ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνολογίες Διαδικτύου για την Συλλογή και Ανάλυση Προσωπικών Δεδομένων Υγείας & Ευεξίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΟΥΣΙΑΔΑ ΔΗΜΗΤΡΙΟΥ

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΈΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνολογίες Διαδικτύου για την Συλλογή και Ανάλυση Προσωπικών Δεδομένων Υγείας & Ευεξίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΟΥΣΙΑΔΑ ΔΗΜΗΤΡΙΟΥ

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Τρίτη 19 Σεπτεμβρίου, 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Παναγιώτης Τσανάκας Καθηγητής Ε.Μ.Π.	Διονύσιος Δ. Κουτσούρης Καθηγητής Ε.Μ.Π.	Κιαμάλ Πεκμεστζή Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2017

(Υπογραφή)

.....

ΧΟΥΣΙΑΔΑΣ ΔΗΜΗΤΡΙΟΣ

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

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας ήταν ο σχεδιασμός και η ανάπτυξη μιας διαδικτυακής εφαρμογής με στόχο τη συλλογή, από διαφορετικές πηγές, δεδομένων που αφορούν την υγεία και την ευεξία του χρήστη, καθώς και την ανάλυσή τους, μέσω της οποίας να μπορεί να αναδειχθεί η ιδιοπροσωπία του, εν σχέσει με ευρύτερα σύνολα.

Υπάρχουν δύο, κυρίως, λόγοι που καθιστούν την ανάπτυξη μιας τέτοιας εφαρμογής χρήσιμη και επίκαιρη. Αφενός, η συνεχώς αυξανόμενη τάση για ποσοτικοποίηση και καταγραφή δεδομένων με στόχο την ανάλυση και την εξαγωγή συμπερασμάτων, η οποία και βρίσκει σημαντικό πεδίο εφαρμογής στον χώρο της υγείας. Αφετέρου, η ύπαρξη, σήμερα, πληθώρας συσκευών και υπηρεσιών λογισμικού, που εξυπηρετούν αυτή την τάση, συνήθως, όμως, αποσπασματικά και ατελώς.

Στα πλαίσια, λοιπόν, της διπλωματικής εργασίας και έχοντας τις παραπάνω παρατηρήσεις κατά νου, αναπτύχθηκε μια εφαρμογή διαδικτύου (web application) που λειτουργεί ως συλλέκτης (aggregator) δεδομένων wellness και fitness. Η συλλογή αυτή γίνεται είτε απευθείας από εξειδικευμένες “έξυπνες” ιατρικές συσκευές, είτε μέσω APIs υπαρχόντων και ευρέως χρησιμοποιούμενων ψηφιακών υπηρεσιών, όπως το Fitbit ή το GoogleFit. Η εφαρμογή δίνει την δυνατότητα στον χρήστη να εγγραφεί στο σύστημα και να παρακολουθεί την εξέλιξη στον χρόνο των βιοσημάτων και των δεδομένων του, καθώς και να βλέπει στατιστικά που αφορούν στον ίδιο ή και στο σύνολο των χρηστών.

Η ανάπτυξη της εφαρμογής έγινε με τη χρήση του MEAN Stack (MongoDB ως βάση δεδομένων, NodeJS ως runtime environment, ExpressJS ως web development framework και AngularJS για front-end development). Επιπλέον, χρησιμοποιήθηκε το Docker για την εγκατάσταση (deployment) της εφαρμογής, καθώς επίσης και το περιβάλλον Gitlab για version control και continuous integration. Επίσης, πραγματοποιήθηκε χρήση του πρότζεκτ ανοιχτού κώδικα Open mHealth, το οποίο έχει αναπτύξει, αφενός, μια τυποποίηση του τρόπου περιγραφής και αναπαράστασης δεδομένων υγείας στο διαδίκτυο, αφετέρου και ένα εργαλείο το οποίο παρέχει πρόσβαση στα APIs εφαρμογών όπως το Fitbit και το GoogleFit.

Τέλος, η εφαρμογή αυτή θα ενσωματωθεί στο πρότζεκτ ανοιχτού κώδικα Agile, το οποίο έχει στόχο να αναδείξει τις λειτουργίες και τις δυνατότητες του Rasberry Pi. Συγκεκριμένα, η εφαρμογή θα εκμεταλλεύεται τις δυνατότητες του Rasberry Pi να επικοινωνεί με άλλες συσκευές στο δίκτυο (Internet of Things) και να αποθηκεύει και να επεξεργάζεται τοπικά πληροφορία, λειτουργώντας έτσι ως ένα τοπικό Cloud.

Λέξεις Κλειδιά: Quantified Self, Wellness & Fitness, Internet of Things, Activity Tracking, Personal Informatics, Web Development

Abstract

The objective of this diploma thesis was the design and development of a web application, that collects and aggregates data, from multiple sources, regarding the user's wellness and fitness, as well as processes it, in order to reveal the user's identity, specially in relation to the wider population.

There are two main factors that make the development of such an application purposeful and timely. On one hand, the growing trend for data quantification and logging, in order to analyze them and extract useful insights, a trend that has significant applications in the health sector. On the other hand, the abundance, today, of devices and software that support this trend, often, however, in an incomplete way.

Bearing in mind the above remarks, the application that was developed provides the necessary technical background and functionality for the aggregation of wellness and fitness data. Specifically, it can collect such data, either directly, through specialized smart medical devices, or through the APIs of popular and widely used applications, such as Fitbit or GoogleFit. This application enables the user to monitor his or her biosignals and data over time, as well as presents him or her with statistics regarding his or her performance or the performance of the wider population of users.

For the development of the application, the MEAN Stack was used, providing MongoDB as the database, NodeJS as the runtime environment, ExpressJS as the web development framework and AngularJS as the front-end development language. Moreover, Docker was used for the deployment of the application and Gitlab for version control, continuous integration and delivery. Furthermore, the schemata developed by the Open mHealth organization were followed as a standard for the representation of health and medical data, and Shimmer, a software tool also developed by Open mHealth, was used to provide connectivity with the APIs of Fitbit and GoogleFit.

In conclusion, this Quantified Self application will be integrated into the AGILE open source project, which aims at pointing out the functionality and potential of the Rasberry Pi. Specifically, the application will make use of the Pi's capabilities to connect to and communicate with other devices in the network (Internet of Things), as well as store and process data locally, creating, thus, a local cloud.

Keywords: Quantified Self, Wellness & Fitness, Internet of Things, Activity Tracking, Personal Informatics, Web Development

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κ. Παναγιώτη Τσανάκα για την εμπιστοσύνη που έδειξε, αναθέτοντάς μου την διπλωματική αυτή εργασία, καθώς και για τις συμβουλές του κατά την εκπόνησή της. Επίσης, θα ήθελα να ευχαριστήσω θερμά τον μηχανικό λογισμικού και ερευνητή κ. Ανδρέα Μενύχτα για την πολύτιμη βοήθεια και καθοδήγηση που μου παρείχε, αναφορικά με τα πιο τεχνικά κομμάτια της διπλωματικής.

Πίνακας Περιεχομένων

Κατάλογος Σχημάτων

Κεφάλαιο 1. Εισαγωγή

1.1 Σκοπός της Διπλωματικής

Οι τεχνολογικές εξελίξεις των τελευταίων ετών έχουν οδηγήσει σε μία εκρηκτική αύξηση του αριθμού των έξυπνων συσκευών και των έξυπνων εφαρμογών, φέρνοντας όλο και πιο κοντά τον φυσικό χώρο με τον χώρο του λογισμικού, σε έναν οικοσύστημα που ονομάζεται Διαδίκτυο Πραγμάτων. Σε ένα τέτοιο περιβάλλον, είναι πιο εύκολο από ποτέ να παρακολουθεί κανείς τον ίδιο του τον εαυτό, να καταγράφει δεδομένα που αφορούν τη ζωή του και την καθημερινότητα του και, μέσα από την ανάλυση των δεδομένων αυτών να αποκτά βαθύτερη γνώση της προσωπικότητας του και των δυνατοτήτων του. Συχνά, μάλιστα, η καταγραφή και η ανάλυση γίνονται αυτόματα, με την υποστήριξη εξειδικευμένων εφαρμογών και συσκευών.

Το φαινόμενο, λοιπόν, της συστηματικής και αυτόματης καταγραφής και ανάλυσης δεδομένων, με χρήση τεχνολογιών Διαδικτύου Πραγμάτων, με σκοπό την ανάδειξη των ιδιαιτεροτήτων του χρήστη ονομάζεται Quantified Self. Τα οφέλη που μπορεί να κερδίσει ο χρήστης, από αυτή την, στην ουσία, ποσοτικοποίηση του εαυτού του είναι πολλαπλά. Από την ορθολογική διαχείριση χρηματικών πόρων έως την προστασία της υγείας, οι Quantified Self εφαρμογές στοχεύουν στο να βελτιώσουν όλες τις πτυχές της ζωής του ατόμου, άλλοτε με περισσότερη, άλλοτε με λιγότερη επιτυχία.

Ωστόσο, οι χρήστες τέτοιων εφαρμογών, συχνά, τις εγκαταλείπουν μετά από σύντομο χρονικό διάστημα [41]. Μερικές από τις πιο σημαντικές αιτίες αυτού του φαινομένου είναι η έλλειψη δυνατότητας εξατομίκευσης της εφαρμογής, η απουσία ολιστικής καταγραφής δεδομένων καθώς και η ελλιπής υποστήριξη που παρέχει η εφαρμογή στις προσπάθειες του χρήστη να βελτιωθεί, κυρίως μέσω της υποτίμησης της κοινωνικής διάστασης (π.χ. ανταγωνισμός με άλλους) που αυτή προϋποθέτει.

Το κενό αυτό στοχεύει να καλύψει η διπλωματική εργασία. Συγκεκριμένα, αντικείμενο της εργασίας είναι η ανάπτυξη ενός τεχνικού υπόβαθρου, με τη μορφή μιας πρότυπης εφαρμογής διαδικτύου, που θα επιτρέπει στον χρήστη να επιλέξει τι είδους δεδομένα θέλει να καταγράφει και από ποιες πηγές, διατηρώντας, όμως, τον ολιστικό χαρακτήρα της καταγραφής, καθώς ο χρήστης θα μπορεί να συνδυάζει ποικίλες και ετερογενείς πηγές δεδομένων στην ίδια πλατφόρμα. Επιπλέον, η εφαρμογή θα περιλαμβάνει και ένα σύστημα υποστήριξης, παρακίνησης και επιβράβευσης του χρήστη, λαμβάνοντας υπόψιν και την παραπάνω κοινωνική διάσταση, αποσκοπώντας στην συνεχή βελτίωση των επιδόσεων του χρήστη.

Η υλοποίηση της εφαρμογής θα γίνει με τη χρήση ορισμένων εξαιρετικά δημοφιλών και διαδεδομένων εργαλείων ανάπτυξης, ενώ θα αποτελέσει κομμάτι ενός ερευνητικού προγράμματος, με στόχο την ανάδειξη των ευκαιριών που παρουσιάζονται στο οικοσύστημα του Διαδικτύου Πραγμάτων. Τα παραπάνω αποτελούν και ενδείξεις της επικαιρότητας και της χρησιμότητας της εφαρμογής.

1.2 Οργάνωση Κειμένου

Στο Κεφάλαιο 2 θα γίνει η παρουσίαση του υπόβαθρου της διπλωματικής. Συγκεκριμένα, θα αναφερθούν οι βασικές θεωρητικές έννοιες, οι τεχνολογίες και τα εργαλεία που αξιοποιήθηκαν για την ανάπτυξη και θα γίνει μια επισκόπηση της τρέχουσας έρευνας στον χώρο του Internet of Things και Quantified Self.

Στο Κεφάλαιο 3 θα γίνει ανάλυση των προδιαγραφών της εφαρμογής και μια υψηλού επιπέδου περιγραφή της αρχιτεκτονικής που ακολουθήθηκε κατά την ανάπτυξη.

Στο Κεφάλαιο 4 γίνεται η περιγραφή της υλοποίησης της εφαρμογής, και εξηγείται λεπτομερώς πως επιτελείται, από τεχνικής άποψης, η κάθε λειτουργία της εφαρμογής.

Στο Κεφάλαιο 5 παρουσιάζεται το αποτέλεσμα της υλοποίησης, δηλαδή η διεπαφή που βλέπει και χρησιμοποιεί ο χρήστης και πως η διεπαφή αυτή εξυπηρετεί τις λειτουργίες της εφαρμογής.

Στο Κεφάλαιο 6, τέλος, γίνεται μια σύντομη σύνοψη της εφαρμογής, καθώς και μια συζήτηση για το πως μπορεί η εφαρμογή να επεκταθεί μελλοντικά.

Κεφάλαιο 2. Ιατρικό, Τεχνολογικό και Ερευνητικό Υπόβαθρο

2.1 Ιατρικό Υπόβαθρο

Στην ενότητα αυτή γίνεται αναφορά, εν συντομία, στους τύπους των ιατρικών δεδομένων που θα καταγράφει και θα αναλύει το σύστημα. Συγκεκριμένα, δύο είναι οι κατηγορίες αυτών των δεδομένων: 1) δεδομένα που αφορούν την φυσική/σωματική δραστηριότητα του χρήστη και 2) δεδομένα βιοσημάτων του χρήστη .

2.1.1 Σωματική Δραστηριότητα

2.1.1.1 Βήματα

Ένα πρώτο μέγεθος σχετικό με την σωματική δραστηριότητα με το οποίο θα ασχοληθούμε είναι ο αριθμός των βημάτων (steps) που κάνει ο χρήστης. Συγκεκριμένα, ο αριθμός βημάτων ανά ημέρα αποτελεί ένα σχετικά αξιόπιστο δείκτη του πόσο καθιστική ή μη είναι η καθημερινότητα του χρήστη.

2.1.1.2 Φυσική Δραστηριότητα

Εν συνεχεία, και σε άμεση σχέση με τα βήματα του χρήστη, θα μας απασχολήσει και η φυσική δραστηριότητα (physical activity) του. Συγκεκριμένα, μας ενδιαφέρει τόσο το είδος της δραστηριότητας (περπάτημα, τρέξιμο κτλ.) όσο και η διάρκειά της (σε λεπτά ανά ημέρα).

2.1.1.3 Θερμίδες

Τέλος, σημαντική παράμετρος για την αξιολόγηση της σωματικής δραστηριότητας του χρήστη είναι και η καύση των θερμίδων (kcal / ημέρα).

2.1.2 Βιοσήματα

2.1.2.1 Σωματικό Βάρος

Ο πρώτος τύπος βιοσήματος είναι το σωματικό βάρος (body weight). Ως μονάδα μέτρησης του βάρους θα χρησιμοποιηθεί το κιλό (kg). Είναι προφανές πως το σωματικό βάρος αποτελεί σημαντική παράμετρο της υγείας ενός ατόμου, κάτι το οποίο καθιστά απαραίτητη την συστηματική καταγραφή και παρακολούθηση του.

2.1.2.2 Καρδιακός Ρυθμός

Ένας άλλος τύπος βιοσήματος με ενδιαφέρον είναι ο καρδιακός ρυθμός (heart rate). Καρδιακός ρυθμός (σφυγμός ή παλμός) λέγεται η μετάδοση του κύματος αίματος, που προκαλείται από την καρδιακή συστολή, στο τοίχωμα των αγγείων. Ο σφυγμός 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 το beats per minute (bpm), δηλαδή οι χτύποι ανά λεπτό.

2.1.2.3 Αρτηριακή Πίεση

Επιπλέον, σημαντικό μέγεθος είναι και η αρτηριακή πίεση (blood pressure), η δύναμη, δηλαδή, που προωθεί το αίμα μέσω των αρτηριών σε όλους τους ιστούς του σώματος, εξασφαλίζοντας την συνεχή κυκλοφορία του. Η αρτηριακή πίεση εκφράζεται με τη βοήθεια δύο επιμέρους τιμών, της συστολικής και της διαστολικής πίεσης. Η συστολική πίεση αντιστοιχεί στην πίεση που ασκείται στις αρτηρίες όταν η καρδιά συστέλλεται και τις τροφοδοτεί με αίμα. Η διαστολική πίεση και αφορά στην πίεση που ασκείται στις αρτηρίες όταν η καρδιά ηρεμεί μετά τη συστολή. Μονάδα μέτρησης και των δύο επιμέρους μεγεθών είναι το mmHg (millimeter of mercury), δηλαδή χιλιοστά στήλης υδραργύρου.

2.1.2.4 Γλυκόζη Αίματος

Τέταρτο σημαντικό βιοσήμα είναι η συγκέντρωση της γλυκόζης στο αίμα (blood glucose) ή, ισοδύναμα, η συγκέντρωση του σακχάρου. Η γλυκόζη είναι απλός μονοσακχαρίτης που βρίσκεται στα φυτά. Είναι ένας από τους τρεις διατροφικούς μονοσακχαρίτες, μαζί με τη φρουκτόζη και τη γαλακτόζη, οι οποίοι απορροφώνται άμεσα στην κυκλοφορία του αίματος κατά τη διάρκεια της πέψης. Αποτελεί τον σημαντικότερο υδατάνθρακα στη βιολογία, αφού τα κύτταρα την χρησιμοποιούν ως την πρωταρχική πηγή ενέργειας και ως μέσο μεταβολισμού. Η γλυκόζη χρησιμοποιείται ως καύσιμο για την κυτταρική αναπνοή. Μονάδα μέτρησης του μεγέθους αυτού είναι το mg/dL (milligrams per deciliter).

2.1.2.5 Κορεσμός Οξυγόνου

Επίσης, θα εξεταστεί και ο κορεσμός οξυγόνου του αίματος, ο οποίος ένας δείκτης που αντικατοπτρίζει το ποσοστό αιμοσφαιρίνης κορεσμένο με οξυγόνο σε σχέση με τη συνολική ποσότητα της αιμοσφαιρίνης που υπάρχει στο αίμα.

2.1.2.6 Διάρκεια Ύπνου

Τέλος, αξία έχει και η καταγραφή των μοτίβων και της διάρκειας του ύπνου του χρήστη (σε ώρες).

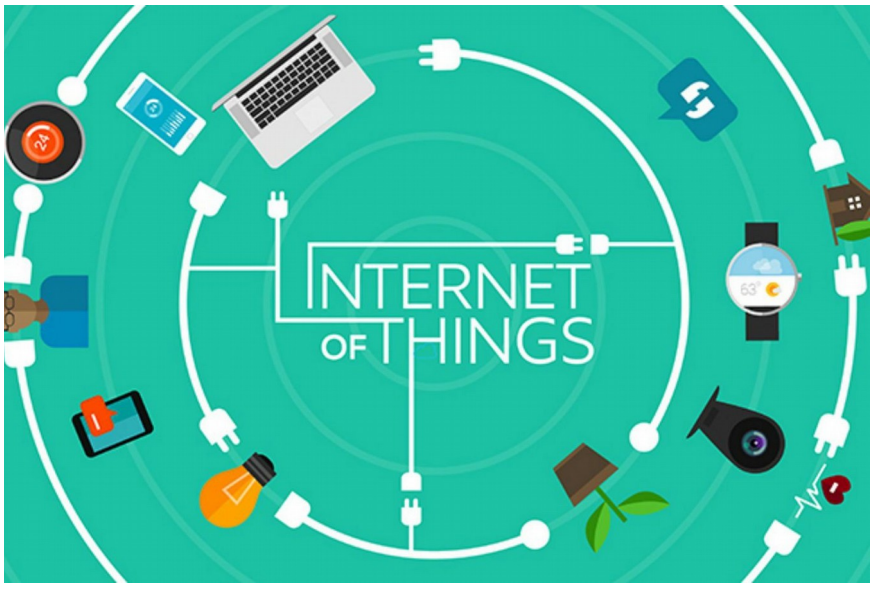
2.2 Τεχνολογίες & Σχετική Έρευνα

Ακολουθεί μια επισκόπηση της τρέχουσας ερευνητικής δραστηριότητας στον χώρο, καθώς και μια περιγραφή των τεχνολογιών και των εργαλείων που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.

2.2.1 Internet of Things

Ο αριθμός των συσκευών με δυνατότητα σύνδεσης στο Διαδίκτυο ξεπέρασε τον αριθμό των ανθρώπων στο Διαδίκτυο το 2008 και εκτιμάται ότι μέχρι το 2020 θα φτάσει τα 50 δισεκατομμύρια [1] ή και παραπάνω. Και φυσικά, ο όρος συσκευές δεν περιλαμβάνει μόνο παραδοσιακούς υπολογιστές, όπως laptop, PC, smartphones και tablets, αλλά κάθε είδους “έξυπνα” αντικείμενα του φυσικού κόσμου, από αισθητήρες και οικιακές συσκευές μέχρι αυτοκίνητα και ολόκληρα κτήρια.

Η έκρηξη αυτή του πλήθους 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). Το Internet of Things, λοιπόν, μπορεί να οριστεί ως η διαδικτύωση φυσικών αντικειμένων, οχημάτων, κτηρίων και, εν γένει, συστημάτων που ενσωματώνουν κάποιο υπολογιστικό σύστημα, έτσι ώστε τα αντικείμενα αυτά να μπορούν να επικοινωνούν, να ανταλλάσσουν και να επεξεργάζονται δεδομένα [2].

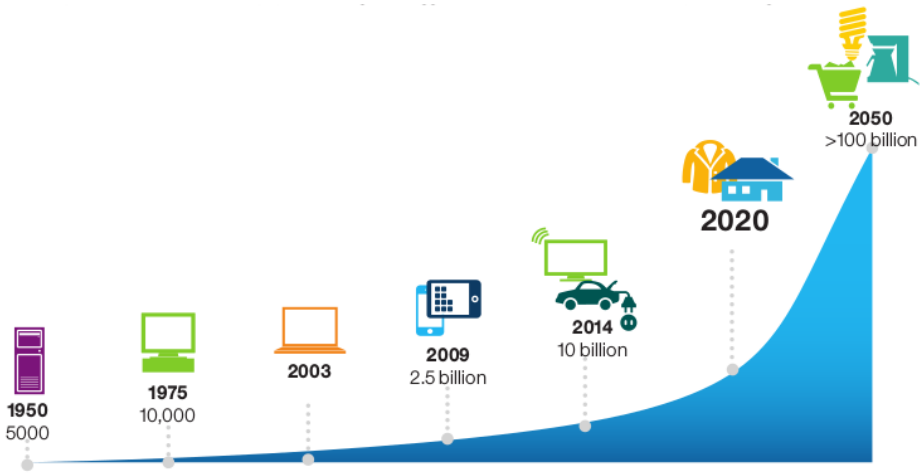


Σχήμα 1: Διαδίκτυο Πραγμάτων

Ιδιαίτερη σημασία, στα πλαίσια της διπλωματικής εργασίας, έχουν οι “έξυπνες” ιατρικές συσκευές, όπως ζυγαριές, πιεσόμετρα, οξύμετρα κ.ά. Πρόκειται για παραλλαγές των αντίστοιχων συμβατικών ιατρικών οργάνων που, αφενός, λαμβάνουν μετρήσεις βιοσημάτων (όπως τα ορίσαμε παραπάνω) και, αφετέρου, έχουν την δυνατότητα να συνδεθούν, συνήθως μέσω της ασύρματης τεχνολογίας BLE (bluetooth low energy) [3], με υπολογιστές που υποστηρίζουν συνδέσεις τύπου bluetooth (π.χ. smartphones) ώστε να μεταδώσουν αυτές τις μετρήσεις.

Επιπροσθέτως, πρέπει να γίνει αναφορά και στα συστήματα καταγραφής δραστηριότητας ή Activity Tracking. Εφαρμογές, δηλαδή, 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. Δύο τέτοιοι Activity Trackers, με τους οποίους θα ασχοληθούμε εκτενώς στην πορεία, είναι το Fitbit και το GoogleFit. Οι εφαρμογές αυτές, συνδεόμενες (μέσω BLE) με ειδικά βραχιόλια-αισθητήρες ή και χρησιμοποιώντας τις λειτουργίες GPS του smartphone, παρακολουθούν και καταγράφουν μετρήσεις σχετικά με την σωματική δραστηριότητα του χρήστη (βλ. παραπάνω) [4].

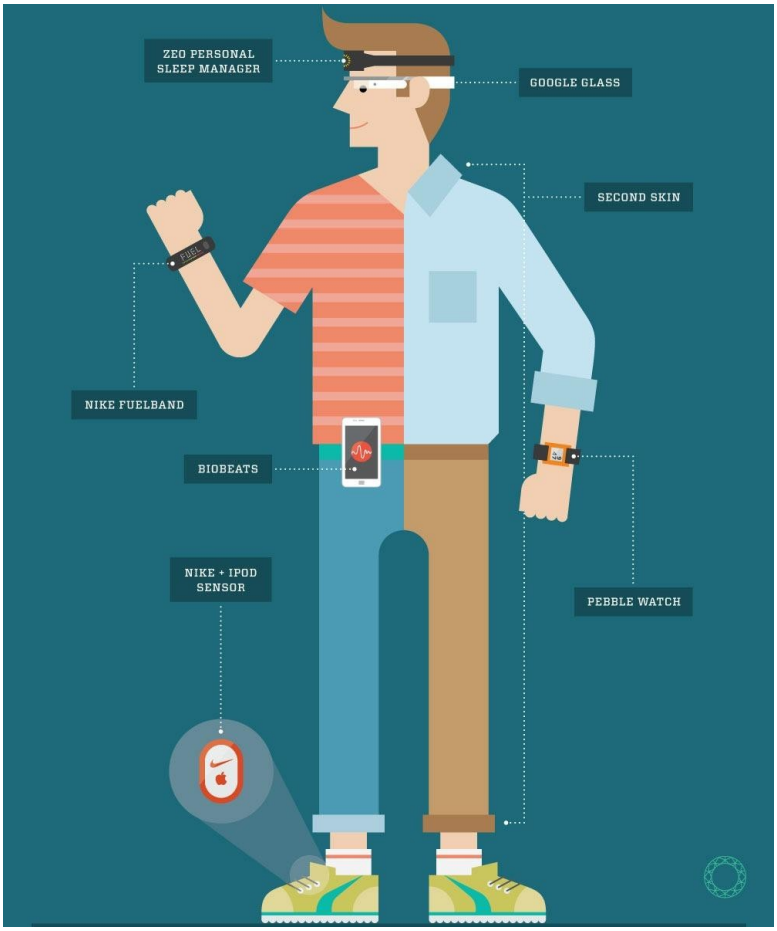
Σχήμα 2: Activity Trackers



Σχήμα 3: Αριθμός Συσκευών στο Διαδίκτυο Πραγμάτων - IBM

2.2.2 Quantified Self

Η ύπαρξη, λοιπόν, τόσο της κατάλληλης τεχνολογίας αισθητήρων, όσο και της πληθώρας "έξυπνων" εφαρμογών (smartphone applications) τύπου Activity Tracking έχει οδηγήσει στην ανάπτυξη του φαινομένου του Quantified Self. Συγκεκριμένα, όλο και περισσότεροι είναι οι χρήστες που χρησιμοποιούν τέτοιες εφαρμογές, ώστε, μέσα από την καταγραφή, ποσοτικοποίηση και ανάλυση των δεδομένων τους, να πάρουν πληροφορίες και συμβουλές για το πως μπορούν να βελτιώσουν τα αποτελέσματα και τις επιδόσεις τους στις διάφορες καθημερινές δραστηριότητες.

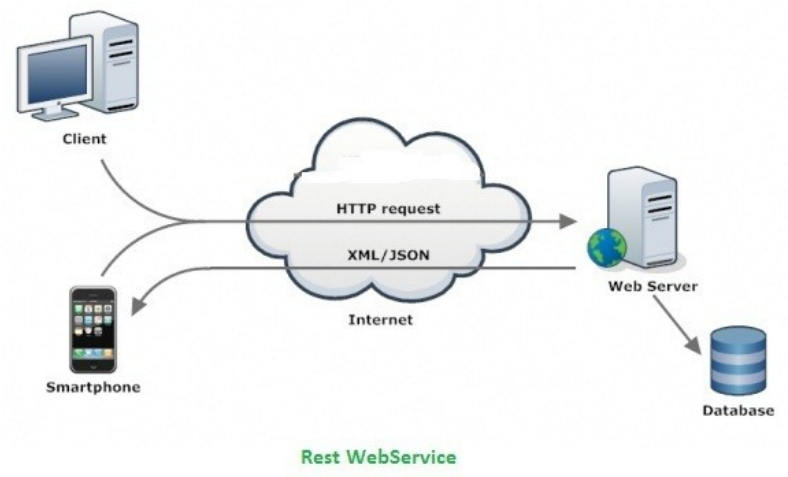


Σχήμα 4: Quantified Self

Συγκεκριμένα, σήμερα υπάρχουν τα 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. Και φυσικά, τα εργαλεία αυτά εξυπηρετούν όχι μόνο την ανάγκη καταγραφής και αποθήκευσης των δεδομένων αυτών, αλλά παρέχουν και την δυνατότητα ανάλυσης τους και εξαγωγής χρήσιμων συμπερασμάτων για τον χρήστη, με στόχο να τον υποστηρίξουν στις δραστηριότητές του και να τον βοηθήσουν να κατανοήσει τον ίδιο του τον εαυτό [5]. 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 δίκτυό του, ενώ, ταυτόχρονα, συγκρίνει τα δικά του αποτελέσματα με αυτά του συνόλου.

2.2.3 REST Web Services & HTTP

Το ReST (Representational State Transfer) είναι μια μορφή αρχιτεκτονικής για τη δημιουργία δικτυακών εφαρμογών, η οποία χρησιμοποιεί το πρωτόκολλο HTTP (βλ. παρακάτω) για την ανταλλαγή δεδομένων. Συγκεκριμένα, σε μια, όπως λέγεται, RESTful εφαρμογή τα πάντα θεωρούνται πόροι (resources), η πρόσβαση στους οποίους γίνεται μέσω δικτύου με ομοιόμορφο τρόπο. Η επικοινωνία και η ανταλλαγή δεδομένων σε ένα σύστημα REST ακολουθούν τις εξής βασικές αρχές [6] :



Σχήμα 5: Αρχιτεκτονική τύπου REST

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 (ID). 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 ReST. Η ομοιόμορφη διεπαφή απλοποιεί και αποσυνδέει την αρχιτεκτονική πελάτη – διακομιστή, επιτρέποντας σε κάθε πλευρά να εξελιχθεί ανεξάρτητα. Η ομοιόμορφη διεπαφή απαιτεί την αντιστοίχιση των πόρων σε μοναδικά IDs για την αναγνώρισή τους, και επιβάλλει την αναπαράσταση της πληροφορίας που μεταδίδεται στον πελάτη σύμφωνα με ένα προκαθορισμένο πρότυπο (συνήθως ως JSON ή XML αντικείμενα).

Χωρίς Κατάσταση (Stateless):
Σύμφωνα με την αρχή αυτή, πληροφορίες που αφορούν στον πελάτη δεν αποθηκεύονται στον εξυπηρετητή και κάθε αίτημα από τον πρώτο προς τον δεύτερο είναι αυτόνομο και περιλαμβάνει όλες τις απαραίτητες πληροφορίες για τον χειρισμό του.

Ένα σύστημα ιεραρχημένο με τη χρήση της cache μνήμης (Cacheable):
Οι αποκρίσεις στα αιτήματα του χρήστη μπορούν να «κρυφτούν», δηλαδή να αποθηκευτούν προσωρινά στη μνήμη cache ή όχι, ώστε οι πελάτες να μην επαναχρησιμοποιούν άκυρα ή ακατάλληλα δεδομένα. Έτσι βελτιώνεται η επεκτασιμότητα και η απόδοση.

Διαστρωματωμένο Σύστημα (Layered System):
Ο πελάτης δεν μπορεί 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 Ζήτηση (Code on Demand):
Οι διακομιστές έχουν τη δυνατότητα να παρατείνουν προσωρινά ή να προσαρμόσουν τη λειτουργία ενός πελάτη με τη μεταφορά εκτελέσιμου κώδικα.

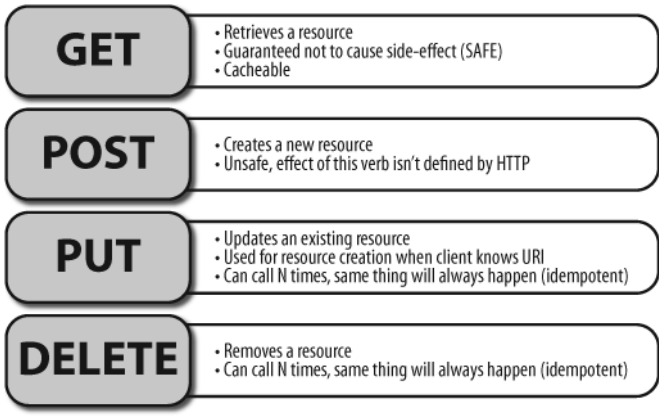
Στην πράξη, στο πρωτόκολλο HTTP (Hypertext Transfer Protocol), κάθε πόρος εντοπίζεται μέσω μιας διεύθυνσης ιστού (web address) η οποία και ονομάζεται URL (Universal Resource Locator ή Ενιαίος Εντοπιστής Πόρων). Ένα παράδειγμα μιας τέτοιας διεύθυνσης είναι το εξής:

http://www.example.com/index.html

Η διεύθυνση αυτή υποδηλώνει το πρωτόκολλο (HTTP), το όνομα του εξυπηρετητή ή Hostname (www.example.com) καθώς και το όνομα του συγκεκριμένου αρχείου (πόρου) που αναζητείται (index.html).

Εκτός από τον εντοπισμό ενός πόρου στο Διαδίκτυο, το πρωτόκολλο HTTP ορίζει και τις μεθόδους, δηλαδή τις ενέργειες που μπορούν να γίνουν στον εντοπισμένο αυτόν πόρο. Οι σημαντικότερες από αυτές είναι οι εξής:

- GET: Με ένα αίτημα GET ο πελάτης ζητάει από τον εξυπηρετητή να του στείλει τον πόρο, χωρίς αυτό να έχει κάποια άλλη παρενέργεια στα δεδομένα
- POST: Με ένα αίτημα POST ο πελάτης στέλνει στον εξυπηρετητή ένα καινούργιο δεδομένο, το οποίο, μετά από κάποια, ενδεχομένων επεξεργασία, θα αποθηκευτεί, ώστε να είναι δυνατή η ανάκτηση του στη συνέχεια, Δημιουργείται, δηλαδή, ένας καινούργιος πόρος
- PUT: Ο πελάτης στέλνει ένα αίτημα PUT όταν θέλει να ενημερώσει, να πραγματοποιήσει δηλαδή κάποια αλλαγή, σε έναν ήδη υπάρχοντα πόρο
- DELETE: Με ένα αίτημα DELETE, ο πελάτης ζητάει την διαγραφή ενός πόρο από το σύστημα



Σχήμα 6: HTTP Methods

2.2.4 Javascript & JSON

Η JavaScript αποτελεί μια ελαφριά, διερμηνευμένη (interpreted), αντικειμενοστρεφή γλώσσα προγραμματισμού και είναι πιο γνωστή ως η γλώσσα σεναρίου για τη δημιουργία ιστοσελίδων. Η JavaScript βασίζεται σε πρωτότυπα (prototype-based), είναι πολυ-παραδειγματική και υποστηρίζει αντικειμενοστρεφή, προστακτικό και συναρτησιακό προγραμματισμό [7].

