαίνεται παρακάτω Σχήμα 75.

```
exports.logout = function (req, res) {
    invalidateSession(req.user_id);
    res.cookie("auth-cookie","",{
        maxAge:0,
        httpOnly: true});
    return sendResponseAndCookie(res);
});
```

Σχήμα 75: Ακύρωση Εγκυρότητας του auth-token στο Cookie

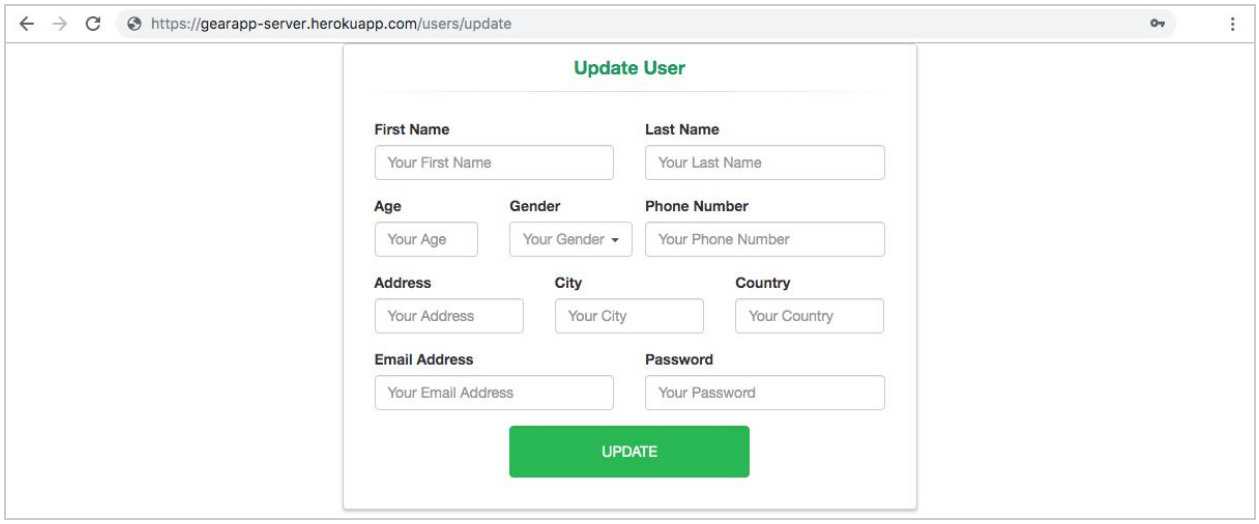
Ο εξυπηρετητης αποστέλλει το νέο cookie το οποίο αντικαθιστά το παλεό στο περιβάλλον του browser του χρήστη. Έτσι, με την λήψη απάντησης από τον server, ο client αποθηκεύει το νέο cookie και ανακατευθήνει τον χρήστη στην αρχική σελίδα του συστήματός. Σε περίπτωση που ο χρήστης προσπαθήσει να εισέλθει στο προσωπικό του προφίλ ή σε άλλες σελίδες που απαιτούν πιστοποίηση, θα λάβει μήνυμα σφάλματος, αφού δεν είναι σε θέση να πιστοποιηθεί με αυτό αναγνωριστικό στο υπάρχον cookie. Για την εκ νέου χρήση των προσωπικών του στοιχείων και πόρων εντός του συστήματος θα πρέπει να ξανα-ταυτοποιηθεί όπως περιγράψαμε στην προηγούμενη ενότητα.

4.2.3.5 Ενημέρωση Στοιχείων Υπάρχοντος Χρήστη

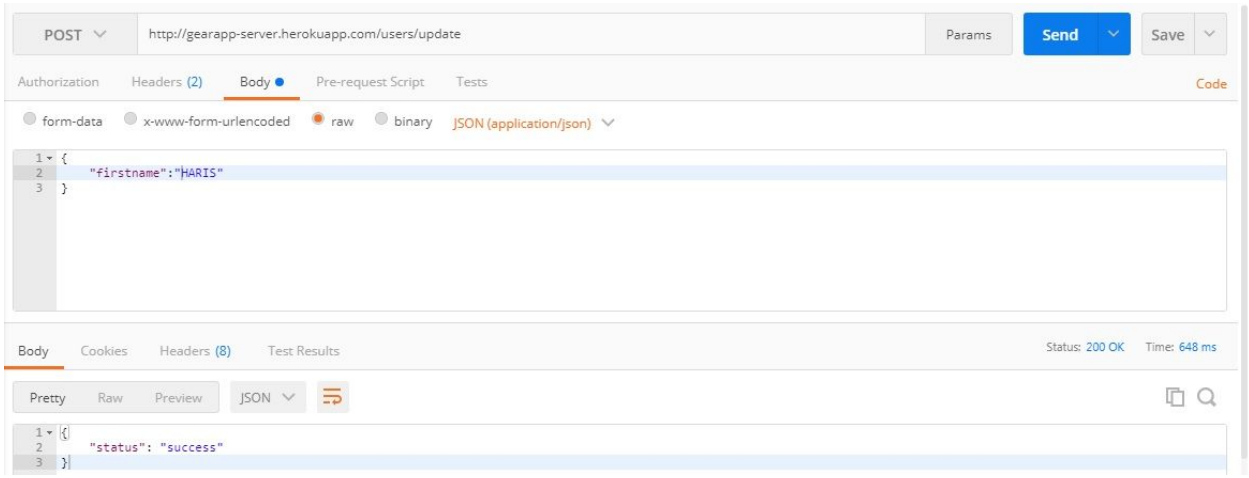
Το σύστημα, παρέχει την δυνατότητα στον χρήστη της αλλαγής των στοιχείων του σε περίπτωση που αυτά αλλάξουν, έτσι ώστε να είναι ενημερωμένο διαρκώς. Συγκεκριμένα, υποστηρίζει την αλλαγή οποιασδήποτε από τις παραμέτρους που έχει εισάγει ο χρήστης κατα την αρχική εγγραφή του. Για την αλλαγή των στοιχείων του, ο χρήστης μεταβαίνει στην αντίστοιχη γραφική διεπαφή, πραγματοποιόντας αίτημα HTTP GET από τον browser στην διεύθυνση:

https://gearapp-server.herokuapp.com/users/update

Για να συντελεστεί οποιαδήποτε αλλαγή στα στοιχεία του χρήστη, θα πρέπει να είναι ήδη συνδεδεμένος στο προσωπικό του προφίλ. Ακολουθεί στο Σχήμα 76 το HTTP POST αιτημα που αλλάζει το όνομα του χρήστη. Ομοίως θα μπορούσε να αλλάξει οποιαδήποτε άλλη παράμετρος του πλην του email και του κωδικού πρόσβασης.



Σχήμα 76: Γραφική Διεπαφή Ενημέρωσης Στοιχείων Χρήστη



Σχήμα 77: Αίτημα Αλλαγής Ονόματος Χρήστη

Για την περίπτωση αλλαγής του email χρήστη, παρότι ακολουθείτε η ίδια ακριβώς διαδικασία, υπάρχει ο περιορισμός που έχουμε θέσει στις λειτουργικές προδιαγραφές του συστήματος, κατα τις οποίες το email χρήστη πρέπει να είναι υπό την κυριότητα του και ως εκ τουτου επαληθευσιμο. Ετσι λοιπόν, ο χρήστης, αλλάζοντας το email του από την γραφική διεπαφή, θα λάβει ένα email στην νέα του διεύθυνση ώστε να την επαληθεύσει. Αυτό έρχεται σε αντιπαράθεση με την ενημέρωση των υπολοίπων στοιχείων του, αφου για αυτά αρκεί η μονομερής ενημέρωση τους στην γραφική διεπαφή. Για την περίπτωση αλλαγής του κωδικού χρήστη, θα πρέπει ο χρήστης να ακολουθήσει την διαδικασία αλλαγής κωδικού πρόσβασης όπως θα αυτή θα περιγραφεί την επόμενη ενότητα.

Στην μερία του εξυπηρετητή, όταν ληφθεί το αίτημα, σύμφωνα με το κομμάτι κώδικα Σχήμα 78, αναζητείται από ποιόν χρήστη προήλθε και το αντικείμενο(object) που αντιστοιχεί σε αυτόν, ανακαλείται από την βάση δεδομένων με την εκτέλεση της συνάρτησης `User.findOne`.

Στην συνέχεια η συνάρτηση *userDataFromRequest* δεχόμενη ως όρισμα το αντικείμενο χρήστη που βρήκαμε και τα νέα ανανεωμένα στοιχεία που εισήχθησαν στην γραφική διεπαφή, δημιουργεί το ενημερωμένο αντικείμενο χρήστη το οποίο και αποθηκεύει στην βάση. Ταυτόχρονα η συνάρτηση αυτή εκτελεί και όλους τους απαραίτητους ελέγχους για την μορφή και τύπο τον δεδομένων. Σε περίπτωση που ο χρήστης άλλαξε το email του, σύμφωνα με τις προδιαγραφές, αποστέλλουμε email για την επιβεβαίωση του και μέχρι τότε θεωρούμε ως εγκυρη την ήδη υπάρχουσα. Τέλος, αποστέλλουμε όλα τα απαραίτητα διακριτικα επιβεβαίωσης στον client, όπως δείξαμε στο παραπάνω αίτημα εντος της εφαρμογής Postman.

```
exports.update = function (req, res) {
   User.findOne({
        '_id': req.body.user_id
   }, function (err, user) {
        if (err || !user) { return sendError(err, res); }
        updatedUser = userDataFromRequest(req.body, user)
        updatedUser.save(function (err) {
            if (err) { return sendError(err, res); }
            if(req.body.email) {
             mailservice.sendRegistrationEmail(updatedUser.email,
               mailtoken, function (){
                return sendResponseAndCookie(res);
             });
            }else{ return sendResponseAndCookie(res); }
}); }) ;};
```

Σχήμα 78: Εξυπηρέτηση Αιτήματος Ενημέρωσης Στοιχείων Χρήστη

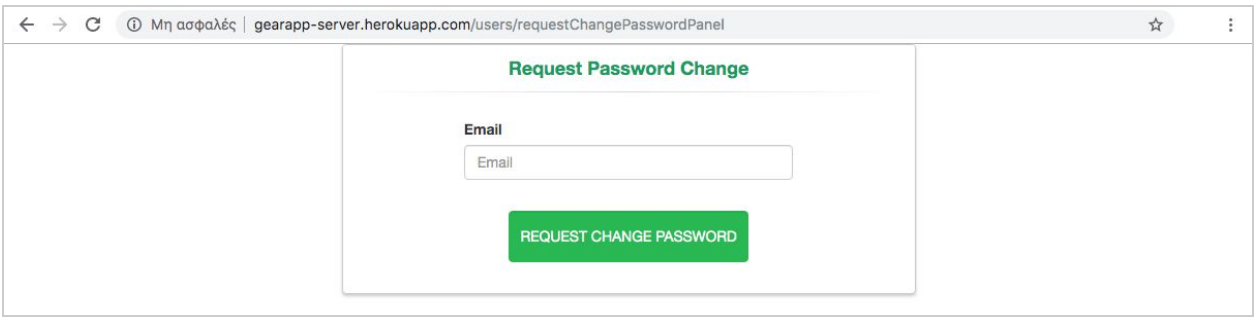
4.2.3.6 Σύστημα Ανάκτησης Κωδικού Πρόσβασης

Στα πλαίσια της παροχής μιας ολοκληρωμένης εμπειρίας χρήσης από τους χρήστες, κρίθηκε απαραίτητη η δημιουργία μηχανισμού ανάκτησης κωδικού πρόσβασης σε περίπτωση που ο χρήστης ξεχάσει τον ήδη υπάρχον. Εν προκειμένω, η εφαρμογή δίνει την δυνατότητα δημιουργίας νέου κωδικού πρόσβασης στον χρήστη μέσω email. Συγκεκριμένα, ο χρήστης μεταβαίνει στην διεπαφή, εκτελώντας αίτημα HTTP GET από τον browser στην διεύθυνση:

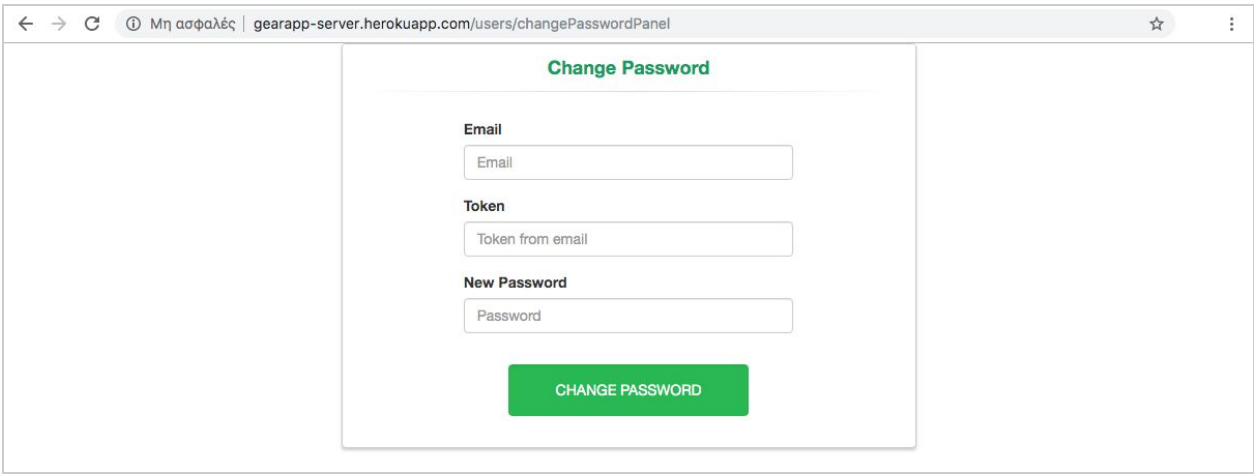
https://gearapp-server.herokuapp.com/users/requestChangePasswordPanel

Εκεί, ο χρήστης εισάγει την διεύθυνση email που έχει καταχωρήσει στο σύστημα, όπως βλέπουμε στο Σχήμα 79 και λαμβάνει email με ένα σύνδεσμο με ένα προσωρινό αναγνωριστικό. Στην συνέχεια ακολουθώντας τον σύνδεσμο, ο χρήστης εισάγει τον νέο κωδικό πρόσβασης του,

μαζί με το προσωρινό αναγνωριστικό που του στάλθηκε στην διεύθυνση του καθώς και το email του όπως φαίνεται στο Σχήμα 80. Εμπιστευόμενοι δηλαδή το γεγονός ότι η διεύθυνση email του χρήστη, βρίσκεται κάτω από την κυριότητα του, με αυτόν τον τρόπο επιτρέπουμε την αλλαγή του κωδικού πρόσβασης.



