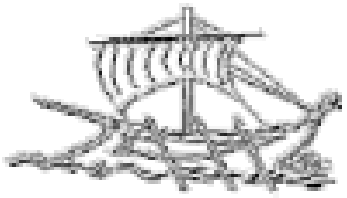


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πλατφόρμα λήψης, αποθήκευσης, επεξεργασίας και προβολής δεδομένων υγείας με χρήση έξυπνων πρακτόρων προγραμματισμού.

Μαστρογιαννόπουλος Νικόλαος, Κακύρης Αλέξανδρος

Εισηγητής: Δρ. Χαράλαμπος Πατρικάκης, Αναπληρωτής Καθηγητής

ΑΘΗΝΑ
Σεπτέμβρης 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πλατφόρμα λήψης, αποθήκευσης, επεξεργασίας και προβολής δεδομένων υγείας με χρήση έξυπνων πρακτόρων προγραμματισμού.

Μαστρογιαννόπουλος Νικόλαος
Α.Μ. 43206

Κακύρης Αλέξανδρος
Α.Μ. 43448

Εισηγητής:

Χαράλαμπος Πατρικάκης, Αναπληρωτής Καθηγητής

Εξεταστική Επιτροπή:

Χαράλαμπος Πατρικάκης, Αναπληρωτής Καθηγητής

Ημερομηνία εξέτασης:

[κενή σελίδα]

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογεγραμμένοι **Κακύρης Αλέξανδρος** & **Μαστρογιαννόπουλος Νικόλαος** με αριθμό μητρώου **43448** και **43206** αντίστοιχα, φοιτητές του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβουμε την εκπόνηση της Πτυχιακής Εργασίας μας, δηλώνουμε ότι ενημερωθήκαμε για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.
Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

[κενή σελίδα]

ΕΥΧΑΡΙΣΤΙΕΣ

Για την επιτυχημένη συγγραφή της εργασίας αυτής θα θέλαμε αρχικά να ευχαριστήσουμε τον κ. Πατρικάκη Χαράλαμπο για την άμεση και σημαντική υποστήριξη όποτε χρειάστηκε αλλά και για τα προσωπικά δεδομένα που μας προμήθευσε και αξιοποιήσαμε για την υλοποίηση.

Επίσης θέλουμε να ευχαριστήσουμε τις οικογένειες μας αλλά και τα κοντινά μας πρόσωπα για την συνεχή στήριξη κατά τη διάρκεια της πραγμάτωσης της εργασίας αυτής.

[κενή σελίδα]

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια μεγάλοι τεχνολογικοί και όχι μόνο κολοσσοί έχουν διανείμει στην αγορά τα smart watches ή αλλιώς wearables ή πολύ πιο απλά τα έξυπνα ρολόγια. Μια από αυτές τις εταιρείες είναι και η Fitbit η οποία έχει λανσάρει μια σειρά από τέτοιες συσκευές.

Η εργασία μας λοιπόν θα αξιοποιήσει τις υπηρεσίες που προσφέρει η εταιρεία αυτή και τα προϊόντα της και συγκεκριμένα το “Charge HR 101” το οποίο συλλέγει δεδομένα που αφορούν τη διατροφή του χρήστη, την καθημερινή του άσκηση, παρακολουθεί τους παλμούς της καρδιάς του καθώς και την ποιότητα του ύπνου του. Συλλέγει όλα αυτά καθημερινά χωρίς διακοπή, τα αποθηκεύει και έπειτα δίνει τη δυνατότητα περισυλλογής αυτών των δεδομένων. Αυτό που κάνουμε λοιπόν είναι να επικοινωνούμε με τον server του Fitbit και να λαμβάνουμε τα δεδομένα του χρήστη με ημερομηνία εκκίνησης τον Δεκέμβρη του 2014, τα αξιοποιούμε, τα επεξεργαζόμαστε και τελικώς τα παρουσιάζουμε με μορφή φιλική προς τον χρήστη.

Τέλος η εφαρμογή δίνει τη δυνατότητα στον χρήστη να λάβει τα δεδομένα σε pdf μορφή, να λάβει ένα e-mail με δεδομένα καρδιάς αλλά και να ανακατευθυνθεί σε μια πλατφόρμα αναλυτικής παρουσίασης των δεδομένων του.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ΔΕΔΟΜΕΝΑ ΥΓΕΙΑΣ, ΕΞΥΠΗΡΕΤΗΤΗΣ, ΔΙΑΔΙΚΤΥΑΚΗ ΕΠΙΚΟΙΝΩΝΙΑ, REST, ΟΠΤΙΚΟΠΟΗΣΗ ΔΕΔΟΜΕΝΩΝ, ΕΞΥΠΝΕΣ ΣΥΣΚΕΥΕΣ

ABSTRACT

In recent years, great technology and not only, giants have been distributing smart watches or wearables or just smart watches to the market. One of these companies is Fitbit, which has launched such devices.

Our work will therefore make use of the services offered by this company and its products, more specifically we use the "Charge HR 101", which collects data on the user's diet, his daily exercise, monitors his heartbeats as well as his quality of his sleep. It collects them every time the user uses the, stores them, and then gives the opportunity to make use of these data. So what we do is to communicate with the Fitbit server and get the user's data with a start date in December 2014, use them, format them, and finally present them in a user-friendly format.

Finally, the application enables the user to download the downloaded data in pdf format, receive an e-mail with user’s heart data, and redirect him to a data presentation platform-site.

SCIENTIFIC FIELD: WEB APPLICATION DEVELOPEMENT
KEY WORDS: HEALTH DATA, WEB SERVER, REST CONTROLLER, DATA PRESENTATION, SMART DEVICES

ΠΕΡΙΕΧΟΜΕΝΑ

1.ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας και γίνεται μια σύντομη περιγραφή της δομής των θεμάτων που εξετάζει.

1.1 Αντικείμενο

Από τότε που ο άνθρωπος ανακάλυψε και αναγνώρισε την έννοια του χρόνου αποφάσισε να προσαρμόσει τα πάντα πάνω σε αυτόν. Αρχικά τον όριζε με την συμπεριφορά της φύσης, την ημέρα κυκλοφορούσε, κυνηγούσε και εξερευνούσε και μετά τη δύση του ηλίου κοιμόταν και κρυβόταν. Με την δύση του ηλίου επίσης σταματάγανε και οι μάχες πολύ αργότερα και με την ανατολή ξανά ξεκινούσαν. Ο ήλιος επίσης έπαιξε σημαντικό ρόλο και στην χρήση του ηλιακού ρολογιού το οποίο όριζε την ώρα ανάλογα τη θέση του και τη σκιά. Από τα ηλιακά ρολόγια πολύ αργότερα φτάσαμε στους δείκτες και στα ξύλινα, στα μεταλλικά και από τη θέση του στον τοίχο το ρολόι κέρδισε και τη θέση του στο χέρι του ανθρώπου και έγινε το αγαπημένο του αξεσουάρ. Έγινε ακόμα και σύμβολο μιας ολόκληρης πόλης και έθνους, αλλά έχει ξεχωριστή θέση στους κατοίκους κάθε πόλης καθώς πολλές φορές αποτελεί σημείο συνάντησης. Πλέον κάποια ρολόγια μπορεί να έχουν αξία μεγαλύτερη και ενός διαμερίσματος και ενός αυτοκινήτου όχι λόγω της χρήσης τους αλλά του υλικού κατασκευής τους. Ωστόσο και η χρήση τους ακόμα έχει αλλάξει καθώς εν έτι 2017 ο κόσμος μας έχει εισέλθει για τα καλά στην ψηφιακή εποχή την οποία ούτε τα ρολόγια όλου του κόσμου δεν αρκούν για να την προλάβουμε. Ο άνθρωπος πλέον έχει και την πολυτέλεια να προσαρμόζει στην τεχνολογία τα πάντα γύρω του ακόμα και στα ρολόγια του. Τα ρολόγια πλέον εκτός από την ώρα μπορεί να παρέχουν και δεκάδες ακόμα υπηρεσίες στην ιδιοκτήτη τους.

Έτσι φτάσαμε λοιπόν στα έξυπνα ρολόγια (smart watches – wearables) τα οποία έχουν τη δυνατότητα να συλλέγουν πληροφορίες από το περιβάλλον τους και τον χρήστη τους ο οποίος μπορεί να χρησιμοποιήσει αυτές τις υπηρεσίες προς όφελος του. Οι πληροφορίες αυτές αποτελούν προσωπικές πληροφορίες για τον χρήστη και ένα μεγάλο εύρος δεδομένων που αφορούν την φυσική του κατάσταση. Η πλειοψηφία των συσκευών αυτών αξιοποιώντας στο μέγιστο την πρόοδο της τεχνολογίας προσφέρουν ακόμα και συμβουλευτικές υπηρεσίες όσον αφορά τη υγεία και την κατάσταση του χρήστη από τον οποίο συλλέγουν δεδομένα συνεχώς.

Τα δεδομένα λοιπόν αυτά στέλνονται σε βάσεις δεδομένων της κάθε εταιρείας ανά τον κόσμο και αποθηκεύονται με ασφάλεια εκεί. **Στόχος** δικός μας λοιπόν είναι να δημιουργήσουμε μια εφαρμογή που θα συλλέγει και θα αξιοποιεί τα δεδομένα του χρήστη από τον Δεκέμβριο του 2014 μέχρι και σήμερα και θα του τα παρουσιάζει με μεθόδους και τρόπους φιλικούς προς αυτόν χρησιμοποιώντας το προϊόν και τις υπηρεσίες της εταιρείας Fitbit αλλά και του προγραμματισμού. Μια εταιρεία η οποία είναι από τις πρώτες που υποστήριξαν αυτά τα προϊόντα όπως και υπηρεσίες αρκετά προχωρημένες όπως τη δυνατότητα για Publish – Subscribe (εκτενέστερη αναφορά στον επίλογο).

1.2 Δομή

Στην αρχή η πτυχιακή μας εργασία κάνει μια εισαγωγή στον ψηφιακό κόσμο και στο πως η τεχνολογία πλέον αποτελεί αναπόσπαστο κομμάτι της καθημερινής μας ζωής και το πώς έχει μπει μέσα στα σπίτια μας. Έπειτα επικεντρωνόμαστε αποκλειστικά στα προϊόντα της Fitbit και αναφέρουμε σημαντικούς ανταγωνιστές της.

Ακολουθεί μια μικρή αναφορά στη δική μας πλατφόρμα και στη συνέχεια εισερχόμαστε στο κομμάτι του προγραμματισμού της εφαρμογής μας με εκτενή αναφορά και στα δύο μέρη της, αυτό της λήψης των δεδομένων και αυτό της προβολής τους. Αφού επεξηγηθεί σε βάθος η πραγμάτωση της εφαρμογής σε όλα τα επίπεδα ακολουθεί μια παρουσίαση των δυνατοτήτων που προσφέρει στον χρήστη βήμα - βήμα.

Τέλος γίνεται μια αναφορά στις δυσκολίες που αντιμετωπίσαμε κατά τη διάρκεια της υλοποίησης της εργασίας τον τελευταίο χρόνο καθώς και στην αντιμετώπιση τους. Καταλήγοντας σημειώνουμε τις βελτιώσεις που μπορεί να δεχθεί ολόκληρη η εφαρμογή και στα δύο μέρη της σύμφωνα με την δική μας οπτική αλλά και τις υποδείξεις του επιβλέποντα καθηγητή μας.

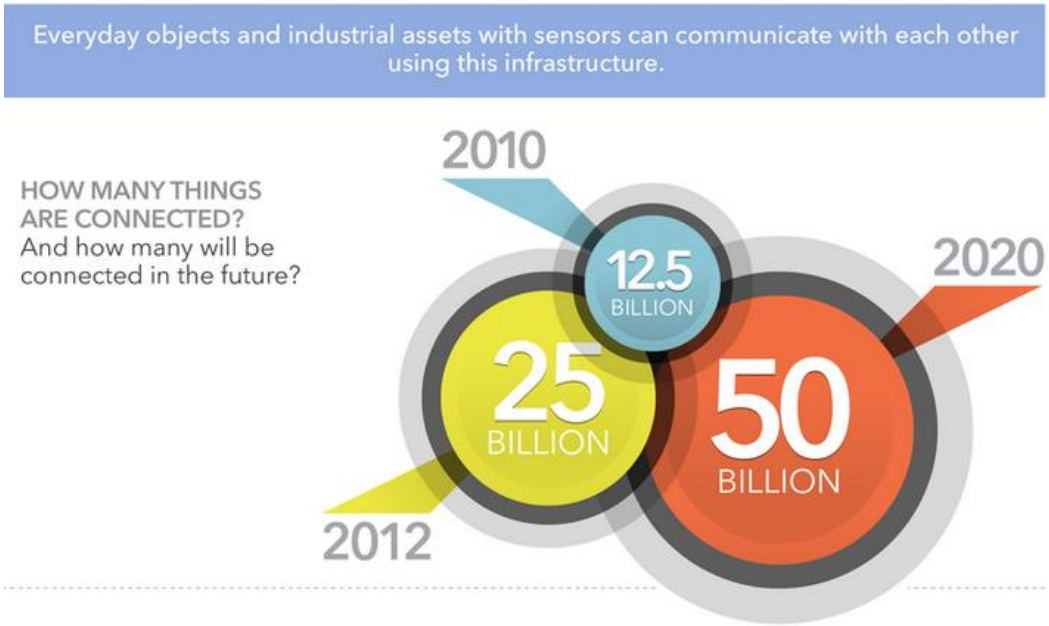
2.ΕΞΥΠΝΕΣ ΣΥΣΚΕΥΕΣ

2.1 Από τα έξυπνα τηλέφωνα (smart phones) στις έξυπνες πόλεις (smart cities)

Η εξέλιξη των τεχνολογικών προϊόντων και των υπηρεσιών που προσφέρουν είναι ραγδαία τα τελευταία χρόνια και θα είναι ακόμη πιο εντυπωσιακή τα επόμενα. Οι διαφορές σε κλίμακα τριακονταετίας ακόμη και εικοσαετίας είναι τεράστιες όσον αφορά την ποιότητα και ταχύτητα ζωής του ανθρώπου. Τα **smart phones** και οι υπηρεσίες που προσφέρουν πλέον είναι συνηθισμένα και η εξέλιξη έχει εισχωρήσει και σε άλλες συσκευές.

Η τεχνολογία έχει πλέον κάνει τη ζωή μας «εξυπνότερη» και μέσω του ίντερνετ και των μεγάλων ταχυτήτων του μπορούμε να εκμεταλλευτούμε άπειρες ευκαιρίες διευκόλυνσης της ζωής μας. Είμαστε στο 2017 και πλέον όλες μας οι συσκευές μπορούν να είναι “smart” και μέσω αυτών να έχουμε τον πλήρη έλεγχο..

Από το κινητό μας και την τηλεόρασή μας μέχρι και το αυτοκίνητο μας και το ίδιο μας το σπίτι. Και από το ίδιο μας το σπίτι μέχρι και την ίδια μας την πόλη (**smart cities**), η τεχνολογία έχει εισέλθει στα πάντα γύρω μας τα οποία μπορούν να αλληλοεπιδρούν με τέτοιο τρόπο που κανένας δεν φανταζόταν μέχρι πριν λίγα μόνο χρόνια.

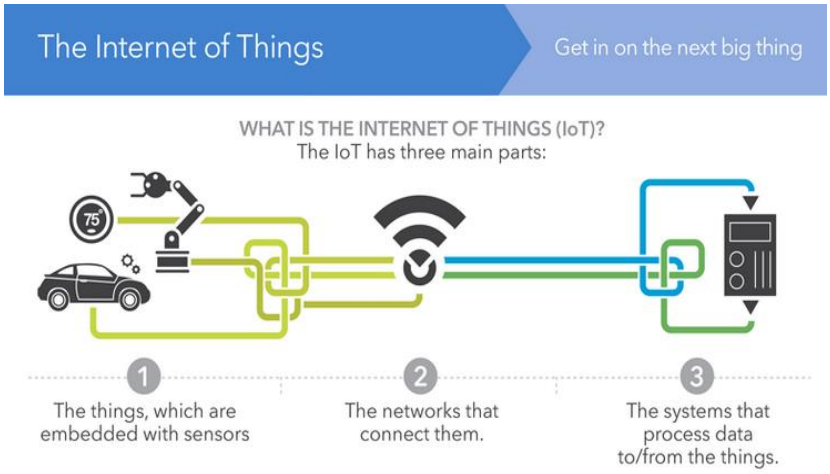


Εικόνα 1: Things connected each other (πηγή: sas.com)

2.2 Διαδίκτυο των πραγμάτων ή αλλιώς Internet of things (IoT)

Το **Internet of Things** είναι μία έννοια που αφορά τα αντικείμενα της καθημερινότητας μας – από βιομηχανικές μηχανές μέχρι wearable συσκευές που χρησιμοποιούν ενσωματωμένους αισθητήρες για τη συλλογή δεδομένων & την ανάληψη κάποιας δράσης σε αυτά μέσα σε ένα δίκτυο.

Κάπως έτσι λειτουργεί ένα κτίριο που χρησιμοποιεί αισθητήρες (sensors) για την αυτόματη ρύθμιση της θέρμανσης ή του φωτισμού. Άλλο παράδειγμα είναι ο ένας εξοπλισμός παραγωγής που προειδοποιεί το προσωπικό συντήρησης για μία επικείμενη βλάβη. Με απλά λόγια το Internet of Things είναι το τεχνολογικό μέλλον που θα κάνει τη ζωή μας πιο εύκολη.

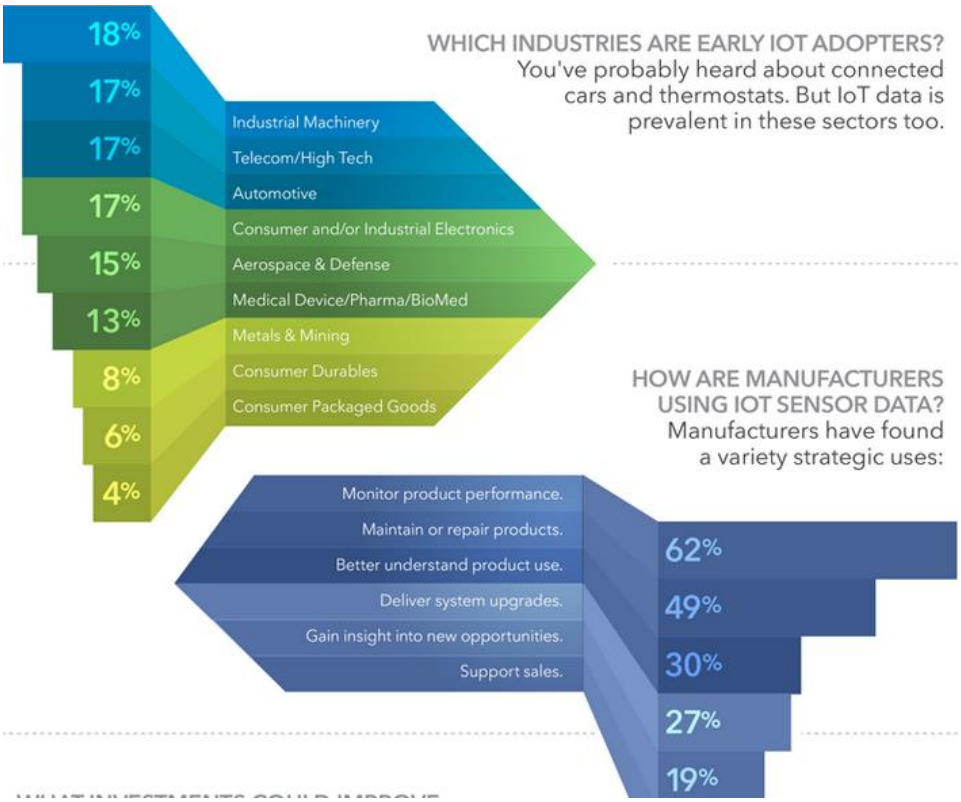


Εικόνα 2: 3 parts of IoT (πηγή: sas.com)

Το μέλλον φαίνεται να είναι λαμπρό όσον αφορά τον τεχνολογικό τομέα τουλάχιστον καθώς πέρα από την διευκόλυνση που προσφέρει το IoT στον άνθρωπο, μπορεί να προσφέρει ακόμα περισσότερα στην ίδια την ανθρωπότητα.

Στην εικόνα καταγράφονται κάποιοι από τους τομείς που εκμεταλλεύονται στο έπακρο αυτήν την υπηρεσία. Τομείς όπως η υγεία, η ασφάλεια, η ενέργεια είναι τομείς που χρησιμοποιώντας την γρηγορότερη και ασφαλέστερη μεταφορά δεδομένων , την ανάλυση και ανάλυση δεδομένων αλλά και την πρόβλεψη των επόμενων μπορούν να βελτιωθούν ραγδαία.

Η επιστήμη μπορεί πλέον να ελπίζει σε μεγάλα πράγματα στο ένα τρίτο του χρόνου που θα έκανε χωρίς το IoT και η ανθρωπότητα να μπει μετά από πολύ καιρό σε τροχιά υγειούς ανάπτυξης. Δεδομένα υγείας και ενέργειας μπορούν να εκτιμηθούν καλύτερα σωστότερα και γρηγορότερα με αποτελέσματα εμφανή και με διάρκεια.



Εικόνα 3: Industries adopt IoT (πηγή: sas.com)

Συνολικά, τα επόμενα χρόνια αναμένεται μία έξαρση του αριθμού των συνδεδεμένων συσκευών, των τοποθεσιών που αυτές βρίσκονται και φυσικά των λειτουργιών που αυτές θα εκτελούν. Ενδεικτικά μπορούμε να αναφέρουμε τα μελλοντικά νοσοκομεία: πέρα από τις standalone συνδεδεμένες συσκευές θα υπάρχουν πληθώρα συσκευών οι οποίες θα βρίσκονται συνδεδεμένες με τους σταθμούς παρακολούθησης ασθενών του νοσηλευτικού προσωπικού.

Σε βιομηχανικές εγκαταστάσεις, όπου απαιτείται η συνεχής παρακολούθηση της ροής των χρησιμοποιούμενων υλικών ώστε να αυξάνεται η παραγωγικότητα. Αισθητήρες προσδιορισμού θέσης θα είναι τοποθετημένοι στα υλικά που κινούνται πάνω σε μία γραμμή παραγωγής και που στη συνέχεια αποθηκεύονται. Οι ίδιοι αισθητήρες μπορούν να βρίσκονται σε περονοφόρα ανυψωτικά μηχανήματα, σε παλέτες και σε εργαζόμενους ώστε μέσω ενός κεντρικά διαχειριζόμενου λογισμικού να δίνονται οδηγίες σε πραγματικό χρόνο.