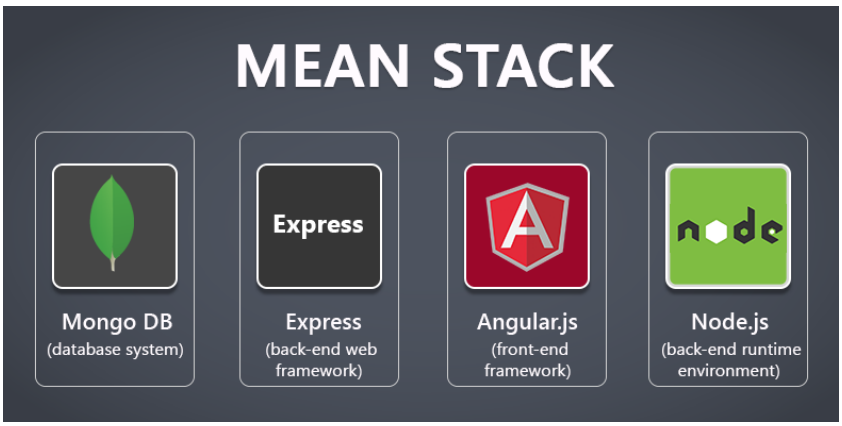
JSON σημαίνει JavaScript Object Notation και πρόκειται για ένα σχήμα, δηλαδή σύνολο συντακτικών κανόνων, για την αποθήκευση και ανταλλαγή δεδομένων. Συγκεκριμένα, κατά την επικοινωνία Πελάτη – Εξυπηρετητή, τα δεδομένα που ανταλλάσσονται πρέπει να έχουν την μορφή κειμένου (text). Μετατρέποντας, λοιπόν, ένα αντικείμενο της γλώσσας JavaScript (JavaScript object) σε κείμενο μορφής JSON, είναι δυνατή η ανταλλαγή του μεταξύ Client-Server, ενώ η μετατροπή μπορεί να γίνει και κατά την αντίθετη κατεύθυνση, δηλαδή από κείμενο τύπου JSON σε αντικείμενο JavaScript. Ένα αντικείμενο JSON αποτελείται από ζεύγη κλειδιού-τιμής (key-value pairs), με την τιμή να είναι κάτι από τα παρακάτω: αριθμός, συμβολοσειρά, αληθοτιμή, πίνακας, αντικείμενο JSON, null [8][9].

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Σχήμα 7: Παράδειγμα αντικειμένου JSON

2.2.5 MEAN Stack

Το MEAN είναι μία ελεύθερη και ανοιχτού κώδικα (open-source) στοίβα λογισμικού (software stack) για την ανάπτυξη δυναμικών εφαρμογών ιστού (web applications). Η στοίβα αυτή αποτελείται από την βάση δεδομένων MongoDB, το περιβάλλον εκτέλεσης NodeJS, το πλαίσιο (framework) ανάπτυξης εφαρμογών ιστού ExpressJS και το πλαίσιο ανάπτυξης εφαρμογών από την πλευρά του πελάτη (client-side ή front-end) AngularJS [10]. Κάθε ένα από αυτά τα συστατικά μέρη αναλύεται παρακάτω.



Σχήμα 8: MEAN Stack

2.2.5.1 NodeJS

Το NodeJS είναι ένα περιβάλλον εκτέλεσης (run-time environment) κώδικα JavaScript από την πλευρά του εξυπηρετητή (server-side), στο οποίο η ροή του προγράμματος καθορίζεται από γεγονότα, όπως πράξεις του χρήστη, είσοδοι/έξοδοι ή μηνύματα από άλλες διεργασίες (event-driven programming) [11].

Ο προγραμματισμός σε NodeJS δεν χρησιμοποιεί νήματα (threading), καθώς χρησιμοποιούνται ευρέως συναρτήσεις επανάκλησης (callbacks) που σηματοδοτούν την ολοκλήρωση ενός έργου (task). Αυτού του είδους ο ασύγχρονος, non-blocking προγραμματισμός επιτρέπει την ανάπτυξη εξαιρετικά κλιμακώσιμων εφαρμογών με αποτελεσματική χρήση της μνήμης [12][13].

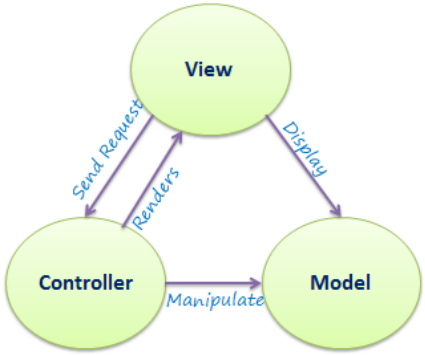
2.2.5.2 ExpressJS

Το ExpressJS αποτελεί ένα Web Application Framework (WAF) [14] για το NodeJS. Είναι κατάλληλο για την ανάπτυξη εφαρμογών ιστού και APIs [15], καθώς υλοποιεί πάρα πολλές βοηθητικές μεθόδους HTTP και Middleware, δηλαδή λογισμικό που μεσολαβεί μεταξύ του λειτουργικού συστήματος ή της βάσης δεδομένων και της εφαρμογής.

2.2.5.3 AngularJS

Η AngularJS είναι ένα Web Application Framework για την ανάπτυξη εφαρμογών από την πλευρά του πελάτη (client-side ή fron-end). Η Angular επεκτείνει το HTML DOM, δηλαδή τον τρόπο με τον οποίο ορίζεται λογικά και περιγράφεται μια ιστοσελίδα και τα περιεχόμενά της, ώστε να διευκολύνει την ανάπτυξη και να καταστήσει την σελίδα περισσότερο αποκρίσιμη [16].

Βασικό χαρακτηριστικό της AngularJS είναι ότι ακολουθεί την αρχιτεκτονική Model-View-Controller ή MVC. Σύμφωνα με την αρχιτεκτονική αυτή, η εφαρμογή χωρίζεται σε τρία διασυνδεδεμένα μέρη: το Model, το View και τον Controller. Το Model αποτελεί το κεντρικό συστατικό της αρχιτεκτονικής, εκφράζοντας την συμπεριφορά της αρχιτεκτονικής, ανεξάρτητα από την διεπαφή του χρήστη, και χειριζόμενο τα δεδομένα, την λογική και του κανόνες της εφαρμογής. Το View αποτελεί την αναπαράσταση της πληροφορίας, όπως την βλέπει ο χρήστης. Ο Controller, τέλος, δέχεται είσοδο και την μετατρέπει σε εντολές για το Model ή το View [17].



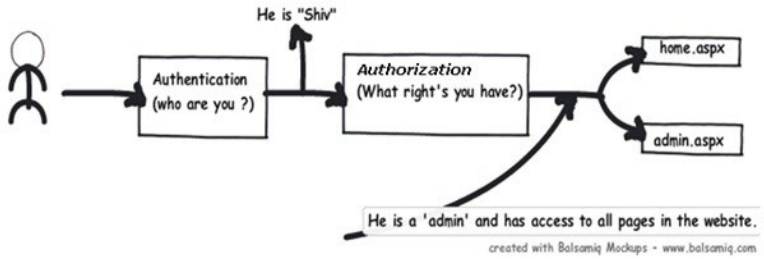
Σχήμα 9: Αρχιτεκτονική MVC

2.2.5.4 MongoDB

Η MongoDB αποτελεί μια εγγραφο-στρεφή ή document-oriented βάση δεδομένων, που ανήκει στην κατηγορία των NoSQL βάσεων. Η MongoDB χρησιμοποιεί έγγραφα που ακολουθούν δομή τύπου JSON, η οποία και περιγράφεται από τα σχήματα (schemas) που ορίζει ο χρήστης. Μια εγγραφή, δηλαδή, στην βάση Mongo είναι ένα έγγραφο-αντικείμενο που περιλαμβάνει ζεύγη από κλειδιά–τιμές (key-value pairs). Δεδομένα αυτής της μορφής καλούνται ημι-δομημένα (semi-structured) δεδομένα [18][19].

2.2.6 User Authentication & Authorization – PassportJS

Σε μια εφαρμογή είναι απαραίτητο να υπάρχει ένας μηχανισμός αναγνώρισης – επαλήθευσης του χρήστη (συνήθως μια φόρμα εισόδου – login), κάτι το οποίο ονομάζεται User Authentication. Ταυτόχρονα, δεδομένου ότι κάθε χρήστης ενδεχομένως να έχει και διαφορετικά δικαιώματα, αναφορικά με τις λειτουργίες που μπορεί να επιτελέσει στην εφαρμογή, χρειάζεται και ένας μηχανισμός ελέγχου και εξουσιοδότησης του χρήστη, κάτι το οποίο ονομάζεται User Authorization. Το PassportJS [20] είναι μια βιβλιοθήκη, γραμμένη σε JavaScript, που αναλαμβάνει να υλοποιήσει τους παραπάνω δύο μηχανισμούς, πάνω στο Node.



Σχήμα 10: Σχηματική επεξήγηση του User Authentication & Authorization

2.2.7 Open mHealth

Το Open mHealth [21] είναι ένας οργανισμός που έχει αναπτύξει διάφορα εργαλεία σχετικά με την αναπαράσταση, συλλογή, επεξεργασία και οπτικοποίηση ιατρικών δεδομένων και δεδομένων υγείας στο διαδίκτυο. Στα πλαίσια της εργασίας έγινε χρήση των ακόλουθων δύο τέτοιων εργαλείων:

2.2.7.1 Open mHealth Schemata

Αρχικά, το Open mHealth έχει αναπτύξει μια βιβλιοθήκη σχημάτων (schemata) με στόχο την τυποποίηση της αναπαράστασης δεδομένων υγείας στο διαδίκτυο, με ομοιόμορφο και συνεπή τρόπο [22]. Ως αποτέλεσμα, εφαρμογές που χρησιμοποιούν αυτή την αναπαράσταση μπορούν, εύκολα και γρήγορα, να ανταλλάξουν δεδομένα μεταξύ τους.

2.2.7.2 Shimmer

Επιπλέον, έγινε χρήση και του εργαλείου Shimmer [23], το οποίο είναι μια εφαρμογή με δυνατότητα σύνδεσης με δημοφιλή third-party APIs, όπως το Fitbit ή το GoogleFit, συλλογής δεδομένων χρήστη από τα APIs αυτά και μετατροπή των δεδομένων αυτών σε μορφή σύμφωνη με τα Open mHealth σχήματα (βλ. παραπάνω).

2.2.8 Cloud APIs

Εν συνεχεία, γίνεται μια σύντομη αναφορά στις δύο δημοφιλείς εφαρμογές που χρησιμοποιήθηκαν για την συλλογή δεδομένων του χρήστη, συλλογή η οποία και έγινε χρησιμοποιώντας τις δυνατότητες του Shimmer (βλ. παραπάνω).

2.2.8.1 GoogleFit

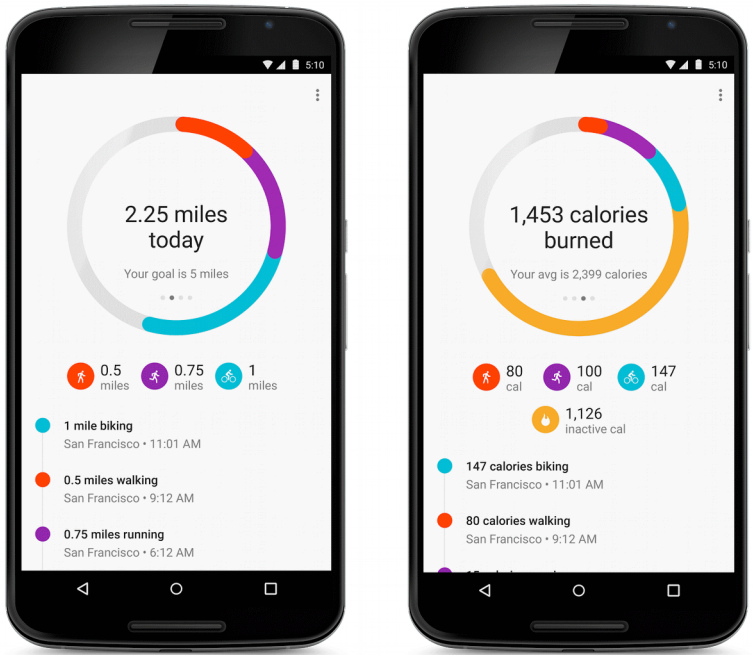
Το GoogleFit είναι μια εφαρμογή που αναπτύχθηκε από την εταιρεία Google για το λειτουργικό σύστημα Android, η οποία δίνει την δυνατότητα στον χρήστη να παρακολουθεί ορισμένα μεγέθη που αφορούν στην υγεία του και στην αθλητική του δραστηριότητα [24].

2.2.8.2 Fitbit

Ομοίως, το Fitbit [25] , είναι και αυτό μια εφαρμογή για την παρακολούθηση και καταγραφή δεδομένων υγείας και ευεξίας του χρήστη.

Τόσο το GoogleFit, όσο και το Fitbit έχουν αναπτύξει ανοικτά και προσβάσιμα APIs, καθιστώντας δυνατή την σύνδεσή τους με τρίτες εφαρμογές, όπως αυτή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας. Όπως ήδη αναφέρθηκε, η σύνδεση με τα APIs των GoogleFit και Fitbit έγινε με χρήση του Shimmer [26].

Σχήμα 11: Fitbit Διεπαφή Χρήστη



Σχήμα 12: GoogleFit Διεπαφή Χρήστη

2.2.9 AGILE

Η Quantified Self εφαρμογή, που αποτελεί το αντικείμενο της διπλωματικής εργασίας αυτής, θα ενσωματωθεί στο ερευνητικό πρόγραμμα AGILE [27]. Πρόκειται για ένα ερευνητικό πρόγραμμα που έχει στόχο την ανάπτυξη μιας αρθρωτής (modular) πύλης (gateway) υλικού και λογισμικού για το Διαδίκτυο Πραγμάτων (Internet of Things ή IoT). Η πύλη αυτή θα υποστηρίζει την διαλειτουργικότητα (interoperability) πρωτοκόλλων, τον χειρισμό συσκευών και δεδομένων, την εκτέλεση IoT εφαρμογών και την επικοινωνία με εξωτερικά υπολογιστικά συστήματα νέφους (Cloud).

Στα πλαίσια, λοιπόν, του AGILE, η εφαρμογή της εργασίας αυτής στοχεύει στο να αναδείξει τις δυνατότητες που προσφέρει η προαναφερθείσα πύλη για την ανάπτυξη και υποστήριξη εφαρμογών για το Διαδίκτυο Πραγμάτων. Συγκεκριμένα, λόγω της συνδεσιμότητας της με έξυπνες συσκευές ή με το ευρύτερο Διαδίκτυο, καθώς και τις δυνατότητες της για τοπική αποθήκευση και επεξεργασία δεδομένων, η πύλη αυτή αποτελεί την κατάλληλη πλατφόρμα στην οποία μπορεί να εγκατασταθεί και να “τρέξει” η Quantified Self εφαρμογή.

2.2.10 Docker

Σε ένα Λειτουργικό Σύστημα (ΛΣ) είναι δυνατόν ο πυρήνας (kernel) να επιτρέπει την συνύπαρξη πολλαπλών απομονωμένων χώρων χρήστη (user-space instances), κάτι το οποίο ονομάζεται εικονικοποίηση σε επίπεδο ΛΣ (Operating-system-level virtualization) ή containerization. Ένα τέτοιο απομονωμένο στιγμιότυπο ή container, ιδωμένο από το εσωτερικό του, συμπεριφέρεται σαν κανονικός υπολογιστής, με τους δικούς του πόρους, χωρίς να έχει επίγνωση των υπόλοιπων στιγμιοτύπων και χωρίς να μοιράζεται τους πόρους του μαζί τους [28]. Το λογισμικό Docker [29] υλοποιεί αυτή ακριβώς την λειτουργία σε περιβάλλον Linux ή Windows .

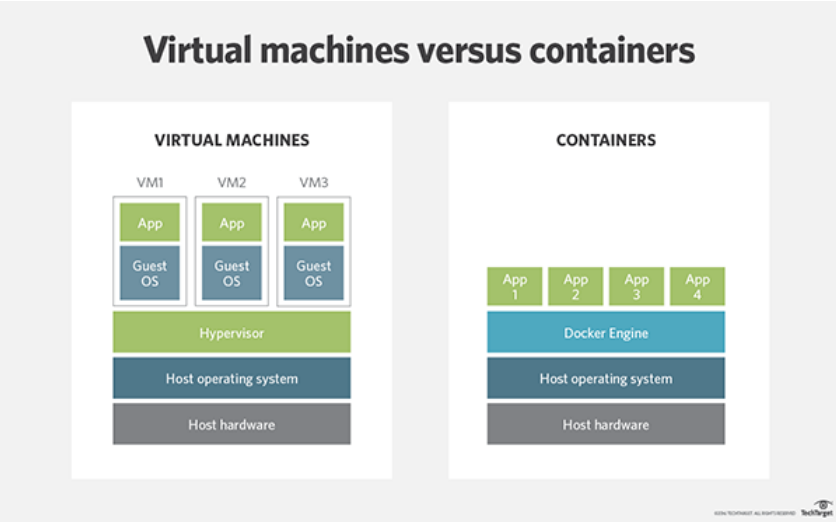
Στο σημείο αυτό αξίζει να γίνει μια σύντομη αναφορά στα σημαντικότερα πλεονεκτήματα και μειονεκτήματα του Docker [30]. Ξεκινώντας από τα πλεονεκτήματα:

- Απομόνωση: μια εφαρμογή που “τρέχει” μέσα σε ένα Docker Container είναι απομονωμένη από το περιβάλλον εκτέλεσης και το πραγματικό ΛΣ του τελικού χρήστη.
- Ελαφριά (lightweight) Εκτέλεση: σε αντίθεση με τις εικονικές μηχανές (virtual machines), οι οποίες χρειάζονται έναν επιβλέποντα (hypervisor) πάνω από το ΛΣ, καθώς και ένα Guest Operating System για την λειτουργία τους, ένα Docker Container χρησιμοποιεί απευθείας τον πυρήνα του ΛΣ, κάνοντας, έτσι, χρήση σημαντικά λιγότερων πόρων.
- Εύκολη εγκατάσταση, μεταφορά στην παραγωγή και “μετανάστευση”: μια εφαρμογή που τρέχει σε ένα Docker Container είναι αυτάρκης και ανεξάρτητη, κάτι το οποίο διευκολύνει σε μεγάλο βαθμό την μεταφορά της από περιβάλλον σε περιβάλλον και λύνει πολλά από τα προβλήματα που προκύπτουν κατά την ενσωμάτωση της στο ευρύτερο σύστημα της παραγωγής.

Αντίθετα, κάποια από τα μειονεκτήματα είναι:

- Στην περίπτωση εξαιρετικά απαιτητικών σε πόρους εφαρμογών, η τεχνολογία των Containers υστερεί σε σχέση με την παραδοσιακή εγκατάσταση τους σε εξυπηρετητές (servers) τους οποίους και χρησιμοποιούν αποκλειστικά.

- Τα διάφορα containers μοιράζονται το ίδιο Host Operating System, κάτι το οποίο τα περιορίζει στην ίδια αρχιτεκτονική.



Σχήμα 13: Σύγκριση Εικονικών Μηχανών με Docker Containers

2.2.11 Περιβάλλον Ανάπτυξης

Τέλος, αξίζει να γίνει και μια αναφορά στο περιβάλλον που χρησιμοποιήθηκε κατά την ανάπτυξη της εφαρμογής.

2.2.11.1 Gitlab

Το Gitlab [31] αποτελεί μια ολοκληρωμένη διαδικτυακή πλατφόρμα διαχείρισης κώδικα και συνεργατικής ανάπτυξης εφαρμογών. Στα πλαίσια της παρούσας εργασίας, ιδιαίτερη σημασία έχουν οι εξής λειτουργίες που παρέχει το Gitlab: Version Control, Continuous Integration και Continuous Delivery:

Version Control
Ως σύστημα ελέγχου έκδοσης ή Version Control System ορίζεται η διαχείριση των αλλαγών του κώδικα που συμβαίνουν κατά τη διάρκεια ανάπτυξης της εφαρμογής. Η διαχείριση αυτή περιλαμβάνει την δυνατότητα παράλληλης ανάπτυξης διαφορετικών εκδοχών του κώδικα, την δυνατότητα συγχώνευσης διαφορετικών εκδοχών σε μία, καθώς και την δυνατότητα ακύρωσης αλλαγών και επαναφοράς του κώδικα σε προγενέστερη εκδοχή. Η πιο γνωστή και διαδεδομένη υλοποίηση ενός τέτοιου συστήματος είναι το Git [32] , στο οποίο και βασίζεται και το Gitlab.

Continuous Integration
Με τον όρο Continuous Integration ορίζεται η πρακτική που συχνά ακολουθείται κατά την ανάπτυξη κώδικα, σύμφωνα με την οποία οι διάφοροι προγραμματιστές που συμμετέχουν στον έργο συγχωνεύουν τον κώδικα που ο καθένας αναπτύσσει ανεξάρτητα από τους άλλους σε μία κοινή εκδοχή, και μάλιστα αρκετές φορές μέσα στην μέρα. Στη συνέχεια, η εκδοχή αυτή ελέγχεται από αυτοματοποιημένα εργαλεία, διευκολύνοντας τον έγκαιρο εντοπισμό προβλημάτων και την επίλυσή τους [33].

Continuous Delivery

Η πρακτική του Continuous Delivery αποτελεί τη φυσική συνέχεια του Continuous Integration, καθώς εξασφαλίζει πως κάθε αλλαγή στον κώδικα μπορεί εύκολα και άμεσα να ενσωματωθεί στο σύστημα παραγωγής και να γίνει διαθέσιμη στον τελικό χρήστη [34].

2.2.11.2 WebStorm

Αρχικά, ως ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για την συγγραφή και επεξεργασία κώδικα επιλέχθηκε το πρόγραμμα WebStorm, το οποίο και αποτελεί προϊόν της εταιρείας JetBrains [35].

2.2.11.3 Postman

Η ανάπτυξη της Quantified Self εφαρμογής περιέλαβε την υλοποίηση ενός REST API, η λειτουργία του οποίου ελέγχθηκε με τη χρήση του εργαλείου Postman [36]. Συγκεκριμένα, το πρόγραμμα Postman υλοποιεί έναν “πελάτη” τύπου REST (REST Client), ο οποίος έχει την δυνατότητα να στέλνει αιτήματα προς και να λαμβάνει απαντήσεις από έναν REST Server (στην προκειμένη περίπτωση, η εφαρμογή Quantified Self), διευκολύνοντας τον προγραμματιστή να επαληθεύσει την ορθή λειτουργία του API και να εντοπίσει και διορθώσει σφάλματα.

2.2.11.4 Okeanos IAAS

Τέλος, κατά την διάρκεια της ανάπτυξης, έγινε χρήση των υπηρεσιών νέφους του okeanos [37]. Συγκεκριμένα, η εφαρμογή εγκαταστάθηκε και “έτρεχε” σε εικονικό μηχάνημα (Virtual Machine) του okeanos, πάντα, βέβαια, μέσα σε Docker container, κάτι το οποίο απέκλειε προβλήματα κατά την αλλαγή περιβάλλοντος εκτέλεσης (από τον okeanos στο Rasberry Pi).

Κεφάλαιο 3. Προδιαγραφές και Αρχιτεκτονική Συστήματος

3.1 Προδιαγραφές Συστήματος

Πριν γίνει αναφορά στην αρχιτεκτονική της Quantified Self εφαρμογής, δηλαδή στα συστατικά μέρη που την αποτελούν και πως αυτά συνδέονται και επικοινωνούν μεταξύ τους, στην παράγραφο αυτή θα γίνει μια σύντομη περιγραφή των προδιαγραφών λειτουργίας του συστήματος. Συγκεκριμένα, θα αναλυθεί ποιος είναι ο χρήστης της εφαρμογής, ποια τα χαρακτηριστικά του και ποιες οι απαιτήσεις του από αυτή.

Έτσι, λοιπόν, στην παρακάτω λίστα φαίνονται τα βασικά χαρακτηριστικά του χρήστη:

- Ο χρήστης αθλείται συστηματικά
- Ο χρήστης παρακολουθεί συστηματικά διάφορους δείκτες που αφορούν στην υγεία του (βάρος, σφυγμοί, κτλ.)
- Ο χρήστης ενδιαφέρεται να καταγράφει τα δεδομένα που προκύπτουν από τις παραπάνω δραστηριότητές, ώστε να μπορεί να σχηματίσει εικόνα για την πορεία της εξέλιξής του και την πρόοδό του.
- Ο χρήστης, ενδεχομένως, διαθέτει συγκεκριμένες “έξυπνες” ιατρικές συσκευές με δυνατότητα σύνδεσης μέσω BLE
- Ο χρήστης χρησιμοποιεί Activity Trackers όπως το Fitbit ή το GoogleFit, ενδεχομένως και τα δύο ταυτόχρονα
- Ο χρήστης ενδιαφέρεται να μπορεί να δει ενοποιημένα τα δεδομένα που προκύπτουν από τις διάφορες πηγές
- Ο χρήστης ενδιαφέρεται να μπορεί να συγκρίνει τα δεδομένα του με αυτά άλλων χρηστών, ώστε να μπορεί να βγάλει ευρύτερα συμπεράσματα για τις επιδόσεις του

Με βάση τα παραπάνω χαρακτηριστικά, ο χρήστης έχει ορισμένες απαιτήσεις από την Quantified Self. Στις απαιτήσεις αυτές θα γίνει μια αρχική αναφορά σε αυτό το σημείο, ενώ στο κεφάλαιο 4 θα αναλυθεί λεπτομερώς πως αυτές υλοποιήθηκαν στο σύστημα.

Οι απαιτήσεις, λοιπόν, που έχει ο χρήστης από το σύστημα είναι οι εξής:

- Να μπορεί να κάνει εγγραφή και να δημιουργήσει προσωπικό λογαριασμό
- Να μπορεί να διασφαλίσει το απόρρητο του λογαριασμού, ώστε μόνο αυτός να έχει πρόσβαση
- Να μπορεί να συνδέσει τον λογαριασμό του με το προφίλ του σε υπηρεσίες όπως το Fitbit και το GoogleFit, ώστε να λαμβάνει τα δεδομένα που αυτές καταγράφουν σχετικά με τις δραστηριότητές του
- Να μπορεί, στην συνέχεια, να αποσυνδέσει το προφίλ του αυτό από το σύστημα, σε περίπτωση που σταματήσει να το χρησιμοποιεί
- Να έχει πρόσβαση σε μια εύχρηστη και αισθητικά ελκυστική διεπαφή, μέσω της οποίας να μπορεί να παρακολουθεί την πορεία του στον χρόνο
- Να μπορεί να αποθηκεύει στο σύστημα μετρήσεις που αφορούν στα βιοσήματά του και, όπως και πριν, να μπορεί να παρακολουθεί πως τα βιοσήματα αυτά εξελίσσονται στον χρόνο
- Να μπορεί να θέσει στόχους προς επίτευξη, αναφορικά με την σωματική του δραστηριότητα
- Να λαμβάνει αποτελέσματα από την ανάλυση των δεδομένων του, τα οποία θα του δίνουν την δυνατότητα να εντοπίσει τις αδυναμίες ή τις ελλείψεις του και να τις αντιμετωπίσει

- Να λαμβάνει μηνύματα σχετικά με την πρόοδό του που θα τον ενθαρρύνουν ή παρακινούν να συνεχίσει την προσπάθεια, και θα τον επιβραβεύουν όταν επιτύχει τους στόχους που έχει θέσει
- Να έχει πρόσβαση σε αποτελέσματα και μετρήσεις που αφορούν το σύνολο των χρηστών, ώστε να μπορεί να συγκρίνει και να προσαρμόσει τις δικές του δραστηριότητές ανάλογα
- Συμπληρωματικά, ο χρήστης ενδιαφέρεται να μπορεί να καταγράψει ορισμένες σκέψεις του και να διατηρεί κάποιο σημειωματάριο

3.2 Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική της εφαρμογής ακολουθεί το μοντέλο Client-Server. Με άλλα λόγια, τα δύο βασικά συστατικά μέρη του συστήματος είναι ο Εξυπηρετητής ή Server από τη μία, και από την άλλη ο Πελάτης ή Client, ο οποίος στέλνει αιτήματα προς διεκπεραίωση στον Server. Συγκεκριμένα, ο Server υλοποιεί μια διεπαφή προγραμματισμού εφαρμογών ή API (Application Programming Interface) τύπου REST, στην οποία έχει πρόσβαση ο Client. Η Διεπαφή αυτή υλοποιεί τις μεθόδους εκείνες που χρειάζεται ο χρήστης, ώστε να ικανοποιηθούν οι απαιτήσεις του, όπως ορίστηκαν παραπάνω.

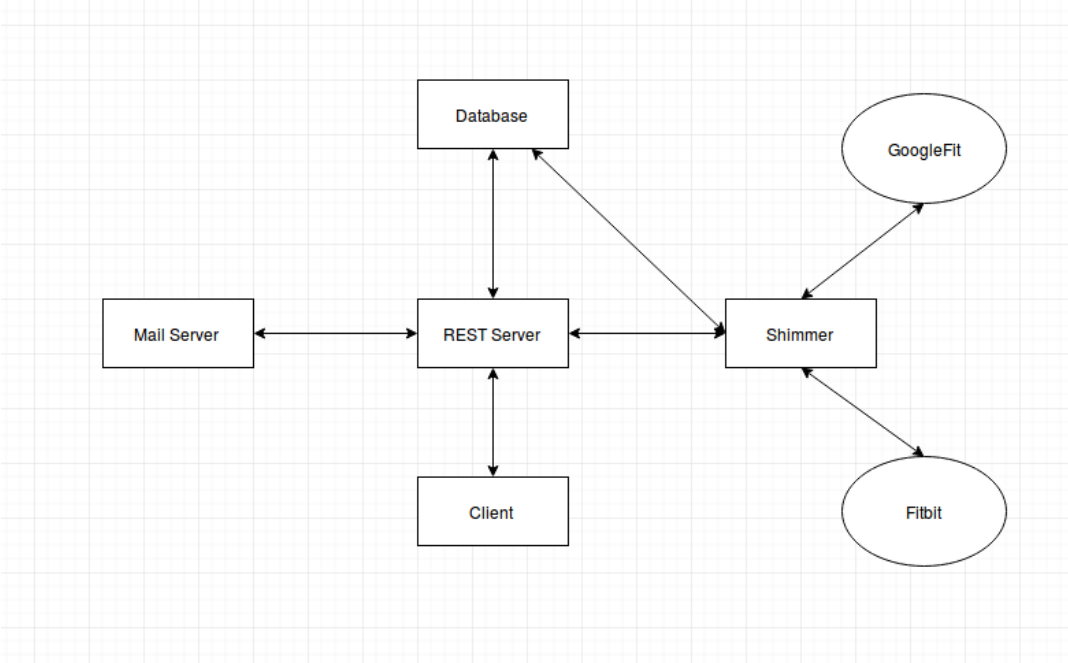
Απαραίτητο κομμάτι της συνολικής αρχιτεκτονικής είναι και η Βάση Δεδομένων, η οποία αποθηκεύει όλες τις απαραίτητες πληροφορίες και τα δεδομένα για τη λειτουργία της εφαρμογής. Σε αυτά περιλαμβάνονται τα δεδομένα των χρηστών της εφαρμογής, όπως η διεύθυνση ηλεκτρονικού ταχυδρομείου (που λειτουργεί ως username ή μοναδικό αναγνωριστικό του χρήστη), οι κωδικοί πρόσβασης τους (αφού έχουν κρυπτογραφηθεί με κατάλληλα εργαλεία) ή τους στόχους που έχει θέσει ο κάθε χρήστης αναφορικά με τις δραστηριότητες του. Επίσης, η βάση περιέχει και τα βιοσήματα και τα δεδομένα δραστηριότητας των χρηστών, καθώς επίσης και τα διάφορα στατιστικά που προκύπτουν από την ανάλυση των δεδομένων αυτών. Τέλος, στην βάση δεδομένων υπάρχουν και οι κατάλληλες πληροφορίες που επιτρέπουν την διασύνδεση της εφαρμογής με τα APIs του Fitbit και του GoogleFit, μέσω του Shimmer.

Με την βάση δεδομένων επικοινωνεί, καταρχάς, ο REST Server, προκειμένου να ανακτήσει δεδομένα που ζητήθηκαν από τον χρήστη, να αποθηκεύσει καινούργια δεδομένα που ο χρήστης χρειάζεται να καταχωρήσει ή να επεξεργαστεί ήδη υπάρχοντα δεδομένα της βάσης ώστε να απαντήσει σε κάποιο αίτημα του χρήστη. Επίσης, και η εφαρμογή Shimmer έχει πρόσβαση στην βάση δεδομένων, από την οποία αντλεί τις πληροφορίες που χρειάζεται κατά την επικοινωνία της με τις υπηρεσίες του Fitbit ή του GoogleFit.

Το σύστημα του πελάτη, αντίθετα, υλοποιεί την διεπαφή χρήστη και όλες τις λειτουργικότητες που αυτή απαιτεί ώστε να επικοινωνεί με τον Server, να λαμβάνει και να στέλνει δεδομένα από και προς αυτόν, και να τα παρουσιάζει με τον κατάλληλο τρόπο στον χρήστη.

Τέλος, υπάρχει και ένας Εξυπηρετητής Ηλεκτρονικού Ταχυδρομείου ή Mail Server, ο οποίος αναλαμβάνει την αποστολή μηνυμάτων προς τον χρήστη. Τέτοια μηνύματα στέλνονται κατά την εγγραφή του χρήστη και την δημιουργία λογαριασμού στην εφαρμογή ή σε περιπτώσεις όπου ο χρήστης έχει ξεχάσει τον κωδικό πρόσβασης του και απαιτείται η επαναφορά του.

Στο επόμενο σχήμα δίνεται μια σχηματική αναπαράσταση της αρχιτεκτονικής αυτής.

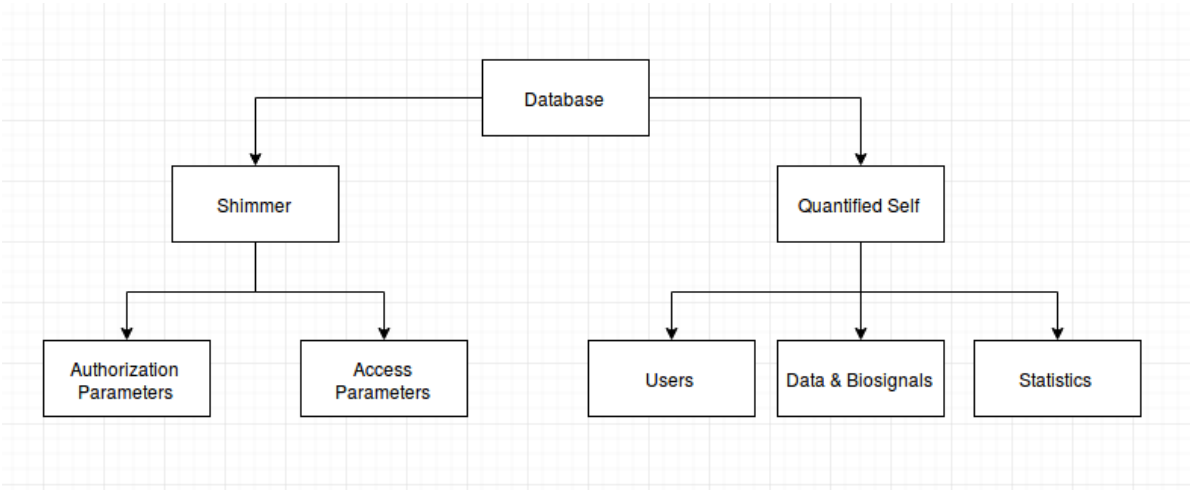


Σχήμα 14: Γενική Αρχιτεκτονική Εφαρμογής

Έχοντας, λοιπόν, την παραπάνω γενική αρχιτεκτονική κατά νου, θα αναλυθούν, στο σημείο αυτό με περισσότερες λεπτομέρειες οι λειτουργίες των δύο σημαντικότερων επιμέρους συστημάτων, δηλαδή της Βάσης Δεδομένων και του Εξυπηρετητή.