Σχήμα 79: Γραφική Διεπαφή Εισαγωγής email Χρήστη για την Ανάκτηση του Κωδικού Πρόσβασης



Σχήμα 80: Γραφική Διεπαφή Καταχώρησης Νέου Κωδικού Πρόσβασης Χρήστη

Ο εξυπηρετητής εξυπηρετεί αιτήματα στις δρομολογησεις που φαίνονται στο Σχήμα 81. Κατ αυτόν τον τρόπο εξυπηρετεί τα GET για την εμφάνιση της γραφικής διεπαφής που αναφέραμε άνωθι καθώς και την εμφάνιση της γραφικής διεπαφής που φαίνεται στο Σχήμα 80. Ο view-controller της πρώτης εκτελεί αιτήματα POST στην διεύθυνση: http://gearapp-server.herokuapp.com/users/requestChangePassword ενώ ο view-controller της δεύτερης εκτελεί αιτήματα POST στην διεύθυνση: https://gearapp-server.herokuapp.com/users/changePassword

```
app.route('/users/requestChangePassword').get(users.requestRecoverPassword);
app.route('/users/requestChangePasswordPanel').get(users.requestRecoverPasswordPanel);
```

```
app.route('/users/changePassword').post(users.recoverPassword);
app.route('/users/changePasswordPanel').get(users.recoverPasswordPanel);
```

Σχήμα 81: Δρομολόγηση Αιτημάτων για την Ανάκτηση του Κωδικού Πρόσβασης Χρήστη

Για την υλοποίηση της προδιαγεγραμμένης συμπεριφοράς, ο εξυπηρετητης λαμβάνοντας το αίτημα για την ανάκτηση κωδικού πρόσβασης για δοθέν email, εκτελεί την συνάρτηση εξυπηρέτησης *requestRecoverPassword*. Όπως φαίνεται στο ακόλουθο κομμάτι κώδικα, Σχήμα 82 ο server εντοπίζει το αντικείμενο χρήστη στην βάση δεδομένων με βάση το δοθέν email και δημιουργεί μια νέα εγγραφή στην οποία αποθηκεύει εάν μοναδικό αναγνωριστικό που θα σταλεί στον χρήστη. Η εκτέλεση της συνάρτησης *mailservice.generateTokenForDB()* δημιουργεί το μοναδικό αναγνωριστικό ενώ η εκτέλεση της συνάρτησης δημιουργεί την χρονοσήμανση κατα την οποία το αναγνωριστικό θεωρείται έγκυρο. Στο Σχήμα 83 φαίνεται ένα παράδειγμα της νέας εγγραφής στο αντικείμενο του χρήστη στην βάση δεδομένων. Εφόσον δημιουργηθεί με επιτυχία, αποστέλλεται το email μέσω της συνάρτησης *mailservice.sendRecoverPasswordEmail* που λαμβάνει ως παράμετρο το αναγνωριστικό, που θα τοποθετηθεί στο περιεχόμενο του email.

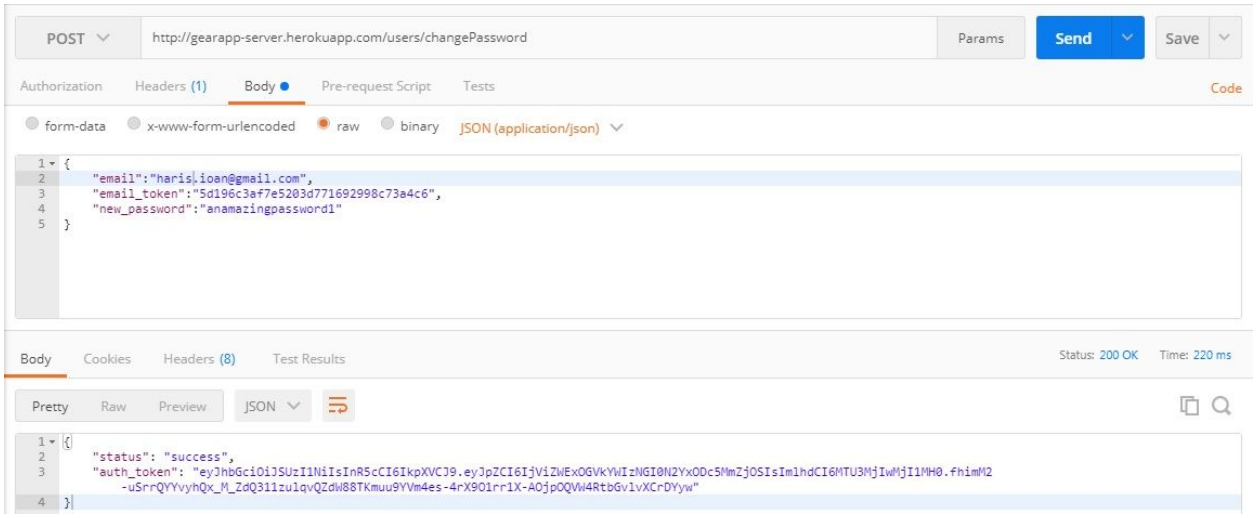
```
exports.requestRecoverPassword = function (req, res) {
   let email  = req.query.email;
   if(!validator.isEmail(email)) invalidInput(res);
   User.findOne({
       'email': email
   }, function (err, user) {
       if (err || !user) { return sendError(err, res); }
       user['change_password_verification'] = {
           token: mailservice.generateTokenForDB(),
           expires: mailservice.expiresIn()};
       user.save(function (err) {
           if (err || !user) { return sendError(err, res); }
            mailservice.sendRecoverPasswordEmail(updatedUser.email,
              user.change_password_verification.token , function (){
               return sendResponseAndCookie(res);
});});});});};
```

Σχήμα 82: Εξυπηρέτηση Αιτήματος Ανάκτησης Κωδικού Χρήστη - Δημιουργία Προσωρινού Αναγνωριστικού

```
46      "change_password_verification": {
47          "token": "cefbe1fe14950c509ffde6527ba1462d",
48          "expires": {
49              "$date": "2018-11-06T16:33:37.806Z"
50          }
51      }
```


Σχήμα 83: Εγγραφή Προσωρινού Αναγνωριστικού στην Βάση Δεδομένων

Στην συνέχεια, ο εξυπηρετητής, λαμβανοντας αίτημα POST απο την γραφική διεπαφή που παρουσιάσαμε στο Σχήμα 80 βρίσκει το αντικείμενο του χρήστη στην βάση δεδομένων απο το δοθεν email και προσωρινού αναγνωριστικού. Τέλος, ενημερώνει το αντικείμενο του χρήστη κατακερματίζοντας τον νεο κωδικο πρόσβασης και το αποθηκεύει στην βάση δεδομένων, ολοκληρώνοντας την διαδικασία.



Σχήμα 84: Αίτημα Ανάκτησης Κωδικού Χρήστη

4.2.3.7 Προβολή και Διαγραφή Συσκευών Χρήστη

Μετα την καταχώρηση τους, οι συσκευές smartwatch του χρηστη, βρίσκονται στην γραφική διεπαφή του προσωπικού του προφίλ, όπως φαίνεται στο Σχήμα 85.

Σχήμα 85: Γραφική Διεπαφή Προβολής και Διαγραφής Συσκευών Χρήστη

Για την διαγραφή κάποιας εκ των καταχωρημένων συσκευών του, ο χρήστης αρκει να πατήσει το αντίστοιχο εικονίδιο που αντιστοιχεί στην συσκευή προς διαγραφή, όπως στο Σχήμα 85. Στην συνέχεια, ο view-controller της διεπαφής θα εκτελέσει αίτημα POST στην διεύθυνση:

https://gearapp-server.herokuapp.com/devices/delete

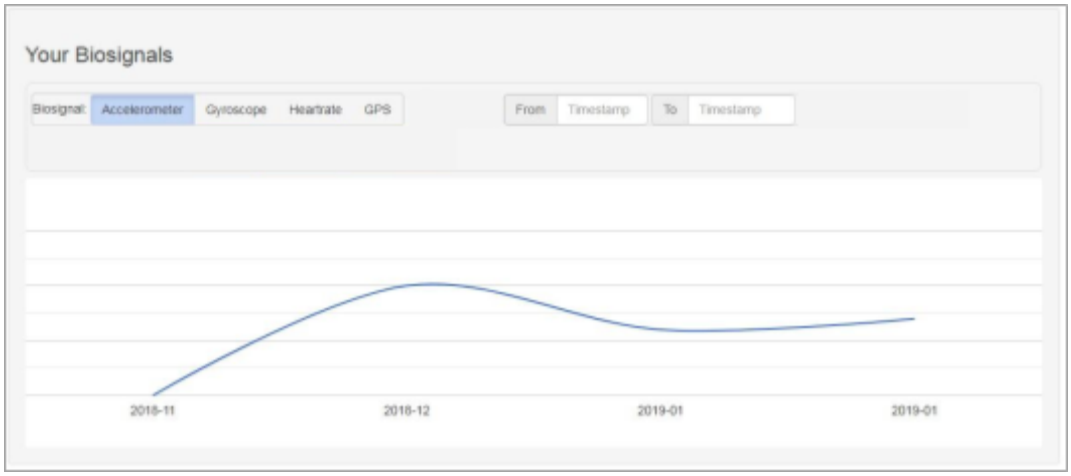
Ο εξυπηρετητής απο την μερία του, για την εξυπηρέτηση του αιτήματος, αρχικά θα λάβει το αναγνωριστικό της συσκευή που επιθυμεί ο χρήστης να διαγραφεί από το κωδικοποιημένο κατα JSON αίτημα, μέσω της εντολής `req.body.deviceId`. Έπειτα εκτελώντας την συνάρτηση `Device.remove,` επί της συλλογής των συσκευών που έχουν καταχωρηθεί στην βάση δεδομένων, διαγράφεται η επιθυμητη συσκευή απο τον χρήστη αλλά και από το σύστημα. Τέλος με την εκτέλεση της συνάρτησης `remainingDevicesForUser` επιστρέφονται με κωδικοποίηση JSON οι υπόλοιπες συσκευές που αντιστοιχούν στον χρήστη, ώστε ο client να ανανεώσει την γραφική διεπαφή.

```
exports.delete = function (req, res, next) {
   let device_id = req.body.deviceId;
   if(device_id === undefined || device_id === ""){ return sendError(err, res);}
   Device.remove({
       '_id': device_id
    }, function(err, result) {
       if (err) {return sendError(err, res);}
       return remainingDevicesForUser(req.user_id, res);
 });};
```

Σχήμα 86: Εξυπηρέτηση Αιτήματος Διαγραφής Συσκευής Χρήστη

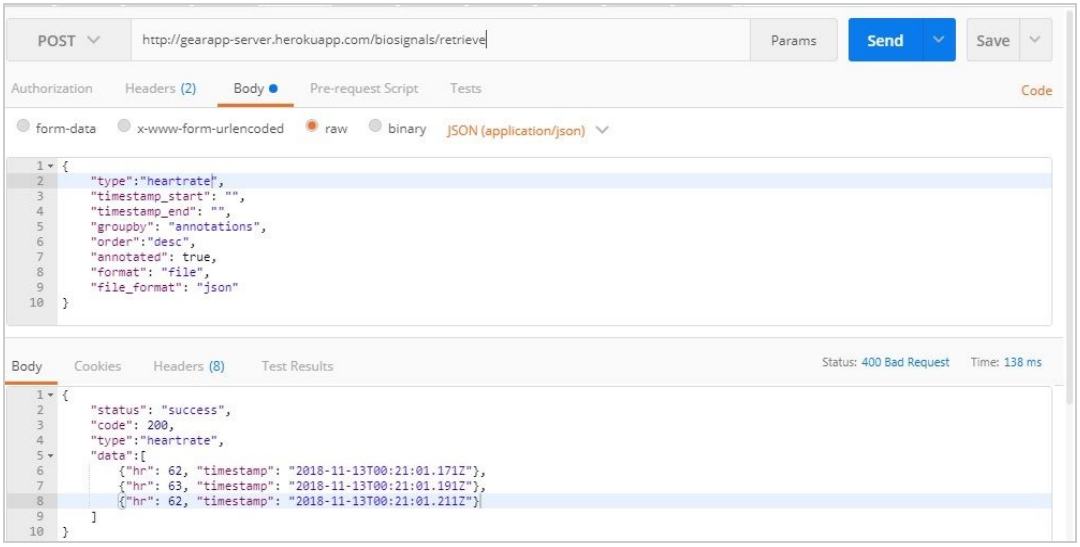
4.2.3.8 Προβολή Ληφθέντων Δεδομένων Κίνησης και Βιοσημάτων

Ο χρήστης είναι σε θέση να οπτικοποιήσει και να φιλτράρει τα ληφθέντα βιοσήματα και δεδομένα κίνησης από τις συσκευές του, μέσω της γραφικής διεπαφής του προσωπικού του προφίλ, όπως φαίνεται στο Σχήμα 87.



Σχήμα 87: Γραφική Διεπαφή Προβολής και Επεξεργασίας Δεδομένων Χρήστη

Ο χρήστης αρχικά πρέπει να επιλέξει τον επιθυμητό προς οπτικοποιήση τύπο βιοσηματος ή δεδομένου κίνησης που υποστηρίζεται από το σύστημα. Στην περίπτωση μας υποστηρίζονται δεδομένα από επιταχυνσιόμετρο, γυροσκόπιο, καρδιακό ρυθμό και G.P.S, όπως έχουμε εξηγήσει. Στην συνέχεια μπορεί να περιορίσει το εύρος της αναζήτησης εισάγοντας το επιθυμητό διάστημα ημερομηνιών. Ο view-controller παρατηρεί διαρκώς την διεπαφή για τυχόν αλλαγές από τον χρήστη στις επιλογές των φίλτρων και εκτελεί αντίστοιχα ερωτήματα προς τον εξυπηρετητη, όπως φαίνεται στο κάτωθι Σχήμα 88.