Σε σπίτια, γραφεία και λοιπούς χώρους εργασίας, αισθητήρες θα μπορούν να παρακολουθούν τα δίκτυα κοινής ωφέλειας και να προσφέρουν έγκαιρη προειδοποίηση σε περίπτωση πτώσης τους ηλεκτρικού ρεύματος, διαρροής

νερού και υπερφόρτωσης του ηλεκτρικού δικτύου. Τα δεδομένα που θα συγκεντρώνονται θα μπορούν να χρησιμοποιηθούν για τη βελτίωση της απόδοσης, να εντοπίζουν ανάγκες και να προβλέπουν ειδικές απαιτήσεις. Χαρακτηριστική είναι η περίπτωση του Όσλο, όπου με τέτοιου είδους έξυπνες λύσεις επιτεύχθηκε μείωση στο κόστος ενέργειας κατά 62%.

Στο ίδιο μήκος κύματος κινούνται και οι χρήσεις των γνωστών σε όλους μας **wearables**. Αισθητήρες παρακολούθησης των καρδιακών παλμών σε συνδυασμό με εφαρμογές κινητών που μετρούν βήματα και αποστάσεις που έχουμε διανύσει είναι από τις πιο κοινές εφαρμογές του «καταναλωτικού» Internet of Things.

2.3 Αξεσουάρ Fitbit (Fitbit Wearables)

Η **Fitbit** λοιπόν είναι μια εταιρεία που ίδρυσαν οι κύριοι Eric and James, οι οποίοι αποφάσισαν να επενδύσουν σε αυτήν την εκθετικά αναπτυσσόμενη τεχνολογία από το 2007 ήδη. Συνειδητοποίησαν ότι η αισθητήρες και η ασύρματη τεχνολογία είναι το μέλλον και δημιούργησαν προϊόντα πρωτοποριακά για την εποχή και απολύτως χρήσιμα για την καθημερινότητα μας. Αποστολή τους όπως υποστηρίζουν είναι να εμπνεύσουν για μια υγιέστερη ζωή και καθημερινότητα σχεδιάζοντας προϊόντα που ταιριάζουν απόλυτα στον χρήστη και τον βοηθούν να καταφέρνει τους στόχους του. Ξεκίνησαν από το San Francisco και πλέον έχουν επεκταθεί σχεδόν σε όλο τον κόσμο απασχολώντας χιλιάδες υπαλλήλους και εξυπηρετώντας χιλιάδες πελάτες ανά τον κόσμο. Τα προϊόντα τους είναι κατά κόρον wearables και συλλέγουν δεδομένα του χρήστη για την υγεία του και το σώμα του βοηθώντας τον να ζει καλύτερα, υγιέστερα και με ασφάλεια. Κάποια από αυτά φαίνονται στο site της εταιρείας όπως και αναλυτικές πληροφορίες για τα παραπάνω. Το παρακάτω είναι το wearable που εμείς αξιοποιούμε τα δεδομένα που συλλέγει από τον χρήστη.

Εικόνα 4: The Fitbit wearable we used (πηγή: google.com)

3.ΠΑΡΟΜΟΙΕΣ ΠΛΑΤΦΟΡΜΕΣ - ΕΦΑΡΜΟΓΕΣ

Πλέον οι περισσότερες μεγάλες εταιρείες τεχνολογίας και όχι μόνο έχουν λανσάρει προϊόντα smart wearables. Apple, Samsung, LG, Motorola αλλά και η Nike είναι κάποιες από τις εταιρείες αυτές που επενδύουν σε αυτές τις συσκευές και μάλιστα με μεγάλη ανταπόκριση καθώς οι χρήστες εμπιστεύονται αυτές τις συσκευές και τις υπηρεσίες που προσφέρουν. Όμως η Fitbit βρίσκεται σταθερά στις λίστες με τις προτιμώμενες εταιρείες για προϊόντα smartwatches και fitness trackers.

3.1 Η σημασία της προβολής (visualization)

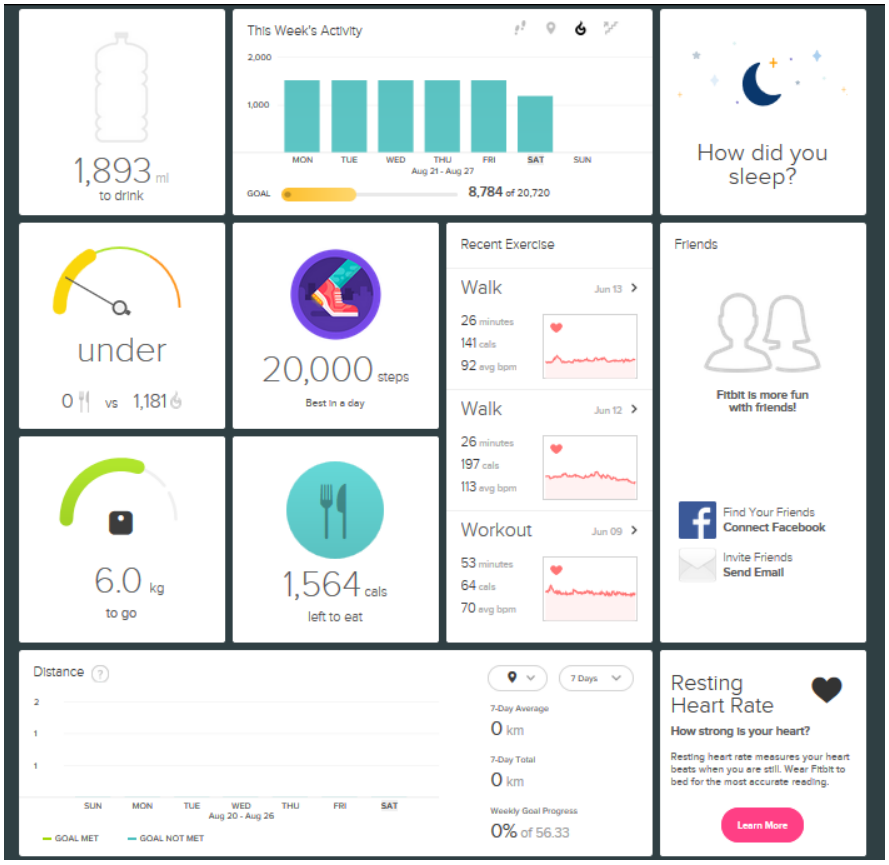
Όλες λοιπόν οι εταιρείες αυτές προσφέρουν στον χρήστη των προϊόντων πλατφόρμες ή και εφαρμογές με αναλυτική παρουσίαση των δεδομένων που συλλέγουν. Γραφήματα, υπολογιστικά στοιχεία, συμβουλές και συνολικά δεδομένα προσφέρουν στον χρήστη μια υπηρεσία άκρως φιλική και προσιτή.

Άλλωστε αυτό απαιτεί και ο ίδιος γιατί πέραν των μετρήσεων και των αριθμών των δεδομένων ο χρήστης ικανοποιείται βλέποντας όλα αυτά τα δεδομένα που συλλέγονται σε διαγράμματα, σχήματα και εικόνες.

Ο χρήστης μπορεί να δει με ακρίβεια και με ευκολία ότι επιθυμεί ο ίδιος κάνοντας συγκρίσεις υπολογισμούς, βγάζει συμπεράσματα και βάζει στόχους για το μέλλον όσον αφορά την υγεία του και το σώμα του. Υπάρχει συνεπώς διαρκής επικοινωνία του χρήστη με τα δεδομένα.

3.2 Πλατφόρμα Fitbit

Η Fitbit προσφέρει αυτή την υπηρεσία στους χρήστες της. Έχει δημιουργήσει μια online πλατφόρμα παραπάνω από ικανοποιητική που προσφέρει την δυνατότητα προβολής των δεδομένων όπως ακριβώς περιγράψαμε παραπάνω.



Εικόνα 5: Fitbit users online platform (πηγή: fitbit.com)

3.3 Η δική μας εφαρμογή – πλατφόρμα

Η δική μας εφαρμογή, με την συμπλήρωση των στοιχείων του ίδιου του χρήστη, λαμβάνει τα δεδομένα από το API που έχει δημιουργήσει η Fitbit και επιτρέπει σε third party εφαρμογές να επικοινωνήσουν μαζί της και να τα λάβουν. (part1 της εργασίας)

Έπειτα αφού τα αποθηκεύσει σε μια βάση δεδομένων τα προσφέρει στην πλατφόρμα που έχουμε δημιουργήσει. Η πλατφόρμα μας παρουσιάζει με διαγράμματα ημέρας αλλά και μήνα δεδομένα καρδιάς, ύπνου και δραστηριοτήτων. Ενώ παρουσιάζει και προβάλει και πληροφορίες του χρήστη. (part2 της εργασίας)

Ο οποιοσδήποτε χρήστης μπορεί με τα credentials του από το site και τις υπηρεσίες του Fitbit να χρησιμοποιήσει την εφαρμογή μας και να περιηγηθεί έπειτα στην online πλατφόρμα. Εννοείται ότι οφείλει να είναι κάτοχος κάποιου Fitbit wearable.

4.ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΦΑΡΜΟΓΗΣ

Ο προγραμματισμός της εφαρμογής μας χωρίζεται σε δύο μέρη. Το πρώτο μέρος αφορά το κομμάτι της λήψης των δεδομένων από το API της Fitbit, την επεξεργασία τους και τέλος την αποθήκευσή τους στην βάση δεδομένων. Το δεύτερο μέρος αφορά την αξιοποίηση των δεδομένων αυτών και την προβολή τους στον χρήστη.

4.1 Part 1 – Λήψη επεξεργασία και αποθήκευση δεδομένων

Όπως πολλές εταιρείες πλέον έτσι και η Fitbit προσφέρει εκτός από τα προϊόντα της, και την ευκαιρία στους προγραμματιστές και όχι μόνο να αξιοποιήσουν τα χιλιάδες δεδομένα που αποθηκεύονται κάθε δευτερόλεπτο στις βάσεις δεδομένων της. Έχει θέσει λοιπόν σε λειτουργία και προς εκμετάλλευση το API της, δηλαδή την δυνατότητα λήψης και όχι μόνο των δεδομένων του εκάστοτε χρήστη με τα credentials του αντίστοιχου χρήστη.

Δίνει την δυνατότητα λοιπόν, με χρήση κάποιων κωδικών και εφαρμογών που ο ίδιος ο χρήστης ορίζει στον λογαριασμό του, σε εφαρμογές να επικοινωνούν άμεσα με το API αυτό. Βέβαια για να επιτευχθεί αυτή η επικοινωνία και ανταλλαγή δεδομένων απαιτούνται κάποιες διαδικασίες ταυτοποίησης καθώς τα δεδομένα αποτελούν προσωπικά δεδομένα και μάλιστα υγείας, οπότε δεν μπορεί να επιτραπεί στον καθένα ή στην οποιαδήποτε εφαρμογή να τα λάβει τόσο εύκολα, εξ ου και οι αυστηρές αυτές διαδικασίες οι οποίες θα περιγραφούν στην συνέχεια.

4.1.1 Λήψη

Ας ξεκινήσουμε λοιπόν με το κομμάτι της λήψης των δεδομένων από το API της Fitbit, ίσως και το δυσκολότερο αυτής της διαδικασίας.

Κάθε API προστατεύεται και δεν αφήνει οποιονδήποτε και οτιδήποτε να επικοινωνήσει μαζί του και να το αξιοποιήσει. Η προστασία αυτή ονομάζεται OAuthentication και έχει πολλές μορφές και λειτουργίες, άλλες πολύ απλές όπως ένα απλό Login με τα credentials του εκάστοτε χρήστη και άλλες αρκετά περίπλοκες και αυστηρότερες όπως το OAuth2 με χρήση JWT token που είχαμε να αντιμετωπίσουμε εμείς.

Ο χρήστης λοιπόν καλείται να επικοινωνήσει με το API και να στείλει κάποια στοιχεία απαραίτητα για την συνέχεια. Το API εφόσον τα στοιχεία που έστειλε είναι σωστά τον ανακατευθύνει σε μια Login page που συμπληρώνει τα στοιχεία του λογαριασμού του.

Ολοκληρώνοντας σωστά το Login το API στέλνει στον χρήστη ένα κωδικό – authorization code το οποίο ο χρήστης στέλνει ξανά στο API για να λάβει το token code. Το token πρέπει να περιλαμβάνεται από εδώ και πέρα σε οποιοδήποτε αίτημα στο API ώστε ο χρήστης να είναι authorized με το σύστημα του API.

Βέβαια το token αυτό έχει ισχύ για κάποιο χρονικό διάστημα. Μετά το πέρας του διαστήματος αυτού ο χρήστης πρέπει να στείλει το refresh token που είχε λάβει μαζί με το token ώστε να ενεργοποιήσει καινούριο token ξανά.

Με την χρήση λοιπόν του token σε κάθε κλήση στο API της Fitbit πραγματοποιήθηκαν όλες οι λήψεις των data διαφόρων κατηγοριών όπως sleep data, heart data, activities data και κάποιες γενικού περιεχομένου όπως profile data, frequency activities data & lifetime activities data. Η λήψη επιτεύχθηκε με rest http calls (πλήρης επεξήγηση σε επόμενη παράγραφο) όπως επίσης και η διαδικασία του authorization της εφαρμογής μας που αναφερθήκαμε παραπάνω.

4.1.2 Επεξεργασία

Με την πετυχημένη ολοκλήρωση της διαδικασίας του authorization πλέον μπορούμε να αρχίσουμε να λαμβάνουμε τα data. Το API στέλνει τις απαντήσεις οι οποίες πέραν των διαφόρων πληροφοριών που θα αναφέρουμε παρακάτω (όπως πχ τα headers), περιλαμβάνουν και data. Η εφαρμογή μας λοιπόν με τη χρησιμοποίηση μιας βιβλιοθήκης που ονομάζεται Jackson κάνει deserialize τα data και τα μετατρέπει σε json format κατάλληλα για διαχείριση και επεξεργασία. Ένα μέρος της επεξεργασίας των data από τα responses περιλαμβάνει και κάποιες μικρές τροποποιήσεις όσον αφορά τη δομή των κάποιων data για μελλοντική διευκόλυνση της αξιοποίησης τους.

Τέλος εφαρμόζουμε μια τελευταία μετατροπή όσον αφορά τη μορφή τους ώστε να είναι κατάλληλη και συμβατή με τις απαιτήσεις της βάσης δεδομένων για αποθήκευση.

4.1.3 Αποθήκευση

Για την αποθήκευση των δεδομένων, που επιτυχώς λαμβάνουμε μετά και την επιτυχή ολοκλήρωση της διαδικασίας του authorization, χρησιμοποιούμε μια NOSQL database την MongoDB (πλήρης αναφορά σε επόμενη παράγραφο). Τα δεδομένα αποθηκεύονται στα εκάστοτε collections (αντίστοιχα των tables στις SQL databases) ανάλογα το είδος τους. Έχουμε τις εξής κατηγορίες :

- Sleep data που περιλαμβάνει collections σχετικά με τη διάρκεια ύπνου του χρήστη, την ποιότητα και τις διαφορετικές φάσεις στην πορεία του ύπνου. Κάθε collection περιλαμβάνει documents με values (σε λεπτά) ανά date (της μορφής YYYY/mm/DD).

- Activities data που περιλαμβάνει τις δραστηριότητες του χρήστη μέσα στην ημέρα όπως πλήθος βημάτων - steps, απόστασης - distance (σε μέτρα) ή και ορόφων - floors.

- Heart - rate data που περιλαμβάνει τους παλμούς της καρδιάς του χρήστη. Περιλαμβάνει τέσσερις ζώνες – επίπεδα και τα λεπτά μέσα στην ημέρα που οι παλμοί βρίσκονταν σε αυτό το επίπεδο. Τα επίπεδα είναι:

    1. Out of range με ελάχιστο όριο τους 30 παλμούς και μέγιστο τους 70.
    2. Fat burn με ελάχιστο όριο τους 70 παλμούς και μέγιστο τους 98.
    3. Cardio με ελάχιστο όριο τους 98 παλμούς και μέγιστο τους 119.
    4. Peak με ελάχιστο όριο τους 119 παλμούς και μέγιστο τους 220.

- Profile data που περιλαμβάνει πληροφορίες και δεδομένα για τον ίδιο τον χρήστη όπως παράδειγμα το πλήρες όνομά του μαζί με άλλα χρήσιμα στοιχεία, πληροφορίες για το ύψος και το βάρος του και πολλά άλλα.

- Lifetime activities data & Frequence activities data. Δύο ξεχωριστά collections που το πρώτο περιλαμβάνει τις καλύτερες επιδόσεις από τα activities γενικότερα αλλά και το σύνολο όλων των τιμών των activities. Το δεύτερο περιλαμβάνει τις συνήθεις δραστηριότητες του χρήστη.

4.1.4 Άλλες υπηρεσίες

Η εφαρμογή μετά και την αποθήκευση των δεδομένων προσφέρει δύο υπηρεσίες στον χρήστη. Η μία είναι να λάβει ένα email με περιεχόμενο ημερομηνίες και values με δεδομένα καρδιάς του χρήστη της κατηγορίας που θέλει για το χρονικό διάστημα που είχε επιλέξει νωρίτερα να λάβει δεδομένα.

Η άλλη υπηρεσία είναι η δυνατότητα να κατεβάσει ένα pdf αρχείο με όλα τα δεδομένα χρήστη που υπάρχουν στη βάση όσον αφορά τις κατηγορίες activities, sleep και heart-rate με την προϋπόθεση να έχουν επιλεγεί κατά τη λήψη νωρίτερα.

4.1.5 Τεχνολογίες και Εργαλεία που χρησιμοποιήθηκαν

Γενικότερες τεχνολογίες και έννοιες

Rest Web Services – Soap Web Services

Ένα web service μπορεί να οριστεί με τους παρακάτω τρόπους:

- Είναι μια client server εφαρμογή ή ένα συστατικό εφαρμογής για επικοινωνία
- Μια μέθοδος/τρόπος επικοινωνίας μεταξύ δύο συσκευών η εφαρμογών μέσω δικτύου
- Είναι μια εφαρμογή λογισμικού για διαλειτουργική επικοινωνία μεταξύ δυο μηχανών
- Είναι μια συλλογή από πρωτόκολλα για ανταλλαγή πληροφοριών μεταξύ δυο συσκευών/εφαρμογών

Τα web services χωρίζονται σε δύο κατηγορίες:

Soap web Services

Soap ή αλλιώς Simple Object Access Protocol. Είναι ένα **XML** based πρωτόκολλο που είναι ανεξάρτητο από τις γλώσσες προγραμματισμού και επιτρέπει την διαδραστική επικοινωνία μεταξύ των εφαρμογών/συσκευών.

- Τα **θετικά** του είναι ότι προσφέρει ένα αρκετά ικανοποιητικό σύστημα ασφαλείας το WS και ότι δεν εξαρτάται από κάποια γλώσσα προγραμματισμού.
- Τα **αρνητικά** του είναι ότι απαιτεί αρκετά resources για την χρήση του καθώς λόγω XML και WSDL πρέπει να τηρούνται κάποια στάνταρ και η προσέγγισή τους είναι ιδιαίτερη.

Rest web services

Rest ή αλλιώς REpresentational State Transfer. Είναι ένα είδος αρχιτεκτονικής και όχι πρωτόκολλο αυτό καθαυτό.

Στα **θετικά** μπορούμε να συμπεριλάβουμε ασφαλώς την ταχύτητα καθώς δεν περιλαμβάνει τόσο σφιχτές απαιτήσεις. Επίσης είναι ανεξάρτητο από γλώσσες προγραμματισμού και αυτό και τέλος υποστηρίζει διάφορες μορφές τύπου δεδομένων όπως **JSON**, Plain Text, XML, HTML κ.α.

HTTP 1.1

Ένα από τα βασικά χαρακτηριστικά των Rest Web Services είναι η υποστήριξη των HTTP requests. Η Rest αρχιτεκτονική απευθύνεται σε One-to-one αντιστοιχία με τέσσερις βασικές διαδικασίες CRUD create, read, update, delete.

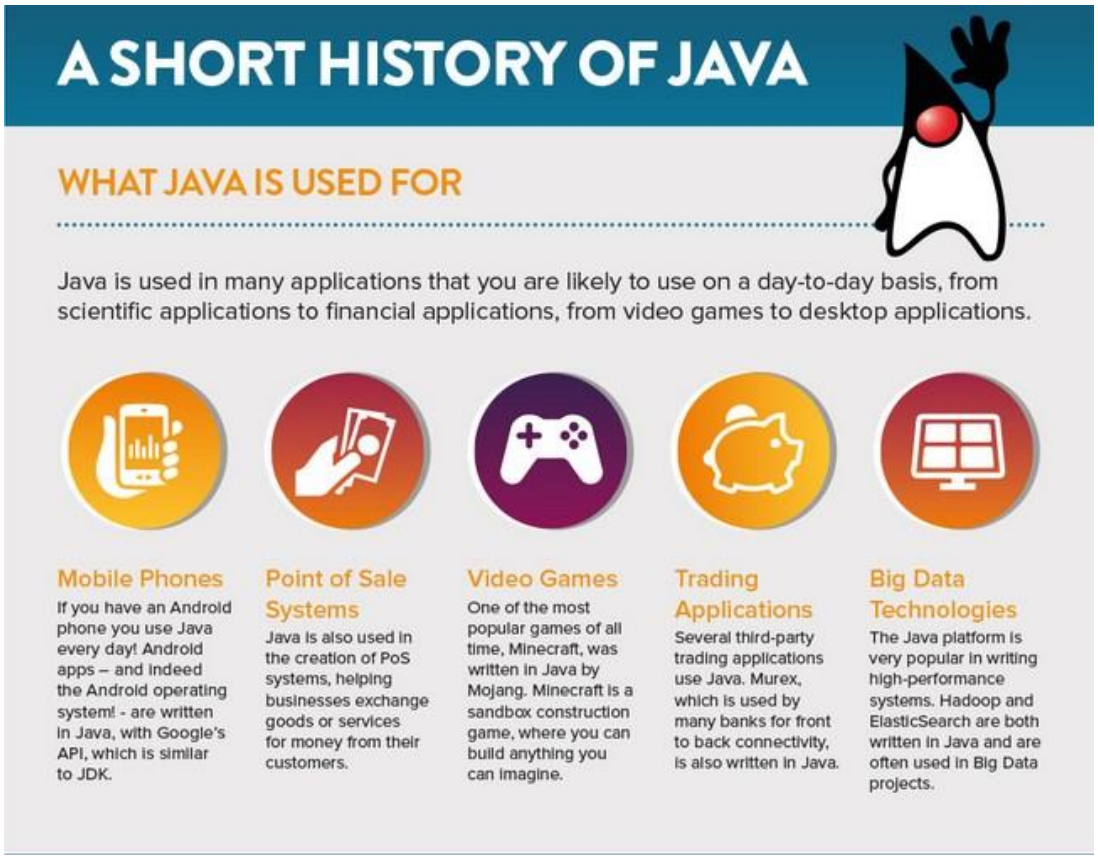
- Create => HTTP POST request
- Read => HTTP GET request
- Update => HTTP PUT request
- Delete => HTTP DELETE request

No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.
5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

Εικόνα 6: Soap vs Rest web services (πηγή: javatpoint.com)

Java 7/8 – Spring Boot – Spring MVC

Μόλις τον περασμένο Μάιο η Java έκλεισε τα 22 της χρόνια στον χώρο του προγραμματισμού. Όπως φαίνεται και στο παρακάτω σχήμα η Java δύναται να χρησιμοποιηθεί σε πολλούς τομείς της τεχνολογίας.