Database

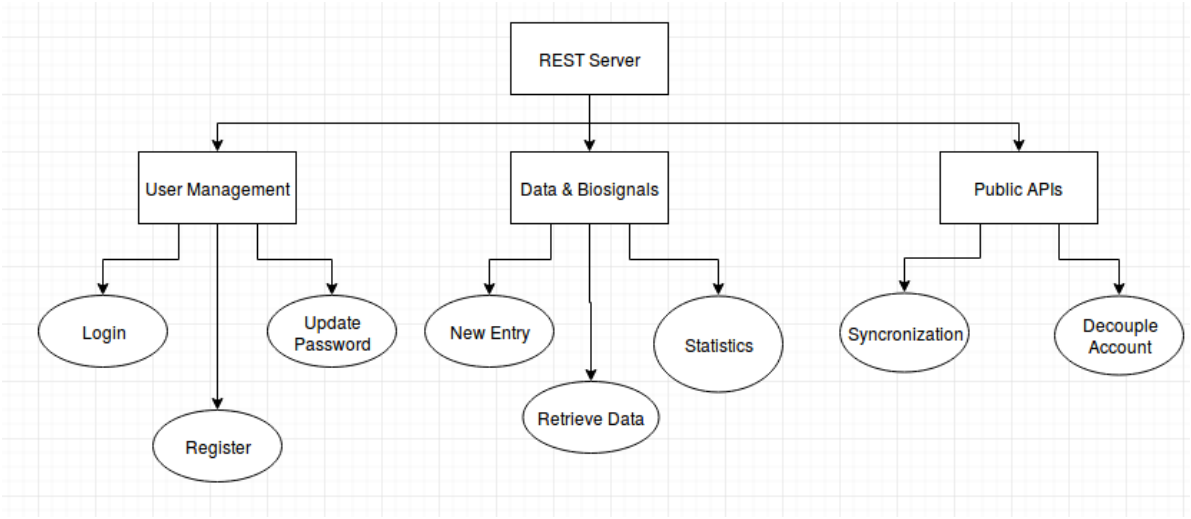
Όπως αναφέρθηκε ήδη, η βάση δεδομένων έχει αποθηκευμένα όλα τα δεδομένα που είναι απαραίτητα για την λειτουργία της εφαρμογής. Σε ένα πρώτο επίπεδο, αυτά χωρίζονται σε δεδομένα που χρησιμοποιεί το Shimmer και σε δεδομένα που χρησιμοποιεί η Quantified Self εφαρμογή καθ΄εαυτή. Τα μεν πρώτα χωρίζονται σε δεδομένα Authorization και σε δεδομένα Access (περισσότερα στο επόμενο κεφάλαιο), ενώ τα δεύτερα χωρίζονται σε τρεις κατηγορίες: σε δεδομένα που σχετίζονται με τους χρήστες (email, κωδικοί κτλ.), σε δεδομένα βιοσημάτων και δραστηριότητας και, τέλος, σε δεδομένα που αφορούν την στατιστική ανάλυση των δεδομένων. Η δομή αυτή φαίνεται σχηματικά ακολούθως:



Σχήμα 15: Δομή της Βάσης Δεδομένων του Συστήματος

REST Server

Ο Εξυπηρετητής αρχιτεκτονικής REST αναλαμβάνει να επεξεργαστεί και να απαντήσει στα αιτήματα του χρήστη (Πελάτη). Τα αιτήματα αυτά μπορούν να χωριστούν σε τρεις κατηγορίες: σε αιτήματα που αφορούν το προφίλ του χρήστη, σε αιτήματα που αφορούν στα βιοσήματα και την δραστηριότητα του χρήστη και σε αιτήματα που σχετίζονται με τις υπηρεσίες του GoogleFit και του Fitbit. Έτσι, λοιπόν, και οι μέθοδοι που υποστηρίζει το API του Server μπορούν να χωριστούν και αυτές ανάλογα. Στο ακόλουθο σχήμα φαίνεται αυτός ο διαχωρισμός, μαζί με κάποια συγκεκριμένα παραδείγματα λειτουργιών που επιτελεί ο Server από κάθε κατηγορία.



Σχήμα 16: Αρχιτεκτονική του REST Server

Συγκεκριμένα, στην κατηγορία του User Management ανήκουν λειτουργίες όπως η εγγραφή του χρήστη στο σύστημα, η αναγνώριση και εξουσιοδότηση του (User Authentication & Authorization) ή αλλαγή κωδικού του χρήστη. Στην κατηγορία Data & Biosignals ανήκουν λειτουργίες όπως καταχώρηση καινούργιας μέτρησης, η ανάκτηση των δεδομένων του χρήστη (με στόχο την οπτικοποίηση τους στην διεπαφή του Πελάτη) ή στατιστική ανάλυση των δεδομένων και η ανάκτηση των συμπερασμάτων. Τέλος, στην κατηγορία Public APIs ανήκουν οι λειτουργίες που σχετίζονται με τον συγχρονισμό του συστήματος με τις πλατφόρμες του Fitbit και του GoogleFit. Κατά τον συγχρονισμό αυτό, ο Server αναθέτει στην εφαρμογή Shimmer να συνδεθεί με τα APIs των πλατφορμών αυτών, ώστε να ανακτήσει τα δεδομένα που αυτές έχουν καταγράψει και να τα μεταφέρει στο σύστημα της Quantified Self εφαρμογής. Ένα άλλο παράδειγμα λειτουργίας του Server που επιτελείται με την διαμεσολάβηση του Shimmer είναι η σύνδεση ή η αποσύνδεση του λογαριασμού του χρήστη με το προφίλ του στο GoogleFit ή το Fitbit.

Κεφάλαιο 4. Υλοποίηση Συστήματος

Στο κεφάλαιο αυτό θα αναλυθεί η υλοποίηση της Quantified Self εφαρμογής. Συγκεκριμένα, θα δειχθεί πως οι προδιαγραφές και απαιτήσεις που συζητήθηκαν παραπάνω υλοποιήθηκαν σε τεχνικό επίπεδο και πως οι τεχνολογίες και τα εργαλεία, στα οποία έγινε αναφορά στο 2ο κεφάλαιο, χρησιμοποιήθηκαν για την υλοποίηση της αρχιτεκτονικής του συστήματος, όπως αυτή σχεδιάστηκε στο 3ο κεφάλαιο.

Γενικά, η ανάπτυξη του όλου συστήματος έγινε στην γλώσσα Javascript, τόσο από την πλευρά του πελάτη, όσο και από την πλευρά του εξυπηρετητή. Όσον αφορά την πλευρά του πελάτη, επιλέχθηκε το framework της AngularJS, το οποίο διαθέτει μερικά αρκετά ελκυστικά χαρακτηριστικά όπως:

- Υποστηρίζει αρχιτεκτονική REST και επικοινωνία με Server
- Ακολουθεί το μοντέλο MVC (Model-View-Controller), το οποίο παρέχει σημαντική ευελιξία και Διαχωρισμό των Ανησυχιών (Separation of Concerns)
- Υποστηρίζει το templating, την αυτόματη παραγωγή, δηλαδή, περιεχομένου βασισμένου σε υπάρχοντα μοντέλα
- Υποστηρίζει το two-way-binding, κάτι το οποίο σημαίνει ότι αλλαγές στο View μεταδίδονται στο Model και αντίστροφα. Αυτό έχει ως αποτέλεσμα την σχεδόν αυτόματη ενημέρωση (update) του περιεχομένου

Για την πλευρά του εξυπηρετητή επιλέχθηκε το περιβάλλον εκτέλεσης (run-time environment) NodeJS, μερικά από τα πλεονεκτήματα του οποίου είναι τα εξής:

- Το NodeJS είναι γρήγορο, καθώς χρησιμοποιεί το Chrome V8 [38] για την μετατροπή της Javascript απευθείας σε γλώσσα μηχανής
- Το NodeJS καθιστά ιδιαίτερα εύκολο τον χειρισμό αιτημάτων που αφορούν είσοδο/έξοδο δεδομένων, π.χ. από την βάση δεδομένων
- Το NodeJS παρέχει διευκολύνει σημαντικά την κλιμακωσιμότητα (scalability) της εφαρμογής
- Το NodeJS είναι Javascript, μια γλώσσα σχετικά εύκολη και με μεγάλη κοινότητα προγραμματιστών

Επιπλέον, “πάνω” από το NodeJS επιλέχθηκε ως Web Framework το ExpressJS, κυρίως λόγω της ευκολίας που παρέχει, μέσω κατάλληλα υλοποιημένων βιβλιοθηκών, στην δρομολόγηση (routing), δηλαδή στον τρόπο με τον οποίο δομείται η κατηγοριοποίηση και η οργάνωση των αιτημάτων (http requests) και των αντίστοιχων μεθόδων του API.

Τέλος, τόσο η εφαρμογή Quantified Self, όσο και η βοηθητική εφαρμογή Shimmer, φιλοξενούνται στο μηχάνημα του Okeanos με διεύθυνση

imeasure.menychtas.com

Η μεν Quantified Self εφαρμογή “τρέχει” στην θύρα 8000 (imeasure.menychtas.com:8000), το δε Shimmer στην θύρα 8083 (imeasure.menychtas.com:8083).

Μετά από αυτά τα εισαγωγικά, θα ακολουθήσει αναλυτική περιγραφή του πως έγινε η ανάπτυξη της εφαρμογής και των λειτουργιών (features) της.

4.1 Σχεδιασμός της Βάσης Δεδομένων Mongo

Αρχικά, θα γίνει αναφορά στην υλοποίηση της βάσης δεδομένων, για την οποία επιλέχθηκε η τεχνολογία MongoDB. Η εγγραφο-στρέφεια της MongoDB την καθιστά κατάλληλη στην περίπτωση δεδομένων όπως αυτά που χειρίζεται η Quantified Self εφαρμογή, καθώς η περιγραφή δεδομένων υγείας και ευεξίας, τόσο πολυπαραμετρικών και ετερογενών όπως το βάδισμα ή ο κορεσμός σε οξυγόνο, απαιτεί ευελιξία στην αποθήκευση, την ομαδοποίηση και την επεξεργασία τους, που δεν διαθέτει το πιο συμβατικό Σχεσιακό Μοντέλο (Relational Model).

Στα πλαίσια της ανάπτυξης σε περιβάλλον NodeJS, έγινε χρήση της βιβλιοθήκης MongooseJS [39], η οποία διευκολύνει την σύνδεση ενός NodeJS Server με μια βάση MongoDB, καθώς επίσης και τον ορισμό συλλογών εγγράφων (document collections) προς αποθήκευση στη βάση. Επίσης, παρέχει εργαλεία για την υλοποίηση ερωτημάτων στη βάση Mongo, τα οποία και χρησιμοποιήθηκαν εκτενώς κατά την ανάπτυξη.

Έτσι, με βάση τα παραπάνω, οι χρήστες της εφαρμογής αποθηκεύονται στην βάση ακολουθώντας ως εξής:

```
const UserSchema = mongoose.Schema({

  email: {
    type: String,
    index: true,
    required: true,
    unique: true,
    dropDups: true,
  },
  password: {
    type: String,
    required: true,
  },
  passwordRecoveryLink: String,
  accountConfirmed: {
    type: String,
    enum: [
      '0',
      '1'
    ]
  },
  confirmationLink: String,

  fitbitActivated: Boolean,
  googleFitActivated: Boolean,
  dataSharing: Boolean,

  fitbitTimestamp: {
    steps: String,
    sleep: String,
    weight: String,
    activity: String
  },
  googleFitTimestamp: {
    step_count: String,
    heart_rate: String,
    body_weight: String,
    activity: String,
    calories_burned: String
  },

  stats: {
      avgSteps: Number,
      avgCalories: Number,
      avgActivity: Number,
      maxSteps: Number,
      maxCalories: Number,
      maxActivity: Number
  },

  today: {
    steps: Number,
    calories: Number,
    activity: Number
  },

  stepsPerDayGoal: Number,
  caloriesPerDayGoal: Number,
  activityPerDayGoal: Number,

  firstName: {
    type: String,
    required: true
  },
  lastName: {
    type: String,
    required: true
  },
  gender: {
    type: String,
    enum: [
      'Male',
      'Female',
      'Other'
    ]
  },

});
```

Το παραπάνω απόσπασμα κώδικα ορίζει το σχήμα (schema) το οποίο περιγράφει τις εγγραφές τύπου User, όπως αυτές αποθηκεύονται στην βάση δεδομένων. Όπως εύκολα διαπιστώνει κανείς, το σχήμα αυτό ακολουθεί μια δομή τύπου JSON, καθώς ένας χρήστης περιγράφεται από ένα σύνολο ζευγών key-value. Κάποια από αυτά τα πεδία είναι υποχρεωτικά, κάποια προαιρετικά, κάποια αποτελούν πεδία index, δηλαδή πεδία με βάση τα οποία γίνεται γρήγορη αναζήτηση στο σύνολο των εγγραφών, κτλ. Όλα αυτά αποτελούν δυνατότητες που παρέχει το MongooseJS.

Ενδεικτικά:

- το πεδίο *email* είναι υποχρεωτικό, είναι τύπου String (συμβολοσειρά) και αποτελεί index, κάτι το οποίο σημαίνει πως οι αναζήτηση χρηστών με βάση το email γίνεται με βέλτιστο τρόπο από την βάση
- το πεδίο *GoogleFitActivated* είναι τύπου Boolean, και παίρνει τιμή True όταν ο χρήστης έχει συνδέσει τον λογαριασμό του με το προφίλ του στο GoogleFit
- το πεδίο *stats* αποτελείται από άλλα υποπεδία, καθένα από τα οποία είναι ένας αριθμός και αντιστοιχεί σε κάποιο στατιστικό αποτελέσματα της ανάλυσης των δεδομένων του χρήστη

Με παρόμοιο τρόπο γίνεται και η υλοποίηση του σχήματος που περιγράφει τα βιοσήματα του χρήστη. Μάλιστα, η υλοποίηση αυτή ακολουθεί τα πρότυπα που έχουν οριστεί από το Open mHealth. Ακολουθεί ο κώδικας:

```
const BiosignalSchema = mongoose.Schema({

  header: {
    creation_date_time: {
      type: Date
    },
    schema_id: {
      name: {
        type: String,
        enum: [
          "heart-rate",
          "body-weight",
          "blood-glucose",
          "oxygen-saturation",
          "blood-pressure",
          "step-count",
          "physical-activity",
          "sleep-duration",
          "calories-burned"
        ],
        index: true,
        required: true
      }
    },
    acquisition_provenance: {
        source_name: {
          type: String
        },
        source_id: {
          type: String
        },
        source_creation_date_time: {
          type: Date
        },
        modality: {
          type: String,
          enum: [
            "sensed",
            "self-reported"
          ]
        }
      },
      user_id: {
        type: String,
        index: true,
        required: true
      }
    },

    body: {
      type: Object,
      required: true
    }

  });
```

Ένα σημείο το οποίο έχει ενδιαφέρον αναφορικά με το παραπάνω σχήμα είναι το πεδίο *body*, το οποίο είναι υποχρεωτικό και είναι τύπου Object, δηλαδή είναι ένα ακόμα αντικείμενο που αποτελείται από ζεύγη key-value. Με αυτόν τον τρόπο, κάτω από το σχήμα αυτό μπορούν να αποθηκευτούν κάθε είδους βιοσήματα, από τον σφυγμό μέχρι την σωματική δραστηριότητα, με αποτέλεσμα να υπάρχει μόνο μια συλλογή για όλα. Αυτό αποτελεί και ένδειξη της ευελιξίας που παρέχει η MongoDB.

Επιπροσθέτως, ορίζονται και δύο ακόμα σχήματα στην βάση. Το πρώτο περιγράφει τα στατιστικά εκείνα αποτελέσματα που αφορούν το σύνολο των χρηστών, ενώ το δεύτερο περιγράφει εγγραφές Σημειωματάριου, δηλαδή σημειώσεις που κρατάει ο χρήστης στο σύστημα. Οι αντίστοιχοι κώδικες φαίνονται παρακάτω.

Global Statistics:

```
const GlobalStatsSchema = mongoose.Schema({

   name: String,
   steps: Number,
   calories: Number,
   activity: Number

});
```

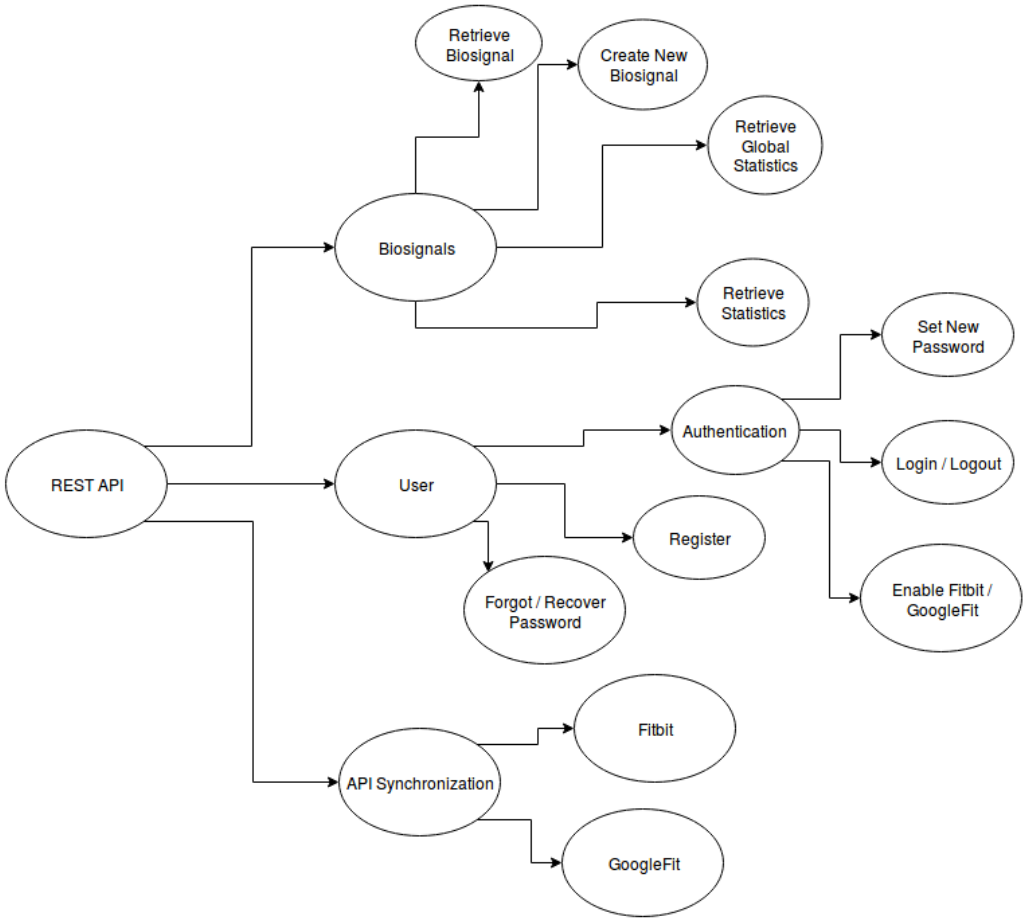
Diary:

```
const DiarySchema = mongoose.Schema({

   user_id: {
      type: String,
      index: true,
      required: true
   },
   items: [{
      date: String,
      text: String
   }]
});
```

Κάτι το οποίο έχει σημασία, τέλος, είναι η εξάρτηση που παρουσιάζουν ορισμένα από αυτά τα σχήματα από άλλα. Συγκεκριμένα, τα σχήματα του Σημειωματάριου και των Βιοσημάτων περιλαμβάνουν ένα πεδίο *user_id*, το οποίο αποτελεί αναφορά στο σχήμα User και ορίζει σε ποιόν χρήστη ανήκει το σημειωματάριο ή το βιοσήμα αντίστοιχα.

4.2 Δρομολόγηση Αιτημάτων

Όπως προαναφέρθηκε, η χρήση του ExpressJS επιτρέπει την δρομολόγηση των αιτημάτων με ομοιόμορφο και απλό τρόπο. Έτσι, η υλοποίηση έγινε ούτως ώστε τα αιτήματα που μπορεί να απευθύνει ο πελάτης στον εξυπηρετητή να χωρίζονται, σε ένα πρώτο επίπεδο, σε τρεις κατηγορίες: σε αιτήματα βιοσημάτων, σε αιτήματα σχετικά με το προφίλ του χρήστη και σε αιτήματα συγχρονισμού του συστήματος με το προφίλ του χρήστη στο Fitbit ή GoogleFit. Στην συνέχεια, η δρομολόγηση προχωράει ένα επίπεδο παρακάτω, είτε καταλήγοντας στην κλήση της τελικής μεθόδου του API, είτε, όπως συμβαίνει στην περίπτωση αιτημάτων τύπου User, περνάει από ένα ακόμα ενδιάμεσο επίπεδο, αυτό του Authentication. Αυτό συμβαίνει, καθώς υπάρχουν αιτήματα, όπως αυτό για την δημιουργία λογαριασμού (Register) τα οποία δεν απαιτούν επαλήθευση του χρήστη. Άλλα αιτήματα, όμως, όπως αυτό για σύνδεση στον λογαριασμό (login) ή για την αντικατάσταση του κωδικού πρόσβασης (Set New Password) απαιτούν πιστοποίηση. Προφανώς, όλα τα αιτήματα τύπου Biosignals και τύπου API Synchronization απαιτούν πιστοποίηση, για αυτό και δεν υπάρχει, σε αυτές τις περιπτώσεις, ενδιάμεσο στάδιο δρομολόγησης. Στο επόμενο σχήμα φαίνεται αυτή ακριβώς η δεντρική δομή της δρομολόγησης των αιτημάτων.



Σχήμα 17: Δρομολόγηση Αιτημάτων

4.3 Υλοποίηση συστήματος Εγγραφής Χρήστη

Για να μπορέσει ο χρήστης να χρησιμοποιήσει την Quantified Self εφαρμογή θα πρέπει, καταρχάς, να κάνει εγγραφή (Register) στο σύστημα. Το πως αυτό υλοποιείται από πλευράς χρήστη (front-end) θα αναλυθεί παρακάτω. Στο σημείο αυτό θα γίνει περιγραφή της υλοποίησης από πλευράς εξυπηρετητή (back-end).

Καθώς το όλο σύστημα είναι βασισμένο στην αρχιτεκτονική REST, η εγγραφή του χρήστη είναι, όπως και όλα τα άλλα αιτήματα, ένα HTTP Request. Συγκεκριμένα, πρόκειται για ένα αίτημα τύπου POST, κατά το οποίο ο χρήστης στέλνει ένα αντικείμενο JSON στον Server, το οποίο φέρει τις απαραίτητες πληροφορίες για την δημιουργία καινούργιου λογαριασμού. Οι πληροφορίες αυτές είναι:

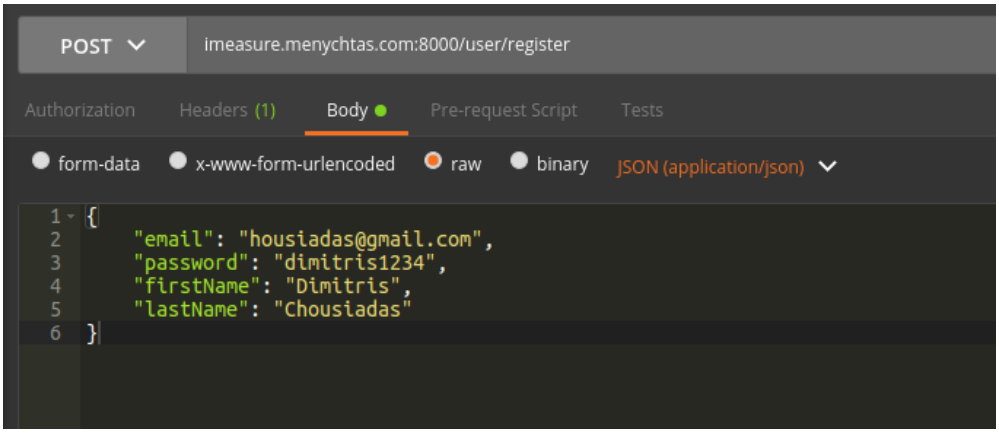
- το email του χρήστη, το οποίο και θα χρησιμοποιείται ως αναγνωριστικό (username)
- ο κωδικός του χρήστη, τον οποίο ο Server θα αποθηκεύσει κρυπτογραφημένο στην βάση δεδομένων
- το όνομα του χρήστη
- το επώνυμο του χρήστη

Το αίτημα αυτό στέλνεται στην διεύθυνση *imeasure.menychtas.com:8000/user/register,* η οποία αποτελεί το, όπως λέγεται, endpoint του λειτουργίας (service) της εγγραφής. Ο Server λαμβάνει το αίτημα αυτό και, στην συνέχεια, κατασκευάζει ένα καινούργιο αντικείμενο, το οποίο ακολουθεί το σχήμα (schema) χρήστη, όπως αυτό περιγράφτηκε παραπάνω (βλ. παράγραφο 4.1).

Το καινούργιο αυτό αντικείμενο αποθηκεύεται στην βάση δεδομένων και, εν συνεχεία, στέλνεται ένα μήνυμα ηλεκτρονικού ταχυδρομείου (email) επιβεβαίωσης στην διεύθυνση που καταχώρησε ο χρήστης. Το email αυτό περιέχει έναν σύνδεσμο (link), τον οποίο πρέπει να ακολουθήσει ο χρήστης, ώστε να ενεργοποιήσει τον λογαριασμό του. Μετά και από αυτό το βήμα, ο χρήστης μπορεί, πλέον, να χρησιμοποιήσει την εφαρμογή.

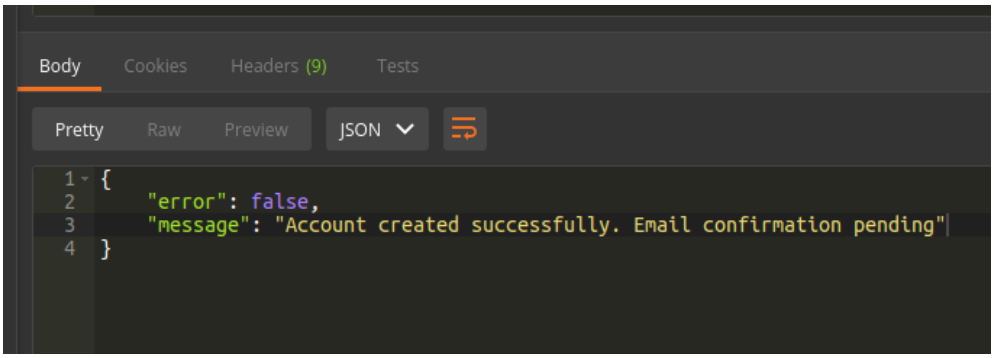
Η πορεία που ακολουθείται για να γίνει η εγγραφή φαίνεται και στα επόμενο σχήματα, που αποτελούν στιγμιότυπα από το πρόγραμμα Postman.

Αρχικά, το αίτημα στέλνεται στο αντίστοιχο endpoint:



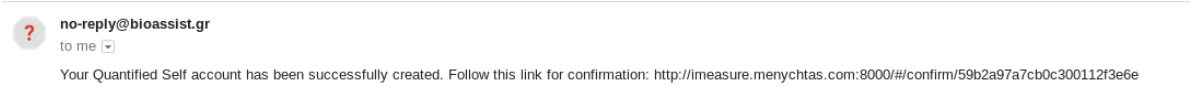
Σχήμα 18: Αποστολή Αιτήματος Εγγραφής

Αφού λάβει και επεξεργαστεί το αίτημα, ο Server απαντάς ως εξής:



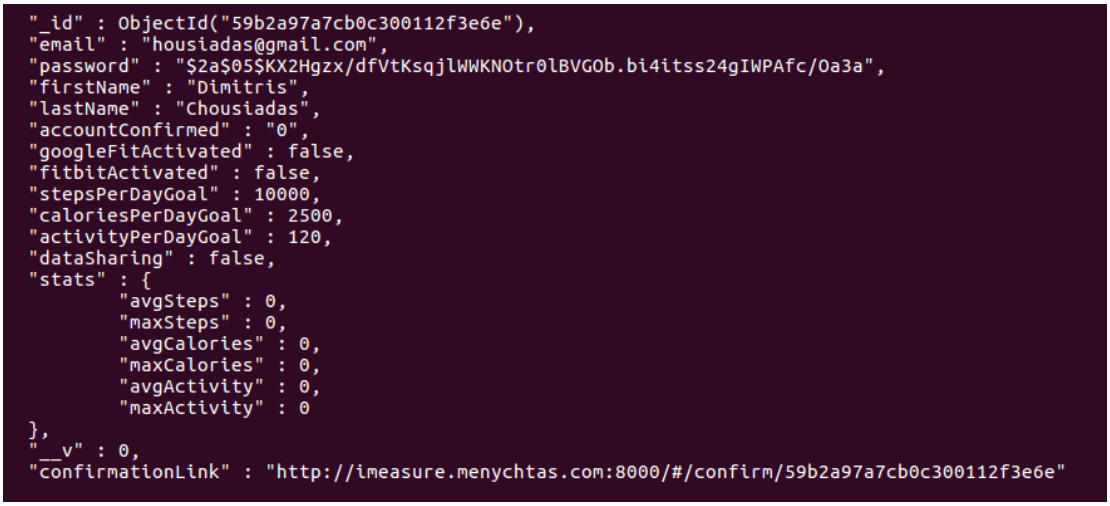
Σχήμα 19: Απάντηση Εξυπηρετητή στο Αίτημα Εγγραφής

Όπως φαίνεται, ο Server έκανε δεκτό το αίτημα και έστειλε email επιβεβαίωσης στον χρήστη. Το email αυτό φαίνεται στη επόμενη εικόνα:



Σχήμα 20: Email Επιβεβαίωσης

Την ίδια στιγμή, στην βάση δεδομένων, ο Server αποθήκευσε μια καινούργια εγγραφή τύπου χρήστη, όπως φαίνεται ακολούθως:



Σχήμα 21: Αναπαράσταση Ανεπιβεβαίωτου Χρήστη στην Βάση

Προς το παρόν, ο λογαριασμός του χρήστη δεν έχει επιβεβαιωθεί, όπως φαίνεται και από το πεδίο *accountConfirmed*, το οποίο έχει την τιμή 0. Όταν ο χρήστης ακολουθήσει τον σύνδεσμο που περιέχεται στο παραπάνω email, ο λογαριασμός θα επιβεβαιωθεί και το πεδίο αυτό θα πάρει την τιμή 1. Η αλλαγή αυτή φαίνεται στην επόμενη εικόνα.

```
"_id" : ObjectId("59b2a97a7cb0c300112f3e6e"),
"email" : "housiadas@gmail.com",
"password" : "$2a$05$KX2Hgzx/dfVtKsqjlWWKNOtr0lBVGOb.bi4itss24gIWPAfc/Oa3a",
"firstName" : "Dimitris",
"lastName" : "Chousiadas",
"accountConfirmed" : "1",
"googleFitActivated" : false,
"fitbitActivated" : false,
"stepsPerDayGoal" : 10000,
"caloriesPerDayGoal" : 2500,
"activityPerDayGoal" : 120,
"dataSharing" : false,
"stats" : {
        "avgSteps" : 0,
        "maxSteps" : 0,
        "avgCalories" : 0,
        "maxCalories" : 0,
        "avgActivity" : 0,
        "maxActivity" : 0
},
"__v" : 0,
"today" : {
        "activity" : 0,
        "calories" : 0,
        "steps" : 0
}
```


Σχήμα 22: Αναπαράσταση Χρήστη Μετά την Επιβεβαίωση Λογαριασμού

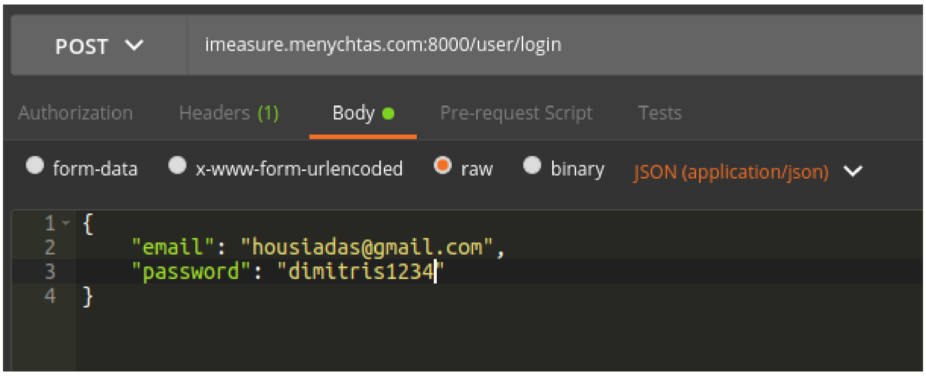
4.4 Είσοδος (login) στο σύστημα και User Authentication & Authorization

Ομοίως με την εγγραφή, ο χρήστης στέλνει ένα αίτημα POST κάθε φορά που θέλει να συνδεθεί στον λογαριασμό. Το αίτημα αυτό φέρει ένα αντικείμενο JSON στον οποίο περιλαμβάνονται το email του χρήστη και ο κωδικός πρόσβασης του. Όταν ο Server λάβει το αίτημα το επεξεργάζεται ως εξής:

- Αρχικά, αναζητά στην βάση δεδομένων κάποια εγγραφή χρήστη με το δοθέν email
- Αν δεν βρει τέτοια εγγραφή απαντά με μήνυμα λάθους, καθώς αυτό σημαίνει πως δεν υπάρχει χρήστης με αυτό το email.
- Αν βρεθεί τέτοια εγγραφή, ο Server κατακερματίζει (hash) τον δοθέν κωδικό πρόσβασης και τον συγκρίνει με τον ήδη κατακερματισμένο κωδικό που βρίσκεται αποθηκευμένος στην βάση. Αν οι δύο κατακερματισμένοι κωδικοί δεν είναι ίδιοι, ο Server δεν επιτρέπει την πρόσβαση στο σύστημα στον χρήστη.
- Διαφορετικά, αν είναι ίδιοι, η είσοδος στην εφαρμογή επιτρέπεται. Στο σημείο αυτό, η ταυτότητα του χρήστη έχει επαληθευθεί, δηλαδή έχει ολοκληρωθεί το στάδιο του User Authentication.

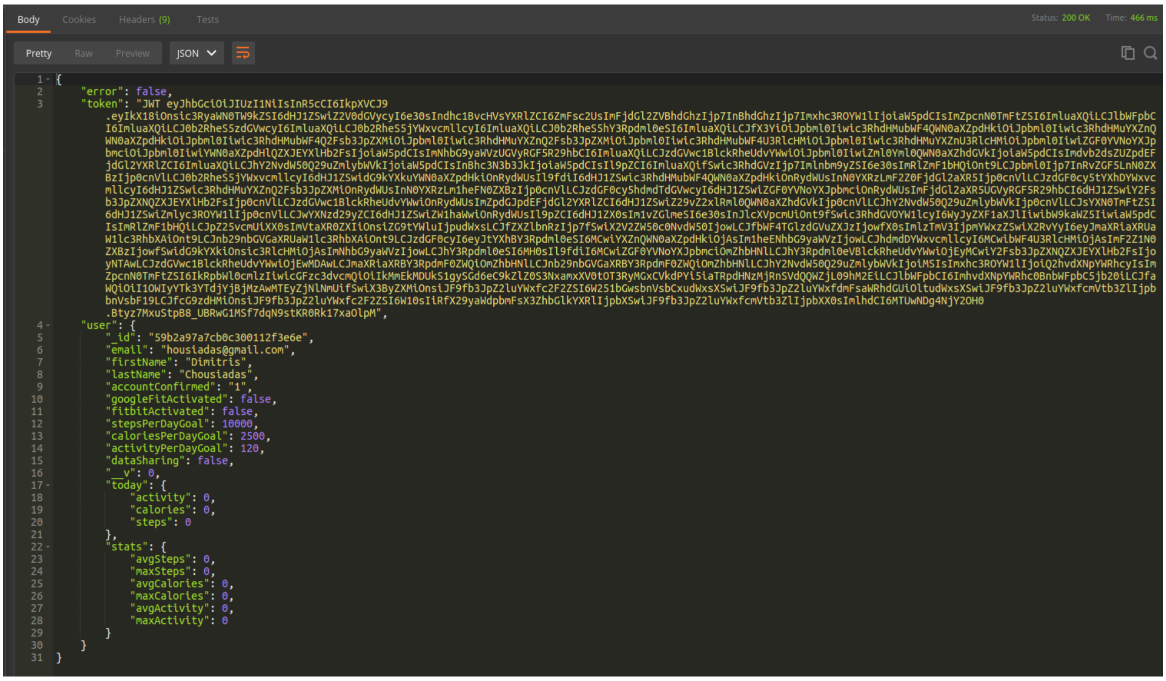
Μετά την επαλήθευση του χρήστη, ο Server επιστρατεύει το PassportJS. Συγκεκριμένα, με τη χρήση κατάλληλων συναρτήσεων, ο Server κατασκευάζει ένα token, μια συμβολοσειρά, δηλαδή, την οποία στέλνει πίσω στον Πελάτη, και την οποία ο Πελάτης οφείλει να χρησιμοποιεί, όσο παραμένει συνδεδεμένος στο σύστημα, κάθε φορά που στέλνει κάποιο αίτημα στον Server. Το token αυτό αποτελεί απόδειξη του ότι ο χρήστης έχει δικαίωμα πρόσβασης στους πόρους του συστήματος. Κάθε αίτημα του χρήστη χωρίς αυτό το token απορρίπτεται από τον Server.

Στις ακόλουθες εικόνες φαίνονται τα διάφορα στάδια ενός αιτήματος Login.