Σχήμα 88: Αίτημα Φιλτραρίσματος Δεδομένων Χρήστη

Ο εξυπηρετητής απο την μερία του, για την εξυπηρέτηση του αιτήματος, αρχικά λαμβάνει τα δεδομενα απο τα φίλτρα που έθεσε ο χρήστης, όπως τον τύπο του βιοσήματος και την χρονική περίοδο που ελήφθησαν κωδικοποιημένα κατα JSON όπως διακρίνεται επίσης στο Σχήμα 88.

Αντίστοιχα με τον επιλεγμένο τύπο δεδομένων εκτελείται η συνάρτηση που αναλαμβάνει να πραγματοποιησει τα αντίστοιχα ερωτήματα στην PostgreSQL βάση δεδομένων και να επιστρέψει την απάντηση στον client, όπως φαίνεται στην συνέχεια στο Σχήμα 89.

```
exports.retrieve = function (req, res, next) {
   let type = req.body.type, timestamp_start = req.body.timestamp_start;
   let timestamp_end = req.body.timestamp_end, user_id = req.body.user_id;
   if (timestamp_start === undefined || timestamp_end === undefined
    || user_id === undefined || user_id === "") { return sendError(err, res); }
   if (type === "accelerometer") {
       return retrieveAccelerometerData(req, res);
   } else if (type === "gyroscope") {
       return retrieveGyroscopeData(req, res);
   } else if (type === "heartrate") {
       return retrieveHeartrateData(req, res);
   } else if (type === "gps") {
       return retrieveGPSData(req, res);
   } else { return sendError(err, res); }};
```

Σχήμα 89: Εξυπηρέτηση Αιτήματος Φιλτραρίσματος Δεδομένων Χρήστη

Παρουσιάζουμε ενδεικτικά την υλοποίηση της συνάρτησης εξυπηρέτησης αιτήματος για το φιλτράρισμα δεδομένων επιταχυνσιομέτρου. Όπως βλέπουμε, χρησιμοποιουμε την `Sequelize` βιβλιοθήκη αφού για την αποθήκευση αυτών των δεδομένων χρησιμοποιούμε την PostgreSQL βάση δεδομένων και ως εκ τούτου τα ερωτήματα προς την βάση δεδομένων είναι σχεσιακά. Συγκεκριμένα, όπως φαίνεται στον κώδικα, δημιουργούμε δυναμικά το ερώτημα προς την βάση δεδομένων χρησιμοποιώντας την συνάρτηση `queryFromParameters(req)`. Η συνάρτηση αυτή λαμβάνει ως παραμέτρους τους περιορισμούς που έθεσε ο χρήστης στην γραφική διεπαφή και παράγει το κατάλληλο ερώτημα προς την βιβλιοθήκη `Sequelize` η οποία με την σειρά της παράγει το κατάλληλο ερώτημα προς την PostgreSQL βάση δεδομένων. Στην ενότητα **4.2.3.3** παρουσιάζουμε αναλυτικά την υλοποίηση της `queryFromParameters` καθώς στην παρούσα ενότητα χρησιμοποιούμε υποσύνολο των διαθέσιμων φίλτρων προς τα δεδομένα, επομένων είναι γόνιμο να παρουσιαστεί στο πλήρες εύρος της.

Εφόσον δημιουργηθεί το αίτημα και εκτελεστει στην βάση δεδομένων, το αποτέλεσμα λαμβανεται απο την συνάρτηση `sendResponse()` η οποία μορφοποιεί τα βιοσήματα στην μορφή που είδαμε στο Σχήμα 88 ώστε να αναπαρασταθούν γραφικά απο τον view-controller.

```
function retrieveAccelerometerData(req, res) {
   sequelize.AccelerometerModel.findAll(
       queryFromParameters(req)
   ).then(biosignals => {
       sendResponse(req, res, biosignals);
   }).catch(function (err) { return sendError(err, res);});
}
```

Σχήμα 90: Υλοποίηση Συνάρτησης Φιλτραρίσματος Δεδομένων Επιταχυνσιομέτρου

4.2.3 Υλοποίηση των Απαιτήσεων Διαχειριστών Συστήματος

Στην ενότητα αυτή θα παρουσιάσουμε αναλυτικα την υλοποίηση της προδιαγεγραμμένης λειτουργικότητας των διαχειριστών συστήματος, όπως την παρουσιάσαμε στην ενότητα **3.1.1** των λειτουργικών απαιτήσεων του συστήματος. Για κάθε λειτουργική απαίτηση θα παρουσιάσουμε το σενάριο χρήσης που επιβάλλει, κύρια σημεία του κώδικα καθώς και στιγμιότυπα από της γραφικές διεπαφές με τις οποίες αλληλεπιδρούν οι χρήστες.

4.2.3.1 Σύνδεση Διαχειριστή Συστήματος

Η σύνδεση του διαχειριστή του συστήματος, επιτελείται με τον ίδιο τρόπο που και οι απλοί χρήστες του συστήματος συνδέονται σε αυτό, εκτελώντας δηλαδή ένα αίτημα HTTP GET από τον browser στην διεύθυνση:

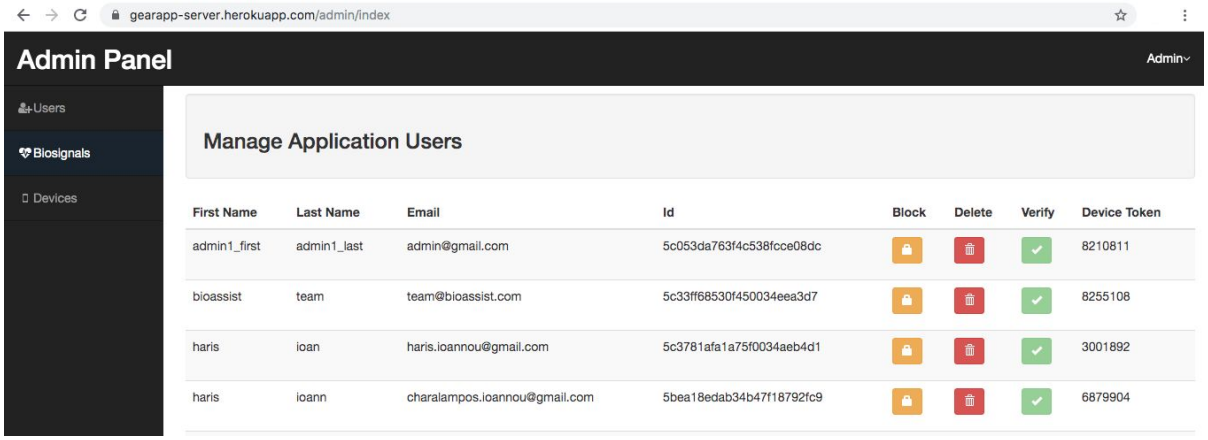
https://gearapp-server.herokuapp.com/users/login

Με την αποστολή του αιτήματος, όπως περιγράψαμε στην ενότητα **4.2.3.3** ο εξυπηρετητής επιστρέφει στον client τον ρόλο(role) του χρήστη που συνδέθηκε με τα στοιχεία του και αντίστοιχα η εφαρμογη τον ανακατευθύνει. Συγκεκριμένα, η διεύθυνση του προφιλ του διαχειριστή συστήματος, βρίσκεται στην στην διεύθυνση:

https://gearapp-server.herokuapp.com/admin/index

4.2.3.2 Προβολή του Συνόλου των Χρηστών

Ο διαχειριστής του συστήματος με την σύνδεση του στο σύστημα αυτόματος ανακατευθύνετε στην κεντρική γραφική διεπαφή η οποία προβάλλει σε μορφή λίστας όλους τους χρήστες της εφαρμογής, όπως φαίνεται στο ακόλουθο Σχήμα 91.



Σχήμα 91: Γραφική Διεπαφή Προβολής του Συνόλου των Χρηστών

Κάθε καταχώρηση χρήστη αντιστοιχεί και σε μία γραμμή του πίνακα, στην οποία παρουσιάζονται το ονοματεπώνυμο και το email του ως μοναδικά αναγνωριστικά σεβόμενοι τα προσωπικά δεδομένα των χρηστών όπως φαίνεται παραπάνω. Για κάθε χρήστη ο διαχειριστής του συστήματος μπορεί: να μπλοκάρει/ξεμπλοκάρει την πρόσβαση του στο σύστημα, να τον διαγράψει ή να επιβεβαιώσει την διεύθυνση email που έχει παρέχει κατα την αρχική του καταχώρηση στο σύστημα. Τέλος, στον διαχειριστή του συστήματος είναι ορατό και το προσωρινό 8-ψήφιο αναγνωριστικό με το οποίο ο κάθε χρήστης καταχωρεί νέες συσκευές στο σύστημα. Αυτό το σενάριο κρίθηκε απαραίτητο για υλοποίηση, ούτως ώστε ο διαχειριστης συστήματος να μπορεί να δηλώσει συσκευές σε χρήστες χωρίς αυτοί να εισέλθουν στο προσωπικό τους προφιλ εντός της διαδικτυακής πλατφόρμας. Η λειτουργία αυτή αποδείχθηκε ιδιαίτερα χρήσιμη σε ηλικιωμένους χρήστες της υπηρεσίας, ελαχιστοποιώντας τον χρόνο που απαιτείται για την καταχώρηση των συσκευών σε ομάδα χρηστών. Σε επίπεδο υλοποίησης απαιτήθηκε κάποια επέκταση για την κάλυψη αυτής της λειτουργίας, μιας και ο διαχειριστής αρκείται στο να γνωρίζει το 8-ψήφιο αναγνωριστικό.

Όσον αφορά, τις αλλαγές των δικαιωμάτων χρηστών, αυτές επιτελούνται πολύ εύκολα πατώντας ο διαχειριστής τα αντίστοιχα κουμπιά στην γραμμή που αντιστοιχεί στον χρήστη που επιθυμεί. Σε επίπεδο υλοποίησης ο view-controller μόλις εντοπίσει το πάτημα ενός από τα κουμπιά εκτελεί ένα αίτημα HTTP POST από τον browser στην διεύθυνση:

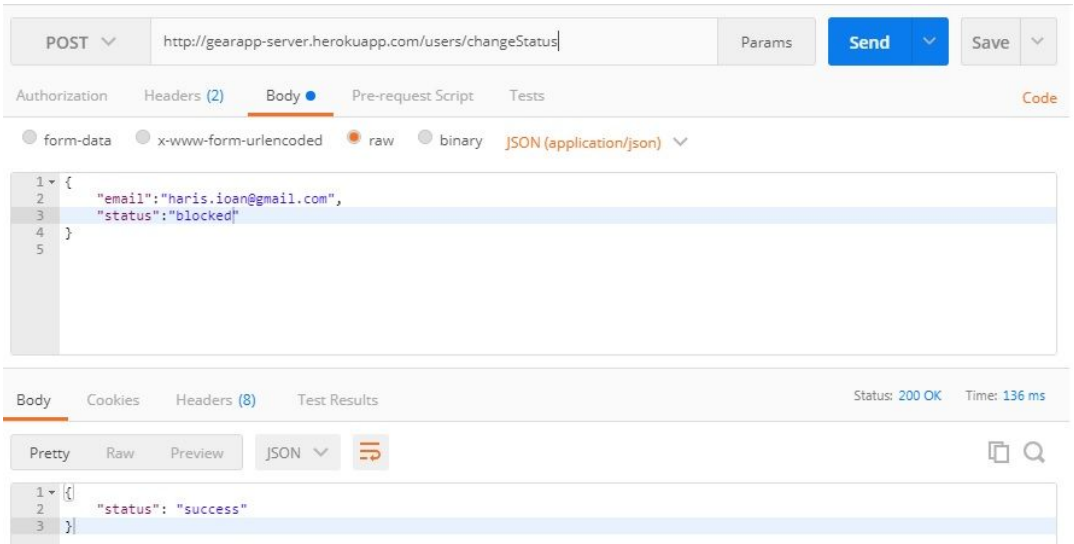
https://gearapp-server.herokuapp.com/users/changeStatus

```
function buttonListener(old_this) {
    let clickedId = old_this.find("button").eq(0).attr('id');
    let currentTableRow = old_this.closest("tr");
    let data = {
        email: "",
```

```
            status: ""
        };
        if (clickedId === "block_btn") {
            data.status = "blocked";
        } else if (clickedId === "delete_btn") {
            data.status = "deleted";
        } else if (clickedId === "verify_btn") {
            data.status = "verified";
        }
        data.email = currentTableRow.children().eq(2).html();+
        search("/users/changeStatus", data, function (response) {
          updateView(response)
        }, function (error) {
            alert("No data received from the server\n" + JSON.stringify(error));
        });
    }
```

Σχήμα 92: View-controller για την Εκτέλεση Αιτήματος Αλλαγής Δικαιωμάτων

Όπως φαίνεται στο άνωθι κομμάτι κώδικα, μολις πατηθεί κάποιο από τα κουμπια αλλαγής δικαιωμάτων χρήστη καλείτε η συνάρτηση `buttonListener`. Μόλις εντοπιστεί το email του χρήστη στον οποίο αναφερόταν η αλλαγή των δικαιωμάτων καθώς και του τύπου της αλλαγής δικαιώματος, εκτελεί το αίτημα προς τον εξυπηρετητη όπως φαίνεται στο ακόλουθο Σχήμα 93.



