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

Ανάπτυξη Αυτόνομης Εφαρμογής σε Έξυπνο Ρολόι για την Συνεχή Καταγραφή Ζωτικών Ενδείξεων του Χρήστη και την Αξιολόγηση των Μετρήσεων Μέσω Υπολογιστικού Μοντέλου

Διπλωματική Εργασία

Θεόδωρος Κ. Χρονόπουλος
Μιχαήλ Πατσάκης

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής ΕΜΠ

Αθήνα, Αύγουστος 2022

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

Ανάπτυξη Αυτόνομης Εφαρμογής σε Έξυπνο Ρολόι για την Συνεχή Καταγραφή Ζωτικών Ενδείξεων του Χρήστη και την Αξιολόγηση των Μετρήσεων Μέσω Υπολογιστικού Μοντέλου

Διπλωματική Εργασία

Θεόδωρος Κ. Χρονόπουλος
Μιχαήλ Πατσάκης

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Αύγουστο του 2022.

……	……	……...
Π.Τσανάκας	Π.Κωττής	Δ. Κουτσούρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

……

Θεόδωρος Κ. Χρονόπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.
Μιχαήλ Πατσάκης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2022

Στους γονείς μου Κώστα και Βάντα
και στον αδερφό μου Παναγιώτη,

Θοδωρής Κ. Χρονόπουλος

Στους γονεις μου Νικόλα και Ελένη,
και στα αδέρφια μου Γιώργο, Απόστολο και Άλκηστη,

Μιχαήλ Ν. Πατσάκης

Ευχαριστούμε τον καθηγητή μας κ. Παναγιώτη Τσανάκα για την αμέριστη υποστήριξη που μας παρείχε καθ' όλη την διάρκεια της υλοποίησης της παρούσας διπλωματικής εργασίας.

Περιεχόμενα

Ευρετήριο Πινάκων

Ευρετήριο Εικόνων

Περίληψη

Τις τελευταίες δεκαετίες, παράλληλα με τη ραγδαία ανάπτυξη του κλάδου της τεχνολογίας, παρατηρείται αξιοσημείωτη εξέλιξη και στις έξυπνες συσκευές, οι οποίες διεισδύουν ολοένα και περισσότερο στην καθημερινότητά μας και προτιμώνται λόγω των συγκριτικών και αδιαμφισβήτητων πλεονεκτημάτων που παρουσιάζουν, διευκολύνοντας την ζωή του χρήστη. Μια τέτοια συσκευή που με το πέρασμα του χρόνου πολλοί άνθρωποι επιλέγουν, είναι τα έξυπνα ρολόγια. Ανάμεσα σε ποικίλες δυνατότητες που προσφέρουν, τα έξυπνα ρολόγια συλλέγουν live δεδομένα για τη ζωτική κατάσταση του χρήστη με σκοπό την παρακολούθηση της πορείας κατά την διάρκεια της αθλητικής δραστηριότητας, την ανίχνευση ατυχημάτων, την έγκαιρη ενημέρωση του γιατρού, αλλά και την συνεχή μελέτη ενός ασθενή. Έτσι, όλες οι μεγάλες τεχνολογικές εταιρείες έχουν στραφεί στην παραγωγή τέτοιων ρολογιών που, μέσω των αισθητήρων που διαθέτουν, δίνουν την δυνατότητα στους προγραμματιστές να αναπτύξουν εφαρμογές που παρακολουθούν τις ζωτικές ενδείξεις του χρήστη και βγάζουν τα κατάλληλα συμπεράσματα. Στο πλαίσιο αυτό, στόχος της παρούσας διπλωματικής ήταν η ανάπτυξη αυτόνομης εφαρμογής για το Apple Watch Series 7 με σκοπό την καταγραφή των μετρήσεων για τους καρδιακούς παλμούς, την μεταβλητότητα τους (HRV), καθώς και το ποσοστό οξυγόνου στο αίμα με την βοήθεια των ειδικών αισθητήρων που διαθέτει ο συγκεκριμένος τύπος ρολογιού. Στη συνέχεια τα δεδομένα αυτά, αποστέλλονται στο Cloud, όπου και υπόκεινται σε συγκεκριμένη επεξεργασία με σκοπό την ενημέρωση του χρήστη σχετικά με την ποιότητα των μετρήσεων που έλαβε. Παράλληλα, δημιουργήθηκε ιστοσελίδα στην οποία τόσο ο χρήστης όσο και επαγγελματίες στο χώρο της υγείας μπορούν να επιβλέπουν τις λαμβανόμενες μετρήσεις.

Στο πρώτο κεφάλαιο, αναλύονται τα εργαλεία λογισμικού που χρησιμοποιήθηκαν για την εκπόνηση της διπλωματικής. Πιο συγκεκριμένα, στην αρχή προβάλλονται τα τεχνικά χαρακτηριστικά του ρολογιού και στην συνέχεια γίνεται αναφορά στις γλώσσες προγραμματισμού Swift, JavaScript και React, καθώς και στο πρόγραμμα Xcode που μας βοήθησε στην προσομοίωση της εφαρμογής του ρολογιού. Ταυτόχρονα, παρουσιάζονται συνοπτικά οι βασικές βιβλιοθήκες που περιέχονται στον κώδικα που αναπτύξαμε και τα κυρίαρχα πλεονεκτήματα του Cloud και του συγκεκριμένου παρόχου που επιλέχθηκε.