Σχήμα 23: Αίτημα Σύνδεσης στην Εφαρμογή

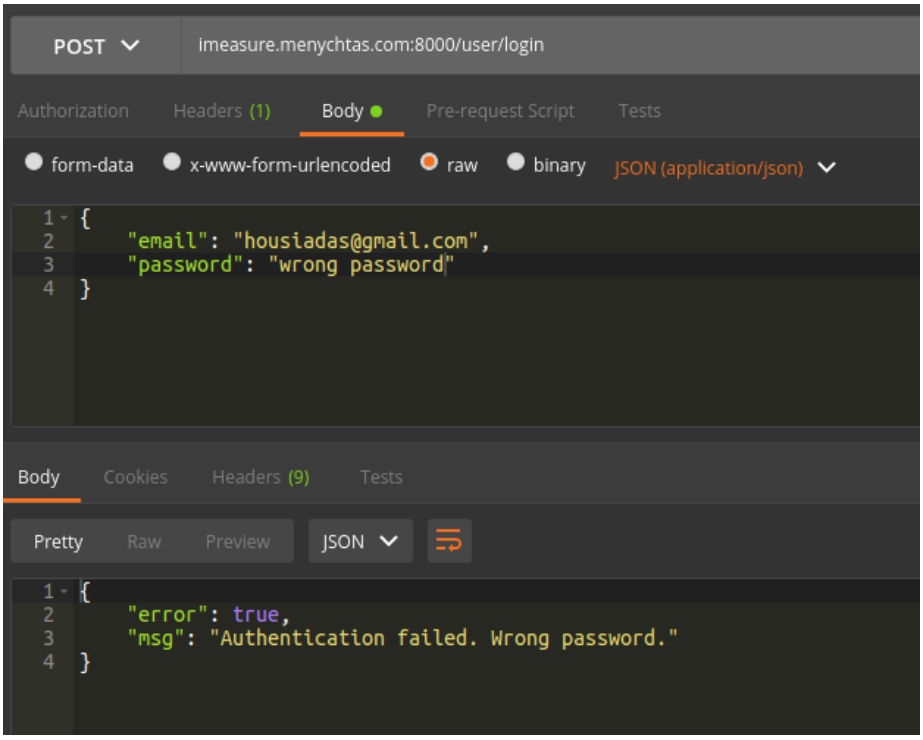
Αρχικά, στέλνεται ένα POST αίτημα που περιλαμβάνει το email του χρήστη και το password του. Αν τα στοιχεία αυτά είναι σωστά, ο Server απαντά στον Client, στέλνοντας, μαζί με τις πληροφορίες του προφίλ του, και ένα token, το οποίο και αφορά τον χρήστη αυτόν και μόνο αυτόν. Η απάντηση αυτή του Server φαίνεται στην επόμενη εικόνα:



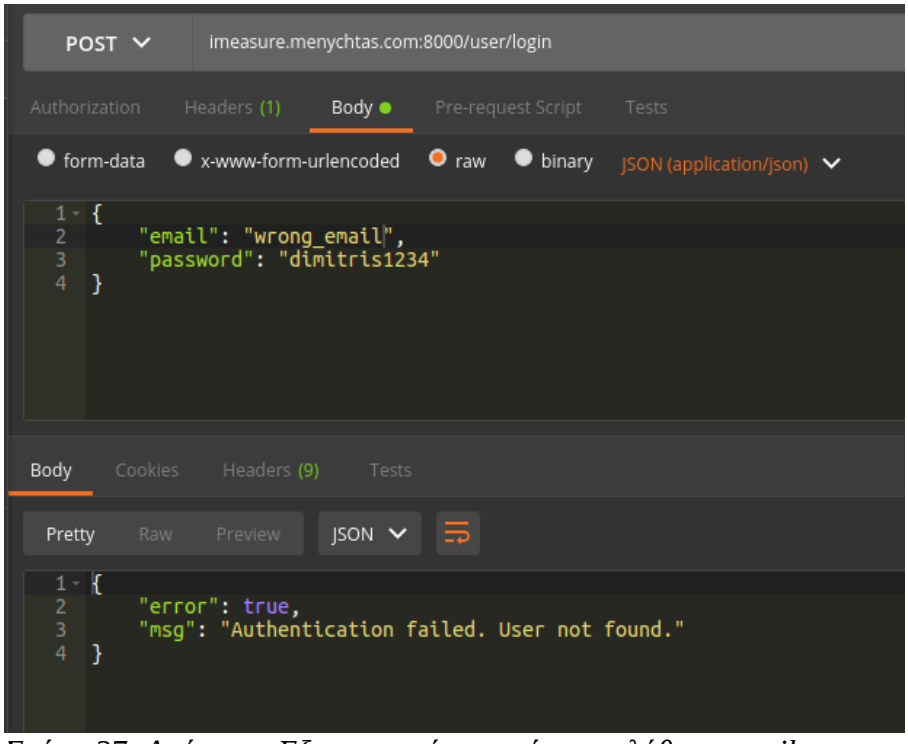
Σχήμα 24: Απάντηση Εξυπηρετητή σε αίτημα Σύνδεσης

Σχήμα 25: Token

Τα παραπάνω ισχύουν στην περίπτωση, φυσικά, που ο χρήστης δίνει τα σωστά στοιχεία. Αν όμως ο χρήστης δώσει λάθος email ή λάθος κωδικό πρόσβασης, ο Εξυπηρετητής θα απαντήσει με μήνυμα λάθους και δεν θα επιτρέψει την είσοδο στον χρήστη, όπως φαίνεται παρακάτω:



Σχήμα 26: Απάντηση Εξυπηρετητή σε περίπτωση λάθους κωδικού



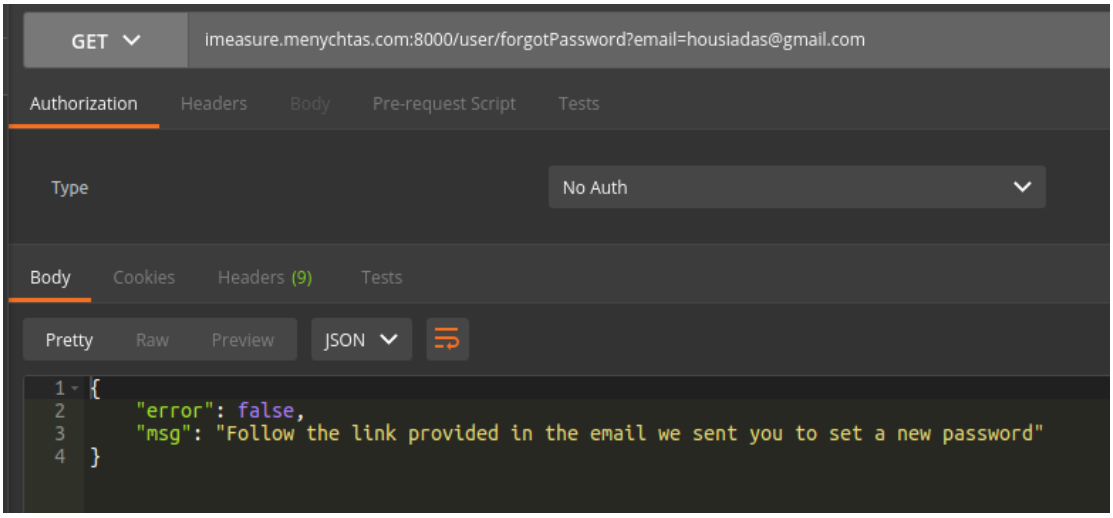
Σχήμα 27: Απάντηση Εξυπηρετητή σε περίπτωση λάθους email

4.5 Σύστημα Ανάκτησης Κωδικού Πρόσβασης

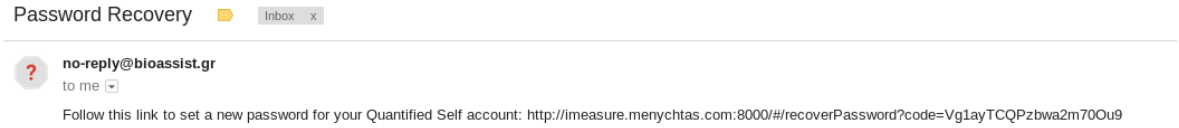
Όπως σε κάθε εφαρμογή με χρήστες, έτσι και στην Quantified Self εφαρμογή, πρέπει να ληφθεί υπόψιν το ενδεχόμενο ο χρήστης να ξεχάσει τον κωδικό πρόσβασης του. Στην περίπτωση αυτή, πρέπει να υπάρχει ο κατάλληλος μηχανισμός ώστε να μπορεί ο χρήστης να ανακτήσει πρόσβαση στον λογαριασμό του.

Εν προκειμένω, η Quantified Self εφαρμογή δίνει την δυνατότητα στον χρήστη να ανακτήσει πρόσβαση μέσω του ηλεκτρονικού ταχυδρομείου. Με την αποστολή ενός GET αιτήματος, το οποίο θα φέρει ως παράμετρο το email του, ο χρήστης λαμβάνει ένα μήνυμα στο οποίο υπάρχει ένας σύνδεσμος, που, στην συνέχεια, τον ακολουθεί ώστε να θέσει έναν καινούργιο password. Στον σύνδεσμο αυτό υπάρχει και ένας κωδικός. Η καταχώρηση του καινούργιου password είναι ένα νέο POST αίτημα προς τον Server, το οποίο έχει ως παράμετρο τον κωδικό αυτόν.

Στα επόμενα στιγμιότυπα φαίνεται η ακολουθία αυτή των βημάτων ανάκτησης κωδικού πρόσβασης:



Σχήμα 28: Αίτημα Ανάκτησης Κωδικού Πρόσβασης



Σχήμα 29: Email με σύνδεσμο ανάκτησης κωδικού πρόσβασης

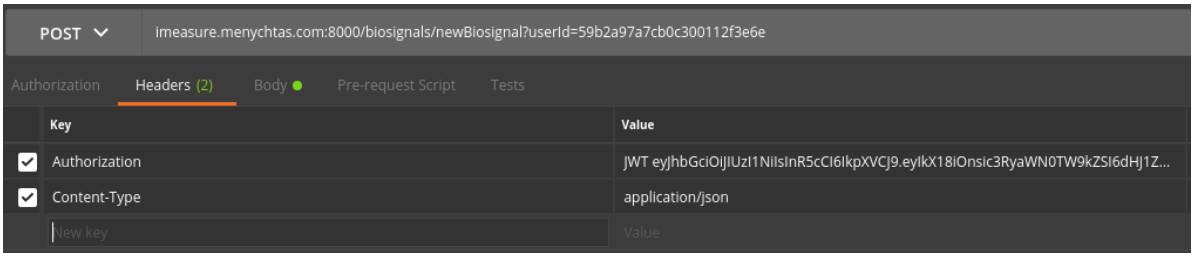
Σχήμα 30: Αίτημα POST για καταχώρηση καινούργιου password

4.6 Αποθήκευση Νέων Βιοσημάτων

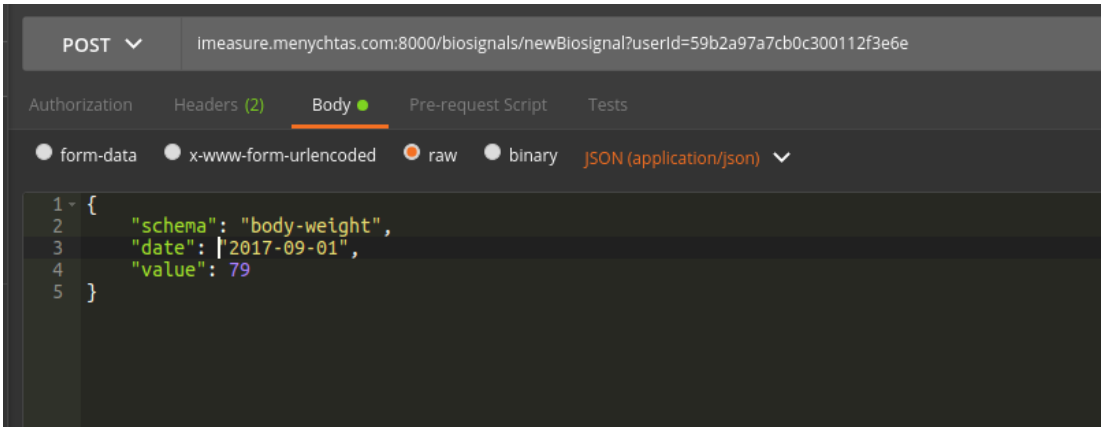
Η εφαρμογή δίνει την δυνατότητα στον χρήστη, μέσα από την διεπαφή (UI), να δημιουργήσει και να αποθηκεύσει καινούργια βιοσήματα (βάρος, καρδιακοί παλμοί, κτλ.). Η λειτουργία αυτή υλοποιείται με ένα αίτημα POST του χρήστη προς τον Server, το οποίο αποτελείται από:

- μια παράμετρο *userId* στο URL του αντίστοιχου endpoint (*/biosignals/newBiosignal?userId=' + <user id>*), η οποία αντιστοιχεί στο μοναδικό ID που δημιουργεί το σύστημα, συγκεκριμένα η βάση Mongo, αυτόματα, κατά την εγγραφή του χρήστη
- το token επαλήθευσης του χρήστη (βλ. παράγραφο 4.5), το οποίο περιλαμβάνεται στις επικεφαλίδες (headers) του αιτήματος
- το σώμα (body) του αιτήματος, το οποίο είναι ένα JSON αντικείμενο με τα εξής πεδία: το σχήμα (schema) του νέου βιοσήματος, την ημερομηνία μέτρησης του βιοσήματος αυτού, καθώς και την τιμή (ή τιμές, αν πρόκειται για βιοσήμα αρτηριακή πίεση, που αποτελείται από συστολική και διαστολική πίεση) τις μέτρησης αυτής

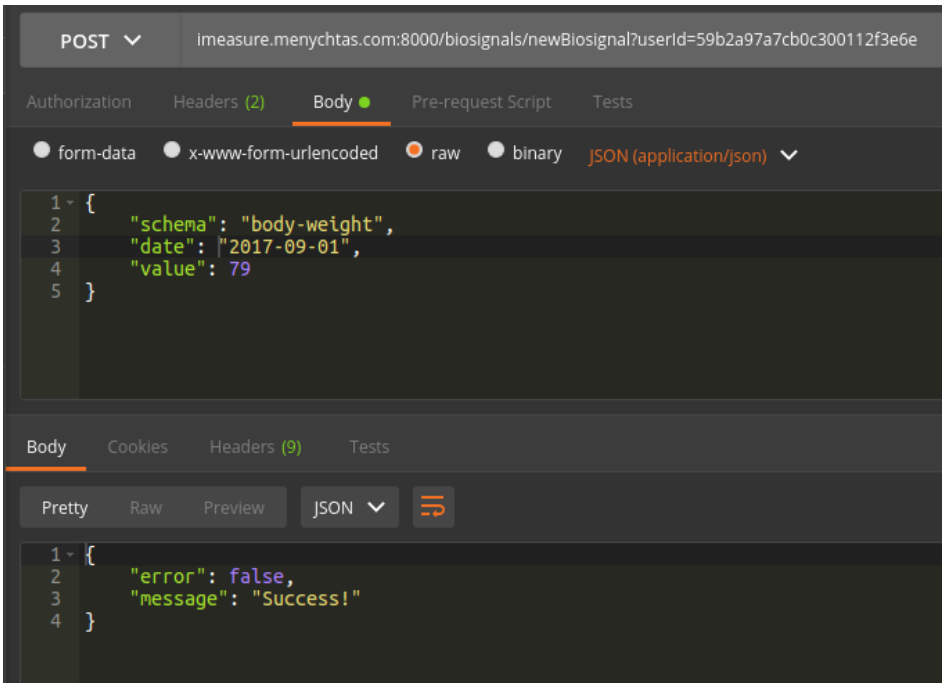
Αφού ο Εξυπηρετητής λάβει το αίτημα, αρχικά, ελέγχει αν ο χρήστης έχει τα αντίστοιχα δικαιώματα, χρησιμοποιώντας το userId και το token. Στη συνέχεια, αφού περάσει το στάδιο του Authorization, ο Εξυπηρετητής εξετάζει εάν τα δεδομένα που περιλαμβάνει το σώμα του αιτήματος είναι σωστά, δηλαδή σύμφωνα με τις προδιαγραφές του Open mHealth. Όταν γίνει και αυτός ο έλεγχος, ο Εξυπηρετητής δημιουργεί ένα καινούργιο αντικείμενο, με βάση το σώμα του αιτήματος, προσθέτοντας μερικά επιπλέον πεδία, που αφορούν μετα-δεδομένα του βιοσήματος, και το αποθηκεύει στην βάση, απαντώντας με μήνυμα επιτυχίας στον Πελάτη. Σε περίπτωση που κάποιος από τους ελέγχους αποτύχει, ο Εξυπηρετητής απαντάει με μήνυμα λάθους. Τα παραπάνω βήματα φαίνονται και στις επόμενες εικόνες, όπου, ενδεικτικά, γίνεται ένα αίτημα αποθήκευσης νέας μέτρησης σωματικού βάρους.



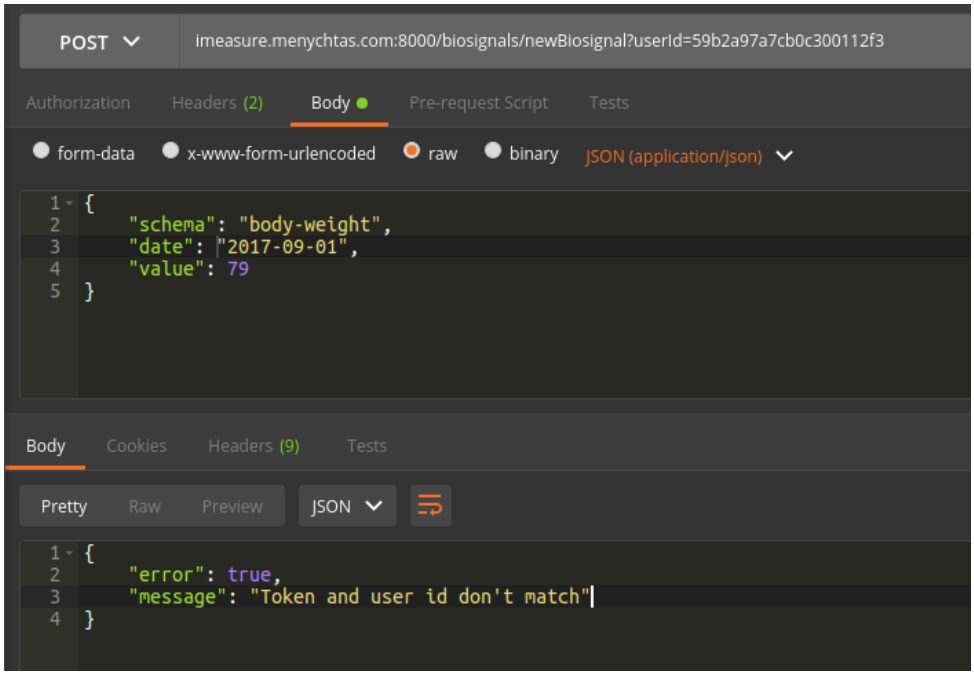
Σχήμα 31: Επικεφαλίδες Αιτήματος και URL parameter



Σχήμα 32: Σώμα Αιτήματος



Σχήμα 33: Μήνυμα Επιτυχίας



Σχήμα 34: Μήνυμα Αποτυχίας

Μετά την ολοκλήρωση του αιτήματος, το νέο βιοσήμα έχει επιτυχώς αποθηκευτεί στην βάση Mongo, κάτι που εύκολα μπορεί να ελεγχθεί με ένα ερώτημα (query):

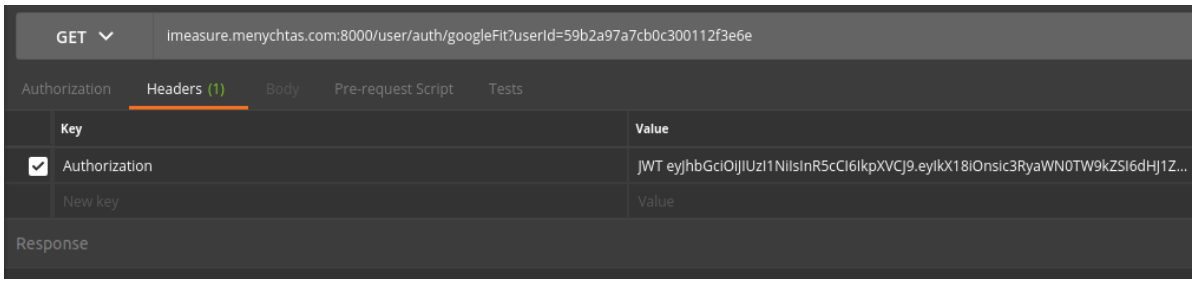
Σχήμα 35: Ερώτημα Αναζήτησης Βιοσημάτων στη Βάση Δεδομένων

4.7 Σύνδεση Λογαριασμού με Fitbit και GoogleFit

Η σύνδεση του λογαριασμού του χρήστη με το προφίλ του στο GoogleFit ή το Fitbit επιτυγχάνεται ως εξής:

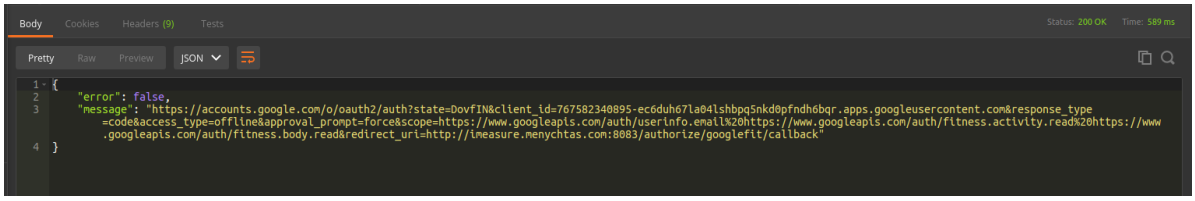
- Αρχικά, μέσω του UI, ο χρήστης στέλνει ένα αίτημα GET στον Εξυπηρετητή, στο endpoint που βρίσκεται το αντίστοιχο service, περνώντας ως παράμετρο το ID του (*/user/auth/googlefit?userId= + <user id>* για ενεργοποίηση του GoogleFit και /user/auth/googlefit?userId= + <user id> για ενεργοποίηση του Fitbit). Στις επικεφαλίδες του αιτήματος περιλαμβάνεται και το token του χρήστη
- Στην συνέχεια, αφού πιστοποιηθεί η ταυτότητα του χρήστη, ο Εξυπηρετητής προωθεί το αίτημα στην εφαρμογή Shimmer. Η προώθηση γίνεται με ένα νέο αίτημα GET, πάλι με παράμετρο το *userId*, αυτή τη φορά όμως στην διεύθυνση *imeasure.menychtas.com:8083//authorize/googlefit?username= + <userId>*, όπου και “ακούει” το Shimmer για το GoogleFit, ή στην διεύθυνση *imeasure.menychtas.com:8083/authorize/fitbit?username= + <userId>, για το Fitbit*
- Η εφαρμογή Shimmer απαντάει στον Server, στέλνοντας έναν σύνδεσμο επιβεβαίωσης (*authorizationUrl*), που πρέπει να ακολουθήσει ο χρήστης. Τον σύνδεσμο αυτό, εν συνεχεία, στέλνει ο Server στον Πελάτη, ως απάντηση στο αρχικό αίτημα
- Μόλις ο Πελάτης λάβει την απάντηση με το authorizationUrl, το ακολουθεί και εγκρίνει την σύνδεση του προφίλ του. Την διεκπεραίωση του συγκεκριμένου βήματος αναλαμβάνει το Shimmer, το οποίο, με την επιτυχή σύνδεση του προφίλ του χρήστη, τον ανακατευθύνει σε μια καινούργια σελίδα
- Η αποκλειστική λειτουργία της καινούργιας αυτής σελίδας είναι να στείλει ένα τελευταίο αίτημα GET στον Server, το οποίο επιβεβαιώνει την επιτυχία της σύνδεσης με το GoogleFit

Στην ακόλουθη εικόνα φαίνεται το αίτημα που στέλνει ο χρήστης για να συνδέσει τον λογαριασμό του με το προφίλ του στο GoogleFit. Αντίστοιχα γίνεται και το αίτημα για το Fitbit.



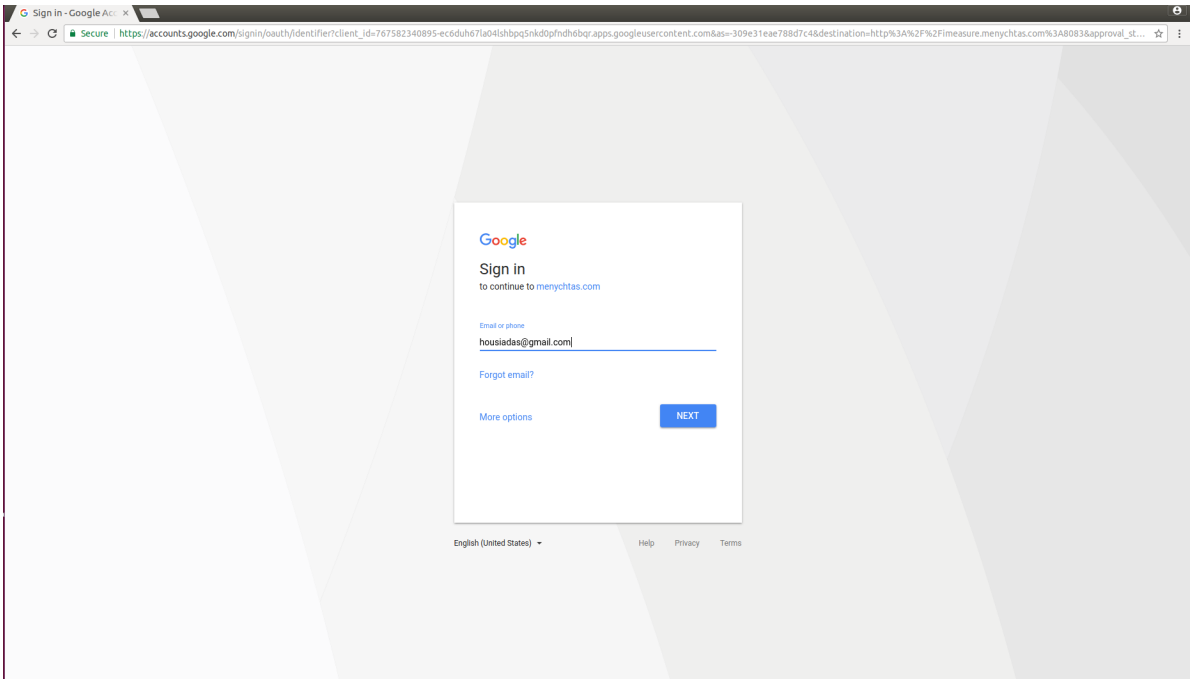
Σχήμα 36: Αίτημα Σύνδεσης με GoogleFit

Η απάντηση που λαμβάνει ο χρήστης από τον Εξυπηρετητή, αφού αυτός προωθήσει το αίτημα στο Shimmer, περιλαμβάνει και τον σύνδεσμο που πρέπει να ακολουθήσει στη συνέχεια, όπως φαίνεται παρακάτω:



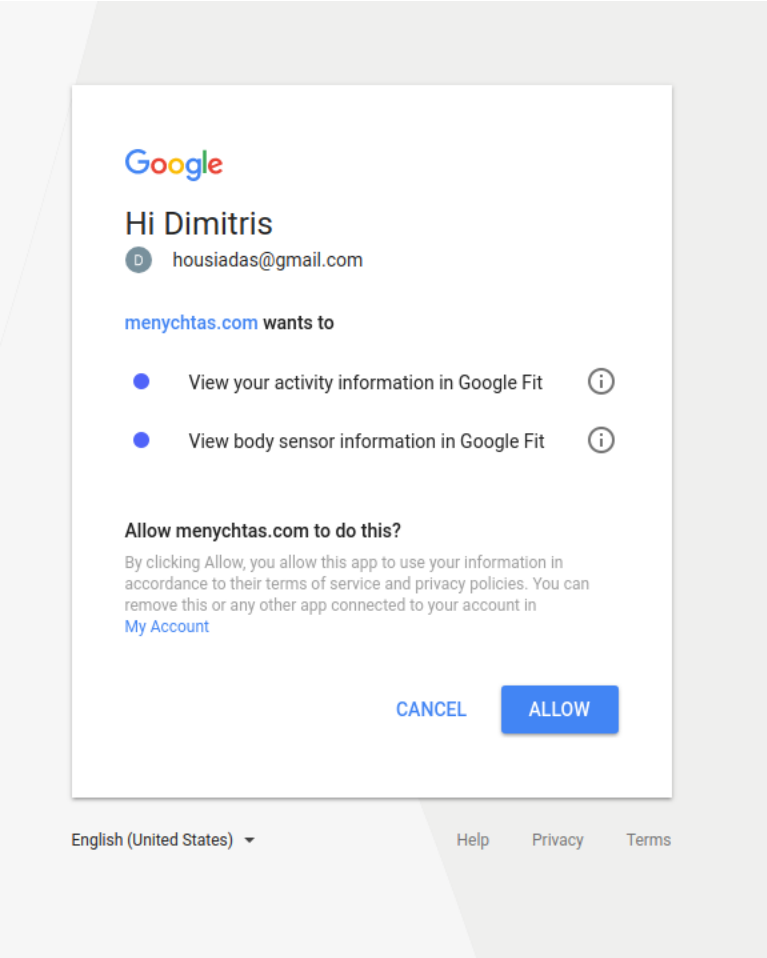
Σχήμα 37: Απάντηση Εξυπηρετητή με τον σύνδεσμο ανακατεύθυνσης

Ο χρήστης ακολουθεί τον σύνδεσμο:



Σχήμα 38: Ανακατεύθυνση χρήστη για σύνδεση με GoogleFit

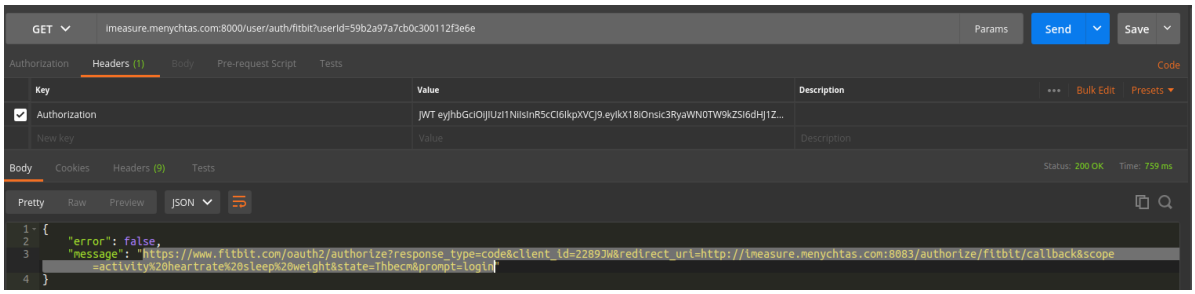
Αφού ο χρήστης εισάγει τα στοιχεία σύνδεσης με το GoogleFit, η εφαρμογή θα του ζητήσει να επιτρέψει την πρόσβαση στο προφίλ του.



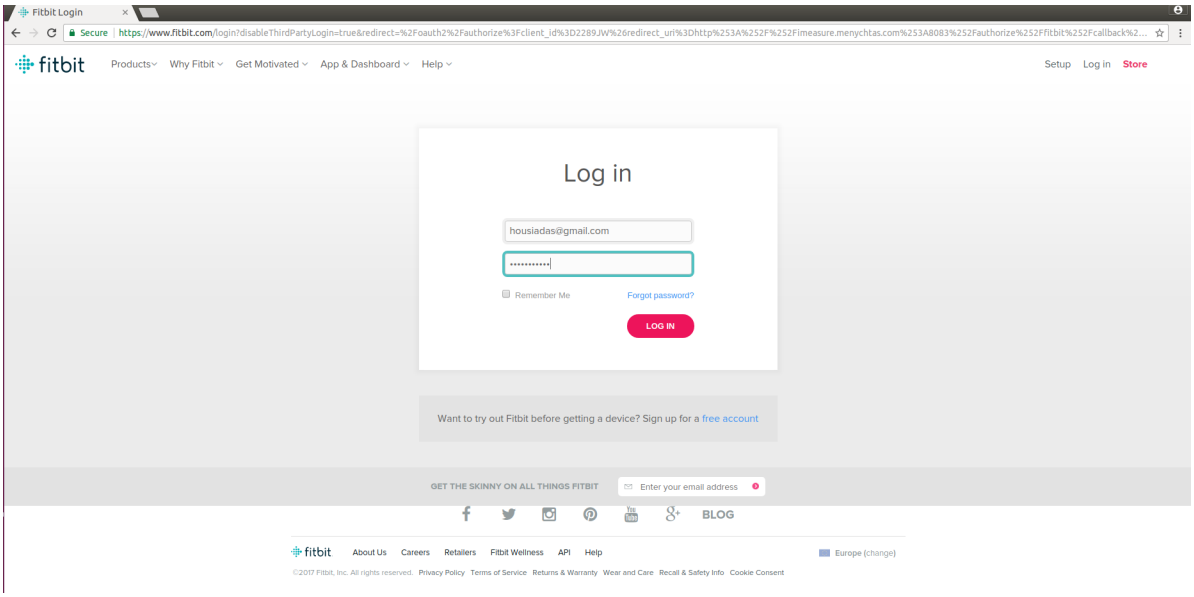
Σχήμα 39: Σύνδεση με GoogleFit

Τέλος, αφού επιτραπεί η σύνδεση από τον χρήστη, θα γίνει ανακατεύθυνση πίσω στην Quantified Self εφαρμογή, η οποία πλέον θα έχει πρόσβαση στα δεδομένα που συλλέγει το GoogleFit για τον χρήστη.

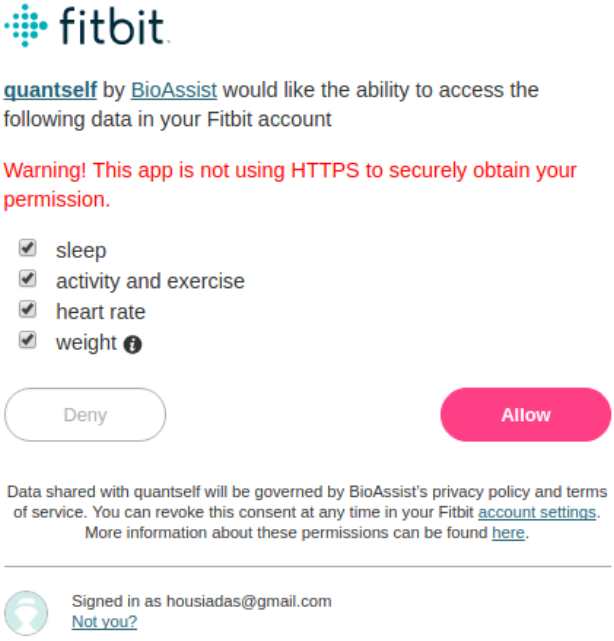
Ακολουθούν οι εικόνες με τα αντίστοιχα για το Fitbit:



Σχήμα 40: Αίτημα Σύνδεσης με Fitbit & Απάντηση Server



Σχήμα 41: Ανακατεύθυνση στη σελίδα του Fitbit



Σχήμα 42: Σύνδεση με Fitbit

4.8 Συγχρονισμός με Fitbit και GoogleFit

Μετά από την σύνδεση με το Fitbit και το GoogleFit, ο χρήστης μπορεί να συγχρονίσει τον λογαριασμό, δηλαδή να μεταφέρει τα δεδομένα που οι υπηρεσίες αυτές καταγράφουν στο σύστημα της Quantified Self εφαρμογής.

Ο συγχρονισμός αυτός γίνεται, είτε περιοδικά και αυτόματα, κάτι το οποίο θα εξετασθεί παρακάτω, είτε χειροκίνητα, μέσω κατάλληλης επιλογής στην διεπαφή. Η λειτουργία αυτή αποτελεί το αντικείμενο αυτής της παραγράφου.