Σχήμα 93: Αίτημα Αλλαγής Δικαιωμάτων Χρηστών

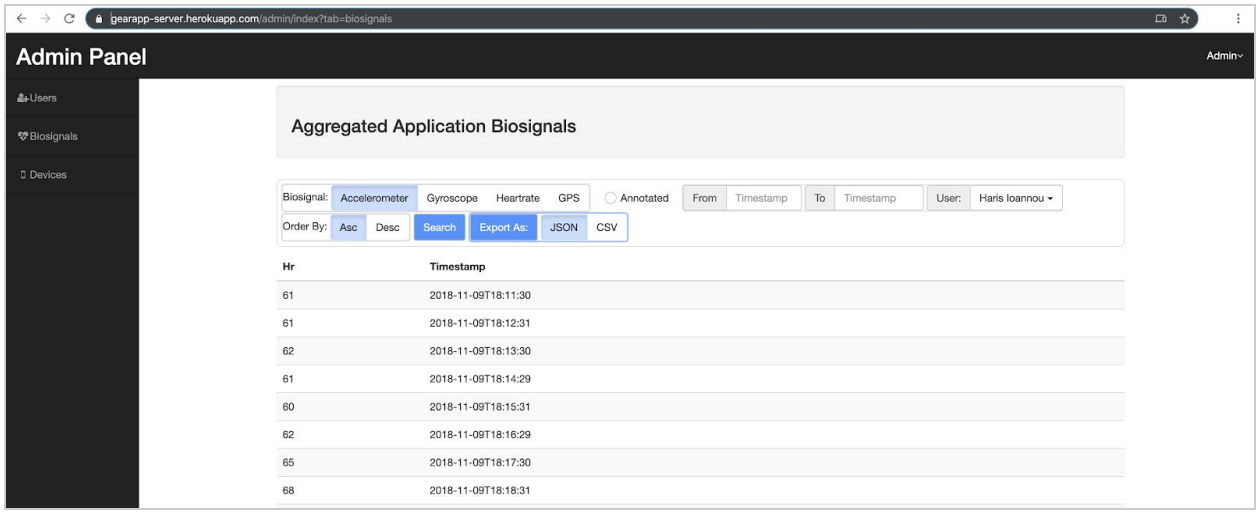
Στην μεριά του εξυπηρετητή, μόλις ληφθεί το αίτημα στην προδιαγεγραμμένη διεπαφή, αναζητάτε ο χρήστης με βάση το email του που συμπεριλαμβάνεται εντός του αιτήματος. Στην συνέχεια και ενημερώνεται το πεδίο στην δομή του χρήστη και επαναααποθηκεύεται στην βάση δεδομένων MongoDB, όπως φαίνεται στην συνέχεια στο Σχήμα 94.

```
exports.changeStatus = function (req, res) {
   let email = req.body.email;
   if(!validator.isEmail(email)) return invalidInput(res);
   User.findOne({
       'email': email
   }, function (err, user) {
       if (err || !user) { return sendError(err, res); }
       let status = req.body.status;
       if( status !== 'pending' && status !== 'verified'
       && status !== 'blocked' && status !== 'deleted'
       && status !== 'suspicious'){ }
       user.status = status;
       user.save(function (err) {
           if (err) { return sendError(err, res); }
           return sendResponse(res);
};});});});
```

Σχήμα 94: View-controller για την Εκτέλεση Αιτήματος Αλλαγής Δικαιωμάτων

4.2.3.3 Προβολή Ληφθέντων Δεδομένων Κίνησης και Βιοσημάτων Χρηστών

Η δυνατότητα φιλτραρίσματος και ταξινόμησης των συλλεχθέντων δεδομένων που παρέχεται στους απλούς χρήστες, παρέχεται και στους διαχειριστές του συστήματος . Η διαφορά του έγκειται στο γεγονός ότι οι διαχειριστές έχουν πρόσβαση σε όλα τα βιοσήματα και δεδομένα κίνησης που έχουν συλλεχθεί από το σύστημα, όπως φαίνεται στο ακόλουθο Σχήμα 95.



Σχήμα 95: Γραφική Διεπαφή Προβολής Δεδομένων Κίνησης και Βιοσημάτων Χρηστών

Σε αντιστοιχία με την δυνατότητα αναζήτησης που παρέχετε στους απλούς χρήστες, ο διαχειριστής στην αναζήτηση του πρέπει αρχικά να επιλέξει τον επιθυμητό προς οπτικοποιήση τύπο βιοσηματος ή δεδομένου κίνησης που υποστηρίζεται από το σύστημα. Στην περίπτωση μας και πάλι υποστηρίζονται δεδομένα από επιταχυνσιόμετρο, γυροσκόπιο, καρδιακό ρυθμό και G.P.S. Στην συνέχεια το εύρος αναζήτησης μπορεί να περιορίσει του με βάση:

- Τον χρήστη από τον οποίο εισήχθησαν στο σύστημα τα δεδομένα.
- Το διάστημα ημερομηνιών που συνελέγησαν τα δεδομένα.

Παράλληλα, τα αποτελέσματα αναζήτησης μπορούν να ταξινομηθούν με βάση την ημερομηνία συλλογής τους σε αύξουσα ή φθίνουσα σειρά ενώ μπορούν να ληφθούν από τον χρήστη για την περαιτέρω ερευνητική επεξεργασία τους σε μορφή JSON ή CSV.

Για την εξυπηρέτηση του αιτήματος αναζήτησης απλών χρηστών και διαχειριστών, βασικό ρόλο επιτελεί η συνάρτηση δυναμικής δημιουργίας των ερωτημάτων χρησιμοποιώντας την βιβλιοθήκη Sequelize. Ο εξυπηρετητής λαμβάνοντας το αίτημα, λαμβάνει τα δεδομενα απο τα φίλτρα που έθεσε ο χρήστης και με την βοήθεια της συνάρτησης `queryFromParameters`, δημιουργεί το Sequelize ερώτημα το οποίο στην συνέχεια εκπληρώνει ο PostgreSQL server.

Η υλοποίηση της συνάρτησης `queryFromParameters` παρουσιάζεται στο Σχήμα 96 και επι της ουσίας δημιουργεί ένα εκτενές “where” σχεσιακό ερώτημα κατά SQL. Αρχικά, δημιουργούμε το υποερώτημα που αφορά το φιλτράρισμα με βάση τις ημερομηνίες με χρήση των τελεστών της Sequelize, `Op.gte: timestamp_start`(gte=Greater Than or Equal) και `Op.lte: timestamp_end`(lte=Less Than or Equal). Έπειτα δημιουργούμε το υποερώτημα που αφορά το φιλτράρισμα με βάση το αναγνωριστικό χρήστη(`'user_id'`). Επιπλέων, δημιουργούμε το υποερώτημα που αφορά το φιλτράρισμα με βάση την ύπαρξη ή απουσία επισημάνσεων(annotations) και σχολίων(comments). Τέλος, δημιουργούμε το υποερώτημα που αφορά την ταξινόμηση σε αύξουσα(`'ASC'`) ή φθίνουσα(`'DESC'`) σειρά.

```
function queryFromParameters(req) {
    let result = {};
    //Handle timestamps
    const {Op} = require('sequelize');
    let timestamp_start = req.body.timestamp_start;
    let timestamp_end = req.body.timestamp_end;
    let user_id = req.body.user_id;
    let whereStatement = {where: {timestamp: {}}};
    if (timestamp_start === '' && timestamp_end === '') {
        whereStatement = {};
    } else if (timestamp_start !== '' && timestamp_end === '') {
        whereStatement.where.timestamp = {
            [Op.gte]: timestamp_start
        };
    } else if (timestamp_start === '' && timestamp_end !== '') {
        whereStatement.where.timestamp = {
            [Op.lte]: timestamp_end
        };
    } else if (timestamp_start !== '' && timestamp_end !== '') {
        whereStatement.where.timestamp = {
            [Op.gte]: timestamp_start,
            [Op.lte]: timestamp_end
        };
     }
    //Handle user_id
    if (user_id !== undefined && user_id !== '') {
        if (whereStatement.where === undefined) {
            whereStatement = {where: {user_id: user_id}};
        } else {
            whereStatement.where['user_id'] = user_id;
        }}
    result['where'] = whereStatement.where;
    //Handle annotations
    let includeStatement = {};
    let annotated = req.body.annotated;
```

```
let type = req.body.type;
if (annotated !== undefined) {
    if (annotated === true || annotated === "true") {
        includeStatement = {
            include: [{
                model: sequelize.AnnotationModel,
                required: false,
                on: {timestamp: '', user_id: ''},
                attributes: ['annotation']
            }]
        };
        if (type === 'accelerometer') {
            includeStatement.include[0].on.timestamp = Sequelize.where(
             Sequelize.col("AccelerometerSignal.timestamp"), "=",
             Sequelize.col("AnnotationModel.timestamp"));
            includeStatement.include[0].on.user_id =
                Sequelize.where(
                 Sequelize.col("AccelerometerSignal.user_id"), "=",
                 Sequelize.col("AnnotationModel.user_id"))
        } else if (type === 'gyroscope') {
            includeStatement.include[0].on.timestamp = Sequelize.where(
             Sequelize.col("GyroscopeSignal.timestamp"), "=",
             Sequelize.col("AnnotationModel.timestamp"));
            includeStatement.include[0].on.user_id =
                Sequelize.where(
                 Sequelize.col("GyroscopeSignal.user_id"), "=",
                 Sequelize.col("AnnotationModel.user_id"))
        } else if (type === 'heartrate') {
            includeStatement.include[0].on.timestamp = Sequelize.where(
             Sequelize.col("HeartrateSignal.timestamp"), "=",
             Sequelize.col("AnnotationModel.timestamp"));
            includeStatement.include[0].on.user_id =
                Sequelize.where(
                 Sequelize.col("HeartrateSignal.user_id"), "=",
                 Sequelize.col("AnnotationModel.user_id"))
```

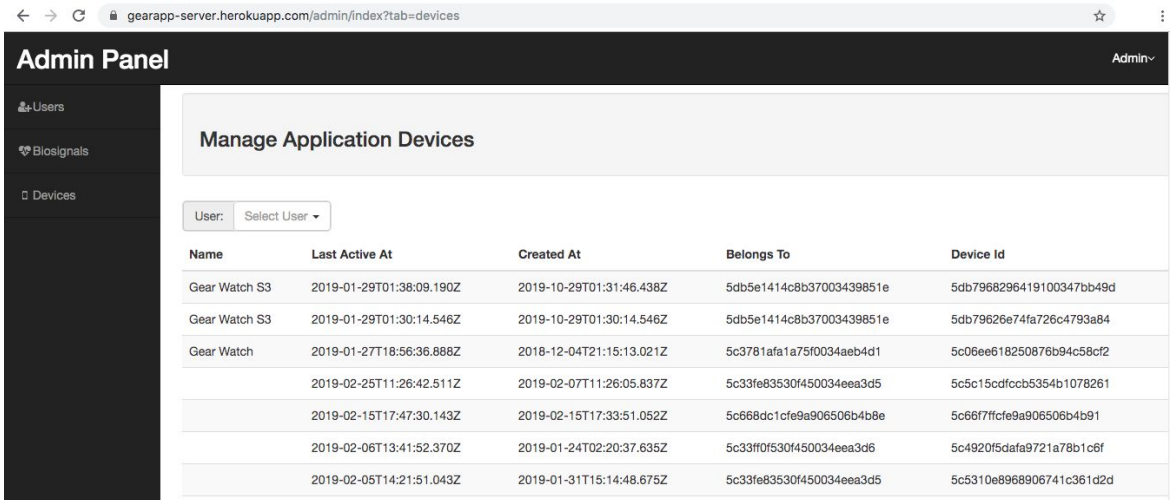
```
        } else if (type === 'gps') {
            includeStatement.include[0].on.timestamp = Sequelize.where(
             Sequelize.col("GPSSignal.timestamp"), "=",
             Sequelize.col("AnnotationModel.timestamp"));
            includeStatement.include[0].on.user_id =
               Sequelize.where(
                Sequelize.col("GPSSignal.user_id"), "=",
                Sequelize.col("AnnotationModel.user_id"))
        } else {
            includeStatement = {};
        }
    } else {
        includeStatement = {};
    }
}
result['include'] = includeStatement.include;
//Handle attributes
let attributesStatement = { attributes: { exclude: ['user_id', 'createdAt'] }};
result['attributes'] = attributesStatement.attributes;
//Handle order-by
let orderStatement = {};
let order = req.body.order;
if (order !== undefined) {
    if (order === 'asc') {
        orderStatement = {
            order: [['timestamp', 'ASC']]
        };
    } else if (order === 'desc') {
        orderStatement = {
            order: [['timestamp', 'DESC']]
        };
    }}
result['order'] = orderStatement.order;
return result;}
```

Σχήμα 96: Υλοποίηση την Συνάρτησης Δυναμικής Κατασκευής Ερωτημάτων Sequelize

4.2.3.4 Προβολή Συσκευών Χρηστών

Ο διαχειριστής του συστήματος έχει πρόσβαση στην καρτέλα στην οποία προβάλλονται σε μορφή λίστας όλες οι συσκευές smartwatch που είναι καταχωρημένες απο τους χρήστες της εφαρμογής, όπως φαίνεται στο ακόλουθο Σχήμα 97. Προσφέρεται η δυνατότητα προβολής συσκευών για συγκεκριμένο χρήστη η οποία λειτουργεί όμοια με όσα έχουμε παρουσιάσει.

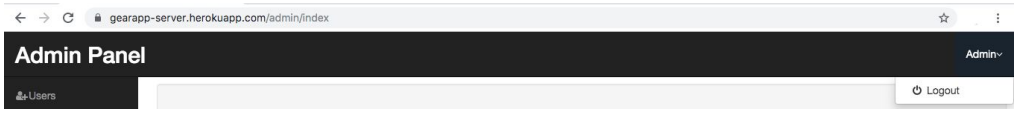
Κάθε καταχώρηση συσκευής αντιστοιχεί και σε μία γραμμή του πίνακα, στην οποία παρουσιάζονται: το όνομα της συσκευής αν αυτό είναι διαθέσιμο, την χρονοσήμανση που καταχωρήθηκε η συσκευή στο σύστημα, την χρονοσήμανση που αλληλεπίδρασε η συσκευή με το σύστημα τελευταία φορά, το αναγνωριστικό της στο σύστημα και το αναγνωριστικό του χρήστη στον οποίο ανήκει.



Σχήμα 97: Γραφική Διεπαφή Προβολής Συσκευών Χρηστών

4.2.3.5 Αποσύνδεση Διαχειριστή Συστήματος

Η αποσύνδεση του διαχειριστή συστήματος από το σύστημα, εκτελείται πατώντας το αντίστοιχο κουμπί όπως φαίνεται στο Σχήμα 98



Σχήμα 98: Γραφική Διεπαφή Αποσύνδεσης Διαχειριστή Συστήματος

Η υλοποίηση στηρίζεται στην ίδια αρχή που στηρίζεται και η αποσύνδεση του απλού χρήστη, στην ακύρωση της εγκυρότητας δηλαδή του auth-token πιστοποίησης που περιέχει το cookie.

Με το πάτημα του κουμπιού ο view-controller που εκτελείται στο front-end θα εκτελέσει ασύγχρονα ένα HTTP POST αίτημα στον εξυπηρετητή στην διεύθυνση: https://gearapp-server.herokuapp.com/users/logout όπως περιγράψαμε στην ενότητα **4.2.3.4**.