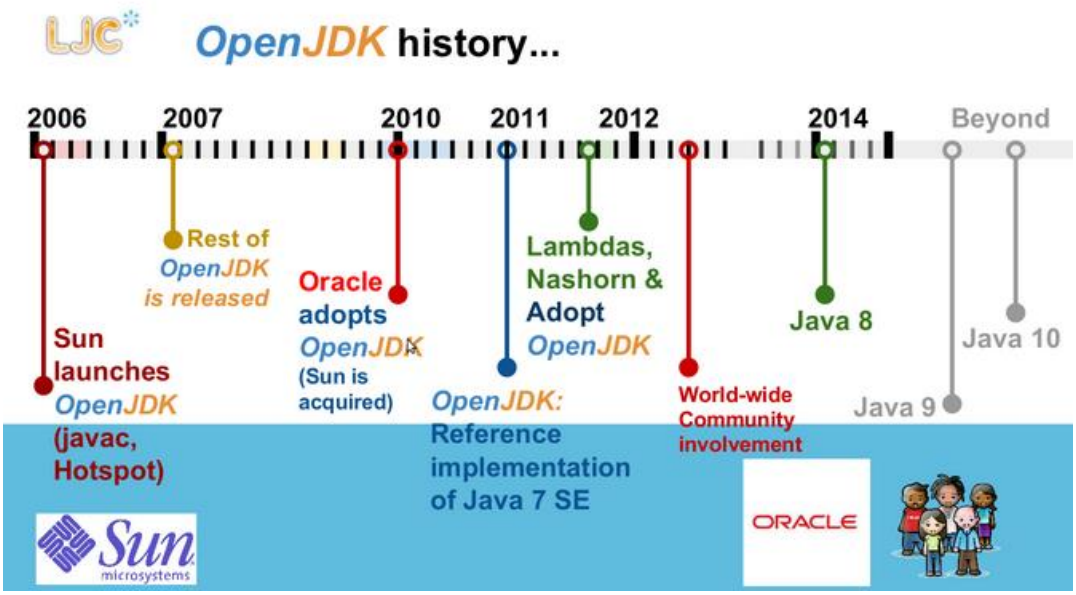
Εικόνα 7: Java main uses (πηγή: dzone.com)

Για την ιστορία, ένας μηχανικός της Sun Microsystems αρχικά και με τη συνεργασία άλλων 2 έπειτα συνεργαστήκαν για την ανάπτυξη ενός νέου εργαλείου – τεχνολογίας που θα συνδύαζε στοιχεία από άλλες δυο γλώσσες την Mesa και την C σε περιβάλλον αντικειμενοστραφές. Διαπίστωσαν λοιπόν ότι τα προβλήματα των ήδη υπάρχων αντικειμενοστραφών γλωσσών ήταν η διαχείριση της μνήμης, η διαχείριση των λαθών (error handling) αλλά και η ανάγκη για μια γλώσσα διαθέσιμη για διάφορες συσκευές τεχνολογίας.

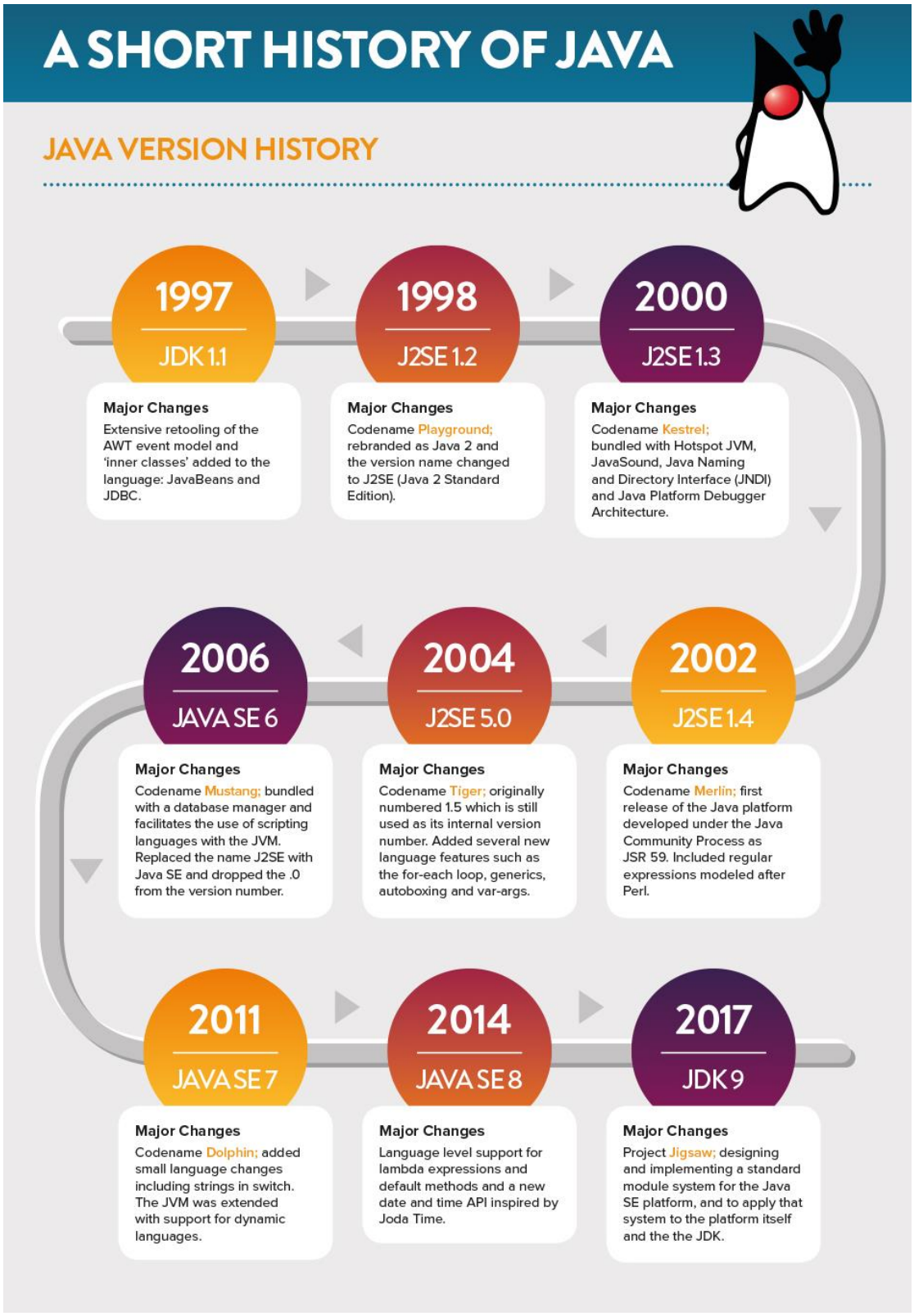
Η επίσημη εμφάνιση της Java αλλά και του HotJava (πλοηγός με υποστήριξη Java) στη βιομηχανία της πληροφορικής έγινε το Μάρτιο του 1995 όταν η Sun την ανακοίνωσε στο συνέδριο Sun World 1995. Ο πρώτος μεταγλωττιστής (compiler) της ήταν γραμμένος στη γλώσσα C και το 1994 ο A.Van Hoff ξαναγράφει τον μεταγλωττιστή της γλώσσας σε Java.

Οι IBM, Borland, Mitsubishi Electronics, Sybase και Symantec ανακοινώνουν σχέδια και αποφασίζουν αν εμπιστευτούν τη Java για την δημιουργία λογισμικού το Δεκέμβριο του 1995 κιόλας. Πλέον η γλώσσα είναι σαφώς ανώτερη και πιο εξελιγμένη από τότε και η πορεία της είναι καθαρά ανοδική όντας σήμερα από τις πιο δημοφιλείς γλώσσες στον χώρο της πληροφορικής.

Μια σημαντική ημερομηνία για την ίδια τη γλώσσα αλλά και τους προγραμματιστές που την εμπιστεύονται είναι η 13 Νοεμβρίου του 2006, καθώς η Java έγινε πλέον μια γλώσσα ανοιχτού κώδικα.



Εικόνα 8: Open JDK history (πηγή: google.com)



Εικόνα 9: Java versions over the years (πηγή: dzone.com)

Java 7 to Java 8

Το 2014 η Java έβγαλε το Java SE 8 και έφερε κάποιες σημαντικές αλλαγές και βελτιώσεις στη γλώσσα που διευκόλυναν πολύ τους προγραμματιστές. 3 από τις πολλές σημαντικές αλλαγές είναι οι παρακάτω.

New Date/Time APIs:

Ένα από τα σημαντικότερα θέματα που προκαλούσε δυσκολίες και προβλήματα στους προγραμματιστές μέχρι και το νέο release ήταν ότι το Java library API δεν προσέφερε ικανοποιητικές υπηρεσίες όσον αφορά το Date/Time API. Θα λέγαμε ότι περισσότερο δυσκόλευε παρά βοηθούσε. Πλέον όμως έχοντας ενσωματώσει το Joda Time και με ακόμη περισσότερες βελτιώσεις, η Java 8 προσφέρει μια καλή λύση στο χρόνιο αυτό πρόβλημα.

Lambda Expressions:

Ίσως η σημαντικότερη όπως αποδείχθηκε προσθήκη στην γλώσσα είναι τα Lambda expressions. Η προσθήκη αυτή έφερε η γλώσσα στο προσκήνιο του Functional Programming μαζί με άλλες JVM γλώσσες όπως την Scala και την Clojure. Αποτέλεσμα λοιπόν είναι πολλές λειτουργίες προστέθηκαν και πολλές γραμμές κώδικα αφαιρέθηκαν πράγμα που εκσυγχρόνισε την Java και την διατηρεί στον αφρό της τεχνολογίας και των προτιμήσεων των προγραμματιστών.

Παραλληλία:

Με την χρήση των lambda expressions έχουμε τη δυνατότητα πλέον να φύγουμε από τα όρια των κλασσικών τρόπων εσωτερικού iteration μέσα σε arrays. Πλέον όπως φαίνεται και στο παρακάτω δείγμα κώδικα μπορούμε να κάνουμε parallel iterate και να κάνουμε μαζί και filtering ή sorting ή και mapping, και όλα αυτά παράλληλα.

```
1 ConcurrentMap<Person.Sex, List<Person>> byGender =
2 roster.parallelStream().collect(
3 Collectors.groupingByConcurrent(Person::getGender))
```

Εικόνα 10: Code snippet example (πηγή: dzone.com)

Προφανώς κάτι τέτοιο μειώνει δραστικά το κόστος και τον φόρτο εργασίας για τον κώδικα και την εκτέλεσή του λύνοντας τα χέρια για τους προγραμματιστές.

Java 8 to Java 9 (New features are coming)

Κάποιες από τις σημαντικότερες νέες προσθήκες είναι οι παρακάτω.

Jshell:

Είναι ένα νέο command line εργαλείο όπου ο προγραμματιστής μπορεί να προσθέσει κώδικα αυτόνομα και μάλιστα χωρίς να πρέπει να τον βάλει σε κάποια μέθοδο.

Full support for HTTP 2.0 Client:

Η Java 9 θα προσφέρει πλήρη υποστήριξη στο νέο HTTP πρωτόκολλο το HTTP 2.0 .

Process API:

Είναι ένα νέο API που προσφέρει τη δυνατότητα απευθείας σύνδεσης των διεργασιών της εφαρμογής να επικοινωνούν δίχως να γραφεί κώδικας. Η συγκεκριμένη προσθήκη θα βελτιώσει την δυνατότητα της Java να επικοινωνεί με τις διεργασίες του λειτουργικού.

Modular Source Code:

Μια εξίσου σημαντική αλλαγή είναι η αναδιοργάνωση του πηγαίου κώδικα του JDK σε modules (ενότητες).

Spring Framework

Το Spring Framework είναι το πιο διαδεδομένο πλέον development framework για Java συστήματα. Είναι μια ανοιχτού κώδικα πλατφόρμα Java προγραμματισμού που ξεκίνησε ο Rod Johnson το 2003 και την εμπιστεύονται εκατομμύρια προγραμματιστές. Το Spring είναι οργανωμένο σε ενότητες (Modules) και πακέτα, πράγμα που διευκολύνει τον προγραμματιστή για να χρησιμοποιεί μονάχα αυτό που τον ενδιαφέρει. Επίσης αυτό που έχει κάνει και έχει καταφέρει να το κάνει τόσο δημοφιλές είναι ότι έχει συλλέξει όλα τα χρήσιμα εργαλεία για τον προγραμματιστή όπως Logging frameworks, ORM frameworks, JEE, Hibernate, JDBC και διάφορες άλλες τεχνολογίες.

Dependency Injection (DI)

Υπάρχει ο constructor-based dependency injection και ο setter-based dependency injection που μπορούν να χρησιμοποιηθούν είτε μαζί είτε χώρια. Η πρώτη μέθοδος επιτυγχάνεται όταν ο container επικαλείται έναν constructor και του περνάει παραμέτρους που το καθένα αποτελεί dependency στην άλλη κλάση. Η δεύτερη μέθοδος επιτυγχάνεται όταν ο **container** καλεί μια setter μέθοδο σε ένα bean μέσω ενός constructor ή μια στατική μέθοδο χωρίς arguments για να κάνει instantiate ένα bean.

Inversion of Control (IoC)

Ο spring container είναι ο πυρήνας του spring framework καθώς είναι αυτός που δημιουργεί, συνδέει και διαμορφώνει τα αντικείμενα μέχρι και την καταστροφή τους. Το dependency injection είναι αυτό που παίζει το ρόλο του διαχειριστή όλων αυτών των εξαρτημάτων για να δημιουργήσει την τελική εφαρμογή. Για την διασύνδεση των εξαρτημάτων (**beans**) χρησιμοποιούνται οδηγίες που δίνονται είτε προγραμματιστικά με annotations ή java code είτε μέσω configurations files.

Spring Bean

Τα beans λοιπόν αποτελούν τα αντικείμενα που αναφέραμε παραπάνω και είναι ο σκελετός μιας εφαρμογής που στήνεται και διαμορφώνεται από τον IoC. Ένας τρόπος για να μπορέσει ο IoC να εντοπίσει αυτά τα αντικείμενα που εμείς θεωρούμε τον σκελετό είναι να τα ορίσουμε με το @Bean annotation είτε να τους δώσουμε τα configurations τους μέσω xml αρχείων.

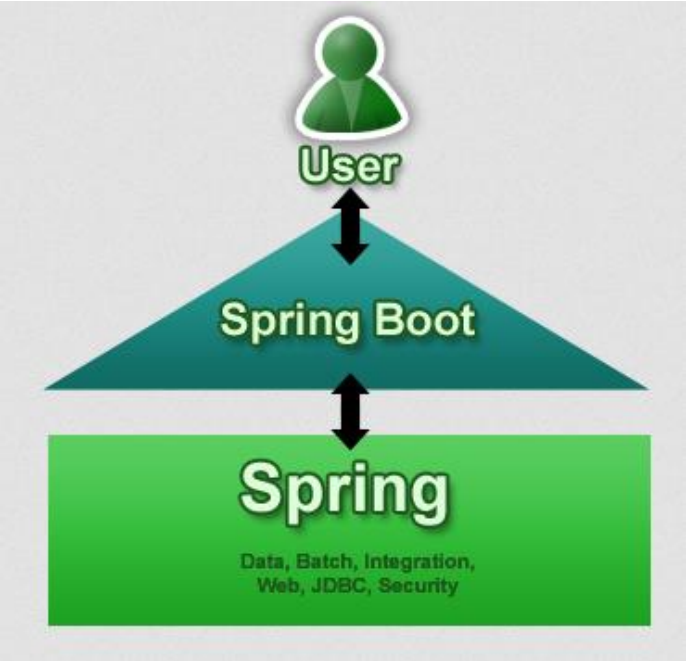
Spring MVC

MVC δηλαδή Model-View-Controller αρχιτεκτονική που προσφέρει έτοιμα components (συστατικά) για την ανάπτυξη ευέλικτων web εφαρμογών. Το model (μοντέλο) είναι αυτό που είναι υπεύθυνο για την ενσωμάτωση των POJOs. Το view (προβολή) είναι υπεύθυνο για να παραχθεί HTML έξοδος και ο controller έχει σκοπό να επεξεργαστεί τα αιτήματα του client (χρήστη) και να κατασκευάσει ένα μοντέλο που θα προβάλει και θα μεταβιβάσει σε αυτόν.

Spring Boot

Από την άλλη το spring boot είναι χτισμένο διαφορετικά. Είναι μια πλατφόρμα-σουίτα με αρκετές έτοιμες διαμορφώσεις και διασυνδέσεις ενσωματώνοντας χρήσιμα εργαλεία που χρησιμοποιούν οι προγραμματιστές στο spring. Ουσιαστικά αφαιρούν την ανάγκη χρήσης xml-configuration θέτοντας έτοιμες διαμορφώσεις απελευθερώνοντας τον προγραμματιστή προσφέροντας του

ταχύτητα, λιγότερες γραμμές κώδικα και όχι περιττές απαιτήσεις για διαμόρφωση σε εργαλεία όπως βάσεις δεδομένων, logging, web-server etc.



Εικόνα 11: Spring & Spring Boot (πηγή: tutorialspoint.com)

SQL – NoSQL Databases

Εικόνα 12: SQL vs NoSQL uses (πηγή: dzone.com)

Η **SQL** λοιπόν είναι μία γλώσσα που πραγματώνει την επικοινωνία με μια σχεσιακή βάση δεδομένων ή ενός συστήματος σχεσιακών βάσεων δεδομένων. Με την SQL ο χρήστης μπορεί να κάνει σχεδόν τα πάντα στη βάση, από το να συνδεθεί μαζί της και να αποκτήσει πρόσβαση σε αυτήν μέχρι και να γράψει και να σβήσει κάτι από αυτήν.

Ο όρος **NoSQL** περιλαμβάνει μια ευρεία γκάμα τεχνολογιών τύπου βάσης δεδομένων προσαρμοσμένες στις νέες τεχνολογίες και τις αυξημένες απαιτήσεις τους. Μπορούν να υποστηρίξουν ένα μεγάλο αριθμό διαφόρων τύπου δεδομένων και μορφής δεδομένων και προσφέρουν υψηλή ταχύτητα και αποδοτικότητα αλλά υστερούν σε σταθερότητα και σε παλαιότερες τεχνικές διαδικασίες.

- SQL databases => MySQL, Oracle SQL, H2 etc.
- NoSQL databases => MongoDB, Redis, Cassandra, CouchDB etc.

<u>Γιατί να χρησιμοποιήσω SQL database;</u>

- Γιατί ίσως χρειάζομαι το λεγόμενο ACID (Atomicity, Consistency, Isolation, Durability) που μόνο μια σχεσιακή βάση μπορεί να μου προσφέρει τουλάχιστον στον μεγαλύτερο βαθμό. Μου προσφέρουν ακεραιότητα και ασφαλείς συναλλαγές (transactions) πράγματα που οι NoSQL βάσεις δεν δίνουν βαρύτητα.

- Γιατί αν τα δεδομένα μου είναι σταθερά και αμετάβλητα χωρίς πολλούς διαφορετικούς τύπους δεν υπάρχει λόγος να μεταφερθώ σε μια βάση που εστιάζει στην ταχύτητα και τις ποικιλία των δεδομένων.

<u>Γιατί να χρησιμοποιήσω NoSQL database;</u>

- Γιατί δεν έχω περιορισμό στο μέγεθος και το scalability των δεδομένων και κανείς δεν με εμποδίζει να χρησιμοποιήσω ότι τύπο απαιτεί η επιχείρησή μου εκείνη τη στιγμή.

- Γιατί χρησιμοποιώ cloud υπηρεσίες και με ενδιαφέρει η ταχύτητα και η ευκολία μεταφοράς και διαμοιρασμού των δεδομένων μεταξύ των server.

- Γιατί η ταχύτητα με ενδιαφέρει ξανά αυτή τη φορά στον τομέα του development που η ομάδα μου θα σπαταλήσει χρήσιμο χρόνο στη προετοιμασία και στο στήσιμο μιας SQL βάσης και σίγουρα όχι στον ίδιο βαθμό σε μια NoSQL.

- Στην εφαρμογή μας χρησιμοποιήσαμε NoSQL βάση για την αποθήκευση των δεδομένων μας και κάποιων άλλων απαραίτητων στοιχείων. Η ταχύτητα ανταπόκρισης αλλά και ο τύπος των δεδομένων ταίριαζε περισσότερο σε αυτού του είδους τη βάση και γι' αυτό την προτιμήσαμε αμέσως χωρίς δεύτερη σκέψη. Πιο συγκεκριμένα χρησιμοποιήσαμε MongoDB και σε μικρό βαθμό και Redis.

Για την λήψη των δεδομένων

Vaadin Framework

```
<!-- vaadin dependency -->
<dependency>
    <groupId>com.vaadin</groupId>
    <artifactId>vaadin-spring-boot-starter</artifactId>
</dependency>
```

Εικόνα 13: Vaadin dependency

Το **Vaadin** Framework είναι ένα web application framework που σε αντίθεση με διάφορες Javascript βιβλιοθήκες και plugins αυτό βασίζεται σε server side αρχιτεκτονική. Αυτό σημαίνει ότι το μεγαλύτερο μέρος της εφαρμογής είναι με ασφάλεια στημένο στον server και όχι στον client.