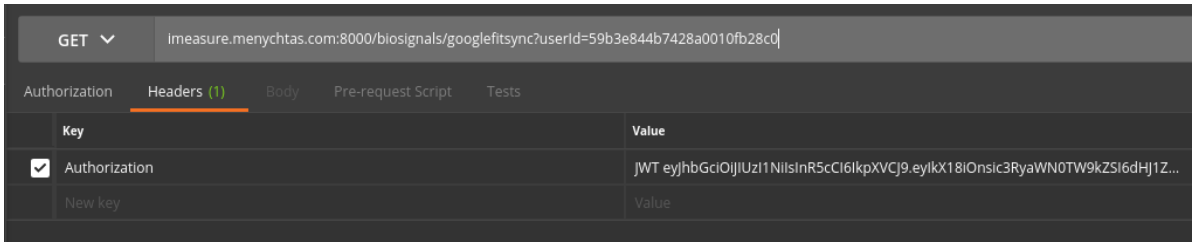
Καταρχάς, για να επιτευχθεί ο συγχρονισμός, είναι απαραίτητο το σύστημα να γνωρίζει πότε συγχρονίστηκε για τελευταία φορά με το Fitbit / GoogleFit, ώστε να αποφευχθεί η επανάληψη δεδομένων (διπλότυπα) και η παράλειψη δεδομένων. Για αυτό τον λόγο, σε κάθε εγγραφή τύπου User, με ενεργοποιημένο το GoogleFit (αντίστοιχα για το Fitbit), υπάρχει ένα πεδίο που δηλώνει πότε έγινε συγχρονισμός για τελευταία φορά. Συγκεκριμένα, συνεχίζοντας το παράδειγμα από την προηγούμενη παράγραφο, μόλις ο χρήστης ενεργοποιήσει το Fitbit και το GoogleFit, η αναπαράσταση του στην βάση δεδομένων είναι η εξής:


```
"_id" : ObjectId("59b3e844b7428a0010fb28c0"),
"email" : "housiadas@gmail.com",
"password" : "$2a$05$A1MbSkIiBzCNY55.2RVSK.jT1MX5CNOhN2vULlxExSpPJrU7tvXa2",
"firstName" : "Dimitris",
"lastName" : "Chousiadas",
"accountConfirmed" : "1",
"googleFitActivated" : true,
"fitbitActivated" : true,
"stepsPerDayGoal" : 10000,
"caloriesPerDayGoal" : 2500,
"activityPerDayGoal" : 120,
"dataSharing" : false,
"stats" : {
        "avgSteps" : 0,
        "maxSteps" : 0,
        "avgCalories" : 0,
        "maxCalories" : 0,
        "avgActivity" : 0,
        "maxActivity" : 0
},
"__v" : 0,
"today" : {
        "activity" : 0,
        "calories" : 0,
        "steps" : 0
},
"googleFitTimestamp" : {
        "calories_burned" : "2016-09-09",
        "heart_rate" : "2016-09-09",
        "body_weight" : "2016-09-09",
        "step_count" : "2016-09-09",
        "activity" : "2016-09-09"
},
"fitbitTimestamp" : {
        "activity" : "2017-09-09",
        "sleep" : "2017-09-09",
        "steps" : "2017-09-09",
        "weight" : "2017-09-09"
}
```


Σχήμα 43: Αναπαράσταση Χρήστη μετά από σύνδεση με Fitbit και GoogleFit

Στην παραπάνω εικόνα φαίνεται πως τα πεδία *fitbitActivated* και *googleFitActivated* έχουν πάρει την τιμή *True*. Ταυτόχρονα, έχουν δημιουργηθεί δύο καινούργια πεδία (*fitbitTimestamp* και *googleFitTimestamp*), τα οποία δηλώνουν πότε έγινε τελευταία φορά συγχρονισμός καθεμίας κατηγορίας δεδομένων (βάρος, βήματα, δραστηριότητα, κτλ.). Η επιλογή αρχικών ημερομηνιών για τα πεδία αυτά βασίζεται με το πόσο συχνά και για τι όγκο δεδομένων επιτρέπουν την πρόσβαση στις υπηρεσίες Cloud τους οι Fitbit και GoogleFit.

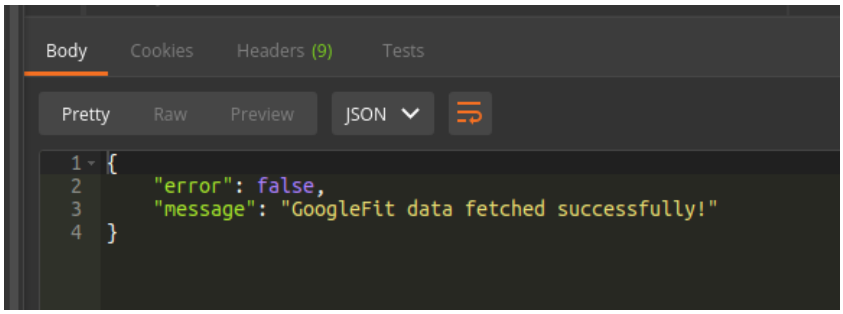
Όταν ο χρήστης επιλέξει να συγχρονίσει το σύστημα, στέλνεται ένα μήνυμα GET στον Server, με παράμετρο το *userId*, μαζί με το token επαλήθευσης.



Σχήμα 44: Αίτημα Συγχρονισμού με GoogleFit

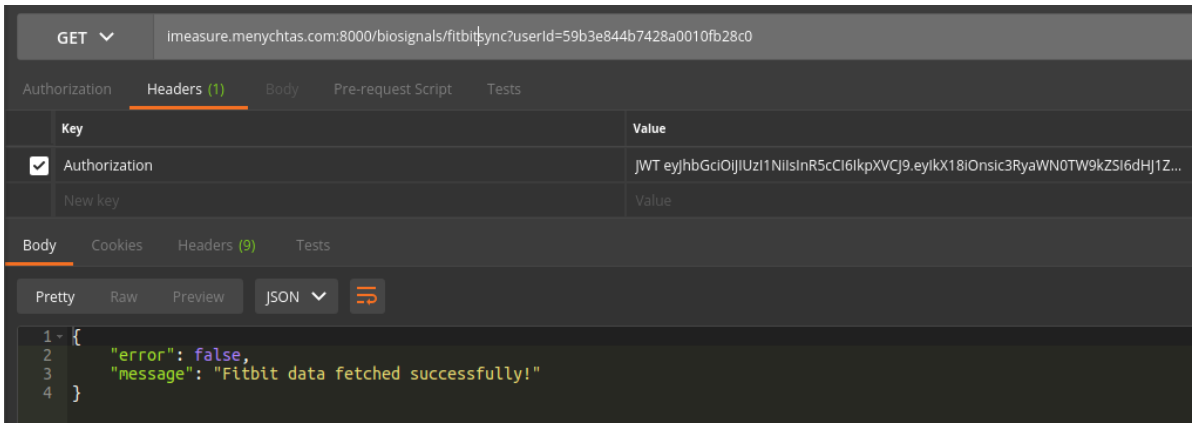
Η επεξεργασία ενός τέτοιου αιτήματος από τον Server γίνεται ως εξής:

- ο Εξυπηρετητής ασχολείται διαδοχικά με κάθε μια κατηγορία δεδομένων
- για κάθε κατηγορία, αρχικά, διαγράφονται από την βάση εγγραφές με ημερομηνία μεγαλύτερη ή ίση από την αντίστοιχη χρονοσφραγίδα (timestamp), η οποία δηλώνει τον τελευταίο συγχρονισμός. Αυτό γίνεται, ώστε να αποφευχθούν οι διπλότυπες εγγραφές
- εν συνεχεία, ο Εξυπηρετητής στέλνει ένα αίτημα GET στην εφαρμογή Shimmer, ζητώντας να τα πιο πρόσφατα δεδομένα
- το Shimmer αναλαμβάνει την επικοινωνία με το GoogleFit (αντίστοιχα και για το Fitbit) και μεταβιβάζει τα δεδομένα στο σύστημα
- μόλις ολοκληρωθεί η μεταφορά των δεδομένων ο Εξυπηρετητής τα αποθηκεύει στην βάση δεδομένων, και παράλληλα ενημερώνει την χρονοσφραγίδα της εκάστοτε κατηγορίας, ώστε ο επόμενος συγχρονισμός να συνεχίσει από εκεί
- μετά και από την ολοκλήρωση αυτού του βήματος, η επεξεργασία προχωράει στην επόμενη κατηγορία δεδομένων ή, αν έχει ολοκληρωθεί, στέλνεται μήνυμα επιτυχίας στον χρήστη



Σχήμα 45: Μήνυμα Επιτυχίας Συγχρονισμού από Εξυπηρετητή

Ομοίως γίνεται και ο συγχρονισμός με το Fitbit, όπως φαίνεται και στην επόμενη εικόνα:



Σχήμα 46: Συγχρονισμός με Fitbit

Μετά, λοιπόν, από τον επιτυχή συγχρονισμό δεδομένων, οι χρονοσφραγίδες έχουν ανανεωθεί, κάτι το οποίο φαίνεται παρακάτω, στην περίπτωση του GoogleFit


```
"_id" : ObjectId("59b3e844b7428a0010fb28c0"),
"email" : "housiadas@gmail.com",
"password" : "$2a$05$A1MbSkIiBzCNY55.2RVSK.jT1MX5CNOhN2vULlxExSpPJrU7tvXa2",
"firstName" : "Dimitris",
"lastName" : "Chousiadas",
"accountConfirmed" : "1",
"googleFitActivated" : true,
"fitbitActivated" : true,
"stepsPerDayGoal" : 10000,
"caloriesPerDayGoal" : 2500,
"activityPerDayGoal" : 120,
"dataSharing" : false,
"stats" : {
        "avgSteps" : 0,
        "maxSteps" : 0,
        "avgCalories" : 0,
        "maxCalories" : 0,
        "avgActivity" : 0,
        "maxActivity" : 0
},
"__v" : 0,
"today" : {
        "activity" : 0,
        "calories" : 0,
        "steps" : 0
},
"googleFitTimestamp" : {
        "calories_burned" : "2017-09-09",
        "heart_rate" : "2016-09-09",
        "body_weight" : "2016-09-09",
        "step_count" : "2017-09-09",
        "activity" : "2017-09-09"
},
"fitbitTimestamp" : {
        "activity" : "2017-09-09",
        "sleep" : "2017-09-09",
        "steps" : "2017-09-09",
        "weight" : "2017-09-09"
}
```


Σχήμα 47: Ανανεωμένες Χρονοσφραγίδες μετά από Συγχρονισμό

4.9 Ανάκτηση Βιοσημάτων

Τα δεδομένα που ο ίδιος ο χρήστης εισάγει στο σύστημα, καθώς και τα δεδομένα που προέρχονται από το Fitbit και το GoogleFit αποθηκεύονται στην βάση δεδομένων, ώστε κάθε φορά που ο χρήστης εισέρχεται στο σύστημα να μπορεί να τα ανακτήσει.

Η ανάκτηση των δεδομένων, λοιπόν, είναι ένα αίτημα τύπου GET στον εξυπηρετητή, το οποίο αποτελείται από:

- το token επαλήθευσης του χρήστη, το οποίο περιλαμβάνεται στις επικεφαλίδες του αιτήματος
- το *userId*, που δίνεται ως παράμετρος στο αίτημα
- μια επιπλέον παράμετρο, που δηλώνει το χρονικό διάστημα αναφοράς. Συγκεκριμένα, δίνονται τρεις επιλογές στον χρήστη: να ανακτήσει τα δεδομένα της τελευταίας εβδομάδας, του τελευταίου μήνα και του τελευταίου έτους.

Αξίζει, στο σημείο αυτό, να γίνει μια σύντομη περιγραφεί του κώδικα που υλοποιεί την διεκπεραίωση ενός τέτοιου αιτήματος. Ενδεικτικά, στην ακόλουθη εικόνα φαίνεται ο κώδικας για την περίπτωση του καρδιακού ρυθμού.

```
exports.heartRate = function (req, res) {

    let span = req.query.span || 'year';
    let start = new Date();

    if (span === 'week') {
        start.setDate(start.getDate() - 6);
    } else if (span === 'month') {
        start.setDate(start.getDate() - 30);
    } else {
        start.setDate(start.getDate() - 360);
    }

    start = start.toISOString();

    Biosignal.find({
        "header.schema_id.name": "heart-rate",
        "header.user_id": req.query.userId,
        "body.effective_time_frame.date_time": {
            $gte: start
        }
    }, null, {
        sort: {
            "body.effective_time_frame.date_time": 1
        }
    }, function (err, biosignals) {
        if (err) {
            console.log(err);
            res.sendStatus(500);
        } else {
            let day;
            let value;
            let x = [];
            let y = [];
            for (let bio of biosignals) {
                day = (bio.body.effective_time_frame.date_time).split("T")[0];
                value = bio.body.heart_rate.value;
                x.push(day);
                y.push(value);
            }
            res.status(200).json({x: x, y: y});
        }
    })

};
```

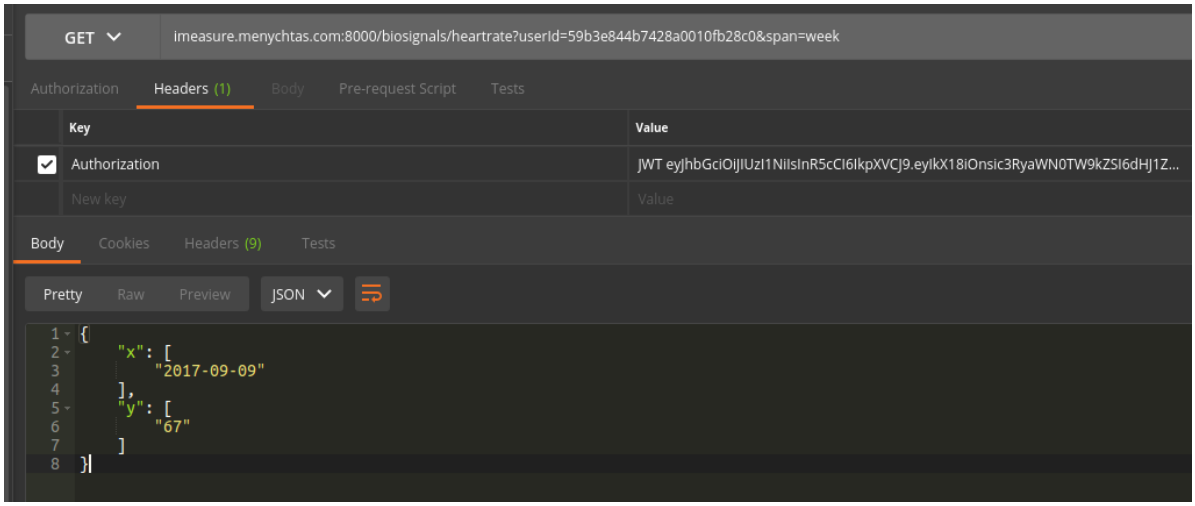
Σχήμα 48: Κώδικας για την διεκπεραίωση του αιτήματος ανάκτησης δεδομένων Καρδιακού Ρυθμού

Φαίνεται πως η απάντηση που θα σταλεί, τελικά, στον χρήστη είναι ένα αντικείμενο JSON με δύο πεδία, το x και το y. Το πεδίο x είναι ένας πίνακας με τις ημερομηνίες όπου έγιναν οι μετρήσεις και το y ένας πίνακας με τις τιμές των μετρήσεων αυτών.

Στον κώδικα έχει γίνει και χρήση των δυνατοτήτων που παρέχει το MongooseJS για την υλοποίηση ερωτημάτων (queries) σε μια βάση Mongo. Συγκεκριμένα, φαίνεται πως είναι δυνατόν να οριστούν τα φίλτρα με τα οποία θα γίνει η αναζήτηση (εν προκειμένω, το *userId*, το όνομα του σχήματος, καθώς και ένα κάτω χρονικό όριο), ενώ, στην συνέχεια, μπορεί να γίνει και κάποια προεπεξεργασία των αποτελεσμάτων (εν προκειμένω, ταξινόμηση σε αύξουσα χρονική σειρά, με την πιο πρόσφατη μέτρηση τελευταία).

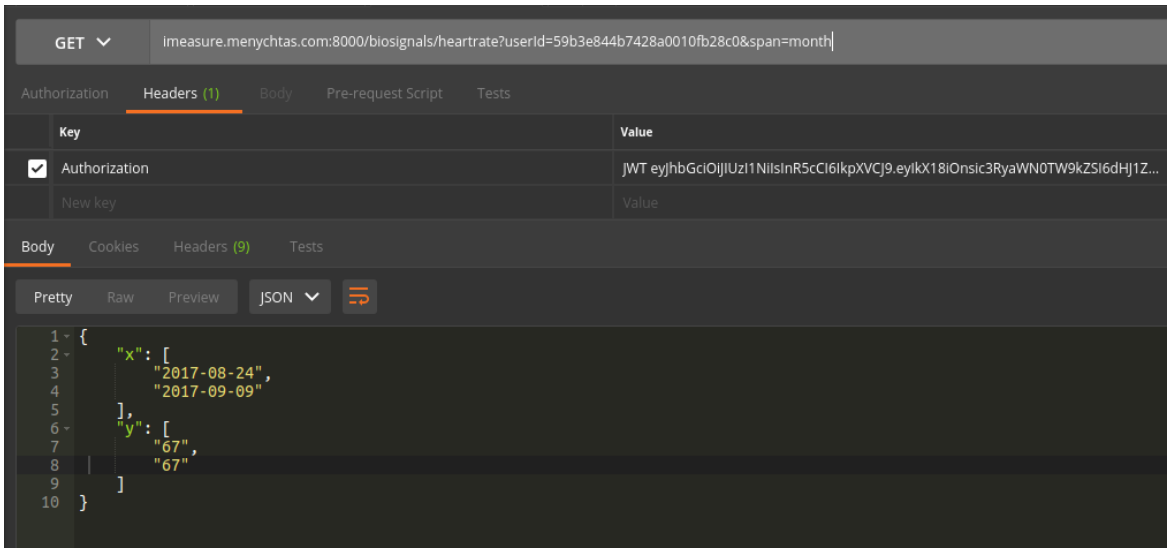
Τέλος, πριν τη υλοποίηση του ερωτήματος, φαίνεται και πως οι τρεις επιλογές που δίνονται στον χρήστη (εβδομάδα, μήνας, χρόνος) “μεταφράζονται” στον κώδικα.

Ένα, λοιπόν, αίτημα ανάκτησης βιοσημάτων στην πράξη είναι αυτό που φαίνεται στην επόμενη εικόνα:



Σχήμα 49: Αίτημα Ανάκτησης Βιοσημάτων Καρδιακού Ρυθμού τελευταίας εβδομάδας

Εδώ, ο χρήστης ζητάει από τον Server τα δεδομένα που αφορούν τον καρδιακό του ρυθμό της τελευταίας εβδομάδας. Στην επόμενη εικόνα, αντίθετα, το διάστημα αναφοράς είναι ο τελευταίος μήνας.



Σχήμα 50: Αίτημα Ανάκτησης Βιοσημάτων Καρδιακού Ρυθμού τελευταίου μήνα

Σε αντίθεση με τα βιοσήματα, όπως ο καρδιακός ρυθμός ή το βάρος, τα βιοσήματα που αφορούν στη σωματική δραστηριότητα του χρήστη χειρίζονται με κάπως διαφορετικό τρόπο. Συγκεκριμένα, σε ένα αίτημα ανάκτησης, για παράδειγμα, του αριθμού των βημάτων του χρήστη, ο Εξυπηρετητής δεν απαντά, στέλνοντας κάθε μια εγγραφή βημάτων. Αντίθετα, οργανώνει τις εγγραφές αυτές ανά ημερομηνία, ώστε ο χρήστης να μάθει πόσα βήματα έκανε (αντίστοιχα, πόσες θερμίδες έκαψε ή για πόση ώρα ασκήθηκε) σε κάθε μια συγκεκριμένη μέρα. Επιπλέον, στην περίπτωση που το διάστημα αναφοράς είναι ο μήνας ή ο χρόνος, ο Εξυπηρετητής οργανώνει τα δεδομένα και ανά ημέρα της εβδομάδας, ώστε ο χρήστης να μάθε πόσα βήματα κάνει, κατά μέσο όρο, τις Δευτέρες, τις Τρίτες, κτλ.

Το πρώτο βήμα της επεξεργασίας αυτής γίνεται όπως φαίνεται στο ακόλουθο απόσπασμα κώδικα, όπου γίνεται χρήση του MongooseJS. Εδώ, γίνεται αναζήτηση των δεδομένων που αφορούν στα βήματα του χρήστη, τα οποία και αθροίζονται ανά ημερομηνία.

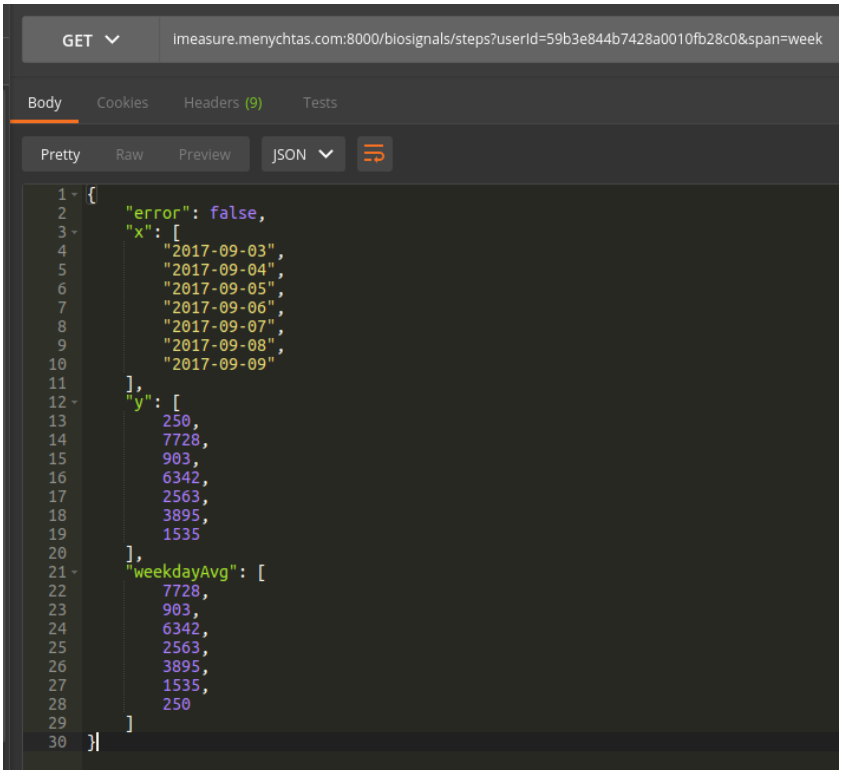
```
Biosignal.aggregate([
    {
        $match: {
            "header.schema_id.name": "step-count",
            "header.user_id": req.query.userId,
            "body.effective_time_frame.time_interval.start_date_time": {
                $gte: start
            }
        }
    },
    {
        $project: {
            "body.step_count": 1,
            date: {
                $substr: [ "$body.effective_time_frame.time_interval.start_date_time", 0, 10]
            }
        }
    },
    {
        $group: {
            _id: "$date",
            total: { $sum: "$body.step_count" }
        }
    },
    {
        $sort: {
            _id: 1
        }
    }
```

Σχήμα 51: Πρώτο Βήμα Επεξεργασίας αιτήματος ανάκτησης δεδομένων βημάτων του χρήστη

Στη συνέχεια, με βάση τα αποτελέσματα του πρώτου αυτού βήματος επεξεργασίας, γίνεται ο υπολογισμός του μέσου όρου βημάτων ανά ημέρα της εβδομάδας. Αυτό υλοποιείται από το απόσπασμα κώδικα που ακολουθεί.

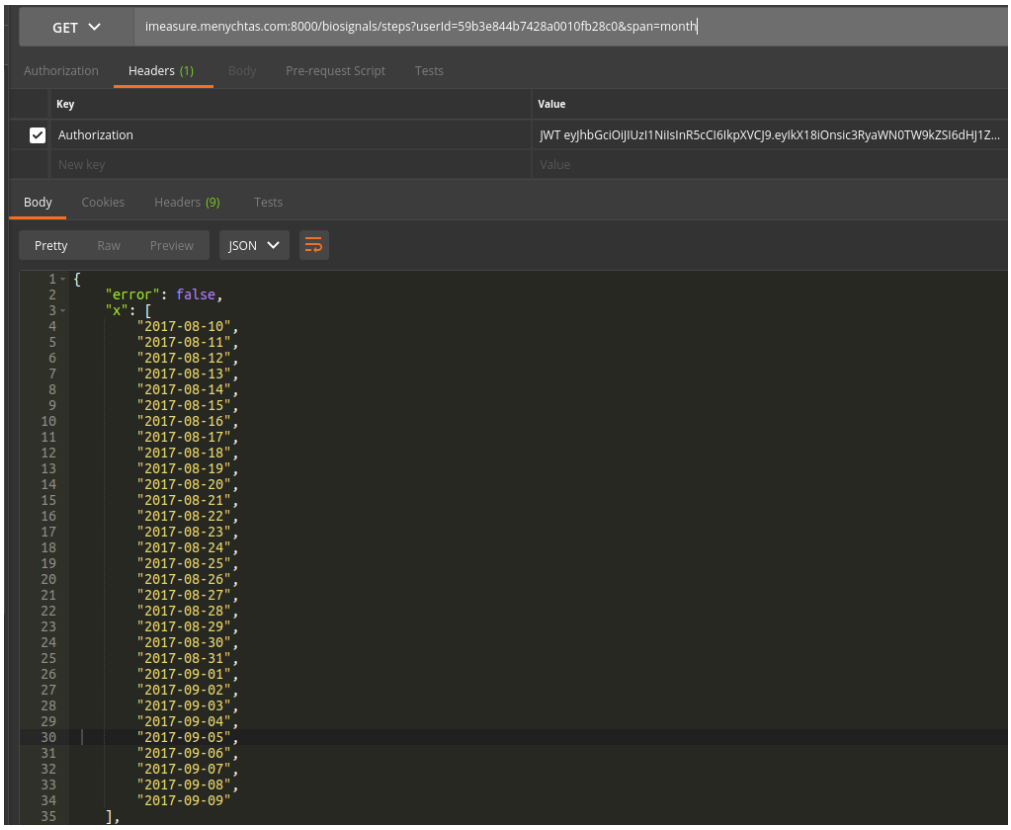
```
], function (err, results) {
    // console.log("Here");
    if (err) {
        console.log(err);
        res.sendStatus(500);
    } else {

        let key;
        let value;
        let x = [];
        let y = [];
        let weekday;
        let weekdayAvg = [0, 0, 0, 0, 0, 0, 0];
        let daysCounter = [0, 0, 0, 0, 0, 0, 0];

        for (let res of results) {
            key = res._id;
            value = res.total;
            x.push(key);
            y.push(value);

            weekday = new Date(key).getDay();
            if (weekday !== 0) {
                weekdayAvg[weekday - 1] += value;
                daysCounter[weekday - 1] += 1;
            } else {
                weekdayAvg[6] += value;
                daysCounter[6] += 1;
            }
        }

        for (let i = 0; i < 7; i++)
            if (daysCounter[i])
                weekdayAvg[i] = Math.round(weekdayAvg[i] / daysCounter[i]);
        console.log("Success");
        res.status(200).json({error: false, x: x, y: y, weekdayAvg: weekdayAvg});
    }
})
```

Σχήμα 52: Δεύτερο Βήμα Επεξεργασίας αιτήματος ανάκτησης δεδομένων βημάτων του χρήστη

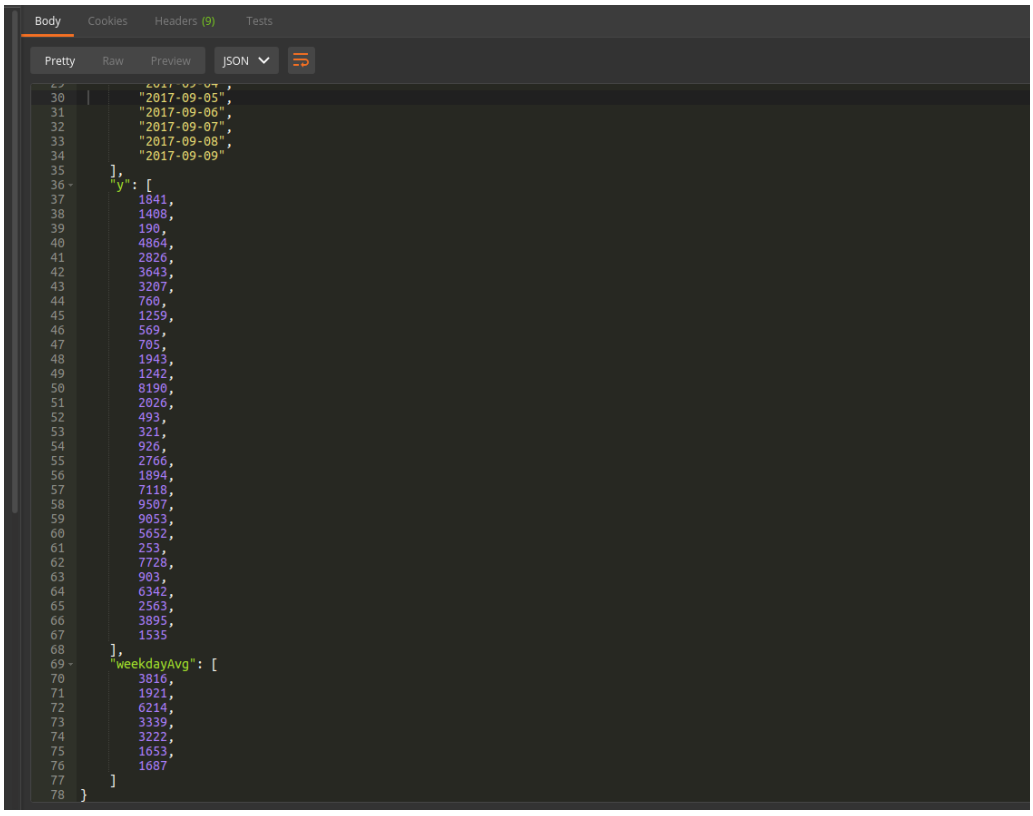


Σχήμα 53: Ανάκτηση Δεδομένων Βημάτων της τελευταίας εβδομάδας

Στην προηγούμενη εικόνα φαίνεται ένα αίτημα χρήστη για ανάκτηση των δεδομένων βημάτων της προηγούμενης εβδομάδας, ενώ στις δύο επόμενες, του προηγούμενου μήνα.



Σχήμα 54: Βήματα Τελευταίου Μήνα (A)



Σχήμα 55: Βήματα Τελευταίου Μήνα (B)

4.10 Περιοδική Επεξεργασία των Δεδομένων Χρήστη

Όπως έχει αναφερθεί κατά την περιγραφή των προδιαγραφών του συστήματος, είναι σημαντικό για τον χρήστη να έχει πρόσβαση, εκτός από την οπτικοποίηση, και σε κάποια στατιστική ανάλυση των δεδομένων του. Έτσι, η Quantified Self εφαρμογή, περιοδικά, για κάθε έναν από τους χρήστες του συστήματος, υπολογίζει δύο στατιστικά μεγέθη που αφορούν στη σωματική τους δραστηριότητα, δηλαδή τις θερμίδες που “καίνε”, τον αριθμό των βημάτων που κάνουν και τη διάρκεια άσκησης τους (ανά μέρα). Τα δύο αυτά μεγέθη είναι η μέση τιμή και η μέγιστη τιμή.

Συγκεκριμένα, έγινε χρήση της βιβλιοθήκης node-schedule [40], η οποία παρέχει την δυνατότητα χρονοδρομολόγησης εργασιών σε περιβάλλον Node. Όπως φαίνεται και στο ακόλουθο απόσπασμα κώδικα, λοιπόν, κάθε μια ώρα (ειδικότερα, στο 40ο λεπτό κάθε ώρας) γίνεται ενημέρωση των στατιστικών των χρηστών.

```
let y = schedule.scheduleJob({ minute: 40 }, function() {
    sync.periodicStatsUpdate();
});
```

Σχήμα 56: Περιοδική Ενημέρωση Στατιστικών

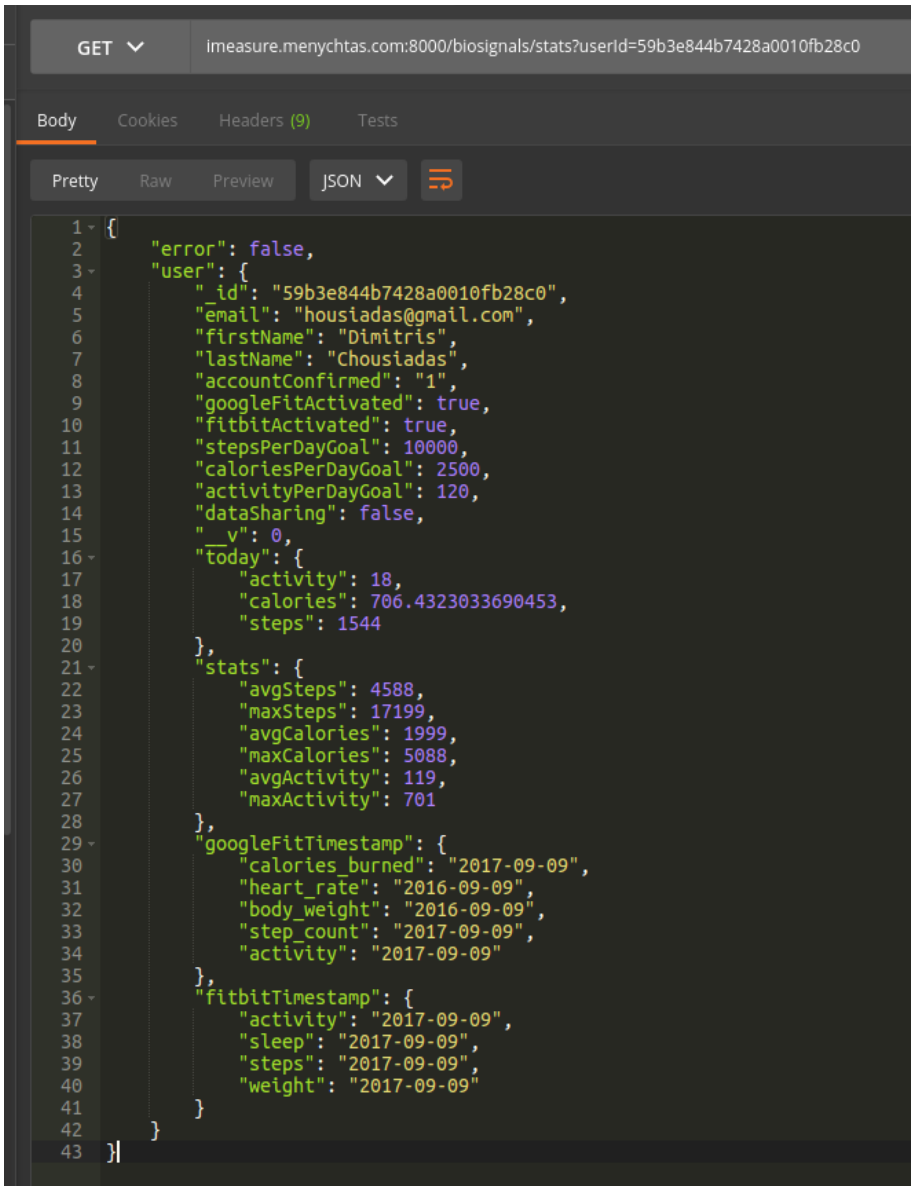
Κατά την ενημέρωση αυτή, διαδοχικά για κάθε ένα χρήστη της εφαρμογής, εκτελούνται ορισμένα ερωτήματα στην βάση δεδομένων. Το πρώτο από αυτά αφορά στον υπολογισμό της μέσης τιμής και του μεγίστου των αριθμών βημάτων που κάνει ο χρήστης μέσα στην μέρα.

```
Biosignal.aggregate([
    {
        $match: {
            "header.schema_id.name": "step-count",
            "header.user_id": user._id.toString()
        }
    },
    {
        $project: {
            "body.step_count": 1,
            date: {
                $substr: [ "$body.effective_time_frame.time_interval.start_date_time", 0, 10]
            }
        }
    },
    {
        $group: {
            _id: "$date",
            total: { $sum: "$body.step_count" }
        }
    },
    {
        $group: {
            _id: null,
            avg_steps: { $avg: "$total" },
            max_steps: { $max: "$total" }
        }
    }
```

Σχήμα 57: Υπολογισμός στατιστικών για τα βήματα του χρήστη

Όπως και στα προηγούμενα ερώτημα που εξετάσθηκαν έως τώρα, ο ζητούμενος υπολογισμός έγινε με χρήση των εργαλείων που παρέχει το MongooseJS. Τον υπολογισμό των στατιστικών για τα βήματα ακολουθεί ο υπολογισμός των στατιστικών για τις θερμίδες που “καίει” ημερησίως ο χρήστης, ενώ, τέλος, υπολογίζονται και τα στατιστικά της ημερήσιας διάρκειας άσκησης του χρήστη. Οι αντίστοιχοι κώδικες ακολουθούν παρόμοια λογική.