Στο δεύτερο κεφάλαιο, αναλύεται ο στόχος που έχει η ανάπτυξη της συγκεκριμένης εφαρμογής και επισημαίνεται η έλλειψη που έρχεται εκείνη να καλύψει. Έτσι, γίνεται συνοπτική παρουσίαση των δεδομένων που αποθηκεύει, του τρόπου επεξεργασίας τους, της αξίας τους στην ιατρική, καθώς και των δυνατοτήτων μια τέτοιας εφαρμογής.

Το τρίτο κεφάλαιο εστιάζει στην αρχιτεκτονική και στην μεθοδολογία ανάπτυξης κώδικα που ακολουθήθηκε για την υλοποίηση της διπλωματικής. Έπειτα, εξηγείται η σύνδεση της εφαρμογής με την ιστοσελίδα και το μοντέλο αξιολόγησης των μετρήσεων του χρήστη, καθώς και ο τρόπος που οι χρήστες και οι επαγγελματίες υγείας θα πλοηγούνται στις εφαρμογές. Ακολούθως, γίνεται μια εκτενής παρουσίαση του κόστους του έργου. Τέλος, αναφέρονται εναλλακτικοί τρόποι υλοποίησης, επιλέγοντας διαφορετικά εργαλεία και γλώσσες προγραμματισμού.

Στο τέταρτο κεφάλαιο, ακολουθεί ανάλυση των βασικών λειτουργιών του κώδικα που υλοποιήθηκε, τόσο για την ανάπτυξη της εφαρμογής στο ρολόι, όσο και την σύνδεσης της με το Cloud. Επίσης, παρουσιάζεται ο κώδικας της ιστοσελίδας που συλλέγει τα δεδομένα, τα προβάλλει και διαχειρίζεται τους χρήστες και τους επαγγελματίες υγείας. Γίνεται αναφορά και στον κώδικα για τον τρόπο λειτουργίας του υπολογιστικού μοντέλου που προβλέπει αν οι τιμές του χρήστη κυμαίνονται σε φυσιολογικά επίπεδα.

Στο πέμπτο κεφάλαιο συνοψίζονται και παρουσιάζονται κωδικοποιημένα όλα τα συμπεράσματα τα οποία αντλήθηκαν συνολικά από όλα τα ενδιάμεσα στάδια εκτέλεσης της συγκεκριμένης διπλωματικής εργασίας. Επίσης, εμπεριέχονται προτάσεις για μελλοντικές χρήσεις και προεκτάσεις ενός τέτοιου έργου και τα προβλήματα που πρέπει να λυθούν. Στο τέλος, επισυνάπτεται η χρησιμοποιούμενη βιβλιογραφία.

Λέξεις κλειδιά

Έξυπνο ρολόι, αυτόνομη εφαρμογή, Apple Watch Series 7, αισθητήρες, οξυγονομέτρηση, μεταβλητότητα καρδιακών παλμών, καρδιακοί παλμοί, Firebase, Swift, JavaScript, Cloud, WatchOs 8, Realtime Database, Cloud Functions Authentication, υπολογιστικό μοντέλο, λειτουργικό σύστημα, ιστοσελίδα.

Λίστα Ακρωνυμίων

HRV:	Heart Rate Variability – Μεταβλητότητα Καρδιακών Παλμών
GB:	Giga Byte
GHz:	Giga Hertz
LED:	Light-Emitting Diode
RAM:	Random Access Memory
GPS:	Global Positioning System
GLONASS:	Global Navigation Satellite System
QZSS:	Quasi-Zenith Satellite
LLVM:	Low Level Virtual Machine
WWDC:	Worldwide Developers Conference
Inc:	Incorporated
IDE:	Integrated Development Environment
macOS:	computer operating system (OS) for Apple desktops and laptops
iOS:	iPhone Operating System
watchOS:	operating system designed specifically for the Apple Watch wearable device
tvOS:	operating system that runs on the 4th and 5th generation Apple TV digital media player
GUI:	Graphical User Interface
NoSQL:	Non Structured Query Language
API:	Application Programming Interface

SaaS:	Software As A Service
PDF:	Portable Document Format
JSON:	JavaScript Object Notation
SDNN:	Standard Deviation of NN intervals
ΧΑΠ:	Χρόνια Αποφρακτική Πνευμονοπάθεια
ARDS:	Acute Respiratory Distress Syndrome
HTML:	Hypertext Markup Language
CSS:	Cascading Style Sheets
UI:	User Interface
CORS:	Cross-Origin Resource Sharing

Abstract

During the last decades, along with the accelerating development of technology, there has been remarkable progress relating to smart devises, which are being more and more used in daily life because of their indisputable comparative advantages that greatly facilitate users' life. Smart watch is such a devise. Among a variety of possibilities offered, smart watches collect live data regarding user's vital situation aiming at checking on their course during sports activity, accident detection, timely notification of the doctor, as well as continuous observance of a patient. As a result, technology companies are focusing on the production of such watches, which, through the right type of sensors, allow programmers to develop applications that follow users' vital indicators and reach relevant conclusions. In this context, the objective of the present thesis is the development of an Apple Watch Series 7 autonomous application aiming at measuring the heart rate, the heart rate variability (HRV) and the oxygen saturation, through specific sensors embodied in the watch. Subsequently, this data is uploaded to Cloud, where it is subjected to specific processing, in order to inform the user about the quality of the measurements received. At the same time, a website was created where both the user and health professionals can monitor the measurements taken.