Το Vaadin λοιπόν περιλαμβάνει ένα μεγάλο αριθμό από διάφορα UI components – εργαλεία που ο προγραμματιστής μπορεί να τα συνθέσει όπως θέλει. Κάποια από αυτά είναι buttons, textboxes, radio buttons, layouts, images και πολλά άλλα. Αυτά προσφέρουν και λειτουργικούς listeners που δίνουν τη δυνατότητα να αξιοποιήσει ο προγραμματιστής για να τα συνδέσει με την υπόλοιπη εφαρμογή.

Έτσι η ταχύτητα ανάπτυξης μιας εφαρμογής με χρήση του Vaadin μπορεί να γίνει αρκετά μεγάλη και ικανοποιητική ώστε να του δώσει μία θέση στη λίστα με τα πιο χρήσιμα εργαλεία για την χρήση του. Η λογική των components σε συνδυασμό με την Java προσφέρει εύκολα συντηρήσιμο κώδικα που μαζί με εύκολη διαχείριση των αντικειμένων αυτών και την πλήρη υποστήριξη από τα IDE μπορούν να δημιουργήσουν κάτι πολύ καλό.

- Το συγκεκριμένο framework το χρησιμοποιήσαμε στο κομμάτι της επιλογής και συμπλήρωσης όλων των δεδομένων που απαιτούνται από τον χρήστη να δώσει ώστε να υλοποιηθεί η εφαρμογή μας.

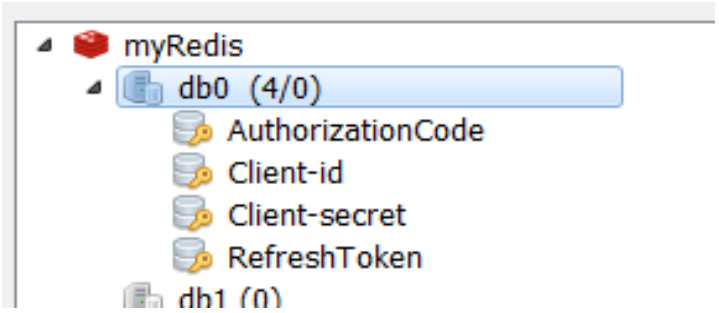
Redis Database

```
<!-- database dependencies -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

Εικόνα 14: Redis database dependency

Το πλήρες όνομά της είναι REmote DIctionary Server και είναι μια NOSQL **database** βασισμένη στην key - value αρχιτεκτονική. Έχει 8 χρόνια ζωής στον κόσμο της τεχνολογίας και από το 2015 και μετά υποστηρίζεται από την Redis Labs. Έχει μεγάλη απήχηση στον προγραμματιστικό κόσμο αφου βρίσκεται σταθερά στις πρώτες θέσεις των προτιμήσεων αλλά και των αποτελεσματικών επιδόσεων. Υποστηρίζει διάφορους τύπους δεδομένων αλλά και τις περισσότερες γλώσσες προγραμματισμού.

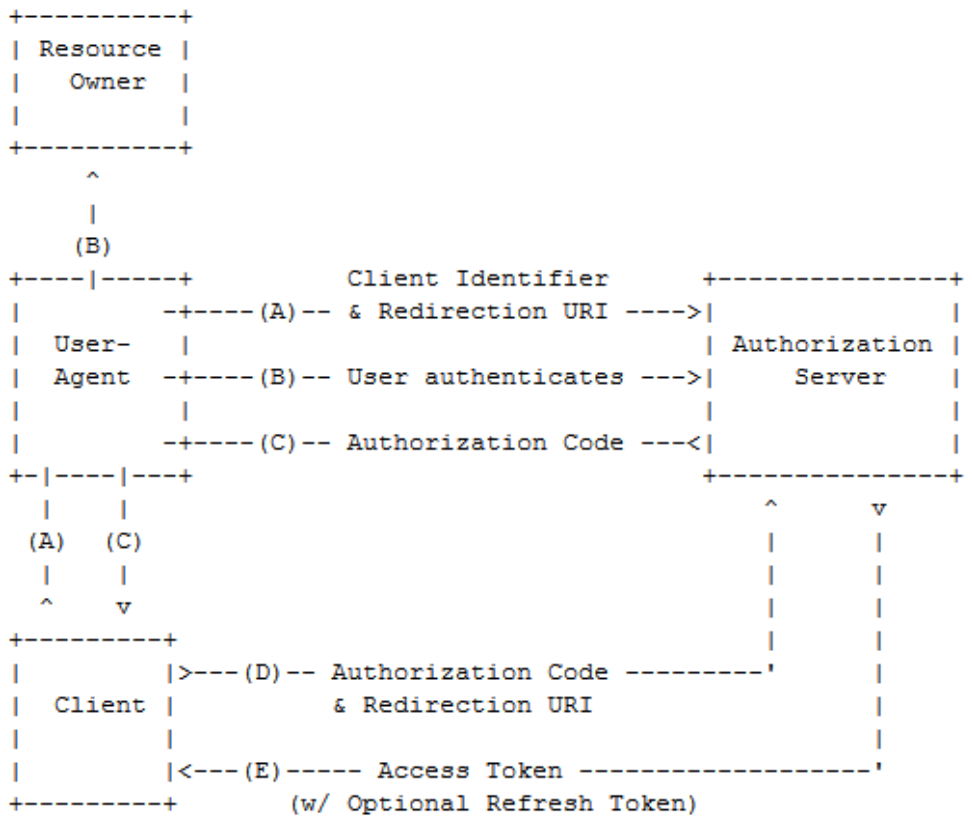
- Τη συγκεκριμένη βάση δεδομένων χρησιμοποιήσαμε στο κομμάτι της ταυτοποίησης του χρήστη όταν και αποθηκεύουμε προσωρινά τον Authorization Code και κάποια άλλα στοιχεία που αφορούν το κομμάτι της ταυτοποίησης για την χρησιμοποίησή του αργότερα με το Access Token και το Refresh Token.



Εικόνα 15: Our Redis database keys

OAuth2 Rest Calls to Fitbit API

Η επικοινωνία προς το API του Fitbit προϋποθέτει την ολοκλήρωση της διαδικασίας ταυτοποίησης της εφαρμογής μας με τα στοιχεία του χρήστη με χρήση της **Oath2** αρχιτεκτονικής. Η διαδικασία λοιπόν της ταυτοποίησης περιγράφεται στο σχήμα παρακάτω.



Εικόνα 16: OAuth2 flow (πηγή: tools.ietf.org)

Ο client του σχήματος είναι η εφαρμογή μας όπου δημιουργεί τον agent ο οποίος στέλνει αρχικά τα client credentials και ένα callback URL ώστε να του σταλεί πίσω ο authorization code. Βέβαια όπως φαίνεται στο σχήμα προηγείται το user authentication όπου ο authorization server του Fitbit ανακατευθύνει τον agent (δηλαδή εμάς και την εφαρμογή μας) σε μια Login φόρμα όπου εμείς Manually την ολοκληρώνουμε και έπειτα αποστέλνεται το code. Το οποίο όπως δείχνει και το σχήμα το ξαναστέλνουμε και λαμβάνουμε το access token που το περιλαμβάνουμε πάντα στις επόμενες κλήσεις προς το API. Όλα αυτά λοιπόν όπως και οι κλήσεις μετέπειτα είναι rest calls της κατηγορίας POST και GET στις περιπτώσεις αυτές.

Κάποια αποσπάσματα από την δική μας εφαρμογή είναι τα παρακάτω:

- GET rest call to Fitbit API for authorization code request

https://www.fitbit.com/oauth2/authorize?redirect_uri=http://localhost:8080/fitbitApp/authCode&response_type=code&client_id=XXXX&scope=activity%20nutrition%20heartrate%20profile%20settings%20sleep%20social%20weight&expires_in=3600&prompt=login

- POST rest call to obtain the access token

Set the parameters:

```
// request parameters
MultiValueMap<String, String> parameters = new LinkedMultiValueMap<String, String>();
parameters.add("clientId", redisTemplate.opsForValue().get("Client-id"));
parameters.add("grant_type", authProp.getGrantType());
parameters.add("redirect_uri", authProp.getRedirectUri());
parameters.add("code", redisTemplate.opsForValue().get("AuthorizationCode"));
```

Εικόνα 17: Request's parameters

Set the headers:

```
// request headers
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
headers.set("Authorization", "Basic " + headerAuth);
headers.set("Accept", authProp.getHeaderAccept());
```

Εικόνα 18: Request's headers

Sent the request with all included as HttpEntity:

```
HttpEntity<MultiValueMap<String, String>> entity = new HttpEntity<MultiValueMap<String, String>>(parameters,headers);
ResponseEntity<String> response = restTemplate.
        exchange(authProp.getTokenUrl(),
                HttpMethod.POST,
                entity,
                String.class);
```

Εικόνα 19: Combine all to create the request

- GET rest call to obtain user data

Μετά την ολοκλήρωση της ταυτοποίησης με τις παραπάνω κλήσεις χρησιμοποιούμε το access token που μας γυρνάει το API και το βάζουμε στους **headers** του call όπως παρακάτω.

```
public HttpEntity<String> getEntity(boolean unauthorized) throws JsonProcessingException, IOException {
    HttpHeaders headers = new HttpHeaders();

    if (unauthorized) {
        headers.set("Authorization", "Bearer " + refreshTokenService.refreshToken());
    } else {
        headers.set("Authorization", "Bearer " + getAccessToken());
    }

    return new HttpEntity<String>(headers);
}
```

Εικόνα 20: Set access token header

Create and send the request:

```
ResponseEntity<String> response = restTemplate.
        exchange(url + month,
                HttpMethod.GET,
                saveOperationsService.getEntity(false),
                String.class);
```

Εικόνα 22: Full request created

```
# urls
fitbitApiUrls:

  #sleep
  timeInBedUrl: https://api.fitbit.com/1/user/-/sleep/timeInBed/date/
  minutesAsleepUrl: https://api.fitbit.com/1/user/-/sleep/minutesAsleep/date/
  minutesAwakeUrl: https://api.fitbit.com/1/user/-/sleep/minutesAwake/date/
  toFallAsleepUrl: https://api.fitbit.com/1/user/-/sleep/minutesToFallAsleep/date/
  afterWakeUpUrl: https://api.fitbit.com/1/user/-/sleep/minutesAfterWakeup/date/
  efficiencyUrl: https://api.fitbit.com/1/user/-/sleep/efficiency/date/

  #activity
  stepsUrl: https://api.fitbit.com/1/user/-/activities/steps/date/
  floorsUrl: https://api.fitbit.com/1/user/-/activities/floors/date/
  distanceUrl: https://api.fitbit.com/1/user/-/activities/distance/date/
  caloriesUrl: https://api.fitbit.com/1/user/-/activities/calories/date/
  heartUrl: https://api.fitbit.com/1/user/-/activities/heart/date/
  profileUrl: https://api.fitbit.com/1/user/-/profile.json
  frequenceUrl: https://api.fitbit.com/1/user/-/activities/frequence.json
  lifetimeUrl: https://api.fitbit.com/1/user/-/activities.json
```

Εικόνα 21: Fitbit endpoints for each user's category data

Όπου **url** είναι το εκάστοτε endpoint στο API του fitbit που αντιστοιχεί στον τύπο δεδομένων που επιθυμούμε να απευθυνθούμε. Στα τρία τελευταία από τα παρακάτω δεν βάζουμε ημερομηνία καθώς είναι γενικές πληροφορίες.

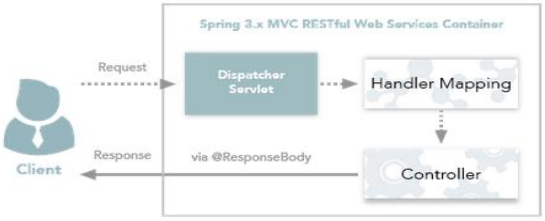
Και **month** είναι το αντίστοιχο χρονικό διάστημα μεταξύ του οποίου ζητάμε τα δεδομένα αυτά από το API.

Tomcat server – Spring Rest Controller

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Εικόνα 23: Tomcat's dependency

Apache **Tomcat** software είναι ένα run-time περιβάλλον από την Apache Foundation που δίνει τη δυνατότητα σε web εφαρμογές να «τρέχουν» σε έναν server. Χρησιμοποιείται κατά κόρον σε Java based τεχνολογίες και προσφέρει δυνατότητες μετατροπής html σελίδας σε java source code (JSP pages, servlets).



Εικόνα 24: Spring controller flow
(πηγή: tutorialspoint.com)

Όσον αφορά τους controllers του spring boot είναι βασισμένοι σε rest υπηρεσίες και η χρησιμοποίηση τους δηλώνεται με χρήση των annotations @Rest controller, @RequestMapping, @GetMapping, @ResponseBody etc. Στο dependency που παραθέτουμε παραπάνω χρησιμοποιείται και ο tomcat web server embedded όμως που σημαίνει ότι περιλαμβάνεται στις βιβλιοθήκες της εφαρμογής χωρίς να χρειαστεί να τον δηλώσουμε με αντίστοιχο dependency.

- Στη δική μας εφαρμογή χρησιμοποιούμε στο πρώτο μέρος της εφαρμογής τον tomcat web server ώστε να σηκώσουμε στο localhost, στην πόρτα 8080 έναν controller που φιλοξενεί τα tabs της εφαρμογής μας αλλά και το endpoint που περιμένει από το Fitbit API να στείλει τον Authorization Code στο κομμάτι της ταυτοποίησης.

```
@RestController
public class AuthorizationCodeController {

    @Autowired
    private RedisTemplate<String, String> redisTemplate;

    @Autowired
    private DailyDataService dailyDataService;

    private final static Logger LOG = LoggerFactory.getLogger("Fitbit application");

    @RequestMapping("fitbitApp/authCode")
    public RedirectView authorization(@RequestParam(value = "code") String code) {
```

Εικόνα 25: Spring Rest controller form

Για την επεξεργασία των δεδομένων

Java 8 Time

Ανέκαθεν οι Java developers είχαν να λένε για τα περιβόητα dates στην Java που τους ταλαιπωρούσε και δυσκόλευε το έργο. Η Java 8 όμως συμπεριέλαβε ένα δικό της Date API και μια ανακούφιση και χαρά επικράτησε στους προγραμματιστές. Το **Java date** λοιπόν πλέον είναι βασισμένο στο Gregorian Calendar και το πακέτο που περιλαμβάνει όλες τις κλάσεις για date και time είναι το java.time package και περιέχει τα εξής παρακάτω.

- **java.time.Period**: This class represents the date part of the datetime. It represents the date part in terms of days, months and years. for example 1 year, 2 months and 5 days
- **java.time.Duration**: This class represents the time part of the datetime. It represents the time part in terms of seconds and nanoseconds. for example '29 seconds'
- **java.time.Instant**: This represents a particular instant in the timeline. It stores the number of seconds through epoch time and also has another field that stores the nanoseconds
- **java.time.LocalDate**: This stores the date part of date-time in terms of years-months-days. It does not store the TimeZone. The class is immutable.
- **java.time.LocalTime**: This class stores the time part of date time without any TimeZone info.
- **java.time.LocalDateTime**: This class stores the LocalDate and LocalTime but no TimeZone
- **java.time.ZonedDateTime**: This class stores the LocalDateTime and the TimeZone info as a ZoneOffset object. The class has access to ZoneRules which can be used to convert to local time
- **java.time.ZoneOffset**:This stores the time zone offset from UTC. The zone rules are stored in ZoneId.
- **java.time.OffsetDateTime**:This stores the local datetime with the offset. This class does not have information about Zone Rules.

Εικόνα 26: Java time libraries (πηγή: studytrails.com)

- Όσον αφορά την εφαρμογή μας αξιοποιήσαμε το καινούριο και απλοποιημένο java date στο μέρος κατά το οποίο παίρνουμε τα date που βάζει ο χρήστης. Ο χρήστης διαλέγει 2 dates ένα για startDate και ένα για endDate τα οποία αποτελούν αρχή και τέλος του range των ημερομηνιών κατά τις οποίες θα ζητήσουμε τα data από το Fitbit API. Όμως το Fitbit API επέτρεπε στον client να λάβει δεδομένα του χρήστη με date range 100 ημέρες. Οπότε λαμβάνοντας τα dates η εφαρμογή με ευρεία χρήση του java date time χώριζε το range αυτό σε τρίμηνα. Πλέον το API δεν έχει αυτόν τον περιορισμό οπότε απλά η εφαρμογή παίρνει τα dates και τα βάζει στα URL στις rest κλήσεις που κάνει.

```java
/**
 * Takes the starting & the ending date and split the range of them to 90-days partitions
 * if it's possible
 *
 * @param startDate
 * @param endDate
 * @return
 */
public List<Map<String, String>> getDates(LocalDate startDate, LocalDate endDate) {

    long addedDays = 0;
    long between = ChronoUnit.DAYS.between(startDate, endDate);
    List<Map<String, String>> dates = new ArrayList<Map<String, String>>();

    while (between > 90) {

        endDate = startDate.plusDays(90);

        Map<String, String> date = new HashMap<>();
        date.put("StartDate", startDate.toString());
        date.put("EndDate", endDate.toString());
        dates.add(date);
        between -= 90;

        startDate = endDate.plusDays(1);
        addedDays += 1;
    }

    if (between > 0) {
        Map<String, String> date = new HashMap<>();
        date.put("StartDate", startDate.toString());
        date.put("EndDate", startDate.plusDays(between).minusDays(addedDays).toString());
        dates.add(date);
    }
    return dates;
}
```

Εικόνα 27: Java time uses in our project

Jackson Mapper - org.json – JSON

```xml
<!-- json dependencies -->
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
</dependency>
<dependency>
    <groupId>com.fasterxml</groupId>
    <artifactId>jackson-xml-databind</artifactId>
    <version>${fasterxml.version}</version>
</dependency>
```

Εικόνα 28: Jackson & org.json dependencies

Ο Jackson Mapper είναι μια βιβλιοθήκη σταθερά στις λίστες με τις πιο χρήσιμες και σημαντικές τα τελευταία χρόνια. Το αντικείμενο της είναι η μετατροπή Java objects σε JSON strings αλλά και το αντίστροφο.

- Η βιβλιοθήκη αυτή αποδείχθηκε πάρα πολύ χρήσιμη ως και απαραίτητη καθώς η εφαρμογή διαχειρίζεται στο μεγαλύτερο της βαθμό JSON strings. Το Fitbit API στέλνει τα data σε μορφή JSON οπότε η διαχείρισή τους είναι απαραίτητη.

```
@Service
public class DailyDataService {

    @Autowired
    private ObjectMapper mapper;

    @Autowired
    private RestTemplate restTemplateGet;

    @Autowired
    private MongoTemplate mongoTemplate;

    @Autowired
```

Εικόνα 30: Code snippet

```
    JsonNode responseDataBody = mapper.readTree(responseWithRefreshToken.getBody());
    DBObject dataToInsert = (DBObject) JSON.parse(responseDataBody.toString());
    mongoTemplate.insert(dataToInsert, collection);

} else if (response.getStatusCodeValue() == 200) {
    JsonNode responseDataBody = mapper.readTree(response.getBody());
    DBObject dataToInsert = (DBObject) JSON.parse(responseDataBody.toString());
    mongoTemplate.insert(dataToInsert, collection);
}
```

Εικόνα 29: Code snippet

- Αλλά χρησιμοποιήσαμε και αρκετές λειτουργίες της org.json βιβλιοθήκης όπως σε φιλτράρισμα και επεξεργασία των υπαρχόντων json strings:

```
JSONArray responseDataArray = heartCallResponse(month).orElseThrow(() -> new IOException("Something went wrong with the requests"));
if (responseDataArray != null) {
    for (int rda = 0; rda < responseDataArray.length(); rda++) {
        JSONArray heartRateZonesArray = responseDataArray.getJSONObject(rda).getJSONObject("value").getJSONArray("heartRateZones");
        for (int hrza = 0; hrza < heartRateZonesArray.length(); hrza++) {
            insertHeartRateValues(responseDataArray, rda, heartRateZonesArray, hrza);
        }
    }
    return true;
}
```

Εικόνα 31: Code snippet

Itext pdf – pdf file export

```
<!-- pdf creator dependency -->
<dependency>
    <groupId>com.itextpdf</groupId>
    <artifactId>itextpdf</artifactId>
    <version>${itext.pdf.version}</version>
</dependency>
```

Εικόνα 32: itext pdf dependency

Η **itext pdf** είναι μια ευρέως χρησιμοποιούμενη βιβλιοθήκη όχι μόνο για java developers καθώς υποστηρίζει εκδόσεις της και σε άλλες γλώσσες. Ο προγραμματιστής δημιουργεί εύκολα την σελίδα που θέλει και προσθέτει πίνακες, κείμενα, εικόνες και ότι άλλο επιθυμεί.

- Παραδείγματα από την δική μας εφαρμογή:

  Εδώ δημιουργούμε μια σελίδα και ορίζουμε το όνομα και το αρχείο που θα χρησιμοποιήσουμε.

```
public void createDocumentWithUserData(List<String> parameters) {
    Document document = new Document();
    try {
        PdfWriter.getInstance(document, new FileOutputStream(downloadingProperties.getExportFileName()));
        Font font = FontFactory.getFont(FontFactory.COURIER_BOLD, 14, BaseColor.BLACK);
        document.open();
```

Εικόνα 33: Code dependency

Εδώ βλέπουμε παράδειγμα δημιουργίας κειμένου και δημιουργία ενός πίνακα.