4.3 Υλοποίηση Εφαρμογής Συλλογής Δεδομένων

Στην ενότητα αυτή θα παρουσιάσουμε αναλυτικα την υλοποίηση της προδιαγεγραμμένης λειτουργικότητας της εφαρμογής συλλογής και αποθήκευσης βιοσημάτων και δεδομένων κίνησης χρηστών, όπως την παρουσιάσαμε στην ενότητα **3.2.2**. Όμοια με τις προηγούμενες ενότητες, για κάθε σενάριο χρήσης, θα παρουσιάσουμε κύρια σημεία του κώδικα και στιγμιότυπα από της γραφικές διεπαφές με τις οποίες αλληλεπιδρούν οι χρήστες.

4.3.1 Υλοποίηση της Εφαρμογής

Η υλοποίηση της εφαρμογής συλλογής και αποθήκευσης βιοσημάτων και δεδομένων κίνησης βασίστηκε σε δύο βασικές λειτουργικές απαιτήσεις.

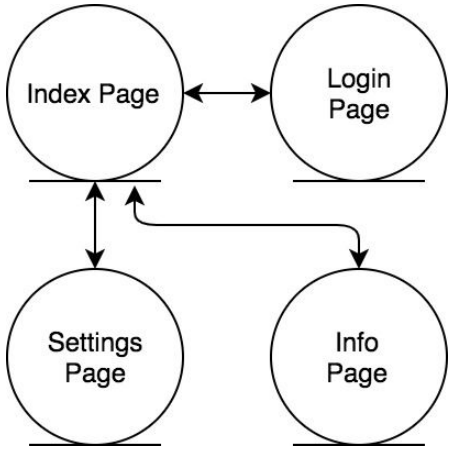
Η πρώτη απαίτηση ήταν η αποδοτική συλλογή δεδομένων με την μεγαλύτερη δυνατή πιστότητα και ανάλυση χωρίς όμως να μειώνεται δραστικά η περίοδος χρήσης της συσκευής. Συγκεκριμένα καθόλη την διάρκεια χρήσης της εφαρμογής η συσκευή λειτούργησε αδιάκοπα για περισσότερες από 24 ώρες με μία μόνο φόρτιση. Αυτό είναι πολύ βασικό χαρακτηριστικό αφού μπορούμε να συλλέγουμε δεδομένα από τους χρήστες μας καθόλι την διάρκεια της ημέρας τους.

Η δεύτερη απαίτηση ήταν η εμπειρία χρήσης της εφαρμογής να είναι ευχάριστη και εύκολη. Όπως έχουμε αναλύσει, οι χρήστες δεν προέρχονται από ένα συγκεκριμένο κοινονικό σύνολο και αυτό επιτάσει την σχεδίαση της εφαρμογής με τρόπο ώστε να είναι χρηστική απο όσο το δυνατόν περισσότερους χρήστες. Συγκεκριμένα, δινουμένου ότι ο χρήστης έχει καταχωρήσει την συσκευή του στο σύστημα με την μεθοδολογία που θα αναλύσουμε στην ενότητα **4.3.2.1**, ο χρήστης απαιτείται μόνο να ανοίξει την εφαρμογή από το κεντρικό menu της συσκευής. Η ενέργεια αυτή αρκεί ώστε η εφαρμογή, να αρχίσει την συλλογή δεδομένων και τον αυτοματοποιημένο συγχρονισμό τους με τους με τον εξυπηρετητή.

Η αποδοτική συλλογή δεδομένων και ο συγχρονισμός τους εκτελείται στο παρασκήνιο της εφαρμογής όπως θα αναλύσουμε στην συνέχεια. Η εμπειρία χρήσης της εφαρμογής όμως υλοποιείται από τέσσερις γραφικές διεπαφές, όπως φαίνεται στο Σχήμα 99.

- Η αρχική σελίδα(index page), είναι η βασική γραφική διεπαφή με την οποία αλληλεπιδρά ο χρήστης και μπορεί να μεταβεί μέσω εικονιδίων στις υπόλοιπες. Επίσης, με την μορφή εικονιδίων παρουσιάζεται στον χρήστη αν η δειγματοληψία είναι ενεργή και αν η συσκευή είναι συνδεδεμένη με το διαδίκτυο.
- Η σελίδα καταχώρησης συσκευής(login page) εμφανίζεται αυτόματα στον χρήστη αν κατα την εκκίνηση ή την εκτέλεση της εφαρμογής διαπιστωθεί αυτοματοποιημένα ότι η συσκευή δεν έχει καταχωρηθεί σε κάποιον χρήστη. Η καταχώρηση της συσκευής είναι απαραίτητη, για ξεκινήσει η συλλογή δεδομένων από την συσκευή.

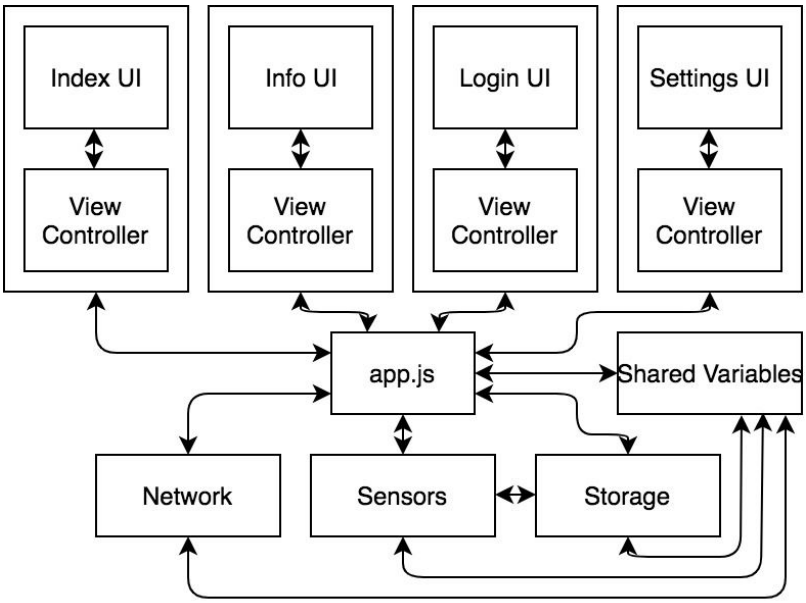
- Η σελίδα ρυθμίσεων της εφαρμογής(settings page) επιτρέπει στον χρήστη με την μορφή κουμπιών να εξαναγκάσει χειροκίνητα την μεταφορά των αποθηκευμένων δεδομένων στον εξυπηρετητή, να αποσυνδέσει την συσκευη απο τον καταχωρημένο χρήστη καθώς και να ολοκληρώσει την εκτέλεση της εφαρμογής.
- Η σελίδα παρουσίασης πληροφοριών της εφαρμογής(info page), παρουσιάζει διάφορες ενημερωτικές πληροφορίες στον χρηστη.



Σχήμα 99: Οντολογική Αναπαράσταση των Διεπαφών της Εφαρμογής Συλλογής Δεδομένων

4.3.2 Υλοποίηση των Απαιτήσεων της Εφαρμογής

Η υλοποίηση της απαιτήσεων της εφαρμογής συλλογής και αποθήκευσης βιοσημάτων και δεδομένων κίνησης από λειτουργικός ανεξάρτητα κομμάτια, όπως φαίνεται στο Σχήμα 100.



Σχήμα 100: Η Τοπολογία των Λειτουργικών Συστημάτων της Εφαρμογής Συλλογής Δεδομένων

Το αρχείο κώδικα app.js αποτελεί την προταρχικότερη λειτουργία του συστήματος για τον λόγο αυτό αποτελεί και ξεχωριστή οντολογία στο σχήμα μας. Συγκεκριμένα κατά την εκτέλεση:

1. Ελέγχετε αν η παρούσα συσκευή έχει καταχωρηθεί σε κάποιον χρήστη, αν όχι εμφανίζεται η διεπαφή για την καταχώρηση της νέας συσκευής.
2. Δρομολογεί την αυτοματοποιημένη συλλογή δεδομένων από τους αισθητήρες.
3. Δρομολογεί την αυτοματοποιημένη αποστολή των συλλεχθέντων δεδομένων μόλις υπάρξει ασφαλής συνδεσιμότητα με τον διαδικτυακό εξυπηρετητη.
4. Εμφανίζει στο προσκήνιο την διεπαφή που επιλέγει κάθε φορά ο χρήστης μέσω της επικοινωνίας της με τους view-controllers κάθε γραφικής διεπαφής.
5. Ενημερώνει τις μοιρασμένες μεταβλητές περιβάλλοντος και τα αρχεία configuration, ώστε όλα οι κώδικες να επικοινωνούν μεταξύ τους χωρίς να εμφανίζονται συνθήκες συναγωνισμού επί των δεδομένων.

```
isDeviceRegistered(function(){
    window.location.replace("login_page.html");
  },function(){
    window.location.replace("index.html");});
sensors.Schedule();
network.Schedule();
environmentVariables.init();
```

Σχήμα 101: Υλοποίηση του app.js

Κάθε γραφική διεπαφή αποτελείται από τα αρχείο αναπαραστασης της(html, css) και του view-controller, οποίος ως κώδικας JavaScript και σε αντιστοιχία με την διαδικτυακή εφαρμογή είναι υπεύθυνος για την δυναμική αλληλεπίδραση της διεπαφής με τον χρήστη και το σύστημα.

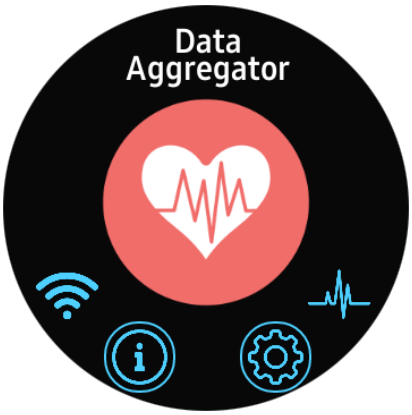
Η οντολογία “Sensors” αποτελείται από κώδικα JavaScript και είναι υπεύθυνη για την χρονοδρομολόγηση(scheduling) και υλοποίηση των συναρτήσεων δειγματοληψίας από τους αισθητήρες. Ταυτόχρονα, παρέχει απευθείας πρόσβαση στις συναρτήσεις αποθήκευσης στην στον τοπικό χώρο της συσκευής για την αποθήκευση των συλλεχθέντων δεδομένων.

Η οντολογία “Network” αποτελείται από κώδικα JavaScript και είναι υπεύθυνη για την ασφαλή μεταφορά των συλλεχθέντων δεδομένων με την συσκευή καθώς και οποιαδήποτε άλλης επικοινωνίας απαιτείται με τον εξυπηρετητή. Αποτελεί το σημείο της εφαρμογής που εκτελεί HTTP REST αιτήματα στον εξυπηρετητή. Ταυτόχρονα, αποτελεί το σημείο που ειδοποιεί το λειτουργικό σύστημα Tizen αυτοματοποιημένα την εφαρμογή σε περίπτωση που η συσκευή βρεθεί εκτός δικτύου με διαδικτυακής πρόσβασης.

Η οντολογία “Storage” αποτελείται από κώδικα JavaScript και είναι υπεύθυνη για την ασφαλή προσωρινή και τοπική αποθήκευση των συλλεχθέντων δεδομένων. Αποτελεί, κρίσιμο κομμάτι της υλοποίησης καθώς υλοποιει ταυτόχρονη πρόσβαση και αποθήκευση δεδομένων στην μνήμη, αποφεύγωντας συνθήκες συναγωνισμού. Ταυτόχρονα, υλοποιεί την απαραίτητη υποδομή για την κωδικοποίηση των δεδομένων σε κατάλληλη μορφή για την μεταφορά τους στον διαδικτυακό εξυπηρετητή.

4.3.2.1 Εκκίνηση Εφαρμογής – Πρόσβαση στην εφαρμογή

Ο χρήστη εκκινώντας την εφαρμογή από το κεντρικό μενού της συσκευής, εισερχεται στην κεντρική οθόνη της εφαρμογής, όπως φαίνεται στο Σχήμα 102. Σε περίπτωση που η συσκευή δεν έχει ακόμη καταχωρηθεί, ανακατευθύνετε αυτόματα στην αντιστοιχεί διεπαφή, όπως είδαμε.



Σχήμα 102: Κεντρική Γραφική Διεπαφή της Εφαρμογής

Η γραφική διεπαφή αποτελείται από δύο εικονίδια που ενημερώνουν των χρήστη για την παρούσα λειτουργικότητα της συσκευής και δύο κουμπιά από τα όποια έχει πρόσβαση στην διεπαφή των πληροφοριών και ρυθμίσεων αντίστοιχα.

Το πρώτο εικονίδιο στα αριστερά της γραφικής διεπαφής ειδοποιεί για την παρούσα κατάσταση της συνδεσιμότητας της συσκευής. Αν η συσκευή βρισκεται υπο την παρουσία δικτύου και είναι συνδεδεμένη με τον διαδικτυακό εξυπηρετητή τότε το εικονίδιο έχει χρώμα μπλε σε αντίθετη περίπτωση έχει χρώμα γκρί και δείχνει ανενεργό.

Το δεύτερο εικονίδιο στα δεξιά της γραφικής διεπαφής ειδοποιεί για την παρούσα κατάσταση της συλλογής δεδομένων από τους αισθητήρες της συσκευής. Αν η εφαρμογή εκτελεί επιτυχώς δειγματοληψία των αισθητήρων τότε το εικονίδιο έχει χρώμα μπλε σε αντίθετη περίπτωση έχει χρώμα γκρί και δείχνει ανενεργό.

Το πρώτο κουμπί στα αριστερά της γραφικής διεπαφής επιτρέπει στον χρήστη να μεταβεί στην διεπαφή πληροφοριών της εφαρμογής, ενώ το δεύτερο κουμπί στα δεξιά της γραφικής διεπαφής, επιτρέπει στον χρήστη να μεταβεί στην διεπαφή ρυθμίσεων της εφαρμογής. Ακολουθεί view-controller της κεντρικής γραφικής διεπαφής στο Σχήμα 103.

```
$("#info_icon").click(function(){ window.location.replace("info_page.html");});
$("#settings_icon").click(function(){window.location.replace("settings_page.html");});
network.on("StateChanged",function(newState){
   $("#network_icon").state(newState);});
sensors.on("StateChanged",function(newState){
   $("#sensors_icon").state(newState);});
```

Σχήμα 103: View-controller της Κεντρικής Γραφικής Διεπαφής

4.3.2.2 Καταχώρηση Νέας Συσκευής σε Χρήστη

Όπως αναφέραμε, σε περίπτωση που η συσκευή δεν έχει καταχωρηθεί σε κάποιον χρήστη, αυτόματα θα ανακατευθυνθεί στην γραφική διεπαφή για την καταχώρηση της, όπως παρουσιάζεται στο Σχήμα 104. Η συλλογή και αποθήκευση δεδομένων από του αισθητήρες της συσκευής είναι ανενεργή έως ότου καταχωρηθεί επιτυχώς η συσκευή στο σύστημα.