In the first chapter, the software components used in the current study are analyzed. More specifically, the technical characteristics of the watch are presented; following that, there is a reference to programming languages Swift, JavaScript and React, as well as to Xcode program, which helped in simulating the watch application. At the same time, the basic libraries contained in the code we developed and the advantages of the Cloud provider we used, are briefly presented.

In the second chapter, we analyze the purpose of developing this specific application through the prism of the need which it is supposed to address. In this way, a brief presentation of data that the application saves, their processing, their value in medicine, as well as, the capabilities of such an application is made.

In the third chapter, there is focus on the architecture applied and in code development methodology, in regards to the realization of this thesis. There is a presentation of the connection of the application in question to the web site and the evaluation model of the user's data as well as the way in which users and health experts will be guided through the application. In addition, the basic cost of the project is highlighted. Alternative ways of implementation are mentioned, through the selection of a variety of tools and programming languages.

In the fourth chapter, there is an analysis of the basic operations of the code that was created, regarding both the development of the watch application and its connection to Cloud. Moreover, the code of the web site is presented, that collects data, projects them, and at the same time manages both users and health experts. There is also a reference to the code of the computational model, which defines whether the user's data lie within the range of normal indications.

In the fifth chapter there is a summary of the conclusions reached through all the intermediary stages of this dissertation. There is also reference to future use cases and expansions of such a project including problems which need to be resolved. At the end, the bibliography that we used, is attached.

Key words

Smartwatch, standalone application, Apple Watch Series 7, sensors, oxygen saturation, heart rate variability, heart rate, Firebase, Swift, JavaScript, Cloud, WatchOs 8, Realtime Database, Cloud Functions Authentication, computational model, operating system, website.

1. Τεχνολογικά Μέσα και Εργαλεία

1.1 Εισαγωγή

Τα τελευταία δέκα χρόνια παρατηρείται αισθητή διείσδυση των έξυπνων συσκευών στη ζωή του ανθρώπου. Από smartphones και τηλεοράσεις μέχρι και έξυπνα ψυγεία, οι συσκευές αυτές γίνονται ολοένα και περισσότερο αναπόσπαστο κομμάτι της καθημερινότητας. Μια κατηγορία αυτών αποτελούν τα έξυπνα ρολόγια, τα οποία τον τελευταίο καιρό φαίνεται να έχουν αυξημένη χρήση και ζήτηση καθώς και ένα συνεχώς διευρυμένο φάσμα δυνατοτήτων που μπορούν να καλύπτουν «ανύπαρκτες» μέχρι τώρα ανάγκες του ανθρώπου. Τα smart watches διαθέτουν αισθητήρες για την μέτρηση σωματικών μεγεθών, εξωτερικών παραγόντων, όπως η θερμοκρασία, το υψόμετρο, η φωτεινότητα καθώς και τη δυνατότητα σύνδεσης και επικοινωνίας, μέσω Bluetooth ή διαδικτύου, για την ανταλλαγή και αποθήκευση δεδομένων σε πραγματικό χρόνο.

Στη σημερινή εποχή, πολλοί άνθρωποι επιθυμούν να έχουν μια καλύτερη εικόνα της σωματικής τους κατάστασης. Οι συνεχώς εναλλασσόμενοι ρυθμοί ζωής, καθώς και οι πολυδιάστατες δραστηριότητες της καθημερινότητας επιφέρουν αλλαγές στη σωματική και ψυχική κατάσταση του ανθρώπου, δημιουργώντας την ανάγκη μιας τακτικής παρατήρησης των αλλαγών και τον προσδιορισμό των παραγόντων που τις προκαλούν. Σε αυτή την ανάγκη τα smart watches και πιο συγκεκριμένα τα Apple Watches, διαθέτουν το απαραίτητο hardware και λογισμικό για να παράξουν μια ολοκληρωμένη εμπειρία χρήστη. Αναλυτικότερα, το Apple Watch μπορεί να πάρει μετρήσεις καρδιακών παλμών, οξυγόνου, δεδομένα για την ποιότητα ύπνου, ηλεκτροκαρδιογράφημα, δίνοντας έμφαση στην ακρίβεια και ταχύτητα λήψης των δεδομένων.