Εκτός από την περιοδική ενημέρωση των στατιστικών κάθε ώρα, οι υπολογισμοί αυτοί γίνονται και κάθε φορά που ο χρήστης εισέρχεται στην εφαρμογή. Στην περίπτωση αυτή, ο κώδικας του Πελάτη στέλνει ένα αίτημα GET στον Server, όπως φαίνεται παρακάτω.



Σχήμα 58: Αίτημα GET για ανανέωση των στατιστικών

Στο αίτημα αυτό, ο Εξυπηρετητής απαντάει, επιστρέφοντας στον Πελάτη ολόκληρο το αντικείμενο JSON που περιγράφει τον χρήστη, στο οποίο περιλαμβάνεται και ένα υπο-αντικείμενο με τα ζητούμενα στατιστικά.

4.11 Περιοδικός Συγχρονισμός και Ενημέρωση της Βάσης

Εκτός από τον περιοδικό υπολογισμό των στατιστικών, η εφαρμογή φροντίζει ώστε να γίνεται και περιοδικός συγχρονισμός με το Fitbit και το GoogleFit. Συγκεκριμένα, κάθε ώρα (στο 20ο λεπτό), το σύστημα εξετάζει διαδοχικά, έναν έναν τους χρήστες και για κάθε χρήστη με ενεργοποιημένο το Fitbit ή και το GoogleFit, όπως περιγράφηκε και στην παράγραφο 4.9, ενημερώνει τα δεδομένα του, χρησιμοποιώντας το Shimmer.

```
let j = schedule.scheduleJob({ minute: 20 }, function() {
    sync.periodicSync();
});
```

Σχήμα 59: Περιοδικός Συγχρονισμός με Fitbit και GoogleFit

```
exports.periodicSync = function () {

    // console.log("syncing");
    User.find({
        $or: [
            { fitbitActivated: true },
            { googleFitActivated: true }
        ]
    }, function (err, users) {

        if (!err && users) {

            for (let u of users) {
                if (u.googleFitActivated) {
                    googleFitSync(u);
                }
                if (u.fitbitActivated) {
                    fitbitSync(u);
                }
            }


        }

    })
};
```

Σχήμα 60: Συγχρονισμός για κάθε χρήστη διαδοχικά

Στην τελευταία εικόνα, οι συναρτήσεις *googleFitSync* και *fitbitSync* επιτελούν τις ίδιες λειτουργίες με αυτές της παραγράφου 4.9.

Τέλος, όπως και στην περίπτωση του υπολογισμού των στατιστικών, έτσι και εδώ, ο συγχρονισμός με το Fitbit και GoogleFit γίνεται, επιπλέον, και κάθε φορά που ο χρήστης εισέρχεται στο σύστημα (login). Μπορεί, εξάλλου, να γίνει και μέσω της διεπαφής (βλ. 4.9).

4.12 Στατιστική Ανάλυση επί του συνόλου των χρηστών & Data Sharing

Μια τρίτη λειτουργία που επιτελείται περιοδικά από την εφαρμογή είναι ο υπολογισμός των στατιστικών που αφορούν στο σύνολο των χρηστών. Συγκεκριμένα, κάθε ώρα το σύστημα υπολογίζει εκ νέου και ενημερώνει τα εξής τρία στατιστικά:

- την μέση τιμή των μέσων τιμών των ημερήσιων βημάτων των χρηστών
- την μέση τιμή των μέσων τιμών των ημερήσιων θερμίδων που καίνε οι χρήστες
- την μέση τιμή των μέσων τιμών της ημερήσιας διάρκειας άσκησης των χρηστών

Όπως και οι προηγούμενοι δύο περιοδικοί υπολογισμοί, έτσι και αυτός εκτελείται στο background, χωρίς την παρέμβαση κάποιου χρήστη. Στην συγκεκριμένη περίπτωση, μάλιστα, δεν υπάρχει άλλος τρόπος να γίνει ο υπολογισμός, σε αντίθεση με τον συγχρονισμό με Fitbit και GoogleFit ή τον υπολογισμό των στατιστικών.

```
let j = schedule.scheduleJob({ minute: 0 }, function() {
    sync.globalStatsSync();
});
```

Σχήμα 61: Περιοδική Ενημέρωση των Συνολικών Στατιστικών

Όπως φαίνεται στο παραπάνω τμήμα κώδικα, κάθε μια ώρα καλείται η συνάρτηση *globalStatsSync*, η οποία και υλοποιεί τον υπολογισμό ως εξής:

```
User.aggregate([
    {
        $match: {
            "dataSharing": true
        }
    },
    {
        $project: {
            "stats.avgSteps": 1,
            "stats.avgActivity": 1,
            "stats.avgCalories": 1
        }
    },
    {
        $group: {
            _id: null,
            steps: { $avg: "$stats.avgSteps" },
            calories: { $avg: "$stats.avgCalories" },
            activity: { $avg: "$stats.avgActivity" }
        }
    }
```

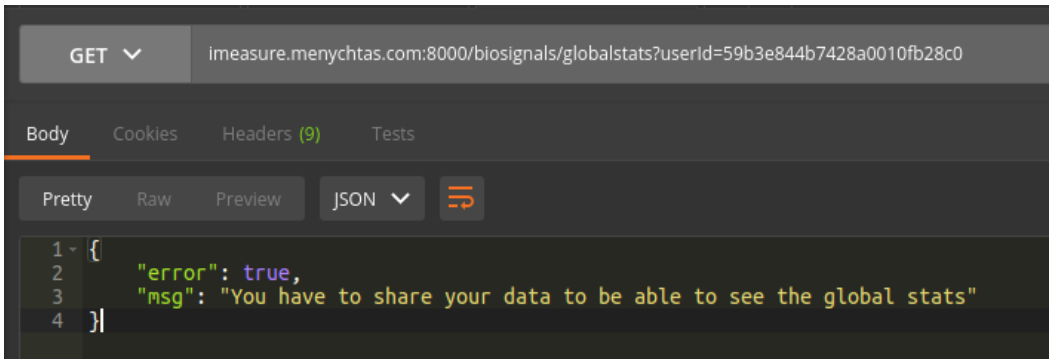
Σχήμα 62: Υπολογισμός Συνολικών Στατιστικών

Ένα σημαντικό σημείο στον παραπάνω κώδικα είναι το φίλτρο που εφαρμόζεται για την επιλογή των χρηστών, των οποίων τα δεδομένα θα χρησιμοποιηθούν για τον υπολογισμό. Συγκεκριμένα, ο χρήστης έχει την επιλογή να μοιραστεί, ανώνυμα, τα δεδομένα του με το σύνολο (Data Sharing). Αυτό σημαίνει, αφενός, πως ο υπολογισμός των συνολικών στατιστικών θα χρησιμοποιήσει τα δεδομένα του χρήστη αυτού, και αφετέρου, πως ο χρήστης θα έχει δικαίωμα να δει τα συνολικά αυτά αποτελέσματα. Μόλις ένας καινούργιος χρήστης δημιουργήσει λογαριασμό στην εφαρμογή, από προεπιλογή, δεν μοιράζεται τα δεδομένα του, κάτι το οποίο υποδεικνύεται και από το πεδίο *dataSharing* στην εγγραφή του χρήστη στην βάση δεδομένων.


```
"_id" : ObjectId("59b3e844b7428a0010fb28c0"),
"email" : "housiadas@gmail.com",
"password" : "$2a$05$A1MbSkIiBzCNY55.2RVSK.jT1MX5CNOhN2vULlxExSpPJrU7tvXa2",
"firstName" : "Dimitris",
"lastName" : "Chousiadas",
"accountConfirmed" : "1",
"googleFitActivated" : true,
"fitbitActivated" : true,
"stepsPerDayGoal" : 10000,
"caloriesPerDayGoal" : 2500,
"activityPerDayGoal" : 120,
"dataSharing" : false,
"stats" : {
        "avgSteps" : 4590,
        "maxSteps" : 17199,
        "avgCalories" : 2001,
        "maxCalories" : 5088,
        "avgActivity" : 120,
        "maxActivity" : 701
},
"__v" : 0,
"today" : {
        "activity" : 18,
        "calories" : 706.4323033690453,
        "steps" : 1544
},
"googleFitTimestamp" : {
        "calories_burned" : "2017-09-09",
        "heart_rate" : "2016-09-09",
        "body_weight" : "2016-09-09",
        "step_count" : "2017-09-09",
        "activity" : "2017-09-09"
},
"fitbitTimestamp" : {
        "activity" : "2017-09-09",
        "sleep" : "2017-09-09",
        "steps" : "2017-09-09",
        "weight" : "2017-09-09"
}
```

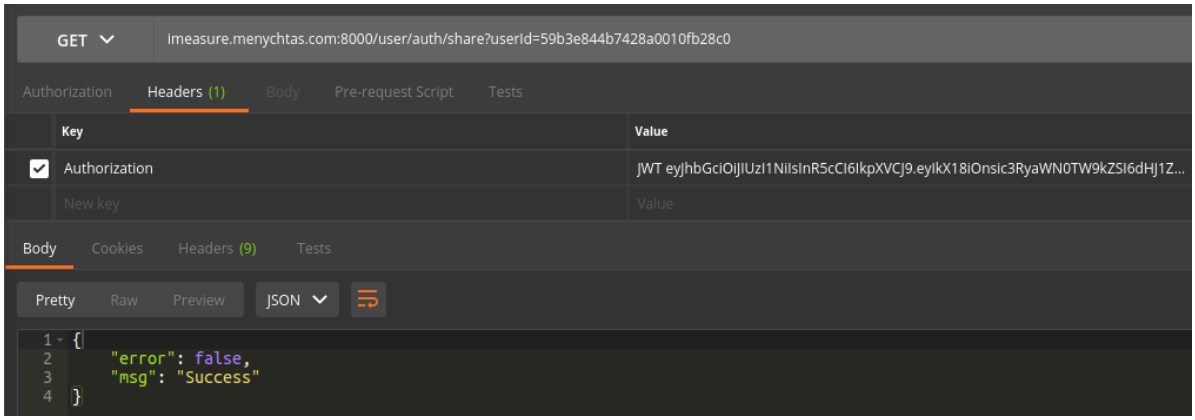

Σχήμα 63: Χρήστης που δεν μοιράζεται τα δεδομένα του

Η εφαρμογή, λοιπόν, δεν χρησιμοποιεί τα δεδομένα ενός χρήστη που δεν έχει επιλέξει να τα μοιράζεται. Και ο χρήστης αυτός, με τη σειρά του, δεν έχει δικαίωμα πρόσβασης στα συνολικά στατιστικά της εφαρμογής, όπως φαίνεται και από το ακόλουθο αίτημα GET.



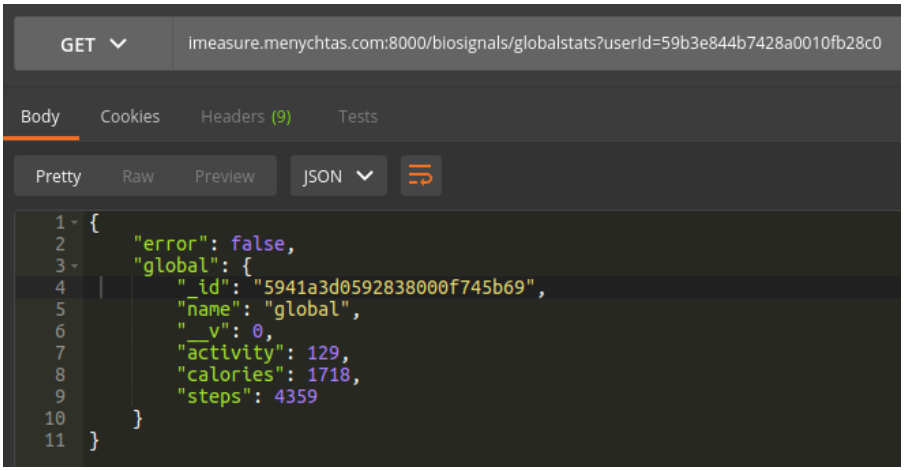
Σχήμα 64: Αίτημα Πρόσβασης στα συνολικά στατιστικά από χρήστη που δεν μοιράζεται τα δεδομένα του

Ο χρήστης, φυσικά, μπορεί να ενεργοποιήσει (και στην πορεία, αν θέλει, να απενεργοποιήσει) το data sharing με ένα αίτημα GET προς τον Server, στο αντίστοιχο endpoint (*imeasure.menychtas.com:8000/user/auth/share/userId= + <user id>*).



Σχήμα 65: Αίτημα Ενεργοποίησης Data Sharing

Έχοντας, πλέον, ενεργοποιήσει την επιλογή αυτή, ο χρήστης αποκτά πρόσβαση στα συνολικά στατιστικά, τα οποία, πλέον, συνυπολογίζουν και τα δικά του δεδομένα. Αυτό φαίνεται και από το ακόλουθο αίτημα:



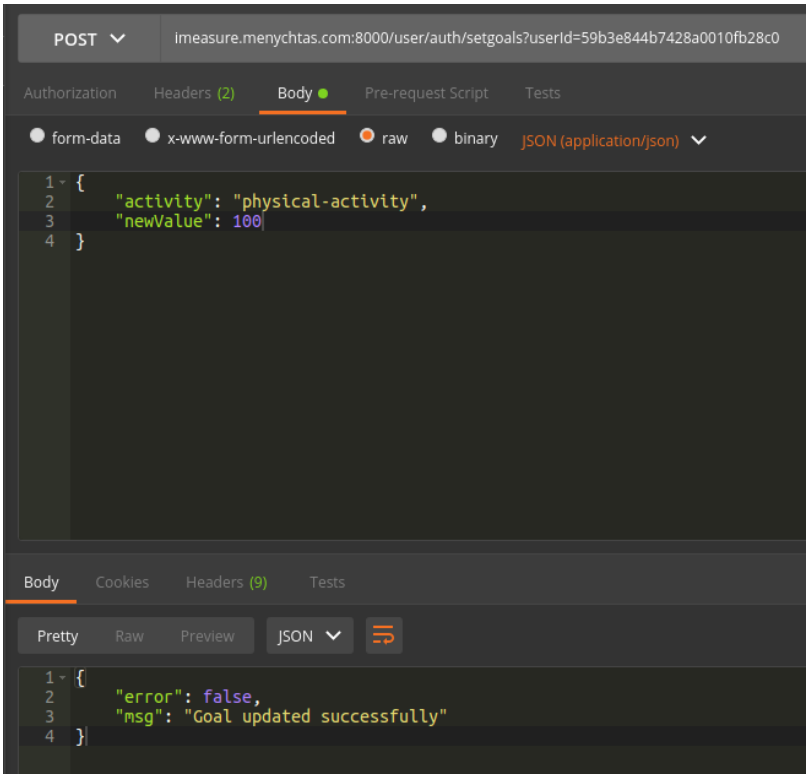
Σχήμα 66: Αίτημα Πρόσβασης στα συνολικά στατιστικά από χρήστη που μοιράζεται τα δεδομένα του

4.13 Στόχοι προς επίτευξη

Στα πλαίσια της Quantified Self εφαρμογής, ο χρήστης μπορεί να θέσει στόχους προς επίτευξη, οι οποίοι και σχετίζονται με την σωματική του δραστηριότητα. Μάλιστα, με την εγγραφή του, ορίζονται αυτόματα κάποιες προεπιλογές (βλ. Σχήμα 62):

- 10,000 βήματα την ημέρα
- 2,500 θερμίδες την ημέρα
- 120 λεπτά άσκησης την ημέρα

Τις επιλογές αυτές, ο χρήστης μπορεί να τις αλλάξει, στέλνοντας ένα αίτημα POST στον Εξυπηρετητή, στο σώμα του οποίου θα δηλώνει ποιόν στόχο θέλει να αλλάξει και ποια είναι η νέα τιμή. Όπως συνήθως, το αίτημα πρέπει να φέρει ως παράμετρο το *userId*, καθώς επίσης και το token στις επικεφαλίδες του.



Σχήμα 67: Αίτημα ορισμού νέου στόχου για την διάρκεια ημερήσιας άσκησης

Μετά την επιτυχή διεκπεραίωση του αιτήματος, η εγγραφή του χρήστη στη βάση έχει μεταβληθεί:

```
"_id" : ObjectId("59b3e844b7428a0010fb28c0"),
"email" : "housiadas@gmail.com",
"password" : "$2a$05$A1MbSkIiBzCNY55.2RVSK.jT1MX5CNOhN2vULlxExSpPJrU7tvXa2",
"firstName" : "Dimitris",
"lastName" : "Chousiadas",
"accountConfirmed" : "1",
"googleFitActivated" : true,
"fitbitActivated" : true,
"stepsPerDayGoal" : 10000,
"caloriesPerDayGoal" : 2500,
"activityPerDayGoal" : 100,
"dataSharing" : true,
"stats" : {
        "avgSteps" : 4590,
        "maxSteps" : 17199,
        "avgCalories" : 2001,
        "maxCalories" : 5088,
        "avgActivity" : 120,
        "maxActivity" : 701
},
"__v" : 0,
"today" : {
        "activity" : 18,
        "calories" : 706.4323033690453,
        "steps" : 1544
},
"googleFitTimestamp" : {
        "calories_burned" : "2017-09-09",
        "heart_rate" : "2016-09-09",
        "body_weight" : "2016-09-09",
        "step_count" : "2017-09-09",
        "activity" : "2017-09-09"
},
"fitbitTimestamp" : {
        "activity" : "2017-09-09",
        "sleep" : "2017-09-09",
        "steps" : "2017-09-09",
        "weight" : "2017-09-09"
}
```


Σχήμα 68: Εγγραφή Χρήστη μετά την αλλαγή του στόχου

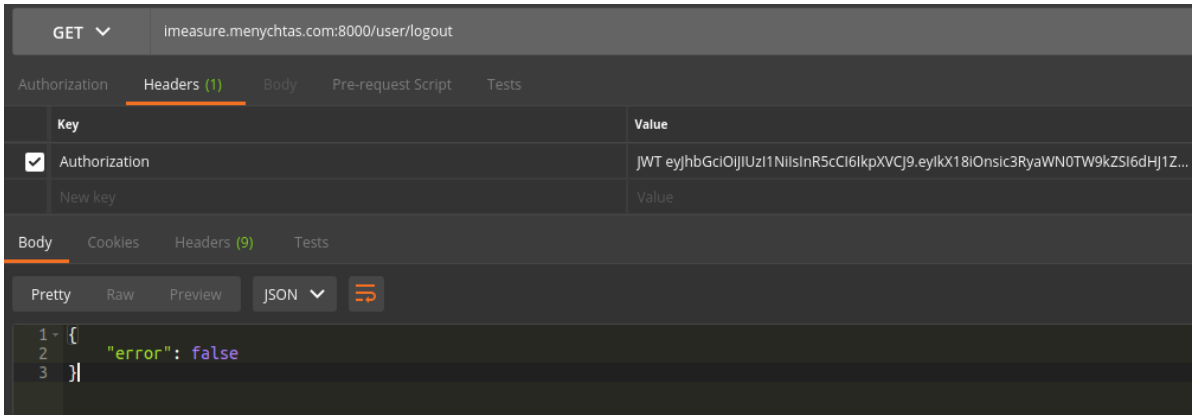
Με βάση τους στόχους αυτούς, το σύστημα Πελάτη παράγει και εμφανίζει στην διεπαφή χρήστη ορισμένα μηνύματα, που, είτε επιβραβεύουν τον χρήστη, όταν αυτός επιτυγχάνει κάποιον στόχο, είτε, σε διαφορετική περίπτωση, τον παρακινούν να συνεχίσει. Στο επόμενο κεφάλαιο θα εξεταστούν και πάλι τα μηνύματα αυτά.

4.14 Επιπλέον Λειτουργίες

Στις προηγούμενες παραγράφους εξετάστηκαν οι σημαντικότερες λειτουργίες της εφαρμογής. Στο σημείο αυτό θα γίνει αναφορά στις υπόλοιπες λειτουργίες, οι οποίες και δρουν συμπληρωματικά στο σύστημα.

Logout

Προφανώς, ένα σύστημα που υποστηρίζει την είσοδο του χρήστη, θα πρέπει να υποστηρίζει και την έξοδο. Εν προκειμένω, η έξοδος υλοποιείται με ένα αίτημα GET στον Εξυπηρετητή, το οποίο και φέρει στις επικεφαλίδες του το token.

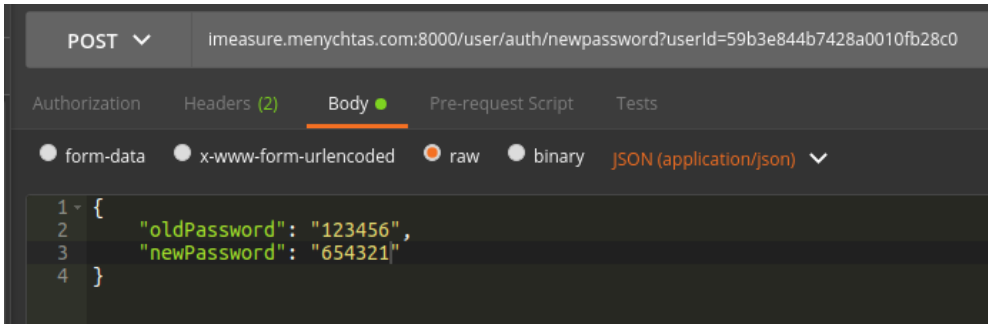


Σχήμα 69: Αίτημα Εξόδου του χρήστη από το σύστημα

Όταν ο Server λάβει το αίτημα, το PassportJS αναλαμβάνει να το χειριστεί, διαγράφοντας το token από την μνήμη, ώστε να μην μπορεί, πλέον, ο κάτοχος του να το χρησιμοποιήσει.

Αλλαγή Κωδικού Πρόσβασης

Επίσης, μέσω της διεπαφής, ο χρήστης μπορεί να αλλάξει, οποιαδήποτε στιγμή, τον κωδικό πρόσβασης του. Αυτό γίνεται με ένα αίτημα POST, στου οποίου το σώμα υπάρχει τόσο ο παλιός (για επιβεβαίωση), όσο και ο καινούργιος κωδικός.



Σχήμα 70: Αίτημα Αλλαγής Κωδικού

Αν ο δοθέν παλιός κωδικός είναι σωστός, κάτι το οποίο ελέγχει ο Server κατακερματίζοντας τον και συγκρίνοντας τον με την αντίστοιχη εγγραφή στη βάση, το αίτημα επιτυγχάνει, διαφορετικά ο χρήστης λαμβάνει κάποιο μήνυμα λάθους.

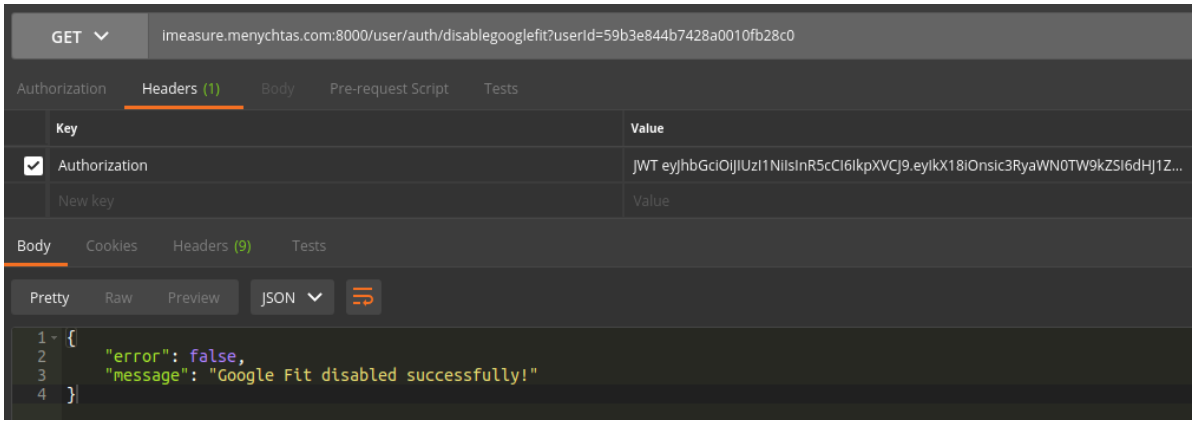
Αποσύνδεση από Fitbit και GoogleFit

Ο χρήστης που έχει συνδέσει τον λογαριασμό του με το προφίλ του στο Fitbit ή το GoogleFit μπορεί, αν θελήσει, να τον αποσυνδέσει (και στη συνέχεια να τον ξανασυνδέσει). Αυτό γίνεται με ένα αίτημα GET, το οποίο ο Εξυπηρετητής διεκπεραιώνει όπως φαίνεται στο επόμενο απόσπασμα κώδικα (στην περίπτωση του GoogleFit):

```
exports.disableGoogleFit = function (req, res) {

    User.findOne({
        _id: req.query.userId
    }, function (err, user) {
        if (err)
            res.status(500).json(err);

        else if (!user)
            res.status(404).json({error: true, message: "The user doesn't exist"});

        else if (!user.googleFitActivated)
            res.status(404).json({error: true, message: "The user has not enabled Google Fit"});

        else {
            user.googleFitActivated = false;
            user.save(function (err) {
                if (err)
                    res.status(500).json(err);
                else
                    res.json({error: false, message: "Google Fit disabled successfully!"})
            })
        }
    })

};
```

Σχήμα 71: Αποσύνδεση από GoogleFit

Ουσιαστικά, αρκεί το πεδίο *googleFitActivated* να πάρει την τιμή *False*, ώστε η εφαρμογή να θεωρεί πως ο χρήστης δεν χρησιμοποιεί GoogleFit. Ταυτόχρονα, το Shimmer θυμάται πως ο χρήστης κάποτε είχε συνδέσει το προφίλ του, με αποτέλεσμα, η επαναφορά, σε περίπτωση που θελήσει ο χρήστης, να είναι αρκετά απλή.

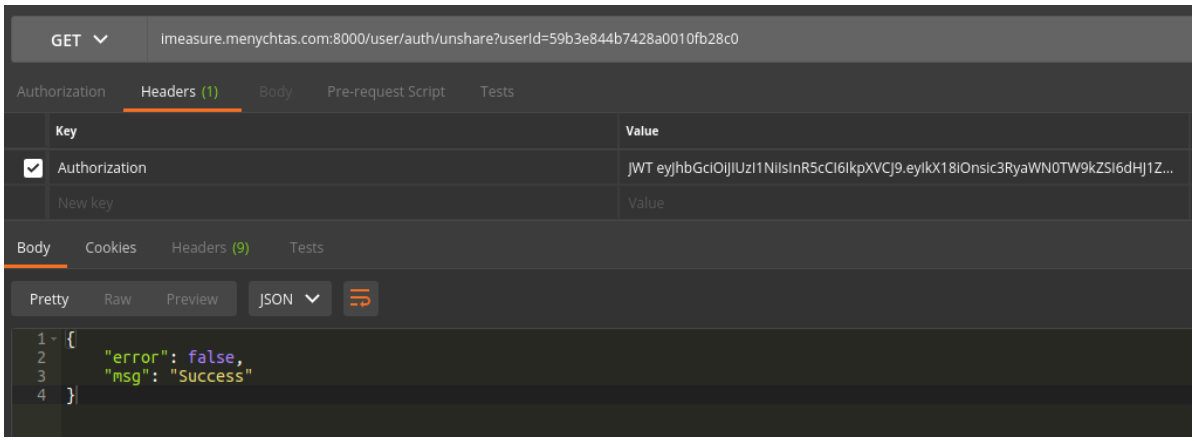


Σχήμα 72: Επιτυχές Αίτημα Αποσύνδεσης από GoogleFit

Ομοίως με τα παραπάνω, γίνεται και η αποσύνδεση από το προφίλ του χρήστη στο Fitbit.

Data Unsharing

Επιπλέον, όπως μπορεί ο χρήστης να μοιραστεί τα δεδομένα του με το σύνολο, έτσι μπορεί και να σταματήσει να τα μοιράζεται. Το παρακάτω αίτημα GET υλοποιεί αυτή ακριβώς την λειτουργία:



Σχήμα 73: Αίτημα με το οποίο ο χρήστης σταματάει να μοιράζεται τα δεδομένα του

Σημειωματάριο

Τέλος, το API παρέχει και δύο μεθόδους, σχετικά με την υλοποίηση της λειτουργίας του σημειωματάριου. Συγκεκριμένα, με ένα αίτημα POST, ο χρήστης μπορεί να καταχωρήσει μια καινούργια σημείωση στην βάση, ενώ με ένα αίτημα GET μπορεί να ανακτήσει όλες τις σημειώσεις που έχει κρατήσει. Όπως αναφέρεται στην παράγραφο 4.1, οι σημειώσεις αυτές αποθηκεύονται μαζί, σε μια εγγραφή τύπου Diary, που αντιστοιχεί και στον εκάστοτε χρήστη. Περισσότερα για την λειτουργία αυτή στο επόμενο κεφάλαιο.

4.15 Containerization, Continuous Integration & Delivery

Στην παράγραφο αυτή θα αναλυθεί το πως χρησιμοποιήθηκε το Docker για την εγκατάσταση και εκτέλεση της εφαρμογής. Αρχικά, θα δοθούν οι εξής δύο ορισμοί

- Docker Image: πρόκειται για ένα ελαφρύ, αυτόνομο, εκτελέσιμο πακέτο, το οποίο περιλαμβάνει οτιδήποτε χρειάζεται ένα κομμάτι λογισμικού για να τρέξει, συμπεριλαμβανομένου του κώδικα, ενός περιβάλλοντος εκτέλεσης (runtime), βιβλιοθήκες, περιβαλλοντικές μεταβλητές και αρχεία ρυθμίσεων (configuration files)
- Docker Container: πρόκειται για ένα στιγμιότυπο ενός Image, το οποίο εκτελείται. Η εκτέλεση γίνεται σε απομόνωση από το host environment, και το container μπορεί να έχει πρόσβαση, μετά από κατάλληλες ρυθμίσεις, μόνο σε αρχεία ή θύρες του host

Στα πλαίσια της εφαρμογής, δημιουργήθηκαν τρία Docker Images:

- ένα Docker Image για τη βάση Mongo
- ένα Docker Image για την εφαρμογή Quantified Self
- ένα Docker Image για το Shimmer

Εγκατεστημένο στο μηχάνημα *imeasure.menychtas.com* είναι και το πρόγραμμα Gitlab, το οποίο περιλαμβάνει και το, όπως λέγεται, Gitlab Registry. Πρόκειται για ένα ιδιωτικό μητρώο από Docker Images, στο οποίο και αποθηκεύονται τα τρία παραπάνω Images.

Αφού ρυθμιστούν κατάλληλα, τα Images της Mongo και του Shimmer εκτελούνται, με τη δημιουργία των αντίστοιχων Containers. Μάλιστα, τα δύο αυτά containers συνδέονται (link), καθώς το Shimmer αποθηκεύει στην βάση τις απαραίτητες πληροφορίες που χρειάζεται. Η σύνδεση δύο containers επιτυγχάνεται μέσω θυρών. Συγκεκριμένα, καθώς ένα container, ιδωμένο από το εσωτερικό του, λειτουργεί σαν ένας ανεξάρτητος υπολογιστής, μπορεί να εκθέσει κάποια θύρα στο εξωτερικό του περιβάλλον και να περιμένει, στην θύρα αυτή, κάποια σύνδεση ή επικοινωνία με άλλον υπολογιστή. Έτσι λοιπόν, τα δύο containers, εκθέτοντας στο εξωτερικό τους τις κατάλληλες θύρες, έρχονται σε επικοινωνία, την οποία εξασφαλίζει το Docker Engine.

Στην συνέχεια, με κάθε αλλαγή στον κώδικα του Quantified Self, πυροδοτείται αυτόματα μια σειρά από ενέργειες, σε σωλήνωση (pipeline):

- Αρχικά, σταματάει η εκτέλεση του Quantified Self Container που, ενδεχομένως, έτρεχε έως εκείνη τη στιγμή, και το container αυτό διαγράφεται από τη μνήμη
- Στη συνέχεια, διαγράφεται το παλιό Quantified Self Image, και δημιουργείται ένα καινούργιο, το οποίο περιλαμβάνει και τις αλλαγές που έγιναν
- Από το καινούργιο Image προκύπτει ένα καινούργιο Container, το οποίο συνδέεται, εκθέτοντας στο εξωτερικό κατάλληλες θύρες, τόσο με τα, ήδη τρέχοντα, containers της Mongo και του Shimmer. Η τελευταία, αυτή, ενέργεια, εναλλακτικά, μπορεί να γίνει και χειροκίνητα, ώστε να αποφασίζει ο προγραμματιστής πότε μια αλλαγή θα ενσωματωθεί στο σύστημα παραγωγής

Συμπερασματικά, οι τρεις λειτουργίες του containerization, του continuous integration και του continuous delivery είναι στενά συνδεδεμένες μεταξύ τους και το περιβάλλον του Gitlab διαθέτει τα κατάλληλα εργαλεία για την αυτοματοποιημένη υλοποίηση τους.

Κεφάλαιο 5. Αποτελέσματα

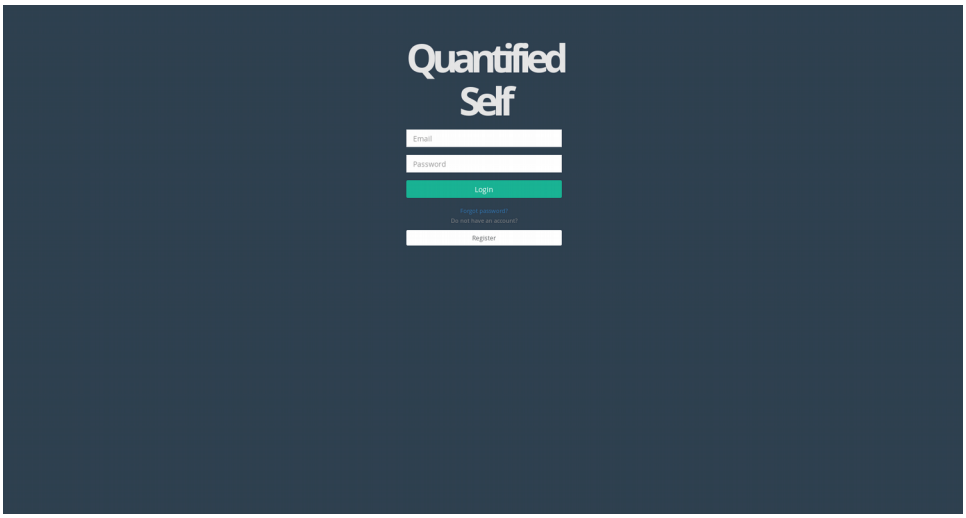
5.1 Λειτουργία

Στο προηγούμενο κεφάλαιο περιγράφηκε εκτενώς η υλοποίηση του back-end της εφαρμογής και των μεθόδων του REST API που αναπτύχθηκε. Οι μέθοδοι αυτές δοκιμάστηκαν με το Postman, ώστε να εξασφαλιστεί η ορθή λειτουργία τους. Αλλά, φυσικά, ο χρήστης δεν χρησιμοποιεί το REST API μέσω του Postman, αλλά μέσω μιας κατάλληλα διαμορφωμένης διεπαφής (UI), η οποία και αποτελεί το αντικείμενο του κεφαλαίου αυτού, καθώς είναι και το τελικό προϊόν της ανάπτυξης.