Σχήμα 104: Γραφική Διεπαφή Καταχώρησης Συσκευής στο Σύστημα

Για την καταχώρηση της νέας συσκευής smartwatch στο σύστημα, σύμφωνα με τις προδιαγραφές ο χρήστης εισάγει στην άνωθι διεπαφή επι της συσκευής το προσωρινό 8-ψήφιο αναγνωριστικό που βρίσκει στην γραφική διεπαφή του προσωπικού του προφίλ. Μόλις το αναγνωριστικό αυτό χρησιμοποιηθεί ή λήξει ο χρόνος εγκυρότητας του, τότε αυτομάτως δημιουργείται νέο.
Μόλις ο χρήστης εισάγει το αναγνωριστικό η συσκευή εκτελεί ένα αίτημα HTTP POST από τον browser στην διεύθυνση:

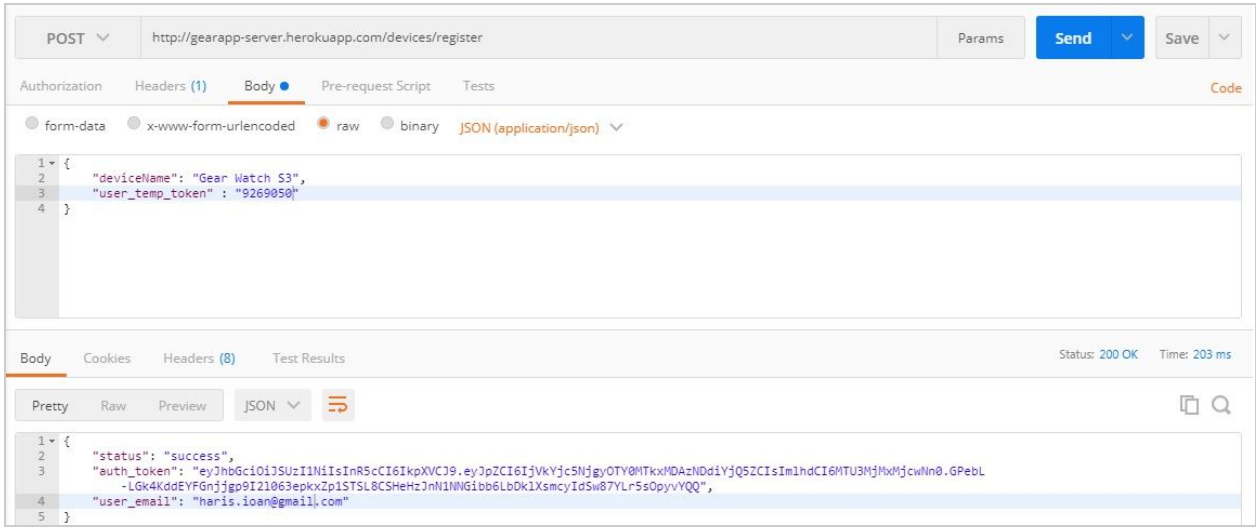
https://gearapp-server.herokuapp.com/devices/register

Στην μεριά της συσκευής ο κώδικάς που εκτελεί το αίτημα παρουσιάζεται στο Σχήμα 105, ενώ υποδειγματικά φαίνεται στο Σχήμα 106 η μορφή του αιτήματος και της απαντησης του απο των εξυπηρετηση, με την χρήση της εφαρμογής Postman.

```
function registerDevice(deviceName, tempKey, successCallback, failureCallback){
    var data = { "device_name": deviceName, "user_temp_token": tempKey };
    $.post(SERVER_URL + "/devices/register", data).done(function (response) {
        successCallback(response);
    }).fail(function (err) {
        failureCallback(err);
    });}
```

Σχήμα 105: Υλοποίηση Αιτήματος Καταχώρησης Συσκευής στο Tizen

Παρατηρούμε στον άνωθι κώδικα την κωδικοποίηση σε αντικείμενο JSON του αιτήματος καθώς και την συνάρτηση που εκτελεί το αίτημα HTTP POST στον εξυπηρετητή. Επίσης φαίνεται και η διαχείριση της απαντησης που λαμβάνετε από τον server μέσω callback συναρτήσεων. Παρατηρούμε επίσης ότι στην απάντηση του αιτήματος από τον server, στο Σχήμα 106 επιστρέφετε το αναγνωριστικό(auth-token) με το οποίο ο εξυπηρετητής θα επαληθεύει την ταυτότητα της συσκευής και θα εξουσιοδοτεί τα αιτήματα της από εδώ και πέρα.



Σχήμα 106: Αίτημα Καταχώρησης Συσκευής μέσω Postman

Στον εξυπηρετητή, βλεπουμε την δρομολόγηση του αιτήματος στο ακόλουθο Σχήμα 107 και την υλοποίηση της συνάρτησης εξυπηρέτησης του *devices.register* στην συνέχεια στο Σχήμα 108.

```
app.route('/devices/register').post(devices.register);
```

Σχήμα 107: Δρομολόγηση Αιτήματος Καταχώρησης Συσκευής

```
exports.register = function (req, res, next) {
   let token = req.body.user_temp_token;
   User.findOne({
       'device.tempToken': token
   }, function (err, user) {
       if (err || !user || user.device.expires < new Date()) {
         return invalidInput(res);
       }
       let device = createDevice(req);
       device.save(function (err, device) {
           if (err) { return invalidInput(res); }
           user.device.tempToken = shortIntId.generate();
           user.device.expires = (new Date()).setHours(
               (new Date()).getHours() + 1);
           user.save(function (err) {
               if(err){ return invalidInput(res); }
           });
           res.setHeader('Content-Type', 'application/json');
           return res.status(200).send({
               status: 'success',
               auth_token: jwtservice.sign({id: device._id}),
               user_email: user.email
   });});});});};
```

Σχήμα 108: Εξυπηρέτηση Αιτήματος Καταχώρισης Συσκευής

Στην εκτέλεση της συνάρτησης εξυπηρέτησης του αιτήματος καταχώρισης νέας συσκευής χρήστη αρχικά λαμβανουμε το προσωρινό 8-ψήφιο αναγνωριστικό που εισήγαγε στην συσκευή ο χρήστης και κωδικοποιήθηκε σε αντικείμενο JSON. Στην συνέχεια, αναζητούμε σε ποιόν χρήστη ανήκει το αναγνωριστικό, καλώντας την συνάρτηση `User.findOne` της βιβλιοθήκης Mongoose επί της δομής των χρηστών της βάσης δεδομένων MongoDB.

Εφόσων βρεθεί επιτυχώς ο χρήστης της συσκευής, δημιουργούμε το αντικείμενο την νέας συσκευής με την συνάρτηση `createDevice` και το αποθηκεύουμε στην βάση δεδομένων, όπως φαίνεται στο Σχήμα 109.

Έπειτα, δημιουργούμε νέο προσωρινό 8-ψήφιο αναγνωριστικό για τον χρήση σε περίπτωση που θελήσει να καταχωρήσει νέα συσκευή. Τέλος, δημιουργούμε το json-web-token

σύμφωνα τους κανόνες που έχουμε περιγράψει στην ενότητα **2.2.5.2.6** και το αποστέλλουμε στην συσκευή που θα χρησιμοποιείται από τον εξυπηρετητή για την επαλήθευση την ταυτότητας της συσκευής και την εξουσιοδότηση των αιτημάτων της.

Σχήμα 109: Εγγραφή Συσκευής στην Βάση Δεδομένων MongoDB

4.3.2.3 Επαλήθευση Ταυτότητας και Εξουσιοδότηση Συσκευής

Έχουμε προδιαγράψει η επικοινωνία της συσκευή με τον διαδικτυακό εξυπηρετητή υλοποιείται με HTTP Post αιτήματα. Για την πιστοποίηση και εξουσιοδότηση των αιτημάτων χρησιμοποιουμε json-web-tokens(JWT), ένα αναγνωριστικό δηλαδη στην κεφαλίδα του αιτήματος το οποίο πιστοποιεί την συσκευή από την οποία προέρχεται.

Όπως αναφέραμε στην προηγούμενη ενότητα, κατα την επιτυχή εγγραφή της συσκευής, λαμβάνετε ως απαντηση απο τον server το αναγνωριστικό το οποίο τοποθετεί η εφαρμογή σε όλα τα αιτήματα της από εκείνο το σημείο και μετά για την ταυτοποίηση και την εξουσιοδότηση τους.

Συγκεκριμένα, με την λήψη του αναγνωριστικού απο τον εξυπηρετητή η εφαρμογή το αποθηκεύει στην τοπική μνήμη της συσκευής κρυπτογραφημένα απο το λειτουργικό σύστημα, όπως φαίνεται στο ακόλουθο κομμάτι κώδικα Σχήμα 110.

```
SecureStorage.setSecureKey('auth_token', response.auth_token, function(){
   successCallback();
}, function(){
   failureCallback();
});
```

Σχήμα 110: Υλοποίηση Αποθήκευσης JWT Τοπικά στην Συσκευή

4.3.2.4 Αποσύνδεση Συσκευής από Χρήστη

Ακολουθώντας την λογική της υλοποίησης που ακολουθήθηκε στην αποσύνδεση απλού χρήστη και διαχειριστή συστήματος, η αποσύνδεση συσκευής χρήστη αποτελεί αντίστοιχη διαδικασία.

Σχήμα 111: Γραφική Διεπαφή Ρυθμίσεων Εφαρμογής

Ο χρήστης για να αποσυνδέσει την συσκευή του απο το σύστημα, πλοηγείται στην γραφική διεπαφή των ρυθμίσεων της εφαρμογής, όπως έχουμε περιγράψει. Πατώντας το αντίστοιχο κουμπί, όπως φαίνεται στο άνωθι Σχήμα 111, η εφαρμογή εκτελεί ένα αίτημα HTTP POST από τον browser στην διεύθυνση:

https://gearapp-server.herokuapp.com/devices/register

Ο εξυπηρετητής αντίστοιχα με ότι έχουμε ήδη παρουσιάσει ακυρώνει το JWT αναγνωριστικό και η εφαρμογή που εκτελείται επι της συσκευής το διαγράφει από την μνήμη του συστήματος.

4.3.2.5 Λήψη Δεδομένων Κίνησης και Βιοσημάτων απο τους Αισθητήρες

Ο κεντρικός λειτουργικος πυρήνας της εφαρμογής είναι η συλλογή δεδομένων από τους αισθητήρες της εφαρμογής. Όπως έχουμε προδιαγράψει, στην παρούσα υλοποίηση χρησιμοποιούμε τους εξής αισθητήρες: επιταχυνσιόμετρο, γυροσκόπιο, G.P.S. και αισθητήρα μέτρησης καρδιακού ρυθμού. Η δειγματοληψία των αισθητήρων σε όσο το δυνατόν μεγαλύτερη συχνότητα είναι στενα συνδεδεμένη με την ποσότητα πληροφορίας που συλλέγουμε για τον κάθε χρήστη, ταυτόχρονα όμως αυξάνεται και η κατανάλωση ενέργειας από την συσκευή. Έτσι λοιπόν επιλέξαμε να δειγματοληπτούμε τους αισθητήρες της συσκευής σε συχνότητες που αποφέρουν dataset με

σημαντική ερευνητική αξία ενώ παράλληλα, επιτρέπουν στην συσκευή να συλλέγει συνεχόμενα δεδομένα για περισσότερες από εικοσιτέσσερις ώρες.

Στον ακόλουθο πίνακα φαίνονται οι συχνότητες δειγματοληψίας που τελικά χρησιμοποιούνται από την εφαρμογή της συσκευής.

Sensor	Accelerometer	Gyroscope	Heart Rate	G.P.S.
Sampling Rate (ms)	50	50	60000(1 minute)	3600000(1 hour)

Πίνακας 3: Συχνότητες Δειγματοληψίας της Εφαρμογής από τους Αισθητήρες της Συσκευής

Όσον αφορά την υλοποίηση της δειγματοληψίας των αισθητήρων, στο ακόλουθο Σχήμα 112 παρουσιάζουμε ενδεικτικά την υλοποίηση για την δειγματοληψία απο επιταχυνσιόμετρο. Συγκεκριμένα, ο αισθητήρας αποτελεί ένα αντικείμενο(*accelerometerSensor*) της γλώσσας JavaScript το οποίο κληρονομεί διάφορα στοιχεία και λειτουργίες από το λειτουργικό σύστημα μέσω της χρήσης του SDK(*tizen.sensorservice.getDefaultSensor()*). Καλώντας την συνάρτηση *start* θέτουμε σε λειτουργία την δειγματοληψία του αισθητήρα, με παράμετρο την μεταβλητή *ACCELEROMETER_SAMPLING* η οποία ορίζει την συχνότητα δειγματοληψίας. Κάθε φορά που λαμβάνετε μια μέτρηση καλείται από το λειτουργικό σύστημα η εμφωλευμένη callback συνάρτηση ***function***(*sensorData*), με την μέτρηση να βρίσκεται εντός της μεταβλητής *sensorData*. Στην συνέχεια, καλώντας την συνάρτηση *DB.createAccelerometerItem* αποθηκεύουμε στην τοπική βάση δεδομένων την μέτρηση που λήφθηκε από το επιταχυνσιόμετρο, αποτελούμενη απο τις χωρικές επιταχύνσεις και την χρονοσήμανση της. Με όμοιο τρόπο υλοποιήθηκε η δειγματοληψία απο τους αισθητήρες του γυροσκοπίου, G.P.S. και καρδιακού ρυθμού.

```
var accelerometerSensor = tizen.sensorservice.getDefaultSensor('LINEAR_ACCELERATION');
    accelerometerSensor.start(function(){
        accelerometerSensor.setChangeListener(function(sensorData){
            DB.createAccelerometerItem({
                x: (sensorData.x).toFixed(2),
                y: (sensorData.y).toFixed(2),
                z: (sensorData.z).toFixed(2),
                timestamp: Date.now()
             }, successCallback);
         }, ACCELEROMETER_SAMPLING,0);});};
```

Σχήμα 112: Υλοποίηση Δειγματοληψίας Αισθητήρα Επιταχυνσιομέτρου στην Συσκευή

4.3.2.6 Τοπική Αποθήκευση Δεδομένων

Η τοπική προσωρινή αποθήκευση των δεδομένων υλοποιείται με την τεχνολογία indexedDB που παρουσιάσαμε στην ενότητα. Βασικό της χαρακτηριστικό είναι η αποθήκευση των δεδομένων της σε μορφή JSON, το οποίο μας επιτρέπει την απευθείας αποθήκευση των συλλεγόμενων δεδομένων χωρίς την ανάγκη μετατροπής τύπων των δεδομένων.