1.2 Το έξυπνο ρολόι Apple Watch Series 7

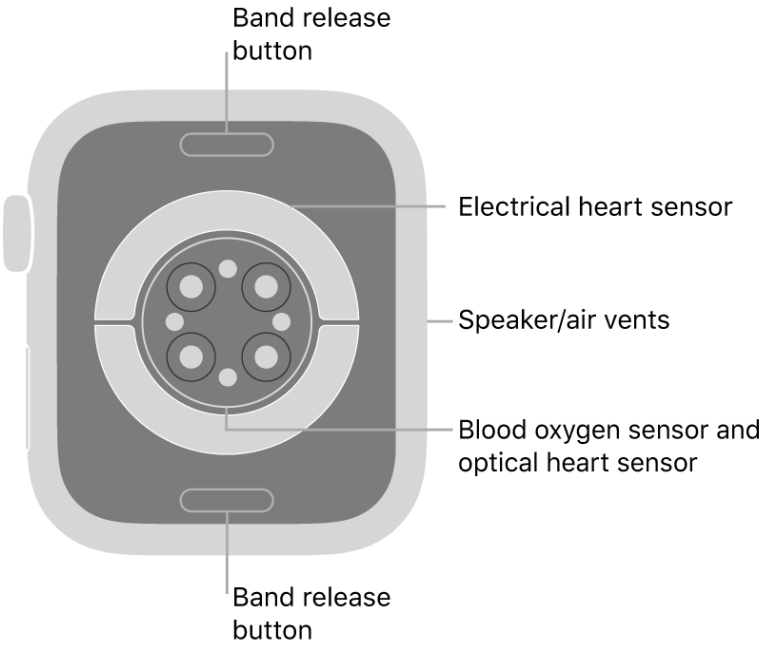
1.2.1 Γενικά

Κυρίαρχη θέση για την εκπόνηση της παρούσας εργασίας έχει η συλλογή των βιοτικών δεδομένων του χρήστη. Για τον σκοπό αυτό επιλέχτηκε η ανάπτυξη μιας εφαρμογής σε έξυπνο ρολόι το οποίο θα διαθέτει τους απαραίτητους αισθητήρες και επιπλέον είναι μία συσκευή φορητή που αρκετοί άνθρωποι στις μέρες μας επιλέγουν να φορούν. Αρχικά, μελετήθηκαν τα διαθέσιμα στην αγορά έξυπνα ρολόγια καθώς και οι δυνατότητες που προσφέρει το καθένα. Η αναζήτησή που πραγματοποιήθηκε, οδήγησε στην επιλογή του Apple Watch Series 7, καθώς έχει αρκετά πλεονεκτήματα σε σύγκριση με άλλα από ανταγωνίστριες εταιρείες και ταυτόχρονα, η Apple διαθέτει μια ευρεία κοινότητα προγραμματιστών, όπου κάποιος μπορεί να απευθυνθεί και να λάβει βοήθεια. Το συγκεκριμένο ρολόι , υποστηρίζει το λογισμικό τελευταίας γενιάς WatchOs 8 και έχει πρόσβαση στο App Store δίνοντας την δυνατότητα εγκατάστασης εφαρμογών. Ταυτόχρονα, διαθέτει υψηλής ακρίβειας αισθητήρες για την μέτρηση των καρδιακών

παλμών, τον υπολογισμό της μεταβλητότητας τους, και επίσης, με την βοήθεια ακτινοβολιών μπορεί και μετρά το ποσοστό οξυγόνου στο αίμα.

1.2.2 Αισθητήρες

Στην παρακάτω εικόνα απεικονίζονται οι αισθητήρες του ρολογιού (1). Πιο συγκεκριμένα, το ρολόι διαθέτει τρείς οπτικούς και ένα ηλεκτρικό αισθητήρα καρδιακών παλμών, αξελερόμετρο και γυρόμετρο νέας γενιάς, καθώς και φωτοδιόδους για τη μετατροπή της ακτινοβολίας φωτός σε ηλεκτρικό σήμα.



Εικόνα 1 Αισθητήρες Apple Watch Series 7

1.2.3 Διαδικασία λήψης μετρήσεων

1.2.3.1 *Καρδιακοί Παλμοί*

Ο οπτικός αισθητήρας καρδιάς στο Apple Watch χρησιμοποιεί τεχνολογία που βασίζεται στην ακόλουθη απλή λογική: Το αίμα είναι κόκκινο γιατί αντανακλά το κόκκινο φως και απορροφά το πράσινο φως. Το Apple Watch χρησιμοποιεί επίσης, πράσινα φώτα LED σε συνδυασμό με φωτοδίοδους για να ανιχνεύει την ποσότητα αίματος που ρέει στον καρπό ανά πάσα στιγμή. Όταν η καρδιά χτυπά, η ροή του αίματος στον καρπό, επομένως και η απορρόφηση του πράσινου φωτός, είναι μεγαλύτερη. Μεταξύ χτύπων της καρδιάς η ροή είναι μικρότερη.

Αναβοσβήνοντας τα φώτα LED εκατοντάδες φορές το δευτερόλεπτο, το Apple Watch μπορεί να υπολογίσει τον καρδιακό παλμό. Επιπλέον, ο οπτικός αισθητήρας καρδιάς έχει σχεδιαστεί για να αντισταθμίζει τα χαμηλά επίπεδα σήματος αυξάνοντας τόσο τη φωτεινότητα των LED όσο και τον ρυθμό δειγματοληψίας. Ο οπτικός αισθητήρας καρδιάς

μπορεί επίσης να εκπέμπει υπέρυθρο φως. Αυτή τη λειτουργία χρησιμοποιεί το Apple Watch όταν μετρά τον καρδιακό ρυθμό στο παρασκήνιο και για ειδοποιήσεις καρδιακού ρυθμού.