Η πορεία που θα ακολουθήσει η περιγραφή του UI, στη συνέχεια, είναι αυτή που θα ακολουθούσε και ένας καινούργιος χρήστης, ο οποίος θέλει να χρησιμοποιήσει την Quantified Self εφαρμογή. Έτσι, επισκεπτόμενος τη διεύθυνση

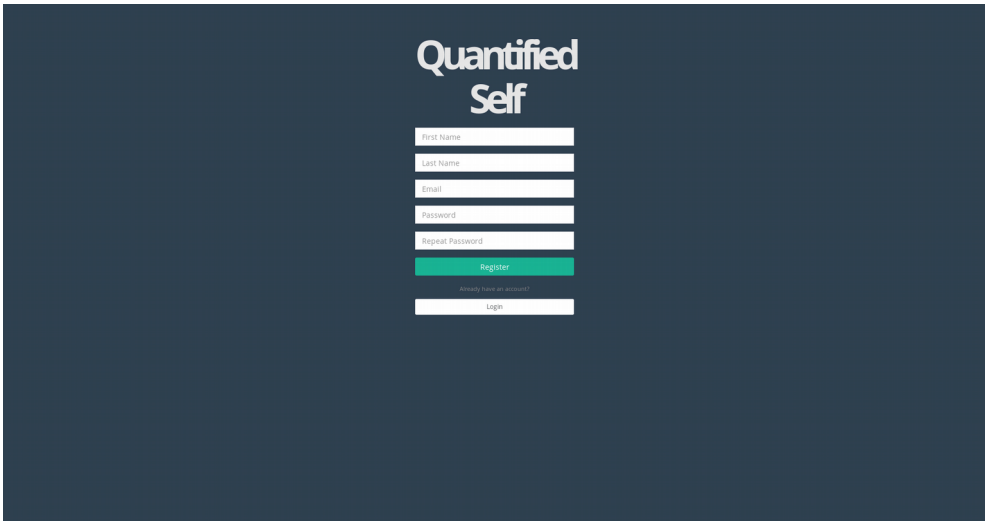
imeasure.menychtas.com:8000

ο χρήστης ανακατευθύνεται στην ακόλουθη σελίδα, όπου μπορεί να κάνει login:



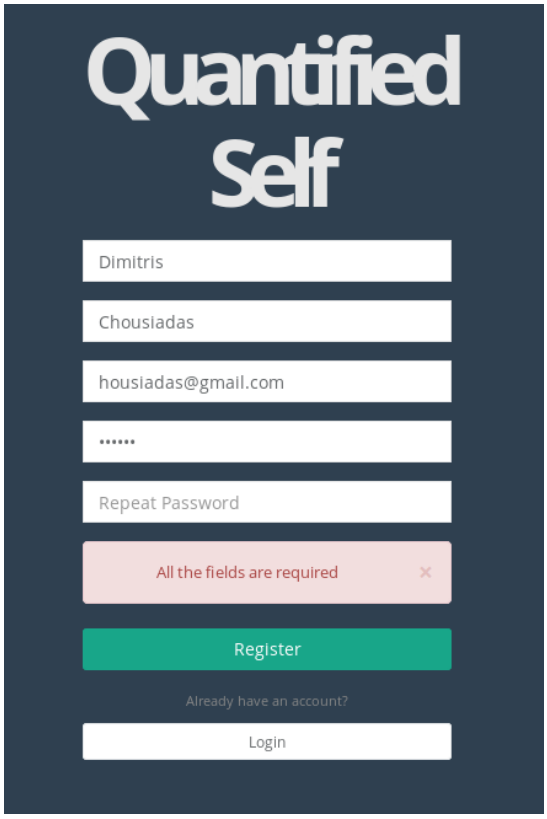
Σχήμα 74: Login Page

Ο νέος χρήστης, που δεν έχει ακόμα λογαριασμό στο σύστημα, πατάει το κουμπί Register, και ανακατευθύνεται στην επόμενη σελίδα:

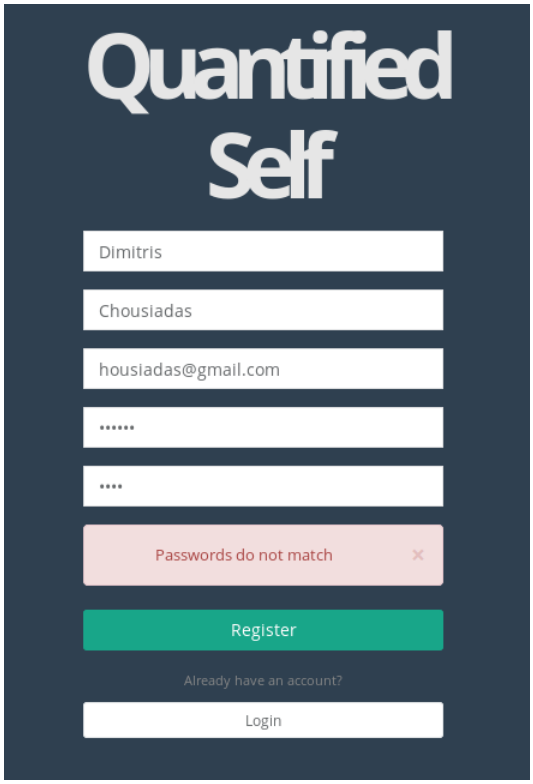


Σχήμα 75: Register Page

Εδώ, μπορεί να κάνει εγγραφή, δίνοντας τα ζητούμενα στοιχεία. Κατά την εγγραφή, φυσικά, μπορεί να προκύψουν κάποια λάθη, όπως φαίνεται παρακάτω:



Σχήμα 76: Ο χρήστης δεν συμπλήρωσε όλα τα απαραίτητα πεδία



Σχήμα 77: Ο χρήστης δεν επιβεβαίωσε σωστά τον κωδικό του

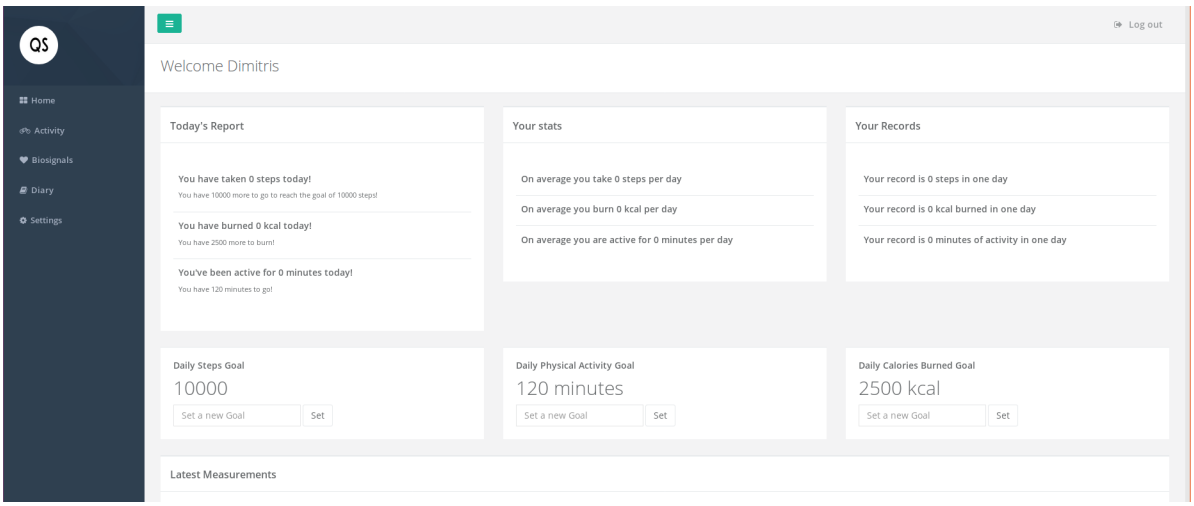
Τελικά, ο χρήστης συμπληρώνει την φόρμα εγγραφής και ανακατευθύνεται στην επόμενη σελίδα:

Σχήμα 78: Επιτυχημένη Εγγραφή

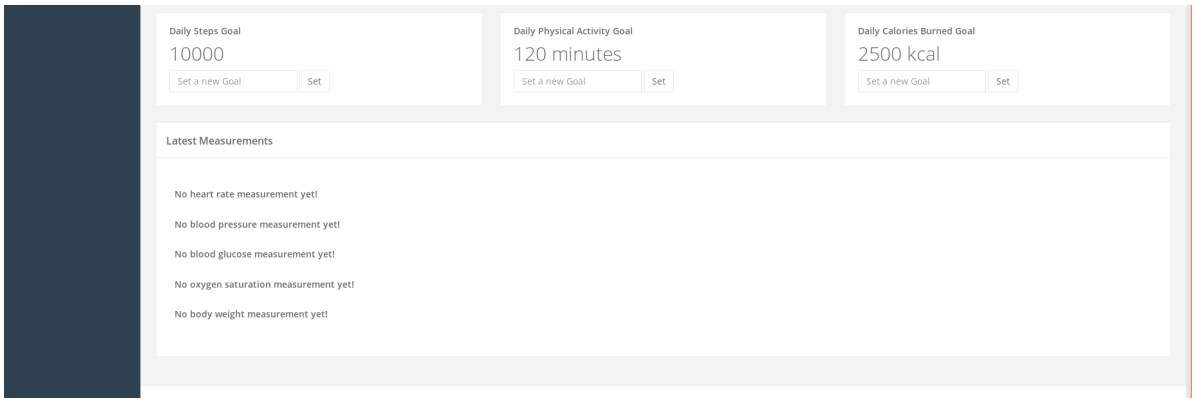
Στη συνέχεια λαμβάνει ένα μήνυμα ηλεκτρονικού ταχυδρομείου με ένα σύνδεσμο, τον οποίο πρέπει να ακολουθήσει ώστε να ενεργοποιήσει τον λογαριασμό του:

Σχήμα 79: Μήνυμα με τον σύνδεσμο ενεργοποίησης

Μετά και από αυτό το βήμα, ο χρήστης ανακατευθύνεται στο homepage της εφαρμογής, το οποίο, βέβαια, για έναν νέο χρήστη δεν φέρει κάποια ιδιαίτερη πληροφορία.

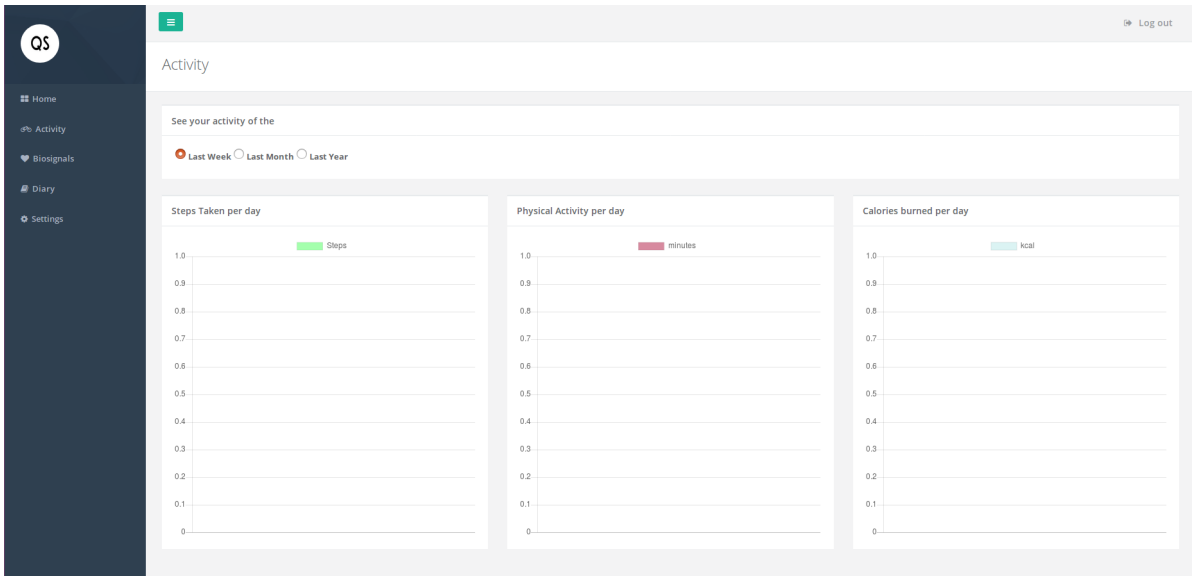


Σχήμα 80: Homepage (A)



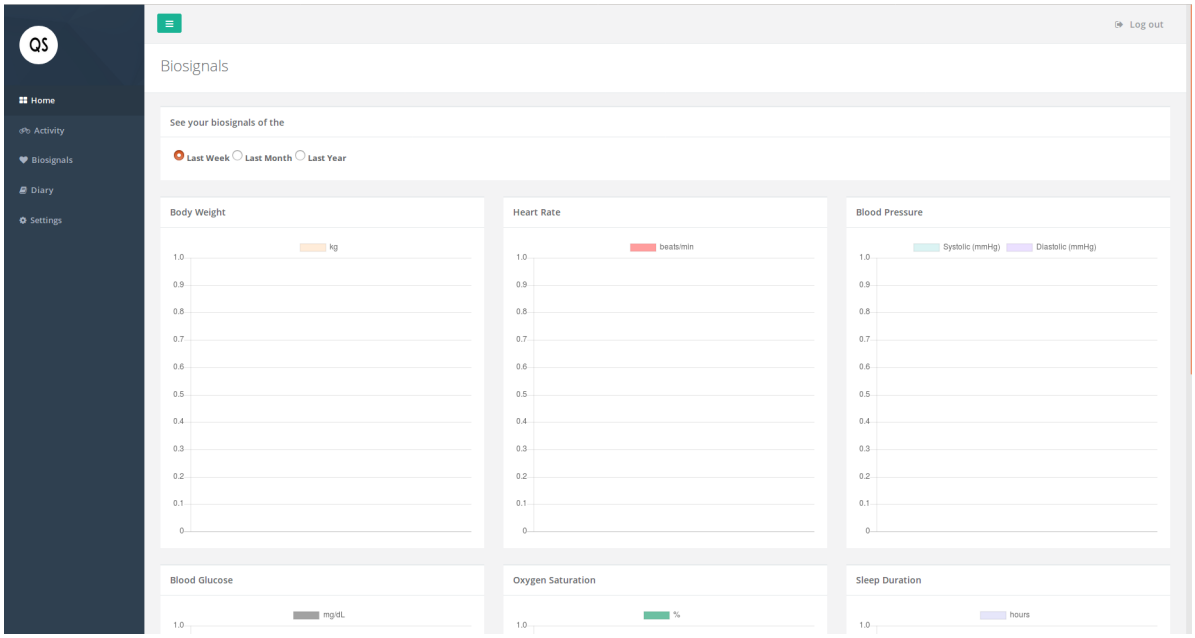
Σχήμα 81: Homepage (B)

Όπως φαίνεται, υπάρχει, στα αριστερά της σελίδας μια μπάρα με επιλογές, εκ των οποίων μία λέγεται Activity, ενώ μία άλλη Biosignals. Η πρώτη οδηγεί στην σελίδα, όπου ο χρήστης μπορεί να δει τα δεδομένα της σωματικής του δραστηριότητας. Στο στάδιο αυτό, φυσικά, δεν υπάρχουν ακόμα δεδομένα.

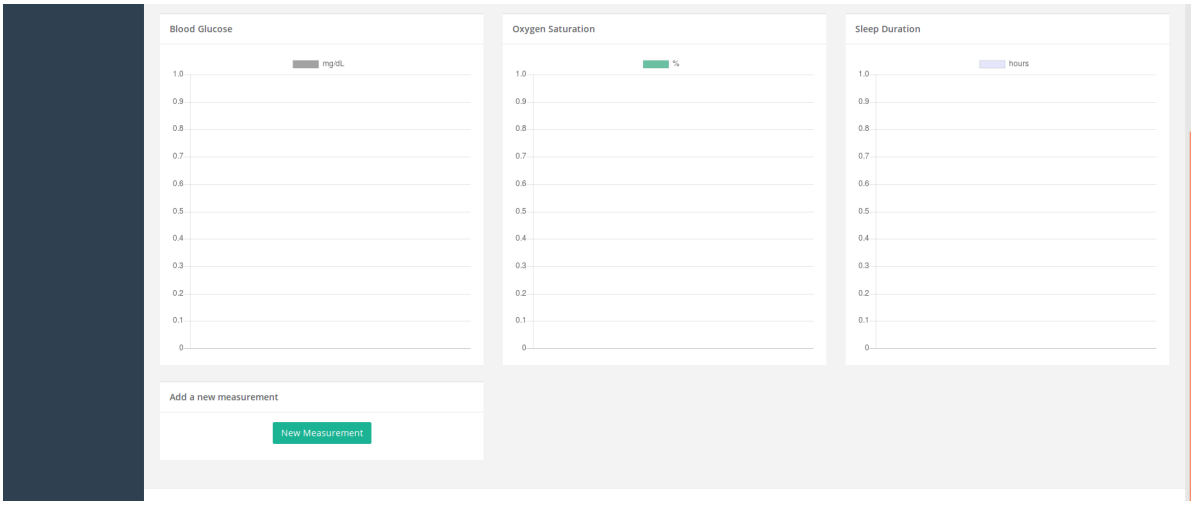


Σχήμα 82: Activity Page

Αντίστοιχα, η δεύτερη επιλογή οδηγεί στη σελίδα με τα βιοσήματα του χρήστη, επίσης κενή, προς το παρόν, όπως φαίνεται στα σχήματα 82 και 83.

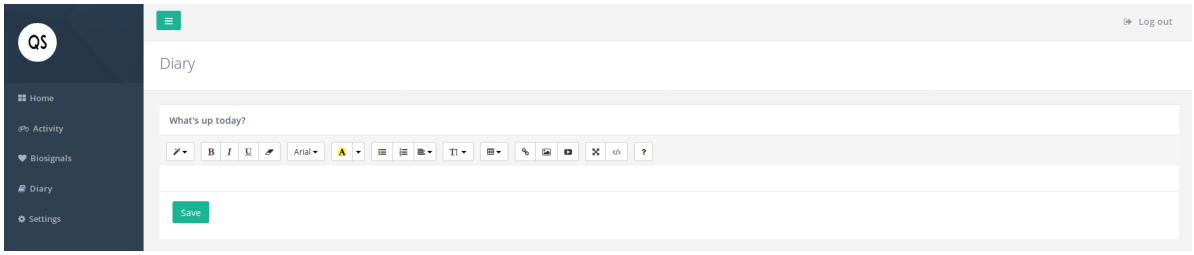


Σχήμα 83: Biosignals Page (A)

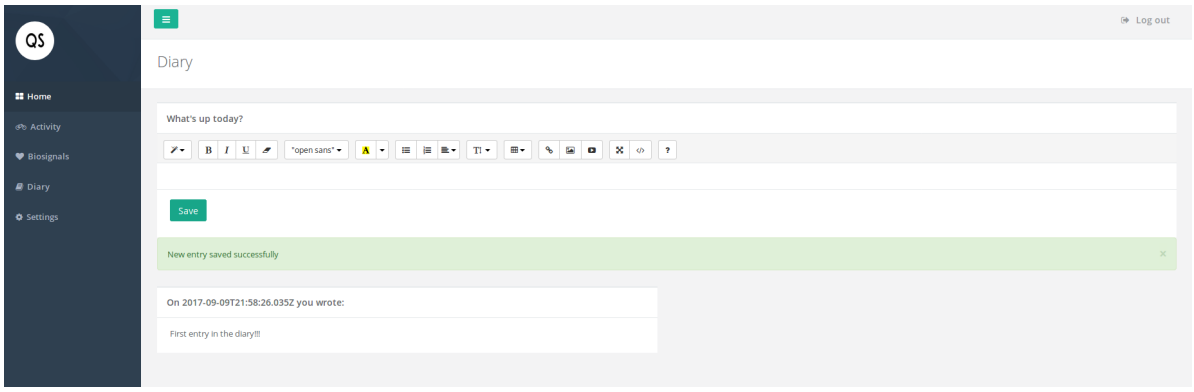


Σχήμα 84: Biosignals Page (B)

Μια τρίτη επιλογή οδηγεί στην σελίδα του σημειωματάριου, όπου ο χρήστης μπορεί να καταχωρήσει την πρώτη του εγγραφή.



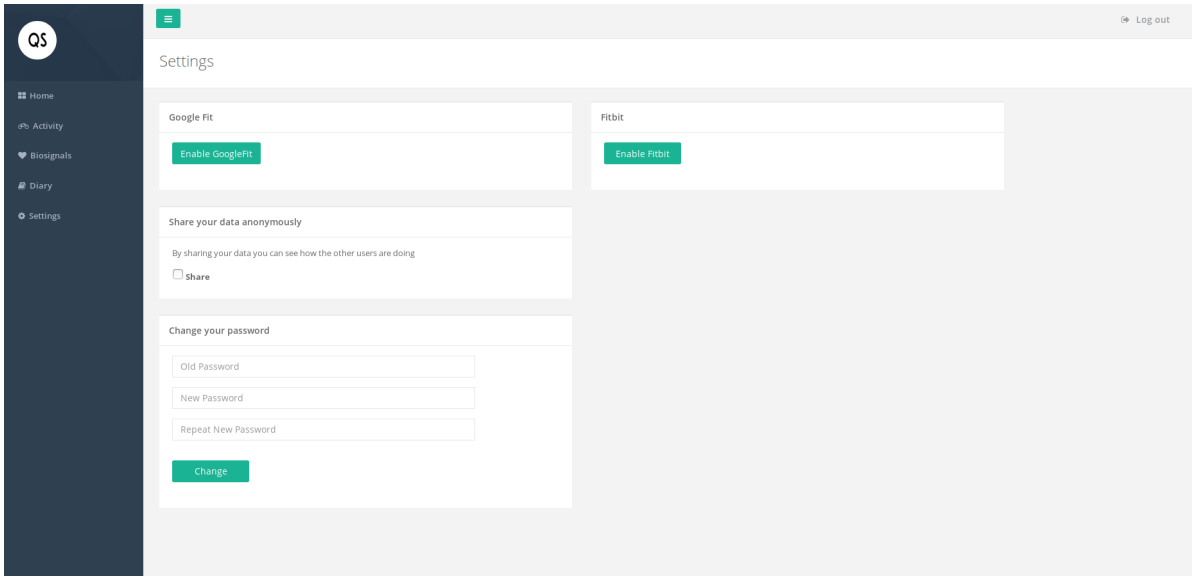
Σχήμα 85: Σελίδα Σημειωματάριου



Σχήμα 86: Καταχώρηση Πρώτης Εγγραφής στο Σημειωματάριο

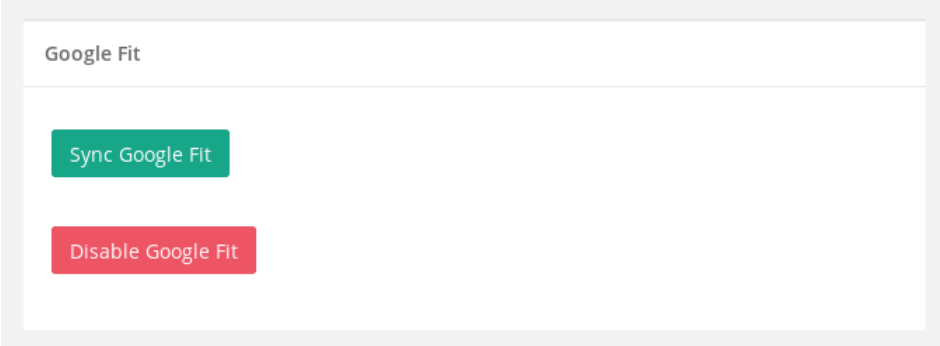
Ο χρήστης, εν συνεχεία, επιλέγει τις ρυθμίσεις (Settings), όπου μπορεί:

- Να συνδέσει τον λογαριασμό του με το προφίλ του στο Fitbit ή το GoogleFit
- Να επιλέξει να μοιράζεται τα δεδομένα του (data sharing)
- Να αλλάξει το password του



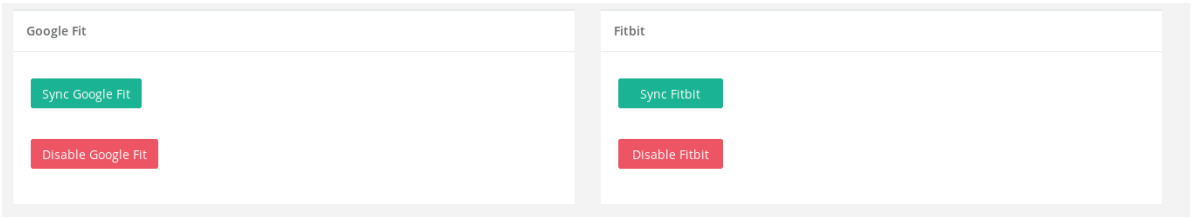
Σχήμα 87: Settings Page

Ο χρήστης επιλέγει να ενεργοποιήσει το GoogleFit και, αφού ολοκληρωθεί η διαδικασία που περιγράφηκε στην παράγραφο 4.7, εμφανίζεται η ακόλουθη επιλογή στην σελίδα των ρυθμίσεων:



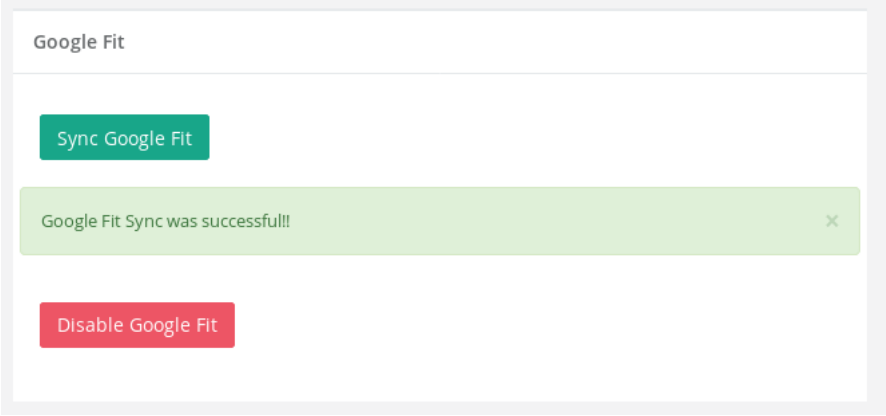
Σχήμα 88: Ενεργοποίηση GoogleFit

Ομοίως, επιλέγει να ενεργοποιήσει και το Fitbit:

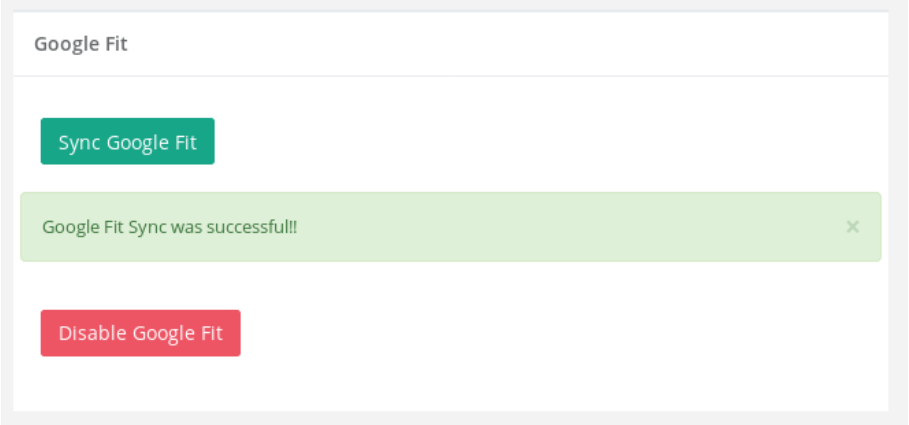


Σχήμα 89: Ενεργοποίηση GoogleFit και Fitbit

Έχοντας κάνει τα παραπάνω, ο χρήστης, τώρα, θέλει να συγχρονίσει την εφαρμογή με τα δεδομένα που έχουν καταγράψει οι Activity Trackers. Το επιτυγχάνει πατώντας το κουμπί *Sync Google Fit.* Αρχικά, εμφανίζεται ένα μήνυμα που ειδοποιεί τον χρήστη ότι ο συγχρονισμός βρίσκεται σε εξέλιξη, ενώ, μόλις ολοκληρωθεί, εμφανίζεται ένα μήνυμα επιτυχίας.

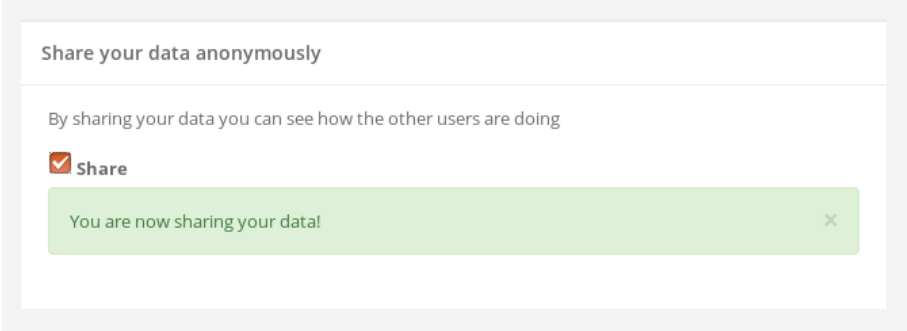


Σχήμα 90: Συγχρονισμός σε Εξέλιξη



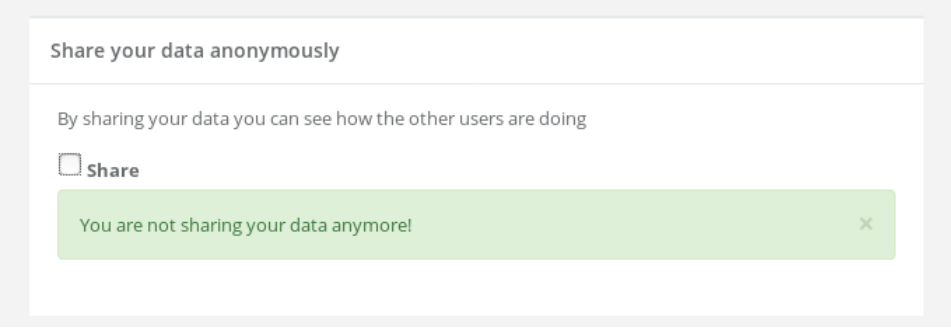
Σχήμα 91: Επιτυχής Συγχρονισμός

Πριν αποχωρήσει από την σελίδα ρυθμίσεων, ο χρήστης επιλέγει να μοιραστεί τα δεδομένα του, ώστε να αποκτήσει πρόσβαση στα συνολικά στατιστικά.



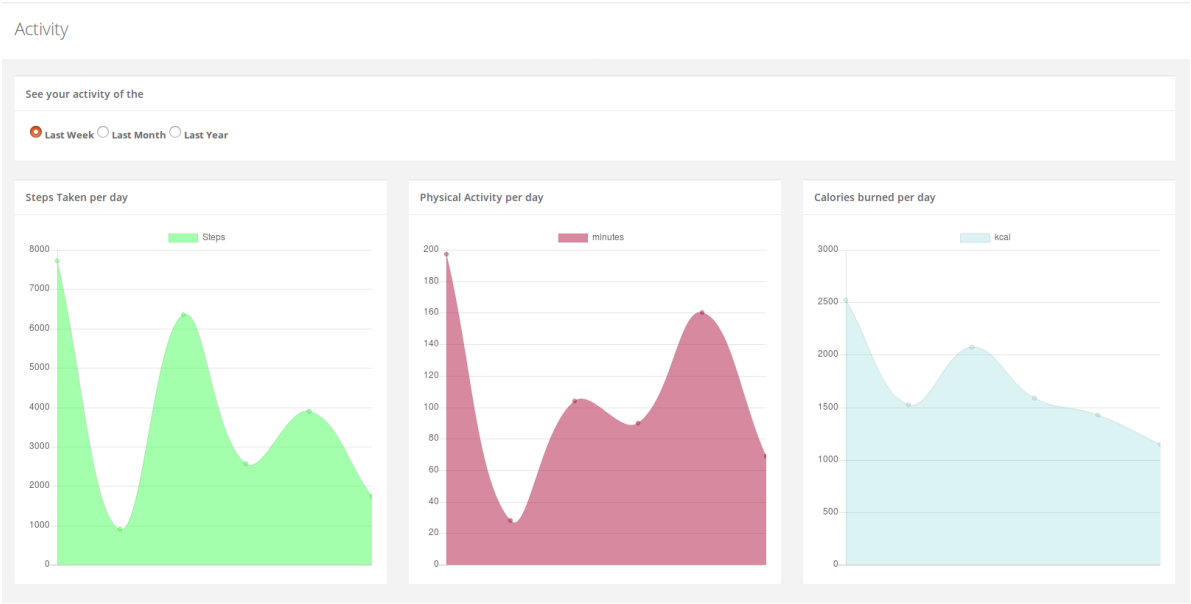
Σχήμα 92: Ο χρήστης μοιράζεται τα δεδομένα του

Οποιαδήποτε στιγμή θελήσει, βέβαια, μπορεί να σταματήσει να τα μοιράζεται:



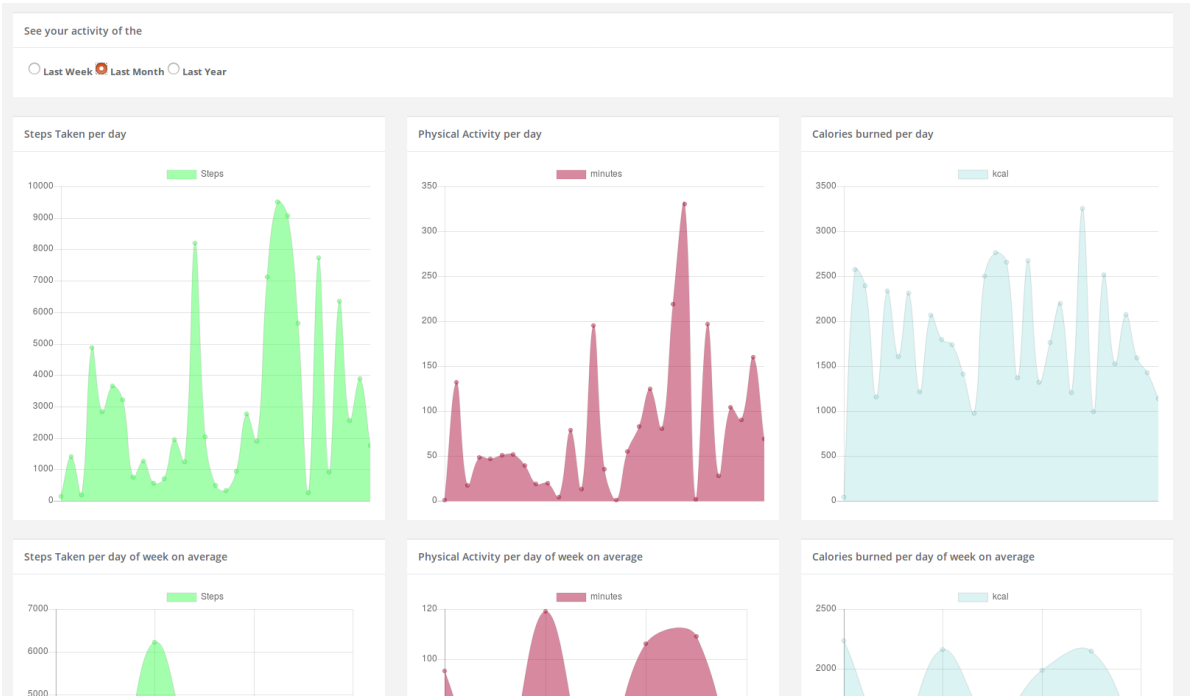
Σχήμα 93: Ο χρήστης σταματάει να μοιράζεται τα δεδομένα του

Ο χρήστης, τώρα, επιστρέφει στη σελίδα Activity, όπου μπορεί να δει, πλέον, τα δεδομένα του, έχοντας κάνει τον συγχρονισμό με το GoogleFit.

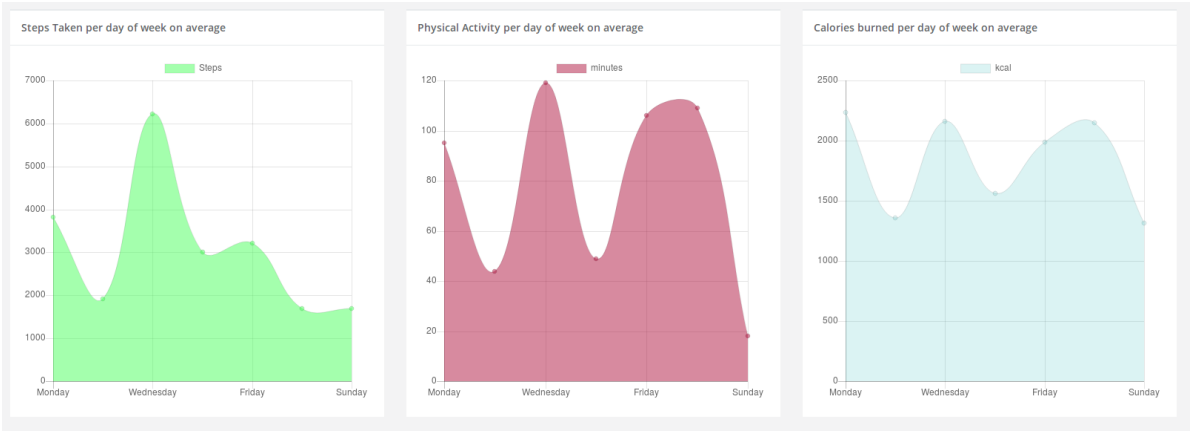


Σχήμα 94: Δεδομένα Δραστηριότητας τελευταίας εβδομάδας

Από προεπιλογή, βλέπει τα δεδομένα της τελευταίας εβδομάδας. Εναλλακτικά, μπορεί να δει τα δεδομένα του τελευταίου μήνα ή χρόνου. Σε αυτήν την περίπτωση, θα δει και μια ανάλυση των επιδόσεων του ανά ημέρα της εβδομάδας.