```
    Chunk steps = new Chunk("Activity - Steps Table", font);
    Chunk floors = new Chunk("Activity - Floors Table", font);
    Chunk distance = new Chunk("Activity - Distance Table", font);
    Chunk calories = new Chunk("Activity - Calories Table", font);

    PdfPTable tableSteps = new PdfPTable(2);
    tableSteps.setSpacingAfter(70);
    tableSteps.setSpacingBefore(50);
    addTableHeader(tableSteps);
    addRows(tableSteps, stepsData);
```

Εικόνα 34: Code snippet

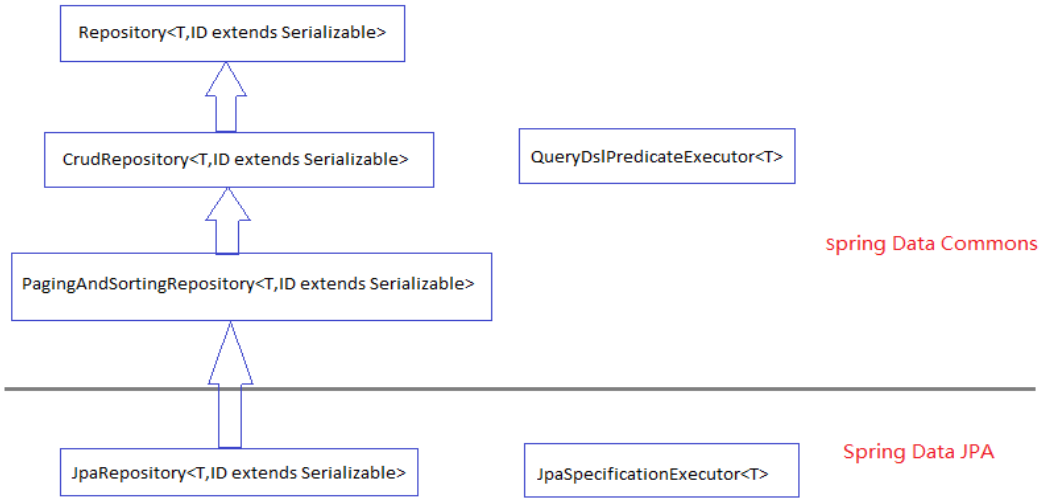
Εδώ ορίζουμε τα headers του πίνακα και έπειτα γεμίζουμε τις γραμμές του με τα δεδομένα.

```
private void addTableHeaderHeartRate(PdfPTable table) {
    Stream.of("Date", "Heart-Rate category", "Minutes", "Minimum heart-rate", "Maximum hea
            .forEach(columnTitle -> {
                PdfPCell header = new PdfPCell();
                header.setBackgroundColor(BaseColor.LIGHT_GRAY);
                header.setBorderWidth(2);
                header.setPhrase(new Phrase(columnTitle));
                table.addCell(header);
            });
}

private void addRowsHeartRate(PdfPTable table, List<HeartRateValue> data) {
    data.forEach(d -> {
        table.addCell(d.getDate());
        table.addCell(d.getName());
        table.addCell(String.valueOf(d.getMinutes()));
        table.addCell(String.valueOf(d.getMin()));
        table.addCell(String.valueOf(d.getMax()));
        table.addCell(String.valueOf(d.getCaloriesOut()));
    });
}

private void addTableHeader(PdfPTable table) {
    Stream.of("Date", "Values")
            .forEach(columnTitle -> {
                PdfPCell header = new PdfPCell();
                header.setBackgroundColor(BaseColor.LIGHT_GRAY);
                header.setBorderWidth(2);
                header.setPhrase(new Phrase(columnTitle));
                table.addCell(header);
            });
}
```

Εικόνα 35: Code snippet

Spring Data

Το Spring Data είναι ένα κομμάτι του Spring γενικότερα και αφορά τη διαχείριση με την βάση δεδομένων. Υποστηρίζει κάθε είδους βάση δεδομένων με τα κατάλληλα configurations είτε αυτή είναι SQL (Oracle, MySQL etc.) αλλά και μη (MongoDB, Redis etc.)



Εικόνα 36: Spring Data layers (πηγή: perficient.com)

Η λογική είναι ότι υπάρχει ένα repository (**interface**) που κάνει extend το **JPA Repository** όπου εκεί βρίσκονται και υλοποιούνται οι περισσότερες βασικές λειτουργίες σχετικά με την βάση δεδομένων. Ο προγραμματιστής το μόνο που κάνει είναι να γράφει και να χρησιμοποιεί μεθόδους σχετικά με τη βάση και το spring data υλοποιεί όλα τα υπόλοιπα.

Παράδειγμα configurations από άλλο project:

```
#database configuration
spring.datasource.url=jdbc:mysql://localhost/hotelReservationDB
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

Εικόνα 37: Spring data configurations for database

Παράδειγμα χρήσης του Spring data JPA από άλλο project:

```
public interface ReservationRepository extends CrudRepository<Reservation, UUID> {

    Reservation findByCustomerId(UUID id);
    Reservation findByRoomNumber(Integer number);

    List<Reservation> findByStatus(ReservationStatus status);
    List<Reservation> findAll();

    List<Reservation> findByFromDateGreaterThanEqual(LocalDate from);
    List<Reservation> findByFromDateLessThanEqual(LocalDate to);
    List<Reservation> findByFromDateBetween(LocalDate from, LocalDate to);

    List<Reservation> findByToDateGreaterThanEqual(LocalDate from);
    List<Reservation> findByToDateLessThanEqual(LocalDate to);
    List<Reservation> findByToDateBetween(LocalDate from, LocalDate to);

    Reservation findByRoomNumberAndToDateIsInOrFromDateIsIn(Integer number, List<LocalDate> datesTo, List<LocalDate> datesFrom);
```

Εικόνα 38: Example of spring data use

- Η δική μας εφαρμογή χρησιμοποιεί ελάχιστα αυτή τη λειτουργία κατά την διαδικασία της αποστολής e-mail στον χρήστη με τα δεδομένα καρδιάς σύμφωνα με τις παραμέτρους που ο ίδιος έχει δώσει.

```
public interface HeartRateZoneRepository extends MongoRepository<HeartRateValue, String> {

    Stream<HeartRateValue> findByMinutesGreaterThanAndNameIs(long minutes, String name);

    HeartRateValue findDistinctByName(String name);

}
```

Εικόνα 40: Code snippet

```
w.write("There are Heart-Rate data when the user's heart-rate was at " + category + " zone!" + '\n' + '\n');
heartRepository.findByMinutesGreaterThanAndNameIs(minutes, category.d()).forEach(d -> {

    try {
        w.write("In " + d.getDate() + " for : "
                + d.getMinutes() + " minutes" + '\n');
    } catch (IOException e) {
        LOG.error("Something went wrong: ", e);
        clearFieldsService.removeAll(content);
    }
});

HeartRateValue heartRateZone = heartRepository.findDistinctByName(category.d());
Long min = heartRateZone.getMin();
Long max = heartRateZone.getMax();

w.close();
sendMailService.email(mail, minutes, category, min, max);
```

Εικόνα 39: Code snippet

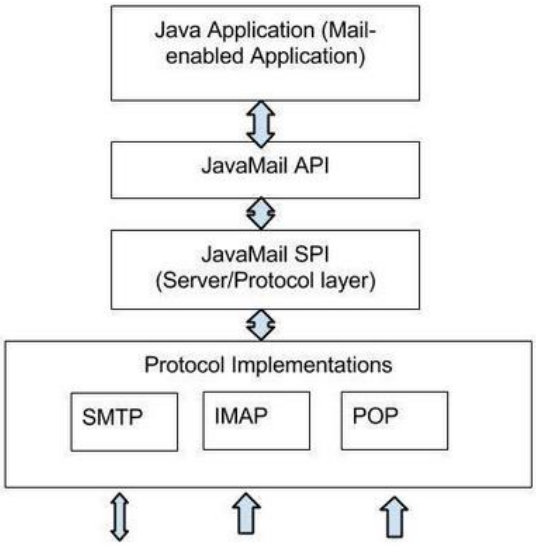
Java Mail API

```
<!-- mail dependency -->
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>${javax.mail.version}</version>
</dependency>
```

Εικόνα 41: Java mail dependency

Το java mail API είναι μια βιβλιοθήκη κατά την οποία μπορεί ο προγραμματιστής με τα εργαλεία που προσφέρει να συντάξει και να στείλει ένα **e-mail**. Υποστηρίζει όλα τα e-mail πρωτόκολλα όπως SMTP, IMAP, POP, MIME, NNTP etc.

Η αρχιτεκτονική είναι η εξής παρακάτω:



Εικόνα 42: Java mail flow (πηγή: tutorialspoint.com)

- Στη δική μας εφαρμογή χρησιμοποιούμε αυτή την υπηρεσία στη διαδικασία του Notification part για τα δεδομένα καρδιάς. Αφού ο χρήστης επιλέξει την κατηγορία (Out of range, Fat burn, Cardio, Peak) και το χρονικό διάστημα που οι παλμοί του βρίσκονταν σε αυτή την κατηγορία, έπειτα επιλέγει που θα στείλει το e-mail με αυτά τα δεδομένα.

Configurations για το java mail api:

```yaml
# e-mail credentials
mailInfo:
  username: [redacted]
  password: [redacted]
  fileName: heartData.txt
  sendFrom: nickma72@gmail.com
  mailSmtpStartEnable: true
  mailSmtpAuth: true
  mailSmtpHost: smtp.gmail.com
  mailSmtpPort: 587
  mailSubject: Fitbit app Info mails
```

Εικόνα 43: Java mail configuration

Απόσπασμα κώδικα που δείχνει τη σύνταξη του e-mail και την αποστολή:

```java
public void email(String mail, Long minutes, HeartRateCategoryEnum category, Long min, Long max)
        throws MessagingException {

    final String subject = mailProp.getMailSubject();
    final String text = "Goodmorning, " + '\n' + '\n' + "These dates declared in this file describe "
            + "the Heart-Rate of the user which was at its " + category.d() + " which means between "
            + min + " and " + max + " " + "for more than " + minutes + " minutes during these days per day." + '\n'
            + "Check it out please as soon as possible and take care." + '\n' + '\n' + "Hope we've helped. Keep on";

    Properties properties = new Properties();
    properties.put("mail.smtp.starttls.enable", mailProp.getMailSmtpStartEnable());
    properties.put("mail.smtp.auth", mailProp.getMailSmtpAuth());
    properties.put("mail.smtp.host", mailProp.getMailSmtpHost());
    properties.put("mail.smtp.port", mailProp.getMailSmtpPort());

    Session session = Session.getInstance(properties, new javax.mail.Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(mailProp.getUsername(), mailProp.getPassword());
        }
    });
```

Εικόνα 45: Code snippet

```java
        Multipart multipart = new MimeMultipart();
        BodyPart messageBodyPart = new MimeBodyPart();
        Message message = new MimeMessage(session);
        DataSource source = new FileDataSource(mailProp.getFileName());

        message.setFrom(new InternetAddress(mailProp.getSendFrom()));
        message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(mail));
        message.setSubject(subject);
        message.setSentDate(new Date());
        messageBodyPart.setText(text);
        multipart.addBodyPart(messageBodyPart);

        messageBodyPart = new MimeBodyPart();
        messageBodyPart.setDataHandler(new DataHandler(source));
        messageBodyPart.setFileName(mailProp.getFileName());
        multipart.addBodyPart(messageBodyPart);
        message.setContent(multipart);

        // sends the notification mail
        Transport.send(message);
```

Εικόνα 44: Code snippet

Για την αποθήκευση των δεδομένων

Mongo Database

```
<!-- database dependencies -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

Εικόνα 46: Mongo database dependency

Η MongoDB λοιπόν είναι μια **NoSQL** document-oriented database που δημιουργήθηκε το 2007 και το τελικό όνομα της που με αυτό έγινε και γνωστή καθιερώθηκε το 2013. Τα **documents**, δηλαδή τα αντίστοιχα rows ενός table της SQL, δημιουργούνται και αποθηκεύονται σαν **BSON** αρχεία, δηλαδή Binary JSON.

Κάποια από τα σημαντικότερα **θετικά** της βάσης αυτής είναι η ταχύτητα αποθήκευσης, το scalability που προσφέρει και τα απολύτως **δυναμικά schemas** που προσφέρει χωρίς τυποποιημένη δομή και προκαθορισμένους τύπους δεδομένων.

Έχει όμως και σημαντικά **αρνητικά** χαρακτηριστικά που μπορούν να απωθήσουν τους ενδιαφερόμενους. Δεν υποστηρίζει λοιπόν επικύρωση των δεδομένων όπως και **transactions** (ασφαλείς καθορισμένες συναλλαγές) πράγμα που καθιστά τον χρήστη υπεύθυνο για τα δεδομένα που εισάγει. Αυτό απαιτεί λοιπόν και αυξημένες ανάγκες συντήρησης και σωστή διαχείριση των δεδομένων.

Μια γρήγορη συντακτική αντιστοιχία με την SQL:

return all book titles over $10

```
SELECT title FROM book
  WHERE price > 10;
```

```
db.book.find(
  { price: { &gt;: 10 } },
  { _id: 0, title: 1 }
);
```

The second JSON object is known as a **projection**: it sets which fields are returned (`_id` is returned by default so it needs to be unset).

count the number of SitePoint books

```
SELECT COUNT(1) FROM book
WHERE publisher_id = 'SP001';
```

```
db.book.count({
"publisher.name": "SitePoint"
});
```

This presumes denormalized documents are used.

return the number of book format types

```
SELECT format, COUNT(1) AS `total`
          FROM book
        GROUP BY format;
```

```
db.book.aggregate([
    { $group:
      {
    _id: "$format",
   total: { $sum: 1 }
      }
    }
]);
```

This is known as **aggregation**: a new set of documents is computed from an original set.

Εικόνα 48: SQL -> Mongo (1) (πηγή: mongodb.com)

insert a new book record

```
INSERT INTO book (
`ISBN`, `title`, `author`
)
VALUES (
'9780992461256',
'Full Stack JavaScript',
'Colin Ihrig & Adam Bretz'
);
```

```
db.book.insert({
  ISBN: "9780992461256",
title: "Full Stack JavaScript",
author: "Colin Ihrig & Adam Bretz"
});
```

update a book record

```
UPDATE book
  SET price = 19.99
WHERE ISBN = '9780992461256'
```

```
db.book.update(
{ ISBN: '9780992461256' },
{ $set: { price: 19.99 } }
);
```

Εικόνα 47: SQL -> Mongo (2) (πηγή: mongodb.com)

Και μια γρήγορη εννοιολογική αντιστοιχία με την SQL:

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	`$lookup`, embedded documents
primary key	primary key
Specify any unique column or column combination as primary key.	In MongoDB, the primary key is automatically set to the _id field.
aggregation (e.g. group by)	aggregation pipeline

Εικόνα 49: SQL -> Mongo (3) (πηγή: mongodb.com)

- Η εφαρμογή μας έχει σαν κύρια βάση την MongoDB και εκμεταλλεύεται τα θετικά της καθώς η ταχύτητα παίζει σημαντικό ρόλο λόγω και του μεγάλου αριθμού δεδομένων που αποθηκεύουμε και ζητάμε κατά την υλοποίηση της. Χρησιμοποιούμε την λειτουργία του MongoTemplate για τα operations μας με τη βάση, το οποίο είναι ένα εργαλείο του Spring για αυτόν τον λόγο, δηλαδή για γρήγορες και εύκολες συναλλαγές και επικοινωνία με τη βάση δεδομένων.

Εδώ βλέπουμε τα configuration για τη βάση.

```
# mongodb configuration
spring:
  data:
    mongodb:
      host: localhost
      database: fitbit
      port: 27017
```

Εικόνα 50: Mongo configuration

Εδώ γίνεται χρήση του find και του insert.

```
private void saveByMonth(String collection, String newCollection) {
    mongoTemplate.findAll(CommonDataSample.class, collection).forEach(v -> {
        v.setDateTime(v.getDateTime().substring(0, 7));
        mongoTemplate.insert(v, newCollection);
    });
}
```

Εικόνα 52: Code snippet

```
    DBObject filteredValue = (DBObject) dataToInsert.get(fcollection);
    mongoTemplate.insert(filteredValue, collection);
} else {
    BasicDBList filteredValue = (BasicDBList) dataToInsert.get(fcollection);
    mongoTemplate.insert(filteredValue, collection);
```

Εικόνα 51: Code snippet

εδώ ένα παράδειγμα χρήσης του findAll που φέρνει όλα τα δεδομένα από τη βάση.

```
if (parameters.contains("sleep")){
    List<CommonDataSample> efficiencyData = mongoTemplate.findAll(CommonDataSample.class, CollectionEnum.S_EFFICIENCY.d());
    List<CommonDataSample> afterWakeUpData = mongoTemplate.findAll(CommonDataSample.class, CollectionEnum.S_MINUTES_AFTER_WAKE_UP.d());
    List<CommonDataSample> asleepData = mongoTemplate.findAll(CommonDataSample.class, CollectionEnum.S_MINUTES_ASLEEP.d());
    List<CommonDataSample> awakeData = mongoTemplate.findAll(CommonDataSample.class, CollectionEnum.S_MINUTES_AWAKE.d());
    List<CommonDataSample> toFallAsleepData = mongoTemplate.findAll(CommonDataSample.class, CollectionEnum.S_MINUTES_TO_FALL_ASLEEP.d());
    List<CommonDataSample> inBedData = mongoTemplate.findAll(CommonDataSample.class, CollectionEnum.S_TIME_IN_BED.d());
```

Εικόνα 53: Code snippet

Και εδώ δημιουργούνται τα νέα collections (αντίστοιχα Tables της SQL) και σβήνονται τα παλιά.