1.2.3.2 *Οξυγονομέτρηση*

Στο Apple Watch Series 6 και Series 7, ο οπτικός αισθητήρας καρδιάς έχει επανασχεδιαστεί για να προσθέσει δυνατότητες μέτρησης του οξυγόνου στο αίμα. Ο αισθητήρας οξυγόνου αίματος είναι ενσωματωμένος στο πίσω μέρος του Apple Watch. Χρησιμοποιεί τέσσερις ομάδες κόκκινων, πράσινων και υπέρυθρων φώτων LED και τέσσερις φωτοδιόδους, συσκευές που μετατρέπουν το φως σε ηλεκτρικό ρεύμα. Τα φώτα λάμπουν στα αιμοφόρα αγγεία στον καρπό και οι φωτοδίοδοι μετρούν πόσο φως αναπηδά πίσω.

1.2.3.3 Γυροσκόπειο και αισθητήρας επιτάχυνσης

Η εξαγωγή δεδομένων από το αξελερόμετρο και το γυροσκόπιο για την παροχή πληροφοριών σε πραγματικό χρόνο είναι επίσης εφικτή. Για παράδειγμα, είναι δυνατή η επεξεργασία των δεδομένων κίνησης για να μετρηθεί το επίπεδο δραστηριότητας του χρήστη και να καταγραφούν συγκεκριμένοι τύποι κίνησης, όπως αυτές των χεριών που γίνονται κατά τη διάρκεια μιας προπόνησης. Το Apple Watch επιτρέπει την πρόσβαση σε δεδομένα αισθητήρων και για μεγαλύτερες χρονικές περιόδους.

1.2.4 Τεχνικά χαρακτηριστικά Apple Watch Series 7

Παράλληλα, ένα άλλο πλεονέκτημα του συγκεκριμένου τύπου ρολογιού είναι το λογισμικό WatchOS 8 που υποστηρίζει. Το ρολόι έχει έναν ταχύτατο διπύρηνο 64-bit επεξεργαστή Apple S7 που ενσωματώνει GPU, 32 GB flash memory, Bluetooth 5.0, 1 GB RAM, 802.11 b/g/n 2.4 και 5 GHz WiFi καθώς και δορυφορικό εντοπισμό θέσης (GPS, GLONASS, Galileo, QZSS). Ο ισχυρός επεξεργαστής του ρολογιού σε συνδυασμό με την πρόσβαση στο App Store, δίνει την δυνατότητα στον χρήστη να κατεβάσει εφαρμογές οι οποίες επιχειρούν δύσκολα και απαιτητικά έργα. (2)