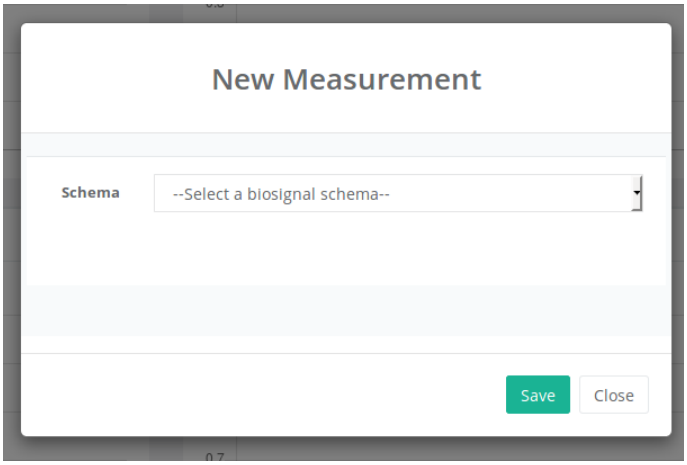


Σχήμα 95: Δεδομένα Δραστηριότητας τελευταίου μήνα (Α)

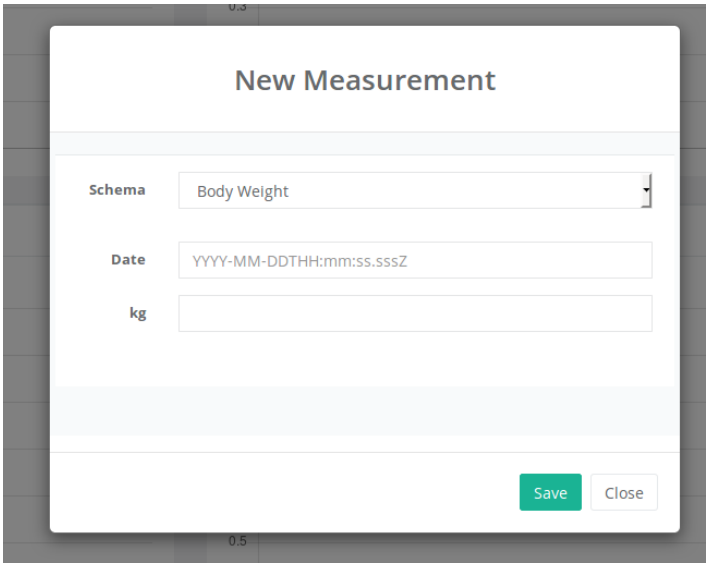


Σχήμα 96: Δεδομένα Δραστηριότητας τελευταίου μήνα (B) - Ανάλυση ανά ημέρα εβδομάδας

Έπειτα, ο χρήστης ενδιαφέρεται να καταχωρήσει μια μέτρηση που αφορά το βάρος του. Έτσι, επιστρέφει στη σελίδα βιοσημάτων και επιλέγει το *Add New Measurement.* Πατώντας το κουμπί αυτό, εμφανίζεται ένα παράθυρο με μια φόρμα προς συμπλήρωση. Αρχικά, ο χρήστης καλείται να επιλέξει τι είδους βιοσήμα θέλει να καταχωρήσει, εν προκειμένω, σωματικό βάρος.

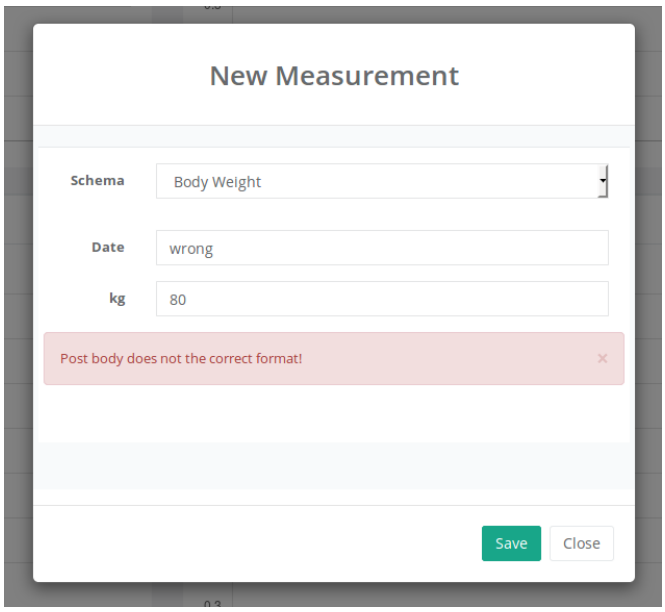


Σχήμα 97: Παράθυρο Καταχώρησης Νέου Βιοσήματος



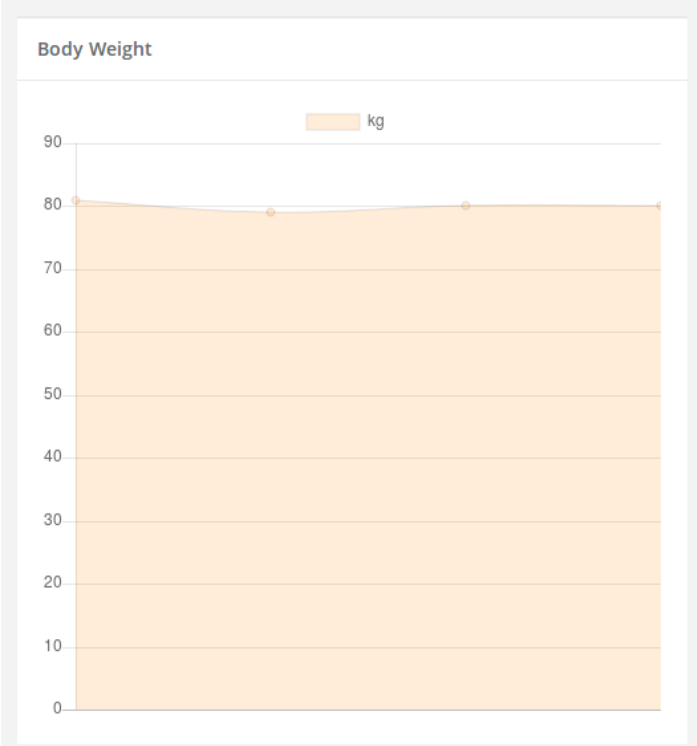
Σχήμα 98: Ο χρήστης επιλέγει να καταχωρήσει βιοσήμα τύπου Σωματικού Βάρους

Αν ο χρήστης συμπληρώσει λάθος τη φόρμα, εμφανίζεται κατάλληλο μήνυμα.



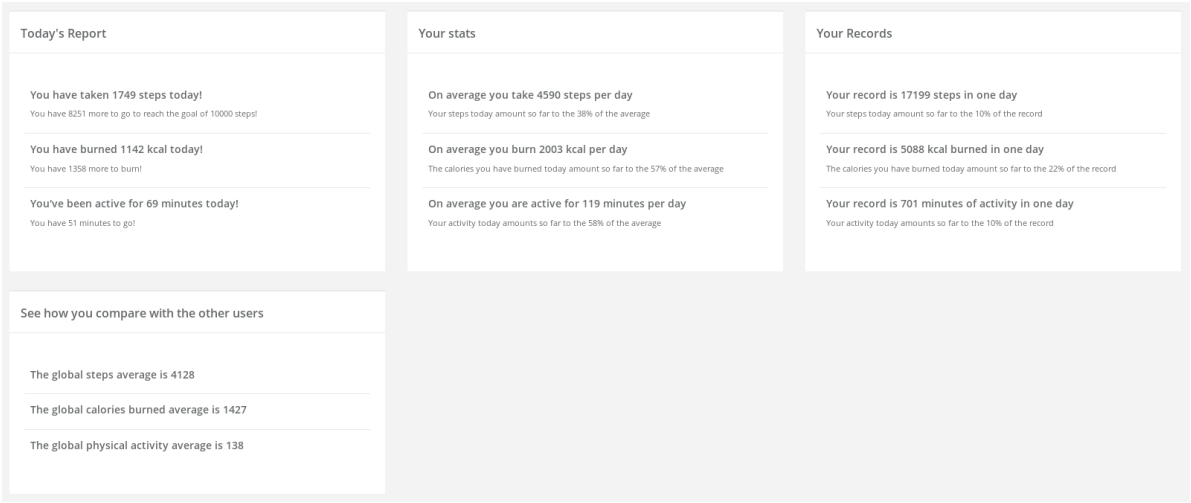
Σχήμα 99: Μήνυμα Λάθους κατά την συμπλήρωση φόρμας νέου βιοσήματος

Διαφορετικά, η νέα μέτρηση αποθηκεύεται επιτυχώς. Μετά από μερικές επαναλήψεις της παραπάνω διαδικασίας, οι νέες μετρήσεις εμφανίζονται στο διάγραμμα ως εξής:



Σχήμα 100: Διάγραμμα Σωματικού Βάρους

Έχοντας κάνει σύνδεση και συγχρονισμό με το GoogleFit και έχοντας επιλέξει να μοιραστεί τα δεδομένα του, ο χρήστης βλέπει ορισμένες αλλαγές στο Homepage του.

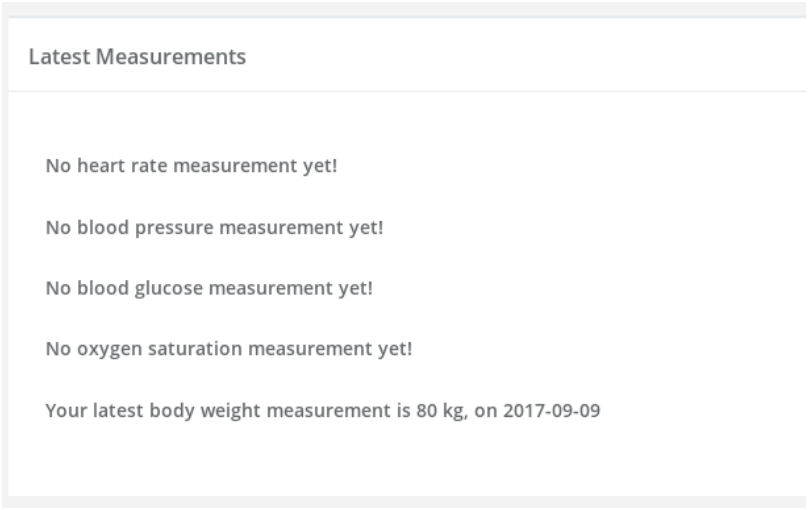


Σχήμα 101: Ανανεωμένο Homepage

Συγκεκριμένα, βλέπει:

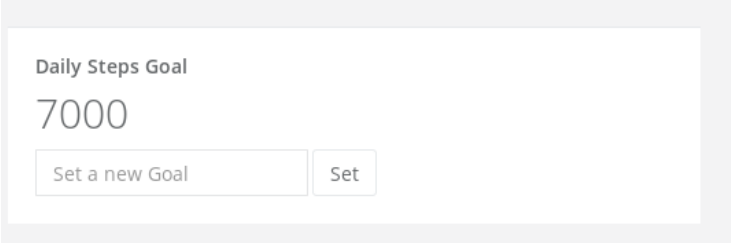
- Today’s Report: επισκόπηση της δραστηριότητάς του σήμερα και πόσο απέχει από το να επιτύχει τους στόχους που έχει θέσει
- Your Stats: στατιστικά αποτελέσματα που αφορούν στις μέσες τιμές των επιδόσεων του και πως αυτά συγκρίνονται με τον μέσο όρο των χρηστών
- Your Records: αποτελέσματα που αφορούν τις μέγιστες επιδόσεις του χρήστη και πως συγκρίνεται με αυτές οι σημερινές του επιδόσεις

Επιπλέον, βλέπει και τα αποτελέσματα της στατιστικής ανάλυσης επί του συνόλου των χρηστών. Παρακάτω, στην ίδια σελίδα βλέπει και μια αναφορά στις τελευταίες μετρήσεις βιοσημάτων του, όπως φαίνεται και στην επόμενη εικόνα.



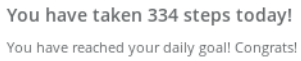
Σχήμα 102: Τελευταίες μετρήσεις βιοσημάτων χρήστη

Στην αρχική σελίδα ο χρήστης μπορεί να αλλάξει και τους προεπιλεγμένους στόχους που έχει ορίσει η εφαρμογή. Επί παραδείγματι, ο default στόχος ημερήσιων βημάτων είναι 10.000 και στην επόμενη εικόνα φαίνεται πως ο χρήστης έθεσε έναν καινούργιο στόχο, στα 7.000 βήματα.



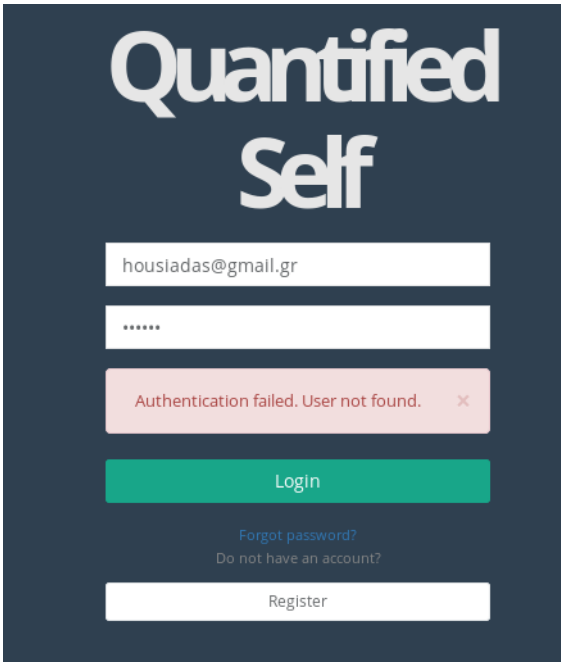
Σχήμα 103: Νέος στόχος ημερήσιων βημάτων

Όταν, τέλος, ο χρήστης επιτύχει τον στόχο αυτό, στο Today's Report εμφανίζεται μήνυμα επιβράβευσης:

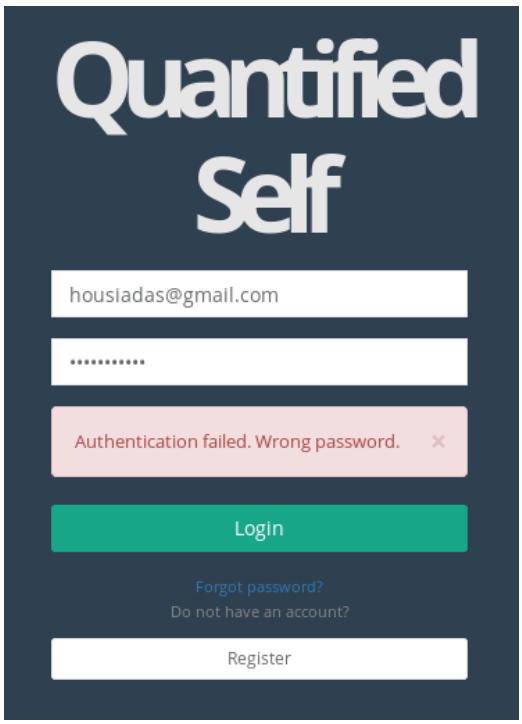


Σχήμα 104: Μήνυμα Επιβράβευσης

Ο χρήστης, κάποια στιγμή, εξέρχεται από το σύστημα (logout), συνεπώς, την επόμενη φορά που θα θελήσει να εισέλθει, θα πρέπει να κάνει login. Κατά το login, όμως, ενδεχομένως, να προκύψει κάποιο σφάλμα, όπως φαίνεται ακολούθως:

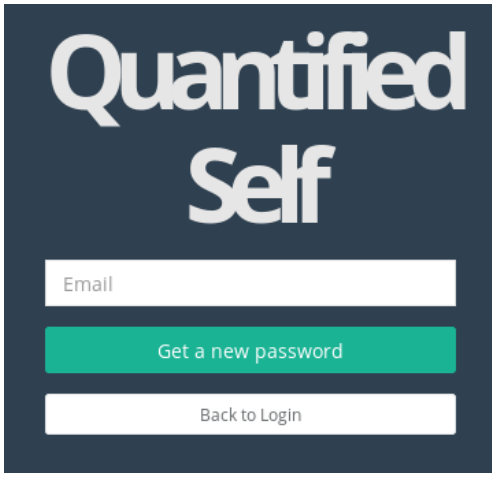


Σχήμα 105: Σφάλμα κατά το login - εισαγωγή λάθους διεύθυνσης ηλεκτρονικού ταχυδρομείου



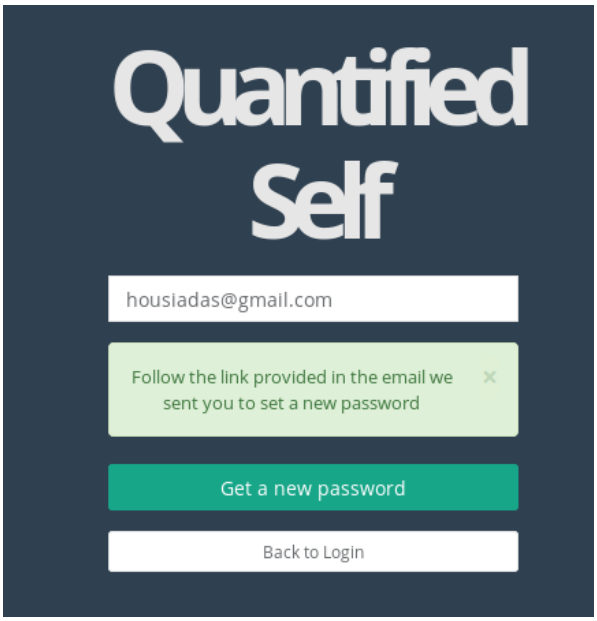
Σχήμα 106: Σφάλμα κατά το login - Εισαγωγή λάθους κωδικού πρόσβασης

Σε περίπτωση που ο χρήστης ξεχάσει τον κωδικό πρόσβασης επιλέγει τον σύνδεσμο *Forgot Password?*, ο οποίος τον ανακατευθύνει στην επόμενη σελίδα:

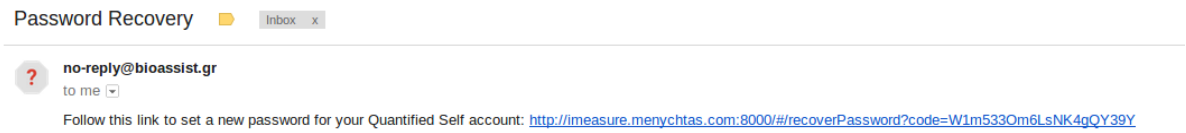


Σχήμα 107: Σελίδα Ανάκτησης Κωδικού Πρόσβασης

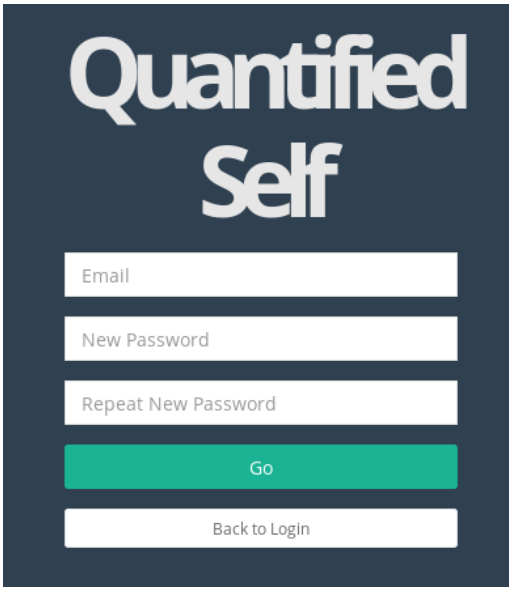
Εδώ, ο χρήστης συμπληρώνει το email του, στο οποίο θα λάβει ένα μήνυμα με έναν σύνδεσμο που θα πρέπει να ακολουθήσει, ώστε να καταχωρήσει έναν καινούργιο κωδικό. Ο σύνδεσμος αυτός τον οδηγεί σε μια νέα σελίδα, όπου και συμπληρώνει μια φόρμα και ανακτά πρόσβαση στον λογαριασμό του. Η διαδικασία αυτή φαίνεται στις επόμενες εικόνες. Στο σημείο αυτό, έχοντας περιγράψει όλες τις λειτουργίες της εφαρμογής, ολοκληρώνεται και η παρουσίαση της διεπαφής χρήστη.



Σχήμα 108: Ο Χρήστης συμπληρώνει τη φόρμα ανάκτησης κωδικού



Σχήμα 109: Email ανάκτησης κωδικού



Σχήμα 110: Φόρμα Καταχώρησης Νέου Κωδικού

5.2 Αξιολόγηση

Από όλες τις παραπάνω λειτουργίες της Quantified Self εφαρμογής, προκύπτουν ορισμένα συμπεράσματα, αναφορικά με τη χρησιμότητα που η εφαρμογή μπορεί να έχει για τον χρήστη, καθώς και για την αξία των ιδιαίτερων χαρακτηριστικών της.

Καταρχάς, κυρίως λόγω της αρχιτεκτονικής που ακολουθήθηκε και των εργαλείων (NodeJS, Docker κτλ.) που επιλέχθηκαν για την ανάπτυξη, η εφαρμογή παρουσιάζει μεγάλη επεκτασιμότητα (Extendability). Είναι εύκολο, με άλλα λόγια, πάνω στο ήδη υπάρχον σύστημα, να προστεθούν επιπρόσθετες λειτουργίες, χωρίς να παρουσιαστούν προβλήματα συμβατότητας ή ορθής λειτουργίας. Ένα τέτοιο χαρακτηριστικό είναι, προφανώς, επιθυμητό καθώς οι κατηγορίες των δεδομένων που κανείς μπορεί να καταγράψει είναι αναρίθμητες, ενώ εξίσου πολλές είναι και οι δυνατότητες που δίνονται για ανάλυση των δεδομένων. Έτσι, οι απαιτήσεις που μπορεί να έχει ο χρήστης από το σύστημα συνεχώς θα αυξάνονται, οδηγώντας στην ανάγκη εύκολης, γρήγορης και ομαλής επέκτασης της εφαρμογής, ώστε οι απαιτήσεις αυτές να ικανοποιούνται.

Ένα δεύτερο αξιόλογο πλεονέκτημα της εφαρμογής αποτελεί και η ολιστική προσέγγιση της απέναντι στα δεδομένα του χρήστη. Συγκεκριμένα, το τεχνικό και λειτουργικό υπόβαθρό της, σε συνδυασμό, μάλιστα, και με την προαναφερθείσα επεκτασιμότητα της, επιτρέπουν την καταγραφή και ποσοτικοποίηση φοβερά ετερογενών δεδομένων, προερχόμενων από πολλές και διαφορετικές πηγές. Η συλλογή, όμως, τέτοιων δεδομένων οδηγεί σε μια συνολική αντιμετώπισή τους και, μέσα από την επεξεργασία τους, διευκολύνεται η δυνατότητα της εφαρμογής να σχηματίζει μαι σφαιρική εικόνα για την ιδιοπροσωπία του χρήστη, παρέχοντας, επομένως, και αντίστοιχα σφαιρικά συμπεράσματα και συμβουλές.

Επιπλέον, και σε άμεση συνάρτηση με τα δύο παραπάνω χαρακτηριστικά, η εφαρμογή παρουσιάζει αυξημένη προσαρμοστικότητα ή δυνατότητα εξατομίκευσης (Customization). Ακριβώς λόγω της ικανότητάς της να υποστηρίζει την επικοινωνία με ετερογενείς πηγές δεδομένων, καθώς και την επεξεργασία τους με διαφορετικούς τρόπους, βρίσκεται στο χέρι του χρήστη να επιλέξει τον ακριβή τρόπο με τον οποίο θα την χρησιμοποιήσει. Προσαρμόζοντας το σύστημα στις δικές του ιδιαίτερες ανάγκες και απαιτήσεις, χωρίς να χάνει, ωστόσο, σε λειτουργικότητα, ο χρήστης εξασφαλίζει μια εξατομικευμένη λειτουργία, η οποία και του παρέχει, κατά συνέπεια, αυξημένης ποιότητας υπηρεσίες.

Ακόμα, η λειτουργία της εφαρμογής στα πλαίσια του προγράμματος AGILE δίνει στον χρήστη απόλυτα δικαιώματα πάνω στα δεδομένα του. Συγκεκριμένα, με την εγκατάστασή της σε ένα προσωπικό, ιδιωτικό υπολογιστή Rasberry Pi, η εφαρμογή παύει να εξαρτάται από έναν κεντρικό εξυπηρετητή. Αντίθετα, ο έλεγχος, πλέον, βρίσκεται στα χέρια του χρήστη, ο οποίος μπορεί να επιλέξει να μοιραστεί ανώνυμα ορισμένα μετα-δεδομένα ή και να μην μοιραστεί καθόλου τα δεδομένα του. Σε μια εποχή, όπου τα δεδομένα αποτελούν σημαντικό συγκριτικό πλεονέκτημα, τόσο για άτομα, όσο και για οργανισμούς, ένα τέτοιο χαρακτηριστικό έχει ιδιαίτερη αξία.

Τέλος, σημαντικό στοιχείο της εφαρμογής είναι και οι τεχνικές που επιστρατεύει, έστω και σε αυτό το αρχικό στάδιο, ώστε να ωθήσει τον χρήστη σε μια περισσότερο θεμιτή, ως προς και τα εκάστοτε κριτήριά του, συμπεριφορά. Ειδικότερα, μέσω από το σύστημα στόχων, μηνυμάτων παρακίνησης ή επιβράβευσης, καθώς και μέσου του κοινωνικού στοιχείου, όπως αυτά περιγράφηκαν στα προηγούμενα κεφάλαια, η εφαρμογή στοχεύει στο να επιφέρει ορισμένες αλλαγές στη συμπεριφορά του χρήστη, που θα του επιτρέψουν να βελτιώσει τις επιδόσεις του.

Κεφάλαιο 6. Επίλογος

6.1 Σύνοψη

Στα πλαίσια της διπλωματικής εργασίας αναπτύχθηκε η εφαρμογή Quantified Self, η οποία εξυπηρετεί ένα χρήστη που ενδιαφέρεται να έχει πρόσβαση σε μια συγκεντρωτική καταγραφή και ανάλυση δεδομένων που αφορούν την υγεία και την ευεξία του. Τα σημαντικότερα χαρακτηριστικά της εφαρμογής είναι τα εξής:

- Η εφαρμογή παρέχει την δυνατότητα στον χρήστη να αποθηκεύσει μετρήσεις που αφορούν στα βιοσήματά του, συγκεκριμένα το σωματικό βάρος, την διάρκεια του ύπνου, την αρτηριακή πίεση, τον καρδιακό ρυθμό, τον κορεσμό του αίματος σε οξυγόνο και την συγκέντρωση της γλυκόζης στο αίμα.
- Η εφαρμογή επιτρέπει της διασύνδεσή της με δύο δημοφιλείς Activity Trackers, συγκεκριμένα το Fitbit και το GoogleFit. Από τις δύο αυτές πλατφόρμες, η εφαρμογή έχει πρόσβαση σε δεδομένα που αφορούν στη σωματική δραστηριότητα του χρήστη. Αυτή περιλαμβάνει τον αριθμό βημάτων που κάνει ο χρήστης ημερησίως, τις θερμίδες που καίει ημερησίως ο χρήστης και την χρονική διάρκεια ημερήσιας άσκησης του χρήστη.
- Εκτός από την καταγραφή των παραπάνω δεδομένων, η εφαρμογή τα οπτικοποιεί για τον χρήστη, ώστε μέσα από την οπτικοποίηση ο χρήστης να αποκτά εποπτεία των επιδόσεων του.
- Η εφαρμογή επιτρέπει στον χρήστη να ορίσει στόχους αναφορικά με τις επιδόσεις που θέλει να επιτύχει και, μέσω κατάλληλων μηνυμάτων, τον παρακινεί και τον επιβραβεύει.
- Η εφαρμογή αναλύει τα δεδομένα του χρήστη, προσφέροντάς του πολύτιμα συμπεράσματα σχετικά με την πορεία του στον χρόνο.
- Η εφαρμογή επιτελεί και μια συνολική ανάλυση των δεδομένων όλων των χρηστών, ώστε μέσα από τα αποτελέσματα της ανάλυσης αυτής, οι επιδόσεις του χρήστη τοποθετούνται σε ένα ευρύτερο πλαίσιο.
- Η εφαρμογή θα ενσωματωθεί στο ερευνητικό πρόγραμμα AGILE, αποτελώντας κομμάτι μιας στοίβας λογισμικού, που προορίζεται για χρήση σε υπολογιστή Rasberry Pi. Ως εκ τούτου, δεν θα υπάρχει ένας κεντρικός Server που θα είναι υπεύθυνος για όλους τους χρήστες. Αντίθετα, κάθε χρήστης που διαθέτει ένα Rasberry Pi, θα το χρησιμοποιεί ως έναν δικό του, προσωπικό Server, στον οποίο θα εγκαταστήσει την εφαρμογή. Έτσι, θα σχηματιστεί ένα κατανεμημένα σύστημα, μέσα στο οποίο κάθε χρήστης θα βρίσκεται, μεν, σε επικοινωνία με τους υπόλοιπους, διατηρώντας όμως, σε μεγάλο βαθμό, αποκλειστικά δικαιώματα στα δεδομένα του.

6.2 Μελλοντικές Επεκτάσεις

Εν κατακλείδι, θα γίνει μια αναφορά στις κατευθύνσεις που μπορεί να πάρει η περαιτέρω ανάπτυξη της εφαρμογής στο μέλλον.

Η πρώτη σημαντική επέκταση που θα μπορούσε να γίνει σχετίζεται με τις πηγές και τους τύπους δεδομένων που συλλέγει η εφαρμογή. Στην παρούσα φάση, τα δεδομένα αυτά είναι τα βιοσήματα, όπως αρτηριακή πίεση ή βάρος, και η σωματική δραστηριότητα (βήματα, θερμίδες και διάρκεια άσκησης). Η τεχνολογία σήμερα (και ακόμα περισσότερο στο μέλλον), όμως, παρέχει την δυνατότητα καταγραφής και ποσοτικοποίησης πολύ περισσότερων κατηγοριών δεδομένων, όπως η διατροφή ή η ψυχική διάθεση. Μάλιστα, στα πλαίσια του Internet of Things, η καταγραφή αυτή, ενδεχομένως, να γίνεται και αυτόματα, χωρίς τη διαμεσολάβηση του χρήστη. Η προσθήκη, λοιπόν, περισσότερων πηγών δεδομένων αποτελεί μια φυσική επέκταση της εφαρμογής.

Ταυτόχρονα, υπάρχουν σημαντικά περιθώρια επέκτασης της στατιστικής ανάλυσης που επιτελεί η εφαρμογή. Μεγέθη όπως η μέση τιμή και η μέγιστη τιμή σαφώς και αποτελούν χρήσιμες ενδείξεις για τον χρήστη, όμως η ανάλυση μπορεί να προχωρήσει παρακάτω, πόσο μάλλον, αν ληφθεί υπόψη και η πιθανή συλλογή περισσότερων και διαφορετικών δεδομένων, όπως αναφέρθηκε παραπάνω. Μάλιστα, η εφαρμογή θα μπορούσε να αξιοποιήσει μεθόδους και εργαλεία, που έχουν προκύψει από το πεδίο της Μηχανικής Μάθησης (Machine Learning), ώστε η ανάλυση των δεδομένων να γίνεται σε μεγαλύτερο βάθος.

Τέλος, ένας τρίτος άξονας μελλοντικής επέκτασης της εφαρμογής αφορά στον τρόπο με τον οποίο ο χρήστης καθορίζει στόχους προς επίτευξη και στο πως το σύστημα τον παρακινεί και επιβραβεύει. Συγκεκριμένα, θα μπορούσε να γίνει χρήση αποτελεσμάτων του χώρου της “Παιχνιδοποίησης” ή Gamification, την εφαρμογή, δηλαδή, στοιχείων και χαρακτηριστικών παιχνιδιών σε γενικότερα περιβάλλοντα, με στόχο την επίτευξη ορισμένων αποτελεσμάτων. Στα πλαίσια του Quantified Self, ένα σύστημα Gamification θα μπορούσε να αυξήσει τα επίπεδα συμμετοχής και δέσμευσης του χρήστη με την εφαρμογή, καθώς και να παρέχει σημαντική βοήθεια στον χρήστη, ώστε να βελτιώνει συνεχώς τις επιδόσεις του.

Βιβλιογραφία

[1] 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.

[2] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things

[3] 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.

[4] Pcmag.com: The Best Fitness Trackers of 2017, Available online at: http://www.pcmag.com/article2/0,2817,2404445,00.asp

[5] Wired.com: Know Thyself: Tracking Every Facet of Life, from Sleep to Mood to Pain, 24/7/365, Available online at: https://www.wired.com/2009/06/lbnp-knowthyself/

[6] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer

[7] MDN Web Docs: About Javascript, Available online at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[8] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/JSON

[9] W3Schools.com: Available online at: https://www.w3schools.com/js/js_json_intro.asp

[10] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/MEAN_(software_bundle)

[11] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Event-driven_programming

[12] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Node.js

[13] W3Schools.com: Available online at: https://www.w3schools.com/nodejs/nodejs_intro.asp

[14] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Web_framework

[15] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Application_programming_interface

[16] TutorialsPoint.com: Available online at: https://www.tutorialspoint.com/angularjs/index.htm

[17] Wikipedia, [Online]. Available:
https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[18] Wikipedia, [Online]. Available:
https://en.wikipedia.org/wiki/MongoDB

[19] Wikipedia, [Online]. Available:
https://en.wikipedia.org/wiki/Document-oriented_database

[20] PassportJS.org: Available online at:
http://passportjs.org/docs/overview

[21] OpenmHealth.org: Available online at:
http://www.openmhealth.org/

[22] OpenmHealth.org: Available online at:
http://www.openmhealth.org/documentation/#/schema-docs/overview

[23] OpenmHealth.org: Available online at:
http://www.openmhealth.org/documentation/#/data-providers/install-shimmer

[24] Google.com: Available online at:
https://www.google.com/fit/

[25] Fitbit.com: Available online at:
https://www.fitbit.com/uk/home

[26] Github.com: Available online at:
https://github.com/openmhealth/shimmer#supported-apis-and-endpoints

[27] AGILE-IOT: Available online at:
http://agile-iot.eu/

[28] Wikipedia, [Online]. Available:
https://en.wikipedia.org/wiki/Operating-system-level_virtualization

[29] Docker.com: Available online at:
https://www.docker.com/

[30] Codeship.com: Available online at:
https://resources.codeship.com/hubfs/resources_PDFs/Codeship_Why_Containers_and_Docker_are_the_Future.pdf?t=1452849717341

[31] Gitlab.com: Available online at:
https://about.gitlab.com/

[32] Wikipedia, [Online]. Available:
https://en.wikipedia.org/wiki/Git

[33] Thoughworks.com: Available online at:
https://www.thoughtworks.com/continuous-integration

[34] Thoughworks.com: Available online at:
https://www.thoughtworks.com/continuous-delivery

[35] Jetbrains.com: Available online at:
https://www.jetbrains.com/webstorm/

[36] Postman.com: Available online at:
https://www.getpostman.com/postman

[37] Okeanos.grnet: Available online at:
https://okeanos.grnet.gr/home/

[38] Wikipedia, [Online]. Available:
https://en.wikipedia.org/wiki/Chrome_V8

[39] MongooseJS.com: Available online at:
http://mongoosejs.com/

[40] Github.com: Available online at:
https://github.com/node-schedule/node-schedule

[41] Daniel Harrison, Paul Marshall, Nadia Bianchi-Berthouze, Jon Bird: "Activity Tracking: Barriers, Workarounds and Customisation". Ubicomp (2015)