```
collections.stream().forEach(collectionName -> {

    if (mongoTemplate.collectionExists(collectionName.d())) {
        mongoTemplate.dropCollection(collectionName.d());
        mongoTemplate.createCollection(collectionName.d());
    } else {
        mongoTemplate.createCollection(collectionName.d());
    }
});
```

Εικόνα 54: Code snippet

4.2 Part 2 – Προβολή των δεδομένων (Data visualization)

4.2.1 Τεχνολογίες και Εργαλεία που χρησιμοποιήθηκαν

Για την υλοποίηση την web εφαρμογής, χρησιμοποιήθηκαν οι ακόλουθες γλώσσες προγραμματισμού:

1. HTML
2. CSS
3. JavaScript
4. Visual Basic .NET

Ενώ , για την καλύτερη διαμόρφωση της εφαρμογής, χρησιμοποιήθηκαν ειδικά εργαλεία – βιβλιοθήκες, όπως

- Bootstrap
- Bootstrap date picker
- Bootstrap select picker
- Font Awesome
- JQuery
- Highcharts JS

Οι πρώτες δυο γλώσσες προγραμματισμού , σχετίζονται με το σχεδιασμό της εφαρμογής, το «front end» κομμάτι, ενώ οι άλλες δυο σχετίζονται με την επικοινωνία του client και του server, «back end» κομμάτι αυτού του μέρους της εφαρμογής. Ας περιγράψουμε, λοιπόν συνοπτικά, τη χρήση και τη σημασία των παραπάνω γλωσσών προγραμματισμού.

HTML

Η **HTML** (HyperText Markup Language) είναι η κυρία γλώσσα για την κατασκευή ιστοσελίδων. Αποτελείται από ετικέτες (tags). Τα tags περικλείονται από τα σύμβολα μικρότερο “<” και μεγαλύτερο “>”, παρουσιάζονται συνήθως σε ζευγάρια, ένα που δηλώνει την έναρξη της συγκεκριμένης ετικέτας και ένα που δηλώνει τη λήξη, και είναι της μορφής (<HTML></HTML>), όπου αναμεσά στις ετικέτες αυτές, οι σχεδιαστές ιστοσελίδων μπορούν να εισάγουν κείμενο , εικόνες, πίνακες ,κουμπιά κ.α.

Ο web browser διαβάζει κάθε έγγραφο HTML, το ερμηνεύει και συνθέτει τις σελίδες απεικόνισης.

CSS

Η **CSS** (Cascading Style Sheets – Διαδοχικά φύλλα στυλ) χρησιμοποιείται για τον έλεγχο εμφάνισης ενός εγγράφου (δηλαδή μιας ιστοσελίδας) που έχει γραφτεί σε γλώσσα σήμανσης (**HTML** , **XHTML**)

Είναι μια γλώσσα υπολογιστή και χρησιμοποιείται για να κάνει πιο όμορφη μια ιστοσελίδα, αναπτύσσοντάς την στιλιστικά , προσθέτοντας χρώματα , στοίχιση , μέγεθος γραμματοσειράς κτλ.

JavaScript

Η **JavaScript** είναι διερμηνευμένη γλώσσα προγραμματισμού, δηλαδή η κανονική υλοποίηση της γλώσσας είναι διερμηνέας ή μεταγλωττιστής, δυναμική, με συναρτήσεις ως αντικείμενα πρώτης τάξης. Χρησιμοποιείται για να υπάρχει επικοινωνία με τον χρήστη , για την ανταλλαγή δεδομένων, και για να αλλάζει δυναμικά την απεικόνιση του εγγράφου.

Είναι μια γλώσσα επηρεασμένη από τη **C**, ενώ κάποια ονόματα και συμβάσεις ονομασίας είναι ίδια με την **Java**, παρόλο αυτά δεν σχετίζονται καθόλου μεταξύ τους. Οι βασικές αρχές της προέρχονται από τις γλώσσες Self και Scheme. Τέλος η JavaScript υποστηρίζει αντικειμενοστραφές ,προστακτικό και συναρτησιακό στυλ προγραμματισμού.

IIS Web Server

Το **IIS** είναι ένα ευέλικτο γενικής χρήσης εργαλείο τύπου **web sever** από την Microsoft για συστήματα windows που έχουν ως σκοπό την εξυπηρέτηση HTML σελίδων ή και αρχείων. Χρησιμοποιείται το **HTTP** πρωτόκολλο επικοινωνίας για ανταλλαγή πληροφοριών. Εμείς τον συγκεκριμένο web server τον αξιοποιούμε στο δεύτερο μέρος της εφαρμογής που αφορά την προβολή των δεδομένων και συγκεκριμένα για την υποστήριξη της σελίδας που δημιουργούμε.

Visual Basic .NET

Η **Visual Basic** , είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου (οι γλώσσες υψηλού επιπέδου χρησιμοποιούν ως εντολές απλές αγγλικές λέξεις και ακολουθούν αυστηρούς κανόνες σύνταξης), η οποία έχει αναπτυχθεί από την Microsoft και αποτελεί τη μετεξέλιξη της παλαιότερης έκδοσης με το όνομα GW BASIC για το λειτουργικό σύστημα MS-DOS. Υποστηρίζει ταχεία ανάπτυξη εφαρμογών (RAD) σε γραφικό περιβάλλον (GUI) , πρόσβαση στη βάση δεδομένων και δημιουργία αντικειμένων . Είναι μία αρκετά πλούσια γλώσσα, η οποία έχει σχεδιαστεί για να είναι εύκολη στην εκμάθηση και τον χειρισμό. Η **Visual Basic .NET** παρουσιάστηκε το 2002 από την **Microsoft** ως διάδοχος

της Visual Basic και υλοποιείται στο **.NET Framework** , στο οποίο θα αναφερθούμε παρακάτω αναλυτικά.

Τέλος , ας εξηγήσουμε με απλά λόγια , τη χρησιμότητα και τον σκοπό των εργαλείων – βιβλιοθηκών που χρησιμοποιήσαμε.

Bootstrap

Το **bootstrap** είναι μία συλλογή εργαλείων , που περιέχει html και CSS και χρησιμοποιείται σαν «βάση» για τη δημιουργία web ιστοσελίδων ή web εφαρμογών. Απαιτεί την λειτουργία του JQuery, όπου σαφώς έχουμε χρησιμοποιήσει και εξηγούμε παρακάτω. Η κύρια λειτουργία του, που το καθιστά σημαντικό, είναι ότι επιτρέπει στην εφαρμογή ή στην ιστοσελίδα να προσαρμόζεται σε διάφορα μεγέθη εικόνας (κινητά, tablet, υπολογιστές). Επίσης παρέχει μεγάλη ποικιλία «έτοιμου κώδικα» που προσθέτει στυλ σε φόρμες , όπως είναι οι πίνακες, τα κουμπιά, λίστες , εικόνες κ.τ.λ.

Bootstrap date picker

Το **Bootstrap date picker** παρέχει ένα ευέλικτο widget ημερολογίου, με ωραίο στυλ και πολλές δυνατότητες για τη διευκόλυνση του προγραμματιστή. Αυτός είναι ο λόγος που το χρησιμοποιήσαμε.

Bootstrap Select picker

Το **bootstrap select picker**, είναι μία βιβλιοθήκη που χρησιμοποιείται για τις λίστες, που απεικονίζονται στην εφαρμογή. Προσθέτει λειτουργικότητα και στυλ στις απλές λίστες που χρησιμοποιεί η html.

Font awesome

Το **Font Awesome** , είναι μία βιβλιοθήκη , η οποία παρέχει έτοιμα επεκτάσιμα εικονίδια (με βάση το CSS και το LESS), όπου μπορούν να προσαρμοστούν άμεσα (μέγεθος , χρώμα κ.α.).

JQuery

JQuery είναι μία βιβλιοθήκη σε JavaScript, η οποία απλοποιεί την υλοποίηση των scripting, κάνοντας, κατά την κατασκευή μια web εφαρμογής ή μίας ιστοσελίδας, ωραία και σύγχρονα πράγματα χωρίς κόπο. Τα τελευταία χρόνια είναι πολύ διαδεδομένη, γιατί έχει σημαντικά πλεονεκτήματα όπως:

- Λειτουργεί σε όλους τους browsers σωστά

- Είναι πολύ ελαφριά
- Αλλαγές στις ιδιότητες της HTML και της CSS , βάσει των ενεργειών του χρήστη
- Έχει μεγάλη επεκτασιμότητα με τη χρήση άλλων βιβλιοθηκών (plug-ins)

Τα **Ajax** request γίνονται εύκολα και γρήγορα. * Ajax είναι μία τεχνολογία που χρησιμοποιείται για την αμφίδρομη επικοινωνία μεταξύ του client και του server. Το σημαντικό χαρακτηριστικό της τεχνολογίας είναι πως επιτρέπει της ανανέωση μέρους μίας σελίδας χωρίς αυτή να γίνει reload.

Highchart.JS

Highchart.JS είναι μία βιβλιοθήκη, που δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν εύκολα και γρήγορα διαδραστικά διαγράμματα, σε μία ιστοσελίδα ή σε μία web εφαρμογή.

4.2.2 Διαδικτυακή εφαρμογή (Web application)

Πριν εξηγήσουμε , λοιπόν, τι είναι η web εφαρμογή, θα πρέπει να κατανοούμε πλήρως τι είναι μία εφαρμογή.

Εφαρμογή ονομάζεται ένα λογισμικό, που εγκαθίσταται σε ηλεκτρονικούς υπολογιστές με σκοπό την εκτέλεση ορισμένων διεργασιών και σκοπών , για να εξάγει το επιθυμητό αποτέλεσμα.

Μία Web εφαρμογή, είναι το ίδιο με μία εφαρμογή, με τη διαφορά ότι δεν χρειάζεται να γίνει εγκατάσταση στον ηλεκτρονικό υπολογιστή , αλλά είναι προσβάσιμη από παντού μέσω του Internet ή κάποιου τοπικού δικτύου. Αυτό όπως είναι λογικό , επιφέρει πλεονεκτήματα αλλά και μειονεκτήματα.

Κάποια από τα πλεονεκτήματα που έχουμε σε σύγκριση με τις τοπικές εφαρμογές, είναι

- Έχεις άμεση πρόσβαση οποιαδήποτε στιγμή και από οποιαδήποτε συσκευή, με μόνη προϋπόθεση την πρόσβαση στο ίντερνετ
- Τρέχουν σε όλα τα λειτουργικά συστήματα, δεν επηρεάζεται δηλαδή από αυτό, καθώς εκτελείται στο διαδίκτυο.
- Είναι πιο ελαφριές, καθώς δεν καταναλώνουν πόρους στο σύστημα και φυσικά ούτε χώρο στο δίσκο του χρήστη
- Παρέχουν γρηγορότερη αναβάθμιση σε σύγκριση με τις τοπικές εφαρμογές, καθώς η αναβάθμιση γίνεται απευθείας στον εξυπηρετητή που φιλοξενεί την εφαρμογή και οι χρήστες έχουν το αναβαθμισμένο πρόγραμμα αμέσως. Στην αντίθετη περίπτωση των τοπικών

εφαρμογών, η αναβάθμιση θα πρέπει να γίνει ξεχωριστά στον κάθε έναν υπολογιστή

Ενώ κάποια από τα μειονεκτήματα των web εφαρμογών, είναι

- Δεν μπορείς να χρησιμοποιήσεις την εφαρμογή εκτός διαδικτύου. (Εφαρμογές με την HTML5 έχουν τη δυνατότητα χρήσης της εφαρμογής εκτός διαδικτύου, αλλά κάτι τέτοιο θα πρέπει να έχει προβλεφθεί από την αρχή ώστε να έχουν ληφθεί και τα κατάλληλα μέτρα, διαφορετικά δεν είναι εφικτό και πάλι).
- Δεν υπάρχει πλήρης συμβατότητα με τους περιηγητές. Αυτό συμβαίνει , γιατί πρώτον , την HTML5 που είναι η τελευταία έκδοση της HTML και παρέχει πολλές δυνατότητες , δεν την υποστηρίζουν, ακόμα, όλοι οι περιηγητές και δεύτερον, γιατί υπάρχει περίπτωση κάποιο χαρακτηριστικό της εφαρμογής να μην δουλεύει σωστά σε όλους τους περιηγητές, ή ακόμα και να μην δουλεύει καθόλου. Τέλος μπορεί μελλοντική έκδοση του περιηγητή να μην υποστηρίζει πλέον κάποια από τα στοιχεία της εφαρμογής, γιατί θεωρεί ότι δεν έχουν μέλλον στις εφαρμογές διαδικτύου και τα εγκαταλείπει. Παρατηρούμε λοιπόν, ότι οι περιηγητές είναι αρκετά ευάλωτοι σε τέτοια θέματα, προκαλώντας συχνά δυσλειτουργίες.
- Η άμεση αναβάθμιση τοποθετείται και στα μειονεκτήματα, καθώς γίνεται χωρίς την έγκριση των χρηστών. Επομένων οποιαδήποτε τυχόν σφάλματα, θα εντοπιστούν από τους χρήστες κατόπιν αναβάθμισης.

Τέλος , θα μπορούσαμε να ερωτηθούμε, ποια είναι η διαφορά της web εφαρμογής με μία ιστοσελίδα.

Η διαφορά τους , λοιπόν, είναι ότι ο βασικός σκοπός μίας ιστοσελίδας είναι η ενημέρωση, δηλαδή να πληροφορήσει τον χρήστη , χρησιμοποιώντας απλά κείμενα , εικόνες κτλ. Με τη web εφαρμογή όμως, ο χρήστης έχει ένα περιβάλλον εργασίας , καθώς μπορεί να ανταλλάξει πληροφορίες , να επεξεργαστεί δεδομένα και να πετύχει τους στόχους του.

4.2.3 Γενικό πλαίσιο ανάπτυξης λογισμικού (Software Framework)

Software Framework, με απλά λόγια είναι, μία πλατφόρμα όπου παρέχει, για ένα σύνολο λειτουργιών, κοινό – έτοιμο κώδικα, με τη μορφή βιβλιοθηκών, όπου ο προγραμματιστής μπορεί να τις χρησιμοποιήσει αυτούσιες ή ακόμα και να τις αντικαταστήσει. Σκοπός του είναι να απλοποιήσει το αναπτυξιακό περιβάλλον , διευκολύνοντας αρκετά το έργο του προγραμματιστή, ώστε να μην αφιερώνει χρόνο για γενικές επαναλαμβανόμενες λειτουργίες, αλλά για τις απαιτήσεις του λογισμικού.

<u>Σε τι διαφέρει τότε από τις κοινές βιβλιοθήκες;</u>

Το Framework , παρέχει ορισμένες λειτουργίες , που το κάνει να διαφέρει, όπως:

1. Προκαθορισμένη Συμπεριφορά

2. Επεκτασιμότητα (όπως αναφέραμε και παραπάνω, ένας χρήστης μπορεί να επεκτείνει ένα framework , αντικαθιστώντας κομμάτια από τον καθορισμένο – έτοιμο κώδικα με δικό του).

3. Μη τροποποίησης στον κώδικα του Framework (ένας χρήστης μπορεί, όπως είπαμε να επεκτείνει το framework για δικό του σκοπό , αλλά όχι να το τροποποιήσει. Ο καθορισμένος κώδικας του Framework παραμένει ως έχει).

4.2.4 Γενικό πλαίσιο περιβάλλοντος .NET (.NET Framework)

Αφού έγινε κατανοητή η σημασία ενός software framework , ας δούμε τώρα το .NET Framework , που χρησιμοποιήσαμε στην εφαρμογή μας. Το Net Framework δημιουργήθηκε από την εταιρεία της Microsoft και χρησιμοποιείται από τους προγραμματιστές για να δημιουργήσουν εφαρμογές πιο εύκολα και παρέχει γλωσσική διαλειτουργικότητα, δηλαδή κάθε γλώσσα να χρησιμοποιεί κώδικα , όπου είναι γραμμένος σε άλλες γλώσσες. Αποτελείται από μία μεγάλη βιβλιοθήκη κατηγοριών που ονομάζεται Framework Class Library (FCL) , καθώς και από το περιβάλλον λογισμικού Common Language Runtime (CLR), μια εφαρμογή εικονικής μηχανής, εκεί όπου εκτελούνται τα προγράμματα που γράφονται για το .NET Framework. Το CLR παρέχει υπηρεσίες όπως η ασφάλεια, η διαχείριση μνήμης και ο χειρισμός εξαιρέσεων. Το FCL είναι αυτό που παρέχει τη διεπαφή χρήστη (user interface), τη σύνδεση με τη βάση δεδομένων, την πρόσβαση δεδομένων, την ανάπτυξη web εφαρμογών , επικοινωνίες δικτύου, αριθμητικούς αλγόριθμους και κρυπτογραφία. Οι προγραμματιστές συνδυάζουν τον δικό τους πηγαίο κώδικα, με άλλες βιβλιοθήκες και με το .NET Framework και έτσι παράγουν το λογισμικό.

[κενή σελίδα]

5.ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

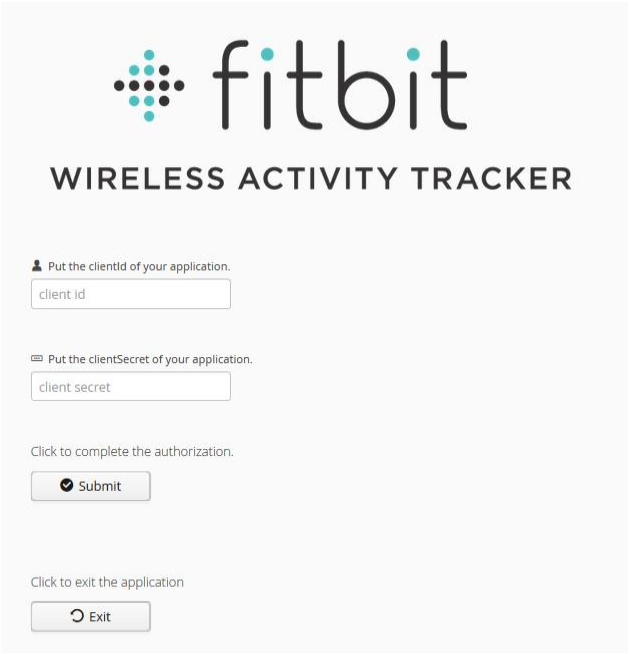
5.1 Part 1 – Λήψη επεξεργασία και αποθήκευση δεδομένων

5.1.1 Σύνδεση και ταυτοποίηση με το Fitbit API

Εικόνα 55: First tab

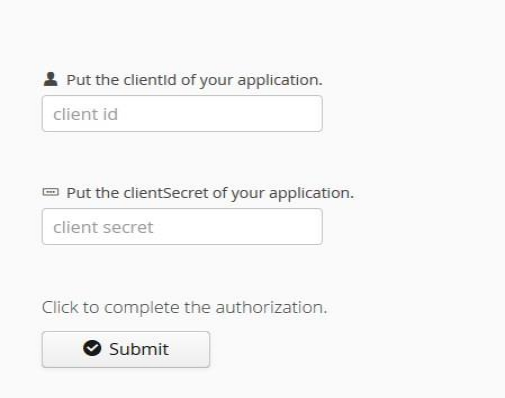
Ξεκινώντας λοιπόν την εφαρμογή μας βρισκόμαστε στον browser μας στην παραπάνω διεύθυνση. Εδώ θα πραγματοποιήσουμε την ταυτοποίηση και θα συνδεθούμε με το Fitbit API ώστε να αποκτήσουμε τον Authorization code όπου θα ξαναστείλουμε στο Fitbit API ώστε να έχουμε το token που χρησιμοποιούμε σε όλες τις κλήσεις μετέπειτα στην εφαρμογή.

Στην παρακάτω φωτογραφία βλέπουμε την αρχική μας σελίδα με τα πεδία που πρέπει να συμπληρώσει ο χρήστης για να προχωρήσει.



Εικόνα 56: User's credentials form

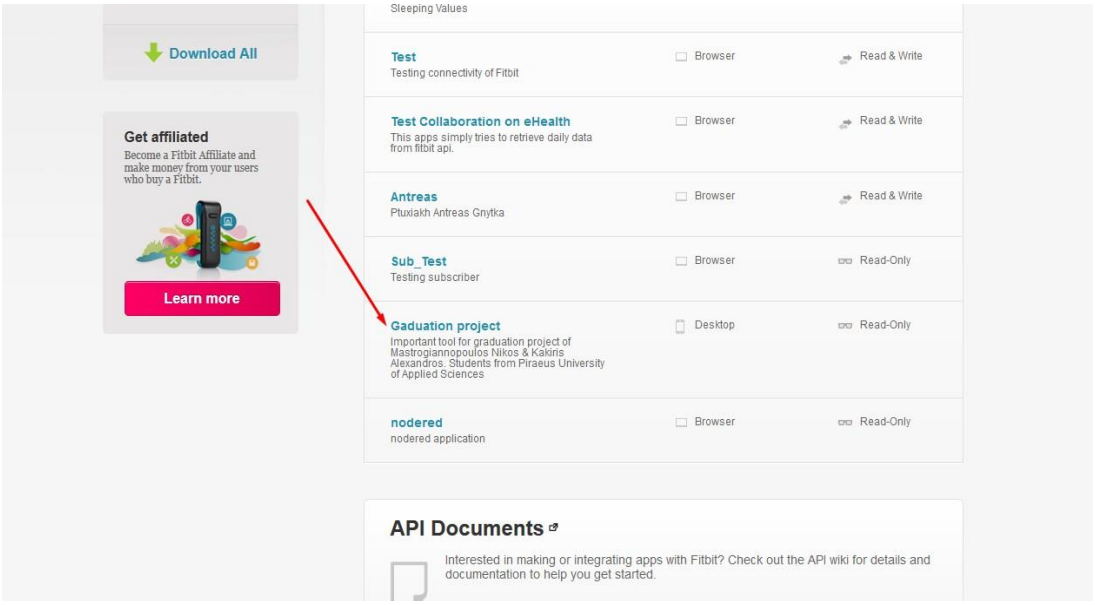
Τα πεδία είναι τα δύο πρώτα textboxes και περιμένουν ένα ClientId και ένα ClientSecret από τον χρήστη.



Εικόνα 57: Client id & client secret form

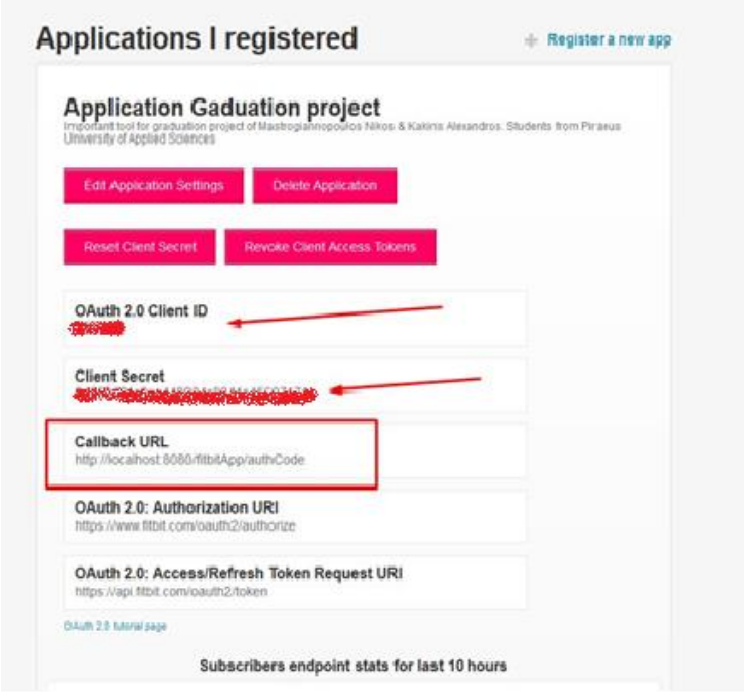
Πατώντας αργότερα το submit ο χρήστης θα ανακατευθυνθεί σε μία Login σελίδα του Fitbit. Θα αναφερθούμε λεπτομερώς παρακάτω.

Αυτά τα credential τα βρίσκει ο χρήστης στην παρακάτω σελίδα “dev.fitbit.com/apps” όπου υπάρχουν όλες οι δηλωμένες εφαρμογές του. Εδώ βρίσκεται και η δική μας εφαρμογή.



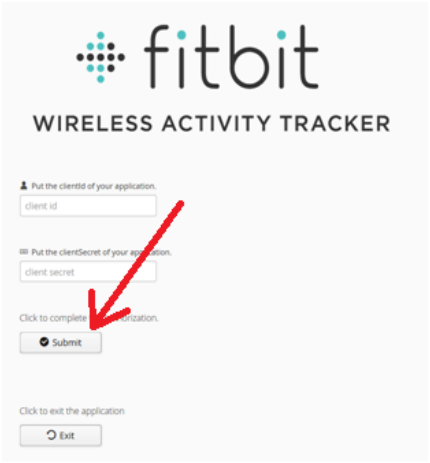
Εικόνα 58: dev.fitbit.com/apps page

Πατώντας πάνω θα δούμε διάφορα στοιχεία για αυτήν και ο χρήστης έχει και την επιλογή να τα αλλάξει αν επιθυμεί.



Εικόνα 59: Our application's credentials

Με βέλη βλέπουμε τα credentials που χρειάζεται ο χρήστης να συμπληρώσει στην εφαρμογή και μέσα σε περίγραμμα το URL, όπου αργότερα αφού συμπληρωθούν τα στοιχεία που πρέπει, θα σταλεί ο Authorization Code που περιμένει η εφαρμογή.



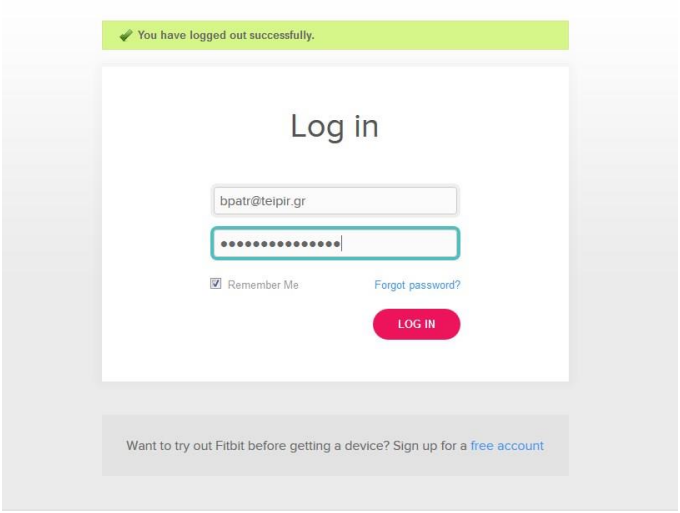
Εικόνα 60: Submit button after complete

Συμπληρώνοντας λοιπόν ο χρήστης τα πεδία που χρειάζονται πατάει το Submit και η εφαρμογή κάνει την κλήση στο Fitbit API και στέλνει εκτός αυτών που συμπλήρωσε μόλις, και τα παρακάτω στοιχεία :