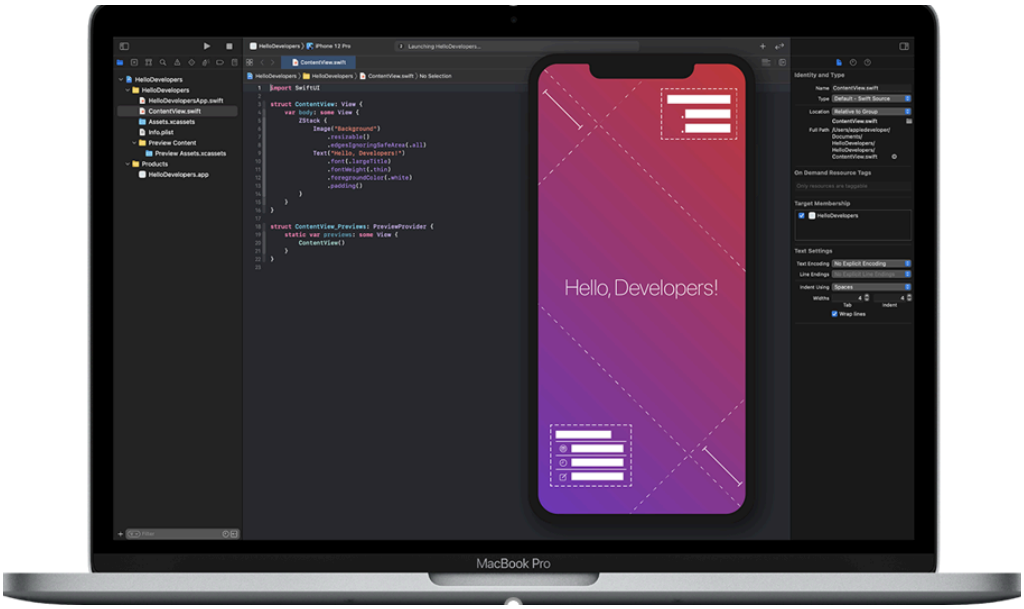
1.3 Swift

Στη συνέχεια, αναζητήθηκαν τα κατάλληλα εργαλεία για την ανάπτυξη εφαρμογής σε λειτουργικό WatchOs 8. Οι εφαρμογές αυτές αναπτύσσονται σε Swift. Πρόκειται για μια αντικειμενοστρεφή γλώσσα προγραμματισμού γενικής χρήσης, που αναπτύσσεται από την Apple Inc. και την κοινότητα ανοιχτού κώδικα. Κυκλοφόρησε για πρώτη φορά το 2014, στο Παγκόσμιο Συνέδριο Προγραμματιστών WWDC, ως αντικαταστάτης της παλαιότερης γλώσσας προγραμματισμού της Apple Objective-C, καθώς η τελευταία ήταν αμετάβλητη από τις αρχές της δεκαετίας του 1980 και δεν είχε σύγχρονες δυνατότητες. Η Swift συνεργάζεται με τα πλαίσια Cocoa και Cocoa Touch της Apple και μια βασική πτυχή του σχεδιασμού της Swift ήταν η δυνατότητα διαλειτουργικότητας με το τεράστιο σώμα του υπάρχοντος κώδικα Objective-C που αναπτύχθηκε για τα προϊόντα της Apple τις

προηγούμενες δεκαετίες. Είναι χτισμένο με το πλαίσιο μεταγλωττιστή LLVM ανοιχτού κώδικα και έχει συμπεριληφθεί στο Xcode από την έκδοση 6, που κυκλοφόρησε το 2014. Στις πλατφόρμες της Apple, χρησιμοποιεί τη βιβλιοθήκη χρόνου εκτέλεσης Objective-C, επιτρέποντας C, C++, Objective-C και Swift code για εκτέλεση σε ένα πρόγραμμα. (3)

1.4 Xcode

Απαραίτητο πρόγραμμα για την ανάπτυξη τέτοιων εφαρμογών, είναι το Xcode που διατίθεται μόνο σε συσκευές iOS και περιέχει όλα τα απαραίτητα εργαλεία για να μπορέσει να προγραμματίσει κανείς σε περιβάλλον Swift. Το Xcode είναι το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) της Apple για macOS, που χρησιμοποιείται για την ανάπτυξη λογισμικού για macOS, iOS, iPadOS, watchOS και tvOS. Κυκλοφόρησε αρχικά στα τέλη του 2003. η τελευταία σταθερή έκδοση είναι η έκδοση 13.4, που κυκλοφόρησε στις 16 Μαΐου 2022 και είναι διαθέσιμη μέσω του Mac App Store δωρεάν για χρήστες macOS Monterey. Οι εγγεγραμμένοι προγραμματιστές μπορούν να κάνουν λήψη εκδόσεων προεπισκόπησης και προηγούμενων εκδόσεων της σουίτας μέσω του ιστότοπου του Apple Developer. Το Xcode περιλαμβάνει εργαλεία γραμμής εντολών (Command Line Tools) μέσω της εφαρμογής Terminal στο macOS. Μπορούν επίσης να ληφθούν και να εγκατασταθούν χωρίς το GUI. Στην πράξη το Xcode χρησιμοποιείται για το build της εφαρμογής μας σε Apple Watch και για το emulation του Apple Watch εντός του MacOS για λόγους πρακτικότητας και παραγωγικότητας στην ανάπτυξη και στην αντιμετώπιση των εμφανιζόμενων προβλημάτων. Μια γενική μορφή του παρουσιάζεται στην παρακάτω εικόνα.



Εικόνα 2 Το περιβάλλον του Xcode

1.5 Firebase – Realtime database - Authentication

1.5.1 Γενικά

Η Realtime βάση δεδομένων του Firebase είναι μια υπηρεσία της Google που επιτρέπει τη δημιουργία εφαρμογών επιτρέποντας την ασφαλή πρόσβαση στη βάση δεδομένων απευθείας από τον κώδικα από την πλευρά του client. Ακόμα και εκτός σύνδεσης, τα συμβάντα σε πραγματικό χρόνο συνεχίζουν να ενεργοποιούνται, ώστε όταν η συσκευή επανακτήσει τη σύνδεση, η βάση δεδομένων συγχρονίζει τις αλλαγές των τοπικών δεδομένων με τις απομακρυσμένες ενημερώσεις που πραγματοποιήθηκαν ενώ ο client ήταν εκτός σύνδεσης, συγχωνεύοντας αυτόματα τυχόν διενέξεις. Παράλληλα, η χρήση της υπηρεσίας Authentication διασφαλίζει ότι η πρόσβαση στα δεδομένα θα γίνεται αποκλειστικά από εγκεκριμένους χρήστες. Με αυτό τον τρόπο δημιουργείται ένα ολοκληρωμένο περιβάλλον για τον προγραμματιστή διευκολύνοντας στην δημιουργία νέων εφαρμογών.