Στα πλαίσια της εφαρμογής μας υλοποιήσαμε τις απαραίτητες συναρτήσεις για την αποθήκευση των συλλεγόμενων δεδομένων αλλά και την ανάκτηση τους, χρησιμοποιώντας τις διεπαφές της indexedDB. Στο ακόλουθο Σχήμα 113 παρουσιάζουμε ενδεικτικά την υλοποίηση των συναρτήσεων για την αποθήκευση και ανάκτηση δεδομένων επιταχυνσιομέτρου, μιας και οι υλοποιησεις για τα δεδομένα των υπόλοιπων αισθητήρων ακολουθούν με όμοιο τρόπο. Οι συναρτήσεις που καλούνται απο το υπολοιπο σύστημα για τα δεδομένα του επιταχυνσιομέτρου είναι η `createAccelerometerItem` που αποθηκεύει μια νέα μέτρηση και η `fetchAllAccelerometer` που ανακαλεί όλα τα αποθηκευμένα δεδομένα που έχουν συλλεχθεί από τον αισθητήρα.

Η κλήση της συνάρτησης `createAccelerometerItem` καλεί με την σειρά της την `createItem`, η οποία αποθηκεύει το νέο δεδομένο στην συλλογή `'AccelerometerModel'` και κάνει γνωστό στον κώδικα εκτέλεσης με την χρήση callback για το αποτέλεσμα της διεργασίας.

Αντίστοιχα, η κλήση της συνάρτησης `fetchAllAccelerometer` καλεί με την σειρά της την `fetchAll`, η οποία ανακαλεί τα δεδομένα που έχουν αποθηκευτεί στην συλλογή `'AccelerometerModel'` και κάνει γνωστό στον κώδικα εκτέλεσης με την χρήση callback για το αποτέλεσμα της διεργασίας.

```
var DB = (function() {
  var tempDB = {};
  var datastore = null;
  tempDB.createAccelerometerItem = function(data, callback){
      createItem(data, 'AccelerometerModel', callback);
  };
  function createItem(item, store, callback) {
      var db = datastore;
      var transaction = db.transaction([store], 'readwrite');
      var objStore = transaction.objectStore(store);
      var request = objStore.put(item);
      request.onsuccess = function(e) { callback(item); };
      request.onerror = tempDB.onerror;
  };
```

```
    tempDB.fetchAllAccelerometer = function(callback) {
        fetchAll('AccelerometerModel', callback);
    };
    function fetchAll(store, callback) {
        var db = datastore;
        var transaction = db.transaction([store], 'readwrite');
        var objStore = transaction.objectStore(store);
        var keyRange = IDBKeyRange.lowerBound(0);
        var cursorRequest = objStore.openCursor(keyRange);
        var results = [];
        cursorRequest.onsuccess = function(e) {
            var result = e.target.result;
            if (!!result == false) { return; }
            results.push(result.value);
            result.continue();
        };
        cursorRequest.onerror = tempDB.onerror;
        transaction.oncomplete = function(e) {
            callback(results);
        };
    };
});
```

Σχήμα 113: Υλοποίηση Αποθήκευσης Δεδομένων Επιταχυνσιομέτρου στην Τοπική Μνήμη της Συσκευής

4.3.2.7 Προβολή Πληροφορίων Περιβάλοντος Εκτέλεσης της Εφαρμογής

Ο χρήστης μπορεί να πλοηγηθεί μέσω του αντίστοιχου εικονιδίου της κεντρικής γραφικής διεπαφης, στην σελίδα πληροφοριών της εφαρμογής, όπως φαίνεται στο Σχήμα 114. Σε αυτήν την γραφική διεπαφή ο χρήστης μπορεί να δει:

1. Την τρέχουσα έκδοση του λογισμικού που εκτελεί η εφαρμογή.
2. Την διεύθυνση του χρησιμοποιούμενου εξυπηρετητή για τον συγχρονισμό των δεδομένων.
3. Το τρέχον μέγεθος της τοπικής βάσης δεδομένων που χρησιμοποιείται και αποθηκεύει τα συλλεχθέντα δεδομένα

Σχήμα 114: Γραφική Διεπαφή Πληροφοριών Εφαρμογής

4.3.2.8 Συγχρονισμός Δεδομένων Κίνησης και Βιοσημάτων με τον Εξυπηρετητή

Η μεταφορά των δεδομένων στον server, εκτελείτε ανα τακτά χρονικά διαστήματα με την παρουσία δικτύου. Όπως παρουσιάσαμε στην υλοποίηση του αρχείου κώδικα app.js η εφαρμογή ειδοποιείται απο το λειτουργικό σύστημα μόλις η συσκευή συνδεθεί με το διαδίκτυο. Σε περίπτωση που η συσκευή είναι διαρκώς συνδεδεμένη με το δίκτυο, τα δεδομένα μεταφέρονται στον server ανα μερικές ώρες. Με τον τρόπο αυτόν εξοικονομούμε ενέργεια στην συσκευή, αφού δεν επεξεργαζόμαστε τα δεδομένα για να μεταφερουμε μερικές πλειάδες σε σύντομα χρονικά διαστήματα. Μόνο όταν έχουν συγκεντρωθεί αρκετές χιλιάδες δεδομένα επιτελείτε η επεξεργασία και αποστολή τους, ώστε να δικαιολογηθεί η κατανάλωση ενέργειας του επεξεργαστή.

Η υλοποίηση της συνάρτησης μεταφοράς δεδομένων `uploadBiosignals`, παρουσιάζεται στον ακόλουθο κώδικα στο Σχήμα 115. Περιορίζουμε την ανάλυση μας όμως μόνο στην αποστολή των δεδομένων από το επιταχυνσιόμετρο, αφού οι υλοποιήσεις της αποστολής των υπόλοιπων τύπων δεδομένων από τους αισθητήρες είναι όμοια.

Αρχικά, με την συνάρτηση `isDeviceOnLine()` ελέγχουμε αν η συσκευή βρίσκεται εντός δικτύου, σύγχρονα αυτή την φορά αφου ετοιμαζόμαστε να στείλουμε δεδομένα. Στην συνεχεια με την συνάρτηση `getServerToken` ανακαλεί από την μνήμη το αναγνωριστικό(JWT) με το οποίο πιστοποιούνται τα αιτήματα προς τον εξυπηρετητή, όπως έχουμε αναλύσει. Έπειτα καλώντας τις συναρτήσεις `DB.open` και `DB.fetchAllAccelerometer` ανακαλούμε όλα τα μέχρι τώρα αποθηκευμένα δεδομένα από την τοπική βάση δεδομένων. Η κλήση της συνάρτησης αποσταλεί όλα τα δεδομένα στον εξυπηρετητή, ενώ σε περίπτωση που το αίτημα είναι ιδιαίτερα μεγάλο, το διαμερίζει σε μικρότερα ώστε να αποφευχθεί ο κίνδυνος απώλειας δεδομένων. Τέλος, με την λήψη επιτυχούς μηνύματος από τον εξυπηρετητή διαγράφουμε τα δεδομένα που αποστείλαμε από την τοπική μνήμη.

```
var Upload = (function() {
    var tempUpload = {};
    tempUpload.uploadBiosignals = function(){
        if(isDeviceOnLine()){
            getServerToken(function(token){
                    DB.open(function(){});
                    DB.fetchAllAccelerometer(function(results){
                        if(results !== undefined && results.length>0){
                            uploadBiosignalsWithRequest("accelerometer",
                             results, token, function(){
                                DB.deleteAllAccelerometerItems(function(){
                                    console.log("Deleted acc data from db")
                                });
                            }, function(err){
                                onError(err)
                            });
                        }else{
                            onError("ACC: db_error");
     });});});};
    return tempUpload; }());
```

Σχήμα 115: Υλοποίηση Αποστολής Δεδομένων Επιταχυνσιομέτρου στον Server του Συστήματος

5 Αξιολόγηση και Συμπεράσματα

Σε αυτό το κεφάλαιο, αρχικά θα συνοψίσουμε το τελικό σύστημα και τα χαρακτηριστικά του. Στην συνέχεια θα παρουσιάσουμε παρατηρήσεις και συμπεράσματα αναφορικά με την χρησιμότητα και την λειτουργικότητα της εφαρμογής που προέκυψαν κατά την υλοποίηση της. Τέλος, θα αναφέρουμε μελλοντικά προτεινόμενες επεκτάσεις της πλατφόρμας.

5.1 Σύνοψη Τελικού Συστήματος

Στην παρούσα διπλωματική εργασία, αναπτύχθηκε μια υβριδική πλατφόρμα συλλογής, ανάλυσης βιοσημάτων και δεδομένων κίνησης χρηστών από συσκευές smartwatch. Το σύστημα σχεδιάστηκε με τέτοιο τρόπο ώστε να καλύπτει όλες τις λειτουργικές προδιαγραφές που τέθηκαν για από τους χρήστες του συστήματος. Η πλατφόρμα συνολικά αποτελείται από δύο συνιστώσες που λειτουργούν συμπληρωματικά, την εφαρμογή συλλογής και προσωρινής αποθήκευσης δεδομένων και την διαδικτυακή εφαρμογή αποθήκευσης και διαχείρισης δεδομένων, χρηστών και των συσκευών τους.

Η πρώτη συνιστώσα του συστήματος ήταν η υλοποίηση της εφαρμογής συλλογής και προσωρινής αποθήκευσης δεδομένων χρηστών στην συσκευή Samsung Gear S3 Frontier. Το σχετικά χαμηλό κόστος της συσκευής σε σύγκριση με τις δυνατότητες λήψης σημάτων απο ετερογενείς αισθητήρες μέτρησης καρδιακού ρυθμού, επιταχυνσιομέτρου, γυροσκοπίου, G.P.S αλλά και συνδεσιμότητας με το διαδίκτυο έκαναν την επιλογή της ιδανική. Ταυτόχρονα λόγω του γεγονότος ότι ο χρήστης φορά ως ρολόι(smartwatch) μια τόσο ισχυρά δειγματοληπτική συσκευή, αναδεικνύει την συλλογή δεδομένων ανά τακτά χρονικά διαστήματα σε συνθήκες καθημερινότητας και όχι εργαστηρίου. Κατ'αυτόν τον τρόπο επιτύχαμε έναν από τους βασικότερους στόχους του συστήματος, την δημιουργία συνόλου δεδομένων(dataset) για την περαιτέρω ανάλυση συμπεριφοράς χρηστών σε συνθήκες καθημερινότητας. Παράλληλα, η υλοποίηση της εφαρμογής επί της συσκευής χρησιμοποιώντας τεχνολογίες και frameworks διαδικτύου(web), κατέστησε δυνατή την παροχή μιας ολοκληρωμένα γραφικά και λειτουργικά εμπειρίας χρήστη. Έτσι, χρήστες χωρίς πρότερη εμπειρία χρήσης του συστήματος μπορούν και επιτελούν τις προδιαγεγραμμένες λειτουργίες χωρίς να απαιτείται εκτενές χρονικό διάστημα εκμάθησης και εξοικείωσης τους με το σύστημα. Η αυτοματοποίηση διαδικασιών όπως αυτή της συλλογής δεδομένων και της μεταφοράς τους στην κεντρική βάση δεδομένων, επιτρέπει την εύκολη χρήση του συστήματος από όσο το δυνατόν περισσότερες κοινωνικές ομάδες χρηστών.

Η δεύτερη συνιστώσα του συστήματος ήταν η υλοποίηση της διαδικτυακής εφαρμογής μόνιμης αποθήκευσης δεδομένων για την μετέπειτα ανάλυση τους. Η χρήση σύγχρονων τεχνολογιών, frameworks και αρχιτεκτονικών κατέστησε την δημιουργία της διαδικτυακής υπηρεσίας σε σύντομο χρονικό διάστημα, καλύπτοντας ταυτόχρονα όλες τις λειτουργικές προδιαγραφές. Συγκεκριμένα, υλοποιήθηκε σε τεχνολογία Node.js ο κεντρικός εξυπηρετητής, υπεύθυνος για την επικοινωνία και διασύνδεση με τις συσκευές χρηστών, την παροχή των

γραφικών διεπαφών για την διαχείριση των χρηστών και των δεδομένων τους. Χρησιμοποιήθηκε ο συνδυασμός σχεσιακών(SQL) και μη-σχεσιακών βάσεων(MongoDB) δεδομένων για την αποθήκευση των ετερογενών δεδομένων χρηστών και των λοιπών δεδομένων του συστήματος. Έτσι λοιπόν παρέχεται η δυνατότητα στους απλούς χρήστες της αυτοματοποιημένης συλλογής, οπτικοποίησης και διαχείρισης των δεδομένων τους. Ταυτόχρονα όμως, παρέχεται η δυνατότητα στους διαχειριστές του συστήματος της συγκεντρωτικής(aggregated) συλλογής δεδομένων από το σύστημα για την περαιτέρω ερευνητική επεξεργασία και εξαγωγή συμπερασμάτων.

Το σύστημα, χρησιμοποιούμενο στα πλαίσια ερευνητικού προγράμματος, για διάστημα τεσσάρων μηνών συνέλεξε πληθώρα δεδομένων από τις συσκευές χρηστών και χρησιμοποιήθηκε εκτενώς από πραγματικούς χρήστες, καλύπτοντας και υπερβαίνοντας τις αρχικές προσδοκίες και στόχους, πράγμα που φέρει την παρούσα διπλωματική σε επιτυχία.