```
authCodeUri1: https://www.fitbit.com/oauth2/authorize?redirect_uri=http://localhost:8080/fitbitApp/authCode&response_type=code&
authCodeUri2: scope=activity%20nutrition%20heartrate%20profile%20settings%20sleep%20social%20weight&expires_in=3600&prompt=login
```

Εικόνα 61: Code snippet

Όπως είπαμε μετά το Submit ο χρήστης ανακατευθύνεται εδώ. Όπου του ζητείται να κάνει Login με τα στοιχεία του λογαριασμού του ώστε να σταλεί ο Authorization Code.



Εικόνα 62: Fitbit login form

Μετά το επιτυχημένο Login λοιπόν μπορούμε να δούμε ότι όντως το Fitbit API έστειλε, στο δηλωμένο URL παραπάνω στα στοιχεία της εφαρμογής μας στο site του Fitbit, τον κωδικό που περιμένουμε.

Εικόνα 63: Authorization Code response from Fitbit

Εδώ λοιπόν σε αυτό το σημείο ολοκληρώθηκε η διαδικασία ταυτοποίησης – authentication που απαιτούν οι διαδικασίες της εφαρμογής μας αλλά και του ίδιου του Fitbit κατά τον διαμοιρασμό των δεδομένων του χρήστη.

Ο χρήστης μετά το Login στη φόρμα του Fitbit και αφού κατευθυνθεί στον controller μας ανακατευθύνεται αυτόματα στο επόμενο βήμα όπου Ακολουθεί η

διαδικασία επιλογής εύρους ημερομηνιών αλλά και δεδομένων που θα ληφθούν.

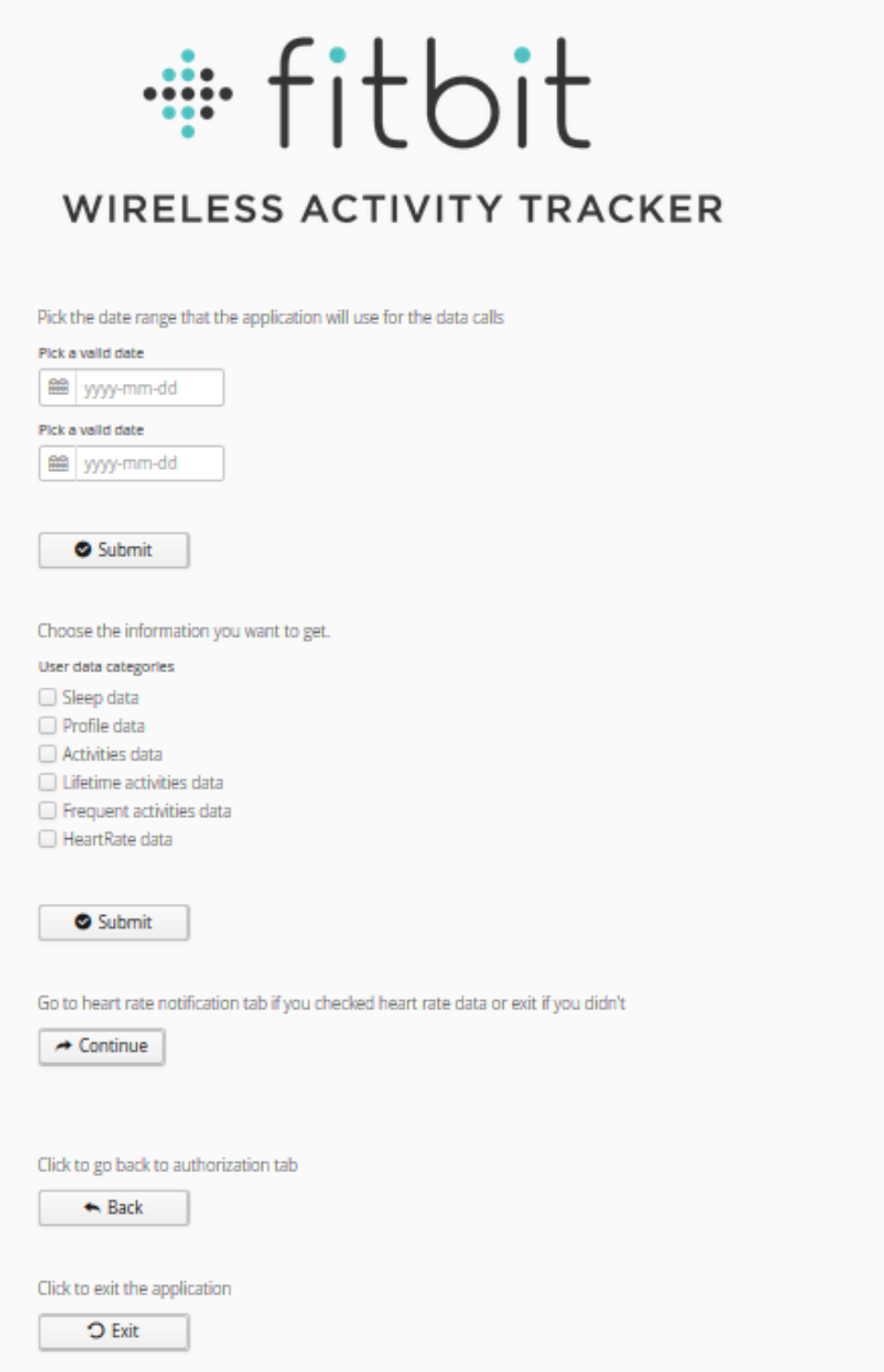
Σημείωση

Να σημειωθεί ότι κατά τη διάρκεια της ολοκλήρωσης αυτού του tab εκτελούνται και άλλες διαδικασίες. Όταν το submit button πατηθεί εκτός του rest call για τον Authorization Code που γίνεται δημιουργούνται και τα collections στην MongoDB που θα χρειαστούν για την μετέπειτα αποθήκευση των δεδομένων που θα ληφθούν. Επίσης αφότου ο controller δεχθεί τον Authorization code και τον αποθηκεύσει εκτελεί κάποιες κλήσεις στο Fitbit API σχετικά με τα δεδομένα steps, floors, distance και heart-rate από την αρχή της ημέρας μέχρι και τη στιγμή που θα εκτελεστεί η εφαρμογή.

Το Exit button ανακατευθύνει τον χρήστη σε άλλο tab στο finalize που θα περιγραφεί λεπτομερώς αργότερα.

Τέλος η διαδικασία είναι αυστηρή και υπάρχει μια συγκεκριμένη σειρά στην εκτέλεσή της οπότε κάθε βήμα για να ξεκινήσει πρέπει να έχει ολοκληρωθεί το προηγούμενο και αυτό ισχύει σε όλα τα tabs της εφαρμογής.

Το δεύτερο tab της διαδικασίας λήψης των δεδομένων είναι της παρακάτω μορφής:



Εικόνα 64: User data tab

5.1.2 Επιλογή εύρους ημερομηνιών

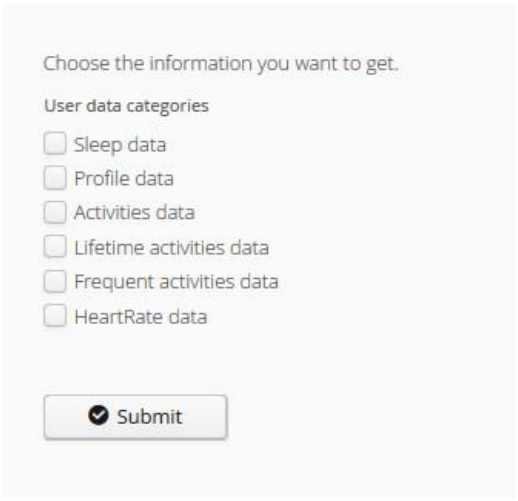
Ο χρήστης δύναται να επιλέξει το εύρος των ημερομηνιών κατά τις οποίες επιθυμεί να συλλέξει τα δεδομένα. Υπάρχει ένα start date και ένα end date. Το start date περιορίζεται από τον Δεκέμβρη του 2014 μέχρι και σήμερα. Το end date πρέπει να ορίζεται χρονικά μετά το start date. Μηνύματα λάθους περιλαμβάνονται σε μη επιθυμητές ενέργειες.

Εικόνα 65: Dates form

Το εύρος των ημερομηνιών που έχουν επιλεγεί αρχικά χωρίζονταν σε τρίμηνα και στέλνονται requests ανά τρίμηνο. Αυτό το κάναμε διότι το Fitbit API δεν πρόσφερε περισσότερα από 100 responses ανά request. Όμως το τελευταίο διάστημα αυτός ο περιορισμός έχει σταματήσει οπότε χωρίζουμε το χρονικό διάστημα σε περιόδους 300 ημερών καθαρά για λόγους ευκολίας διαχείρισης κάποιου σφάλματος.

5.1.3 Επιλογή κατηγοριών για τη λήψη & λήψη

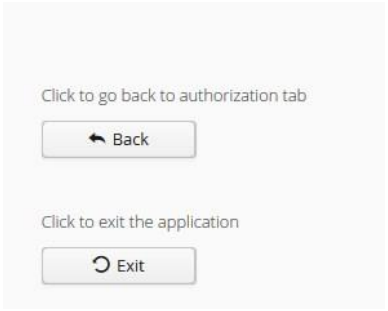
Με την ολοκλήρωση της επιλογής του εύρους ημερομηνιών και αφού ο χρήστης έχει πατήσει το Submit button, ακολουθεί η επιλογή των κατηγοριών των δεδομένων που θα ληφθούν για αυτές τις ημερομηνίες.



Εικόνα 66: User data category form

Ο χρήστης τσεκάρει τις κατηγορίες που επιθυμεί και πατάει το Submit. Οι κατηγορίες αυτές είναι γενικού περιεχομένου και η κάθε μια περιλαμβάνει υποκατηγορίες (βλ. Προγραμματισμός Εφαρμογής -> 1.Λήψη και αποθήκευση δεδομένων -> Αποθήκευση)

Στο τέλος της σελίδας υπάρχουν τα εξής buttons.



Εικόνα 67: Down page buttons

Το **exit** button όπως έχουμε πει και για το προηγούμενο tab οδηγεί στο Finalize tab.

Το **back** button οδηγεί τον χρήστη στο αρχικό tab με τις διαδικασίες ταυτοποίησης.

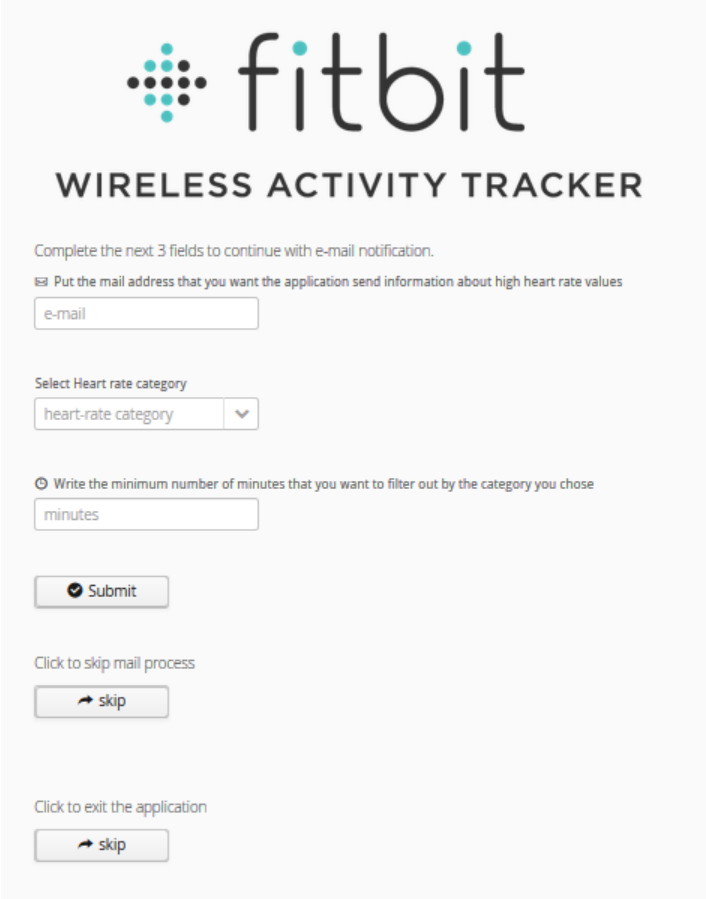
Σημείωση

Οι διαδικασίες είναι και εδώ αυστηρές όσων αφορά τη ροή της εκτέλεσης. Μηνύματα λάθους και ειδοποιήσεις καθοδηγούν τον χρήστη ώστε να λειτουργήσει η εφαρμογή σωστά.

Κατά το Continue button λοιπόν μετά την ολοκλήρωση της λήψης των δεδομένων υπάρχει ένας περιορισμός. Στην περίπτωση που ο χρήστης έχει συμπεριλάβει στις κατηγορίες που επέλεξε να λάβει από το Fitbit API, την κατηγορία Heart Data τότε θα ανακατευθυνθεί στο Notification tab που θα αναλυθεί αναλυτικά στην επόμενη παράγραφο. Σε περίπτωση όμως που δεν περιλαμβάνεται στις επιλογές το Heart Data category πατώντας το Continue button θα ανακατευθυνθεί στο Finalize tab.

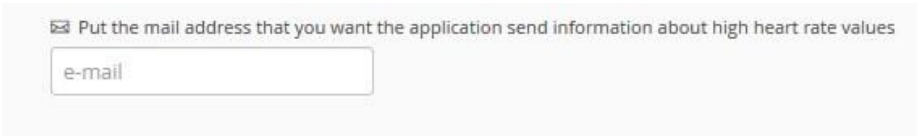
5.1.4 Αποστολή e-mail στον χρήστη και λήψη .pdf αρχείου

Το τρίτο tab είναι το Notification tab και είναι της παρακάτω μορφής.



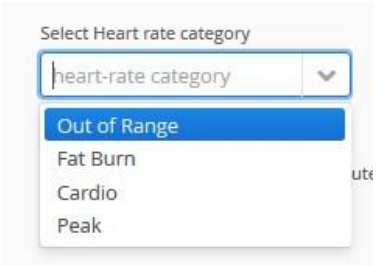
Εικόνα 68: Notification page

Στο πρώτο πεδίο ο χρήστης πρέπει να συμπληρώσει το mail στο οποίο επιθυμεί να σταλεί το ειδοποιητήριο e-mail.



Εικόνα 69: E-mail form

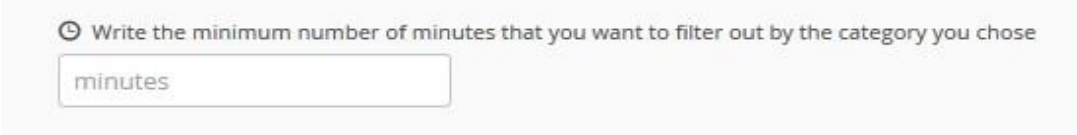
Στο δεύτερο πεδίο ο χρήστης δύναται να επιλέξει την κατηγορία κατά την οποία επιθυμεί φιλτράρει και να περιλαμβάνει το e-mail.



Εικόνα 70: Heart rate category form

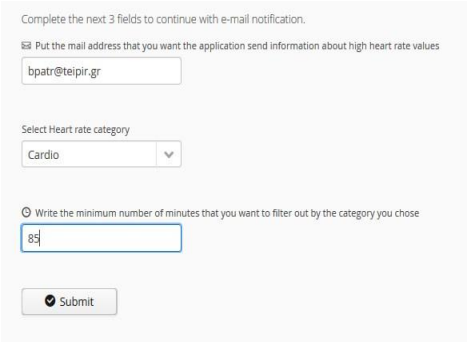
Κάθε κατηγορία επεξηγείται αναλυτικά στο κεφάλαιο «Προγραμματισμός Εφαρμογής»

Τέλος στο τελευταίο πεδίο ο χρήστης δύναται να γράψει τον αριθμό των λεπτών κατά τα οποία θα γίνει το φιλτράρισμα των δεδομένων μαζί με την κατηγορία που έχει δηλωθεί στο προηγούμενο πεδίο.



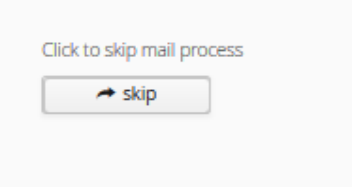
Εικόνα 71: Minutes form

Έχοντας συμπληρώσει και τα τρία πεδία πατάει το Submit button και ολοκληρώνει τη διαδικασία.



Εικόνα 72: Form complete

Παράλληλα ο χρήστης μπορεί απλά να προχωρήσει στο finalize tab της εφαρμογής παραλείποντας για τον οποιοδήποτε λόγο το κομμάτι του notification mail αν επιθυμεί.

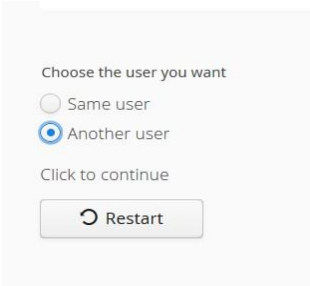


Εικόνα 73: Skip notification tab

Το τελευταίο tab είναι το Exit tab.

Σε αυτό το tab ο χρήστης μπορεί να οδηγηθεί οποιαδήποτε στιγμή από οποιοδήποτε άλλο tab. Πέρα από το ευχαριστήριο μήνυμα ο χρήστης έχει την επιλογή να εκτελέσει ξανά τις διαδικασίες είτε με τον ίδιο χρήστη δηλαδή με χρήση των ίδιων credentials είτε με άλλον.

Εικόνα 74: Finalize tab



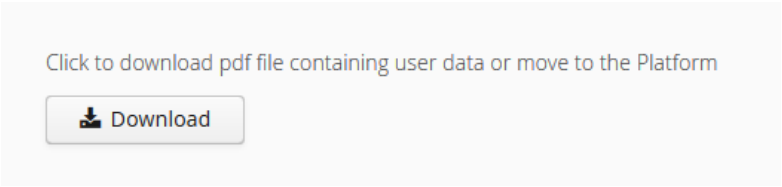
Εικόνα 75: Choose user to continue

Στη περίπτωση λοιπόν που ο χρήστης επιλέξει να κάνει Restart με χρήση των ίδιων credentials δηλαδή επιλέξει το Same user, τότε θα ανακατευθυνθεί στο δεύτερο tab αυτό του User data . Λόγω του ότι τα Credentials είναι τα ίδια η διαδικασία ταυτοποίησης δεν είναι αναγκαία.

Στην περίπτωση όμως που επιλέξει να επανεκτελέσει τη διαδικασία με χρήση credentials άλλου user θα ανακατευθυνθεί στο πρώτο tab αυτό της ταυτοποίησης του χρήστη.

Όμως σε περίπτωση που κατευθυνθεί εδώ από το πρώτο tab δεν έχει δικαίωμα να διαλέξει κάτι καθώς δεν έχει ολοκληρωθεί καν η διαδικασία του authentication οπότε ανακατευθύνεται πίσω στο dashboard tab απευθείας.

Τέλος ο χρήστης έχει τη δυνατότητα να λάβει ένα αρχείο με δεδομένα κατηγορίας activities, sleep και heart-rate για το χρονικό διάστημα που έκανε την λήψη νωρίτερα στο κομμάτι της λήψης. Βέβαια θα λάβει πίνακες με δεδομένα που υπάρχουν στη βάση δεδομένων, που δηλαδή είχε επιλέξει νωρίτερα να ληφθούν. Σε περίπτωση που νωρίτερα δεν είχε επιλέξει καμία από τις τρείς κατηγορίες για λήψη, θα λάβει ένα κενό pdf αρχείο με αντίστοιχο μήνυμα.



Εικόνα 76: Download button for pdf export

Το τελευταίο Button θα τον οδηγήσει απευθείας στην πλατφόρμα όπου πλέον τα δεδομένα παρουσιάζονται σε διαγράμματα και πίνακες αναλυτικά και φιλικά προς τον χρήστη. (το δεύτερο μέρος της εργασίας μας)

5.2 Part 2 – Προβολή των δεδομένων (Data visualization)

Αφού πρώτα έχει προηγηθεί το κομμάτι της λήψης , αποθήκευσης και επεξεργασίας των δεδομένων , ακολουθεί η απεικόνιση τους σε web εφαρμογή.

5.2.1 Είσοδος

Η είσοδος στην εφαρμογή γίνεται αυτόματα μετά την ολοκλήρωση του πρώτου μέρους και αφού έχουν ληφθεί τα δεδομένα.

5.2.2 Αρχική σελίδα

Στην αρχική σελίδα της εφαρμογής , ξεκινώντας από πάνω δεξιά , υπάρχει η επιλογή της αποσύνδεσης, όπου πατώντας το κουμπί επιστρέφετε ξανά στο site του fitbit. Αριστερά υπάρχει το μενού , από το οποίο ο χρήστης , μπορεί να περιηγηθεί στην εφαρμογή. Από το μενού υπάρχει πρόσβαση σε όλες της σελίδες της εφαρμογής , ξεκινώντας γενικά και καταλήγοντας σε πιο αναλυτικές απεικονίσεις .

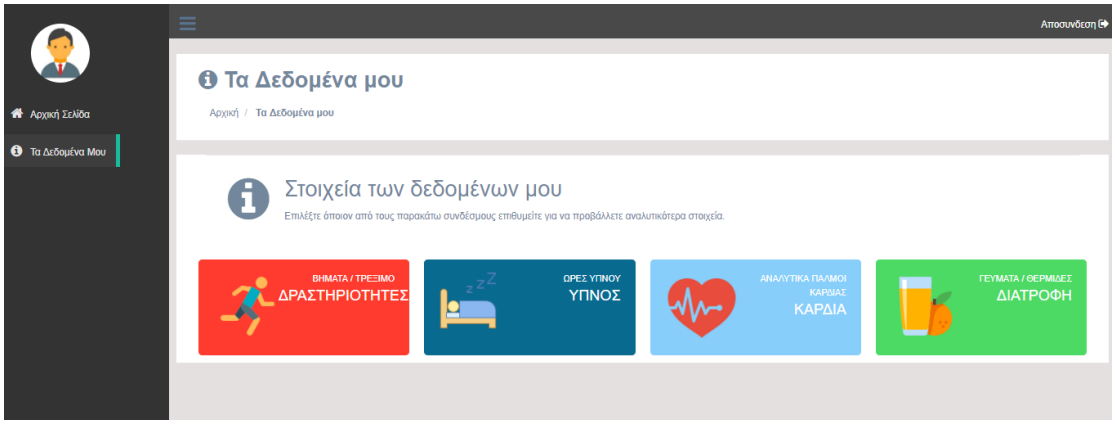
Εικόνα 77: Start page

Τέλος, στο μεγαλύτερο μέρος της αρχικής σελίδας , απεικονίζονται εικόνες σχετικές με την εφαρμογή.

5.2.3 Μενού

Το μενού , στα αριστερά της οθόνης, είναι ορατό σε κάθε σελίδας της εφαρμογής. Ξεκινώντας από πάνω , η πρώτη επιλογή είναι η «Αρχική», όπου πατώντας τη, ο χρήστης, μπορεί να επιστρέψει οποιαδήποτε στιγμή, στην αρχική σελίδα της εφαρμογής.

Στη δεύτερη επιλογή, «Τα δεδομένα μου», απεικονίζονται οι κατηγορίες για τις οποίες υπάρχουν στατιστικά απεικόνισης .

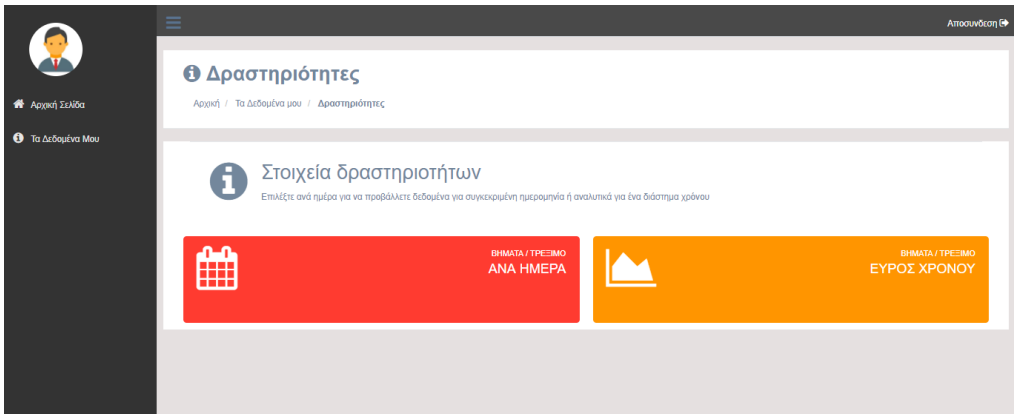