1.5.2 Δυνατότητες

Η βάση δεδομένων της Firebase παρέχει μια ευέλικτη γλώσσα κανόνων (Firebase Realtime Databse Security Rules) η οποία χρησιμοποιείται για τον καθορισμό του τρόπου δόμησης των δεδομένων καθώς και του πότε μπορούν να διαβαστούν ή να εγγραφούν. Η παραπάνω δυνατότητα όταν ενσωματωθεί με τον έλεγχο ταυτότητας της Firebase (Authentication) δίνει την ικανότητα στους προγραμματιστές να ορίσουν το ποιος έχει πρόσβαση, σε ποια δεδομένα και πώς μπορούν να έχουν πρόσβαση σε αυτά. Η βάση δεδομένων σε πραγματικό χρόνο είναι NoSQL και ως εκ τούτου έχει διαφορετικές βελτιστοποιήσεις και λειτουργικότητα σε σύγκριση με μια σχεσιακή βάση δεδομένων. Το API της βάσης δεδομένων έχει σχεδιαστεί για να επιτρέπει μόνο λειτουργίες που μπορούν να εκτελεστούν γρήγορα. Αυτό δίνει τη δυνατότητα να δημιουργηθεί μια εξαιρετική εμπειρία σε πραγματικό χρόνο που μπορεί να εξυπηρετήσει εκατομμύρια χρήστες χωρίς συμβιβασμούς στην ανταπόκριση.

1.5.3 Πλεονεκτήματα

1.5.3.1 *Realtime συγχρονισμός για JSON δεδομένα*

Πρόκειται για μία βάση δεδομένων που επιτρέπει την εγγραφή και το διάβασμα δεδομένων με απλές εντολές. Τα δεδομένα αποθηκεύονται σε μορφή JSON tree και έτσι αφήνεται στην ευχέρεια του προγραμματιστή η διαμόρφωση της βάσης ώστε η ανάγνωση και η εγγραφή να γίνεται με αποδοτικό τρόπο.

1.5.3.2 *Συνδεσιμότητα μεταξύ συσκευών*

Ο συγχρονισμός σε πραγματικό χρόνο επιτρέπει στους χρήστες την άμεση και σχεδόν ταυτόχρονη πρόσβαση στα δεδομένα. Το γεγονός αυτό επιτρέπει όταν πραγματοποιούνται αλλαγές στην βάση από την εφαρμογή του ρολογιού να γίνονται ορατές άμεσα από την ιστοσελίδα.

1.5.3.3 *Authentication*

Το Firebase Authentication στοχεύει να διευκολύνει τη δημιουργία ασφαλών συστημάτων ελέγχου ταυτότητας, βελτιώνοντας παράλληλα την εμπειρία σύνδεσης και ενσωμάτωσης για τους τελικούς χρήστες. Παρέχει μια λύση ταυτότητας από άκρο σε άκρο, υποστηρίζοντας λογαριασμούς email και κωδικών πρόσβασης, έλεγχο ταυτότητας τηλεφώνου ακόμα και σύνδεση μέσω Google, Twitter, Facebook και GitHub. Έτσι η Εγγραφή / Σύνδεση χρηστών καθώς και η εξασφάλιση της προστασίας των ευαίσθητων δεδομένων γίνεται μέσω της χρήσης της βιβλιοθήκης Firebase Auth απαλλάσσοντας τον προγραμματιστή από μια δύσκολη και πολύπλοκη διεργασία.

1.6 Firebase – Cloud Functions

1.6.1 Γενικά

Το Google Cloud Functions είναι ένα serverless περιβάλλον εκτέλεσης για τη δημιουργία και τη σύνδεση υπηρεσιών Cloud. Με τις Cloud Functions δίνεται η δυνατότητα ανάπτυξης στοχευμένων συναρτήσεων για την εξυπηρέτηση λειτουργιών που σχετίζονται με συμβάντα που παράγονται από την υποδομή και τις υπηρεσίες στο Cloud. Ο κώδικας των συναρτήσεων παρέχεται σε μορφή SaaS, δηλαδή εκτελείται σε ένα πλήρως διαχειριζόμενο από τη Google περιβάλλον, ώστε να μην υπάρχει ανάγκη παροχής υποδομής ή ανησυχίας για τη διαχείριση servers.

Οι Cloud συναρτήσεις της Google υποστηρίζουν πολλαπλές γλώσσες προγραμματισμού, καθώς και τη δυνατότητα εκτέλεσης και δοκιμής σε τοπικό περιβάλλον, γεγονός που τις καθιστά φορητές και επεκτάσιμες. Η Google προσφέρει δύο εκδόσεις προϊόντων: Cloud Functions (1st gen), την αρχική έκδοση και Cloud Functions (2nd gen), μια νέα έκδοση που βασίζεται στο Cloud Run και το Eventarc για να παρέχει ένα βελτιωμένο σύνολο δυνατοτήτων. (4)

Στα πλαίσια της διπλωματικής χρησιμοποιήθηκαν οι συναρτήσεις Cloud που παρέχονται στην πλατφόρμα του Firebase. Όπως αναφέρθηκε, πρόκειται για ένα serverless framework που επιτρέπει την αυτόματη εκτέλεση κώδικα ως απόκριση σε συμβάντα που ενεργοποιούνται από τις λειτουργίες του Firebase και τα αιτήματα HTTPS. Η υλοποίηση του κώδικα γίνεται σε JavaScript ή TypeScript και αποθηκεύεται στο Cloud. Εκτελείται σε διαχειριζόμενο περιβάλλον το οποίο δε χρειάζεται διαχείριση και συντήρηση server. Οι συναρτήσεις χωρίζονται σε δύο κατηγορίες. Η πρώτη είναι HTTPS συναρτήσεις και η δεύτερη background και θα αναλυθούν παρακάτω.