5.2 Παρατηρήσεις και Συμπεράσματα

Κατά την υλοποίηση των λειτουργικών προδιαγραφών του συστήματος και μέσα από την εκτεταμένη χρήση του σε πραγματικό περιβάλλον χρήσης, προέκυψαν ορισμένες παρατηρήσεις και συμπεράσματα αναφορικά με την χρησιμότητα και την λειτουργικότητα του.

Το πρώτο αξιόλογο πλεονέκτημα του συστήματος είναι η ολιστική συλλογή δεδομένων και η επεκτασιμότητα του. Σύμφωνα με τα στοιχεία βιβλιογραφίας που παρουσιάσαμε, το πλήθος των εφαρμογών για απομακρυσμένη συλλογή και παρακολούθηση των βιοσημάτων των χρηστών, έχει αυξηθεί σημαντικά τα τελευταία χρόνια. Στα πλαίσια της εργασίας αυτής, δημιουργήσαμε ένα ενοποιημένο σύστημα συλλογής δεδομένων από διαφορετικές πηγές δεδομένων. Επι του παρόντος, χρησιμοποιήσαμε μόνο ένα είδος smartwatch συσκευής, παρ'όλα αυτά ακολουθώντας συγκεκριμένες τεχνολογικές αποφάσεις στην υλοποίηση των διεπαφών επικοινωνίας, δείξαμε ότι οποιαδήποτε συσκευή υλοποιεί το προδιαγεγραμμένο πρωτόκολλο επικοινωνίας με την εφαρμογή μπορεί εύκολα να συλλέξει δεδομένα εκμεταλεύσιμα απο την πλατφόρμα. Παράλληλα, λόγο των αρχιτεκτονικών(REST) αποφάσεων που πάρθηκαν, αλλά και των τεχνολογιών(Node.js, Postgres, MongoDB) που χρησιμοποιήσαμε η εφαρμογή χαρακτηρίζεται από σημαντική επεκτασιμότητα. Είναι εύκολο δηλαδή, επι του αρχικού συστήματος να προστεθούν επιπρόσθετες λειτουργίες χωρίς να επηρεαστεί η ορθή του λειτουργία ή να παρουσιαστούν προβλήματα συμβατότητας. Το χαρακτηριστικό αυτό είναι επιθυμητό και εντός των αρχικών προδιαγραφών του συστήματος, αφού εν γένει θέλουμε να προσθέτουμε διαρκώς νέες κατηγορίες βιοσημάτων και δεδομένων που σχετίζονται με το επίπεδο διαβίωσης των χρηστών. Δεν πρέπει επίσης να παραλείψουμε το γεγονός ότι καθώς αυξάνονται οι ετερογενείς πηγές δεδομένων, αυξάνεται και το ερευνητικό ενδιαφέρον για την ανάλυση αυτών, καθιστώντας την γρήγορη και ομαλή επέκταση του συστήματος σημαντική.

Το δεύτερο αξιόλογο πλεονέκτημα του συστήματος είναι η κλιμακωσιμότητα. Το τεχνικό και λειτουργικό υπόβαθρο της εφαρμογής υλοποιήθηκε σε περιβάλλον που επιτρέπει την διαρκή του κλιμάκωση, χωρίς την απαίτηση νέας υλοποίησης του συστήματος. Φιλοξενώντας τον εξυπηρετητή σε περιβάλλον υπολογιστικού νέφους, μπορούμε όπως έχουμε αναφέρει να

δημιουργούμε νέα στιγμιότυπα διεργασιών σε περίπτωση που οι ήδη υπάρχουσες είναι κατειλημμένες. Σε χρονικές περιόδους που η χρήση της υπηρεσίας είναι αυξημένη, μπορούμε δυναμικά να αντιμετωπίσουμε το εκτεταμένο επεξεργαστικό φορτίο. Συνδυάζοντας την δυνατότητα αυτή του εξυπηρετητή με το γεγονός του διαχωρισμού των βάσεων δεδομένων απο την λογική επεξεργασίας των δεδομένων, μπορούμε να επεκτείνουμε δυναμικά το σύστημα ώστε αφενός να υπάρχει αρκετός αποθηκευτικός χώρος και αφετέρου να διασφαλίσουμε τον ελάχιστο χρόνο εξυπηρέτησης των αιτημάτων.

Το τρίτο αξιόλογο πλεονέκτημα του συστήματος είναι η ασφάλεια των δεδομένων του. Στην εποχή που τα δεδομένα αποτελούν σημαντικό συγκριτικό πλεονέκτημα τόσο για χρήστες όσο και οργανισμους και ερευνητές, η ασφαλής διαχείριση αυτών αποτελεί καίριο μέλημα του συστήματος. Ειδικότερα όπως παρουσιάσαμε, υιοθετώντας αρχιτεκτονικές και λειτουργικές επιλογές, στα πλαίσια του συστήματος οι χρήστες και οι διαχειριστές έχουν πλήρη έλεγχο επι των δεδομένων. Οποιαδήποτε λειτουργία επιτελείται σε αυτά είναι αυτοματοποιημένη και σε συμφωνία με τις προδιαγραφές κρυπτογράφησης και αποθήκευσης που έχουν τεθεί.

Το τέταρτο αξιόλογο πλεονέκτημα του συστήματος είναι η προσφορά του στην ερευνητική κοινότητα. Συνολικά το σύστημα προσφέρει ένα σημαντικό επίπεδο αφαίρεσης της υλοποίησης συλλογής και διαχείρισης δεδομένων. Παρέχει με αυτόν τον τρόπο στους ερευνητές ένα χρήσιμο εργαλείο, για την εύκολη, ασφαλή και αποδοτική μελέτη των δεδομένων. Σημαντικό γνώρισμα της διαδικασίας αυτής είναι πως η πλατφόρμα αφαιρεί την ανάγκη για οποιαδήποτε τεχνική υλοποίηση, από τον ερευνητή, εξοικονομώντας πόρους σε πολλά επίπεδα.

5.3 Μελλοντικές Επεκτάσεις

Ακολουθώντας τις παρατηρήσεις και τα συμπεράσματα, θεωρούμε ότι η υπάρχουσα υλοποίηση μπορεί να αποτελέσει μια λειτουργική και αξιόπιστη βάση, η οποία μελλοντικά να εμπλουτιστεί με νέες δυνατότητες που θα παρέχουν στους χρήστες μεγαλύτερη προστιθέμενη αξία.

Σημαντική επέκταση της λειτουργικότητας και χρησιμότητας του συστήματος είναι η αύξηση των συλλεγόμενων δεδομένων που συλλέγει η πλατφόρμα. Στην παρούσα υλοποίηση συλλέγετε το βιοσήμα του καρδιακού ρυθμού καθώς και τα δεδομένα κίνησης απο επιταχυνσιόμετρο, γυροσκόπιο και G.P.S. Στο εμπόριο όμως είναι ευρέως διαθέσιμες “έξυπνες” συσκευές που μπορούν να ποσοτικοποιήσουν μια μεγαλύτερη πληθώρα βιοσημάτων. Η προσθήκη περισσότερων πηγών δεδομένων και κατ’επέκταση νέων συσκευών που υλοποιούν το πρωτόκολλο επικοινωνίας του συστήματος, αποτελεί μια φυσική τάση της εφαρμογής.

Σημαντική επέκταση της λειτουργικότητας και χρησιμότητας του συστήματος είναι η μελλοντική υποστήριξη δυναμικής στατιστικής ανάλυσης των δεδομένων των χρηστών. Δεδομένου του πλήθους δεδομένων που συλλέγονται από το σύστημα η εφαρμογή θα μπορούσε να αξιοποιήσει μεθόδους μηχανικής μάθησης(machine learning), ώστε να αναλύει σε βάθος την κινητικότητα και την συμπεριφορά των χρηστών.

6 Βιβλιογραφία

[1] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Activities_of_daily_living [Accessed: 10-Dec-2018]
[2] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Accelerometer [Accessed: 10-Dec-2018]
[3] Tizen Documentation, [Online]. Available: https://developer.tizen.org/zh-hans/development/guides/native-application/location-and-sensors/device-sensors?langredirect=1#gravity [Accessed: 10-Dec-2018]
[4] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Gyroscope [Accessed: 10-Dec-2018]
[5] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System [Accessed: 10-Dec-2018]
[6] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Photoplethysmogram [Accessed: 10-Dec-2018]
[7] Department of Engineering Science, The University of Oxford, [Online]. Available: https://www.robots.ox.ac.uk/~neil/teaching/lectures/med_elec/notes6.pdf [Accessed: 10-Dec- 2018]
[8] Wikipedia, [Online]. Available: https://el.wikipedia.org/wiki/Καρδιά [Accessed: 10-Dec-2018]
[9] Swan, Melanie. "Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0." Journal of Sensor and Actuator Networks 1.3 (2012): 217-253.
[10] Wired.com: Know Thyself: Tracking Every Facet of Life, from Sleep to Mood to Pain, 24/7/365, [Online]. Available: https://www.wired.com/2009/06/lbnp-knowthyself/ [Accessed: 10-Dec-2018]
[11] Wild D, Nayak US, Isaacs B (1981) How dangerous are falls in old people at home? Br Med J(Clin Res Ed) 282(6260): 266–268.
[12] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things [Accessed: 10-Dec-2018]
[13] Gomez, Carles, Joaquim Oller, and Josep Paradells. "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology." Sensors 12.9 (2012): 11734-11753.
[14] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cloud_computing [Accessed: 10-Dec-2018]
[15] Samsung, [Online]. Available: https://www.samsung.com/us/mobile/wearables/smartwatches/samsung-gear-s3-frontier-sm-r760ndaaxar/ [Accessed: 10-Dec-2018]
[16] Wikipedia, [Online]. Available: https://el.wikipedia.org/wiki/Web_browser [Accessed: 10-Dec-2018]
[17] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Chrome_V8 [Accessed: 10-Dec-2018]

[18] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Node.js [Accessed: 10-Dec-2018]
[19] Express.js, [Online]. Available: https://expressjs.com/ [Accessed: 10-Dec-2018]
[20] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [Accessed: 10-Dec-2018]
[21] TutorialsPoint, [Online]. Available: https://www.tutorialspoint.com/http/http_requests.htm [Accessed: 10-Dec-2018]
[22] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer [Accessed: 10-Dec-2018]
[23] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Application_programming_interface [Accessed: 10-Dec-2018]
[24] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Ajax_(programming) [Accessed: 10-Dec-2018]
[25] Medium, [Online]. Available: https://medium.com/dev-bits/a-guide-for-adding-jwt-token-based-authentication-to-your-single-page-nodejs-applications-c403f7cf04f4 [Accessed: 10-Dec-2018]
[26] Medium, [Online]. Available: https://medium.com/@paul.senon/node-express-js-cookies-set-get-secure-884311606148 [Accessed: 10-Dec-2018]
[27] MDN Web Docs: About Javascript, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript [Accessed: 10-Dec-2018]
[28] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Event-driven_programming [Accessed: 10-Dec-2018]
[29] Medium, [Online]. Available: https://medium.com/front-end-weekly/javascript-event-loop-explained-4cd26af121d4 [Accessed: 10-Dec-2018]
[30] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/JSON [Accessed: 10-Dec-2018]
[31] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/JSON#Using_JSON_in_JavaScript [Accessed: 10-Dec-2018]
[32] OpenmHealth.org: [Online]. Available: http://www.openmhealth.org/ [Accessed: 10-Dec-2018]
[33] OpenmHealth.org: [Online]. Available: http://www.openmhealth.org/documentation/#/schema-docs/overview [Accessed: 10-Dec-2018]
[34] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/PostgreSQL [Accessed: 10-Dec-2018]
[35] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/MongoDB [Accessed: 10-Dec-2018]

[36] Mozilla, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API [Accessed: 10-Dec-2018]
[37] Medium, [Online]. Available: https://medium.com/@codeshifu/object-relational-mapping-concepts-e2ff0838590c [Accessed: 10-Dec-2018]
[38] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/HTTPS [Accessed: 10-Dec-2018]
[39] Tizen Studio, [Online]. Available: https://developer.tizen.org/ko/development/tizen-studio/download [Accessed: 10-Dec-2018]
[40] Jetbrains.com: [Online]. Available: https://www.jetbrains.com/webstorm/ [Accessed: 10-Dec-2018]
[41] Postman.com: [Online]. Available: https://www.getpostman.com/postman [Accessed: 10-Dec-2018]
[42] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Git [Accessed: 10-Dec-2018]
[43] Bitbucket.com: [Online]. Available: https://bitbucket.org/product/ [Accessed: 10-Dec-2018]
[44] pgAdmin.com: [Online]. Available: https://www.pgadmin.org/ [Accessed: 10-Dec-2018]
[45] Maglogiannis, Ilias, Ioannou, Charalampos, Tsanakas, Panayiotis:"Fall detection and activity identification using wearable and hand-held devices". Integrated Computer-Aided Engineering, vol. 23, no. 2, pp. 161-172, (2016)
[46] Ilias Maglogiannis, Charalampos Ioannou, George Spyroglou, Panayiotis Tsanakas: "Fall Detection Using Commodity Smart Watch and Smart Phone":" Artificial Intelligence Applications and Innovations 2014", Conference Records, Springer Volume 436(2014)
[47] Alban Maxhuni, Angélica Muñoz-Meléndez, Venet Osmani, Humberto Perez , Oscar Mayora , Eduardo F. Morales: "Classification of bipolar disorder episodes based on analysis of voice and motor activity of patients". Elsevier (2015)
[48] Simon Rosenbaum, Anne Tiedemann, Catherine Sherrington, Hidde P van der Ploeg: "Assessing physical activity in people with posttraumatic stress disorder: feasibility and concurrent validity of the International Physical Activity Questionnaire– short form and actigraph accelerometers". Biomedcentral (2014)
[49] Heroku.com: [Online]. Available: https://www.heroku.com/ [Accessed: 10-Dec-2018]
[50] Wikipedia: [Online]. Available: https://en.wikipedia.org/wiki/Tizen [Accessed: 10-Dec-2018]
[51] Okeanos: [Online]. Available: https://okeanos.grnet.gr/home/ [Accessed: 10-Dec-2018]
[52] Sequelize.com: [Online]. Available:

https://sequelize.org/ [Accessed: 10-Dec-2018]
[53] MongooseJS.com: [Online]. Available:
http://mongoosejs.com/ [Accessed: 10-Dec-2018]