Εικόνα 78: User data category select

Οι κατηγορίες, είναι οι εξής:

1. Δραστηριότητες
2. Ύπνος
3. Καρδιά
4. Διατροφή

Για την καθεμία , από τις παραπάνω κατηγορίες, δίνεται η δυνατότητα απεικόνισης στατιστικών ημέρας, αλλά και εύρος χρόνου (διάστημα μηνών).

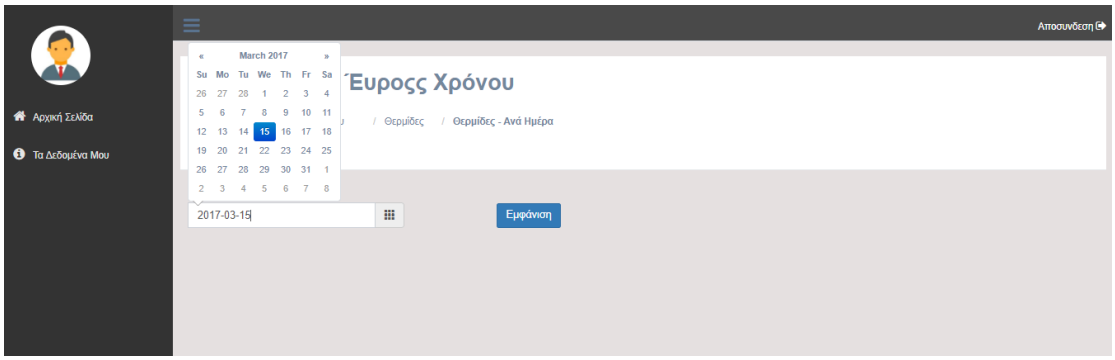


Εικόνα 79: Per day or per month

5.2.4 Προβολή δεδομένων ανά ημέρα

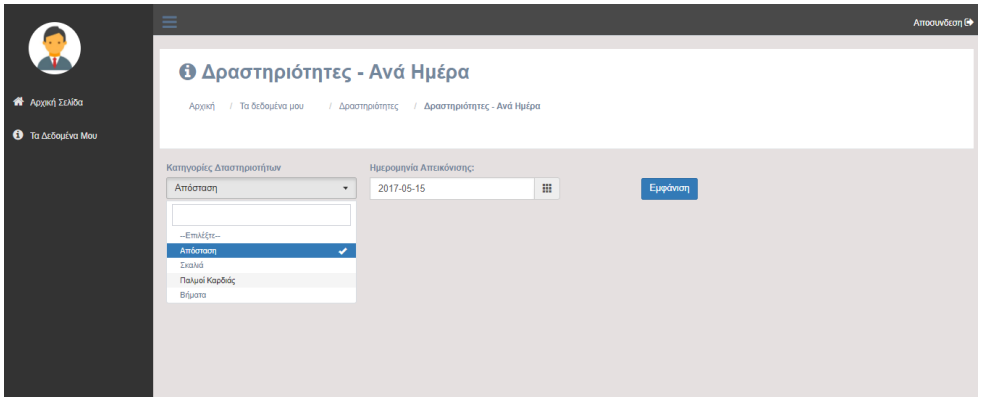
Στην επιλογή «Ανά ημέρα» , η οποία είναι διαθέσιμη σε όλες της κατηγορίες των δεδομένων , ο χρήστης έχει στη διάθεση του να απεικονίσει δεδομένα για οποιαδήποτε ημέρα επιθυμεί.

Στην σελίδα υπάρχει ένα ημερολόγιο , από το οποίο ο χρήστης επιλέγει την ημέρα για την οποία θέλει να δει δεδομένα και έπειτα πατώντας το κουμπί αναζήτησης, αυτά εμφανίζονται στην οθόνη.



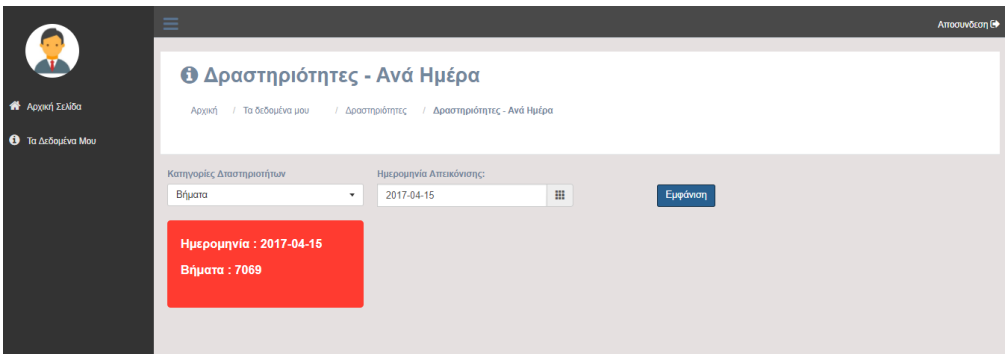
Εικόνα 80: Date picker example

Σε όσες κατηγορίες υπάρχουν και υποκατηγορίες για την απεικόνιση πιο αναλυτικών δεδομένων, υπάρχει αντίστοιχο πεδίο λίστας για να μπορεί να επιλέξει όποια επιθυμεί ο χρήστης.



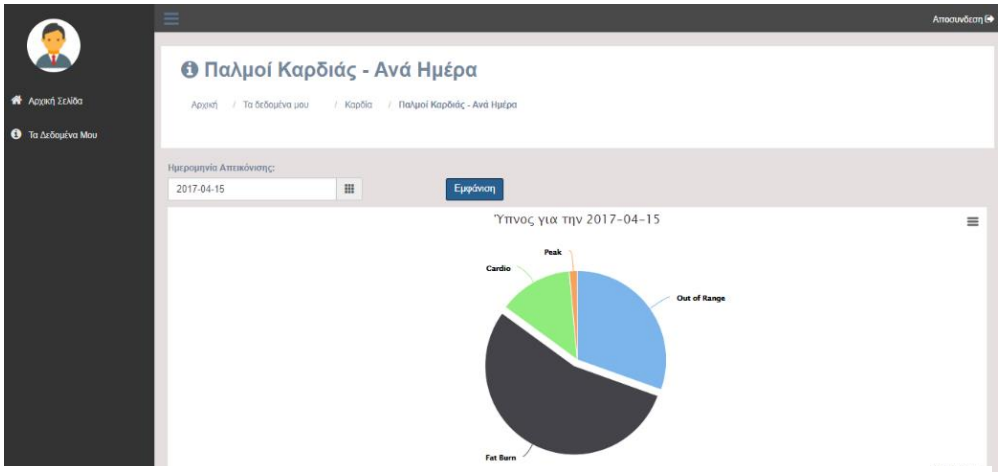
Εικόνα 81: Activities category select example

Στη συγκεκριμένη επιλογή, τα δεδομένα εμφανίζονται με αριθμούς μέσα σε κουτάκια , δείχνοντας επιγραμματικά σε ποια κατηγορία ανήκουν και για ποια ημερομηνία,



Εικόνα 82: Per day results example

ενώ σε ορισμένες περιπτώσεις , ενδέχεται να υπάρχουν και διαγράμματα με τη μορφή πίτας, όπου απεικονίζονται για τη συγκεκριμένη ημέρα διάφορα δεδομένα.

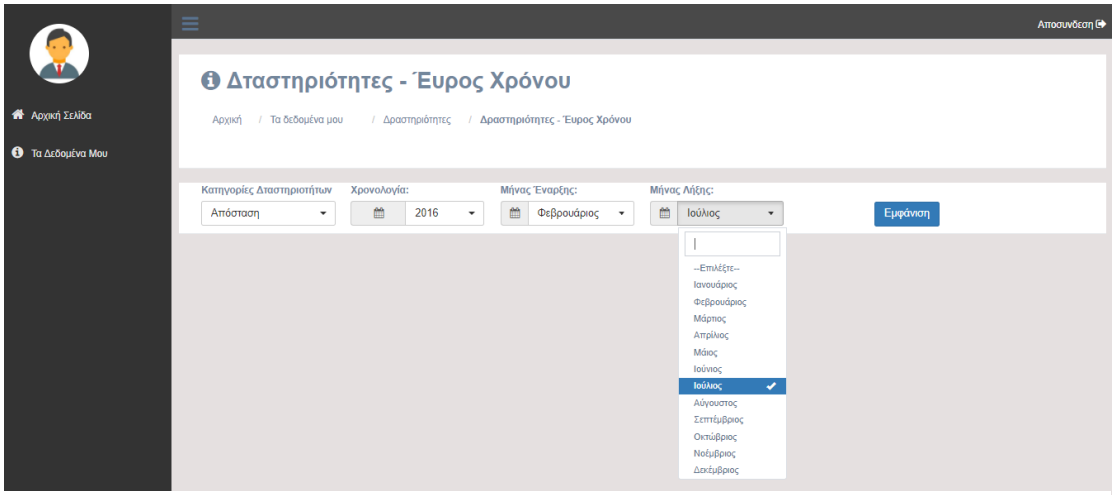


Εικόνα 83: Per day heart rate results example

5.2.5 Προβολή δεδομένων ανά μήνα

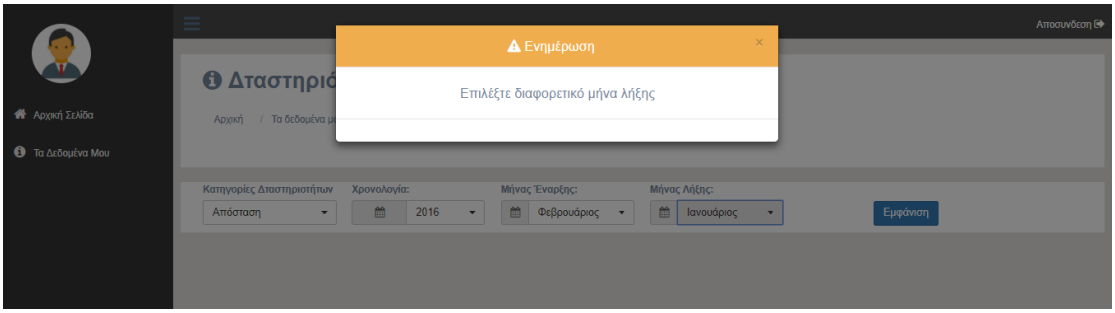
Στην επιλογή «Εύρος χρόνου», όπου δεν διατίθεται σε όλες τις κατηγορίες των δεδομένων , ο χρήστης έχει τη δυνατότητα να απεικονίσει δεδομένα ανά μήνα , για τη χρονική περίοδο που επιθυμεί.

Στη σελίδα τώρα, υπάρχουν δύο ημερολόγια , όπου στο ένα επιλέγεις τον μήνα από όπου θε ξεκινήσει η αναζήτηση και στο άλλο επιλέγεις τον μήνα που θα σταματήσει.



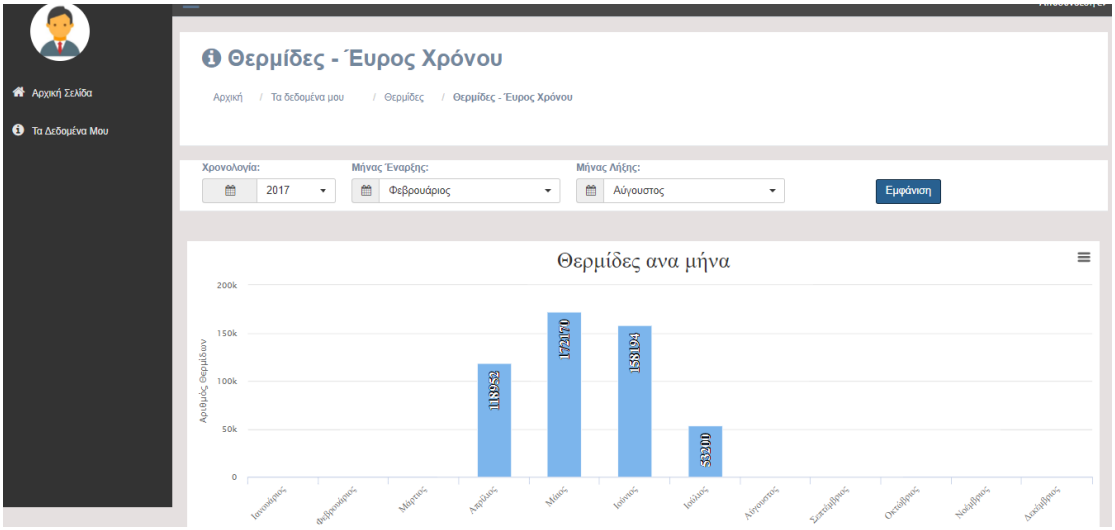
Εικόνα 84: Month select example

Φυσικά ο μήνας που θα σταματήσει η αναζήτηση , δεν μπορεί να είναι μικρότερος του μήνα έναρξης. Αν είναι, εμφανίζεται κατάλληλο μήνυμα.



Εικόνα 85: Zero data notification

Εφόσον , λοιπόν, έχουμε επιλέξει σωστά τους μήνες, θα εμφανιστούν τα αποτελέσματα στην οθόνη μας. Σ’ αυτή την περίπτωση τα αποτελέσματα εμφανίζονται με την μορφή διαγραμμάτων. Συνήθως έχουμε διαγράμματα με μπάρες , όπου κάθε μπάρα αντιπροσωπεύει τα αποτελέσματα για κάθε μήνα ξεχωριστά. Για τους μήνες που δεν γεμίζει η μπάρα, φυσικά δεν υπάρχουν δεδομένα.



Εικόνα 86: Monthly results example

6.ΕΠΙΛΟΓΟΣ

6.1 Προβλήματα και αντιμετώπιση

Η αλήθεια είναι ότι το πρώτο εξάμηνο της υλοποίησης ήταν τελείως αναγνωριστικό καθώς επικεντρωθήκαμε στο διάβασμα και στην κατανόηση κάποιων εννοιών άγνωστες για εμάς μέχρι τότε. « Rest Web Services, OAuth2, APIs » είναι κάποιες από αυτές όπως δυσκολίες αντιμετωπίσαμε και με τις γλώσσες που χρησιμοποιήσαμε όπως « Java Spring Boot, VB.NET, JavaScript » ώστε η ποιότητα του αποτελέσματος και του κώδικα που θα παρουσιάσουμε να είναι αξιόλογη. Επίσης μικρές δυσκολίες μας πρόσφερε και η « MongoDB » που είναι η κύρια βάση δεδομένων που χρησιμοποιήσαμε καθώς δεν είχαμε έρθει σε επαφή μέχρι τότε με NO-SQL databases.

Ακόμη αντιμετωπίσαμε σημαντικά προβλήματα όσον αφορά το κομμάτι της επεξεργασίας των δεδομένων και τον συνδυασμό MongoDB – JSON Objects – Visualization όπως και με το μέρος της επικοινωνίας (Oath2) με τον server και το API του Fitbit ώστε η επικοινωνία να είναι όσο πιο φιλικό γίνεται για τον χρήστη της εφαρμογής.

Τα προβλήματα τελικώς τα αντιμετωπίσαμε επιτυχώς και μάθαμε πολλά πράγματα μέσα από αυτά για την λειτουργία όλων αυτών των εννοιών και τεχνολογιών και τη σωστή χρήση τους. Σε αυτό βοήθησε η επιμονή μας αλλά και η καθημερινή ενασχόληση μας σε συνδυασμό με τις γνώσεις που μας προσφέρουν πλέον οι εργασίες μας οι οποίες μας έδωσαν μεγάλη ώθηση.

6.2 Μελλοντικές αναβαθμίσεις εφαρμογής

Αξίζει να αναφερθούμε σε κάποιες σημαντικές μελλοντικές αναβαθμίσεις όσον αφορά κάποια μέρη της υλοποίησης της εργασίας.

Όσον αφορά το κομμάτι της λήψης και της επεξεργασίας των δεδομένων, θα μπορούν τα δεδομένα που λαμβάνονται να γίνονται « mapping » σε « domain java models » με χρήση του « Spring Data » και να μην αποθηκεύονται κατευθείαν στη βάση δεδομένων χωρίς να έχουν οριστεί νωρίτερα. Αυτό θα βοηθήσει στην ευκολότερη και καλύτερη αξιοποίηση τους. Επίσης η καλύτερη αξιοποίηση των δυνατοτήτων του « Vaadin Framework » θα δημιουργήσει ποιοτικότερα αποτελέσματα όσον αφορά το UI κατά την λήψη των δεδομένων και την αποθήκευση τους ως προς τον χρήστη.

Τώρα όσον αφορά το visualization της εφαρμογής, αντί να λαμβάνει τα δεδομένα από την βάση δεδομένων, θα δημιουργεί «rest calls» προς το backend API και αυτό με τη σειρά του να πρόσφερε τα δεδομένα προς αξιοποίηση. Αυτό θα προσφέρει άμεση σύνδεση του back end με το front end παραλείποντας τον ενδιάμεσο κρίκο με τη βάση δεδομένων που υπάρχει τώρα.

Ακόμη στο μέλλον μπορεί να αξιοποιηθεί αυτός ο μεγάλος αριθμός των δεδομένων με χρήση της γλώσσας προγραμματισμού R. Η συγκεκριμένη γλώσσα είναι η κατάλληλη για στατιστική ανάλυση και visualization δεδομένων καθώς περιλαμβάνει και υποστηρίζει τεχνικές ανάλυσης αριθμητική και όχι μόνο, μαθηματικές συναρτήσεις και γραφικές αρχιτεκτονικές. Σκοπός λοιπόν αυτής της αξιοποίησης είναι η επιτυχημένη διαδρομή στο open data και στην μαζική αξιοποίηση δεδομένων.

Τέλος το Fitbit προσφέρει την δυνατότητα του « Publish - Subscribe » δηλαδή της σύνδεσης του API με την εφαρμογή μας μέσω κάποιων endpoints που θα παρέχουμε προς το Fitbit ώστε να μας στέλνει δεδομένα ανά ορισμένα χρονικά διαστήματα με τη μορφή notifications ώστε να γίνεται update η βάση δεδομένων αυτόματα και με τη σειρά του και το front end site της εφαρμογής.

7.ΒΙΒΛΙΟΓΡΑΦΙΑ

Η βιβλιογραφία μας αποτελείται από διαδικτυακές πηγές και ιστοσελίδες που μας βοήθησαν αρκετά κατά τη διάρκεια της αξιοποίησης και καταγραφής όλων των εργαλείων.

1) http://javabeat.net/java-8-lambda-expressions-example/
2) https://medium.com/javascript-scene/master-the-javascript-interview-what-is-functional-programming-7f218c68b3a0
3) https://www.romexsoft.com/blog/java-8-vs-java-9/
4) https://dzone.com/articles/why-springboot
5) https://www.tutorialspoint.com/spring/spring_beans_autowiring.htm
6) http://www.bestwebframeworks.com/web-framework-review/java/132/vaadin/
7) http://searchsqlserver.techtarget.com/definition/application-server
8) https://dzone.com/articles/spring-boot-embedded-tomcat-example
9) https://javapipe.com/hosting/blog/tomcat-application-server/
10) http://www.studytrails.com/java/java8/java8_date_and_time/
11) https://dzone.com/articles/deeper-look-java-8-date-and
12) http://blogs.perficient.com/delivery/blog/2017/04/19/basic-usage-of-spring-data-jpa/
13) https://cloud.google.com/appengine/docs/standard/java/mail/sending-mail-with-mail-api
14) https://www.tutorialspoint.com/javamail_api/javamail_api_overview.htm
15) https://dzone.com/articles/sql-or-nosql-that-is-the-question
16) https://dzone.com/articles/a-comparison-of-sql-and-nosql-to-simplify-your-dat
17) https://dzone.com/articles/sql-nosql-newsql-comparative-advantages-and-disadvantages
18) https://conceptainc.com/blog/mongodb-a-review-of-a-document-database-used-for-unstructured-data/

19) https://hackernoon.com/mongodb-vs-mysql-comparison-which-database-is-better-e714b699c38b

20) https://dzone.com/articles/mongodb-the-good-the-bad-and-the-ugly

21) https://www.sitepoint.com/sql-vs-nosql-differences/

22) https://docs.mongodb.com/manual/reference/sql-comparison/

23) http://proov.io/2016/08/17/smartphones-to-smart-cities-how-technology-is-impacting-our-lives/?preview=true

24) http://www.naftemporiki.gr/story/1022645/i-epoxi-tou-internet-of-things

25) https://www.sas.com/el_gr/insights/big-data/internet-of-things.html

26) https://www.javatpoint.com/soap-vs-rest-web-services

27) https://www.ibm.com/developerworks/library/ws-restful/index.html

28) https://en.wikipedia.org/wiki/HTTP/2#cite_note-Pratt-14

29) http://searchwindowsserver.techtarget.com/definition/IIS

30) https://www.r-project.org/about.html

31) http://r4ds.had.co.nz/data-visualisation.html

32) bootstrap-datepicker.readthedocs.io

33) silviomoreto.github.io

34) searchwindevelopment.techtarget.com

35) support.microsoft.com

36) bootstrap-datepicker.readthedocs.io

37) www.intel-soft.gr

Όπως μπορούμε να διαπιστώσουμε κύριες πηγές για την εργασίας μας ήταν κάποια σημαντικά και δημοφιλή site με χρήσιμες συμβουλές και καθοδηγήσεις όπως:

mongodb.com, spring.io, dzone.com, microsoft.com, ibm.com, sitepoint.com, sas.com, wikipedia.org, tutorialspoint.com, javabeat.net, naftemporiki.gr, developer.mozilla.org, stackoverflow.com, w3schools.com, techopedia.com, insomnia.gr, howtogeek.com, techtarget.com etc.

[κενή σελίδα]

[κενή σελίδα]