1.6.2 Κατηγορίες Συναρτήσεων

1.6.2.1 HTTPS Functions

Τα HTTPS Functions αποτελούν την πρώτη κατηγορία των Google Cloud συναρτήσεων. Οι συγκεκριμένες συναρτήσεις καλούνται μέσω HTTPS requests. Η υλοποίηση τους ακολουθεί την παρακάτω δομή. (5)

```
exports.date = functions.https.onRequest((req, res) => {
  // ...
});
```

Εικόνα 3 Παράδειγμα κλήσης HTTPS function

Το “req” αναπαριστά το request και το “res” το αποτέλεσμα που επιστρέφει η συνάρτηση στο αρχικό HTTPS αίτημα.

1.6.2.2 Background Functions – Cloud Event Functions

Οι συναρτήσεις αυτές έχουν αυτοματοποιημένη λειτουργία. Πιο συγκεκριμένα, η λειτουργία τους ενεργοποιείται από κάποια αλλαγή που έχει πραγματοποιηθεί στη βάση δεδομένων ή σε οποιαδήποτε άλλη δομή του Google Cloud. Σε αντίθεση με την πρώτη κατηγορία, οι background συναρτήσεις δεν λειτουργούν με HTTPS request. (6)

1.7 CocoaPods

Το CocoaPods είναι ένας διαχειριστής των εξαρτήσεων σε επίπεδο εφαρμογής για Objective-C και Swift που παρέχει μια τυπική μορφή για τη διαχείριση εξωτερικών βιβλιοθηκών. Η πρώτη δημόσια κυκλοφορία του έγινε την 1η Σεπτεμβρίου 2011. Το CocoaPods εστιάζει στη διανομή κώδικα τρίτων και στην αυτόματη ενσωμάτωσή τους σε έργα Xcode. Τα αναγκαία για μια εφαρμογή pods εγκαθίστανται από τη γραμμή εντολών (terminal). Με αυτό τον τρόπο, επιλύει τις εξαρτήσεις (π.χ. βιβλιοθήκες) για μια εφαρμογή βάσει προδιαγραφών που έχει, γλυτώνοντας έτσι τον προγραμματιστή από την μη αυτοματοποιημένη αντιγραφή αρχείων και κώδικα που ενδεχομένως να οδηγήσει σε λάθη και παραλείψεις.

1.8 HealthKit

1.8.1 Βασικές αρχές της HealthKit

Το 2014, η Apple παρουσίασε τη HealthKit παράλληλα με το iOS 8. Η HealthKit είναι η λύση της Apple για αποθήκευση, διαχείριση και κοινή χρήση δεδομένων υγείας. Η βιβλιοθήκη αυτή, επιτρέπει στους προγραμματιστές να διαβάζουν και να γράφουν δεδομένα υγείας σε μια βάση δεδομένων, την οποία διαχειρίζεται το λειτουργικό σύστημα.

Είναι αυτονόητο ότι αυτό ανοίγει ένα ευρύ φάσμα δυνατοτήτων για τους προγραμματιστές. Ήδη χιλιάδες εφαρμογές έχουν ενσωματωθεί στο περιβάλλον της βιβλιοθήκης αυτής, η οποία ανά τακτά χρονικά διαστήματα εμπλουτίζει το φάσμα δυνατοτήτων της. (7)

1.8.2 Δυνατότητες της HealthKit

Η HealthKit είναι κάτι περισσότερο από έναν απλό container για αποθήκευση και κοινή χρήση δεδομένων υγείας. Από τη μεριά του χρήστη, είναι ένας βολικός τρόπος για τη συγκέντρωση και οργάνωση δεδομένων που σχετίζονται με την υγεία, τη φυσική και ψυχική του κατάσταση. Από τη μεριά του προγραμματιστή, είναι μια πλατφόρμα που υλοποιεί τη συλλογή και διαχείριση δεδομένων υγείας με εύκολο και βολικό τρόπο.

1.8.3 Κοινή χρήση δεδομένων

Ένα από τα βασικότερα χαρακτηριστικά της HealthKit είναι η δυνατότητα κοινής χρήσης δεδομένων υγείας με άλλες εφαρμογές. Ορισμένες εφαρμογές συλλέγουν δεδομένα μέσω των αισθητήρων, ενώ άλλες συγκεντρώνουν τις μετρήσεις που έχουν ήδη πραγματοποιηθεί. Η βιβλιοθήκη της HealthKit βρίσκει εφαρμογή και στις δύο αυτές κατηγορίες. Οποιαδήποτε εφαρμογή ενσωματώνεται με τη συγκεκριμένη βιβλιοθήκη έχει τη δυνατότητα να εισάγει και να διαβάσει δεδομένα από την κοινόχρηστη βάση. Επομένως, ο χρήστης πρέπει να εκχωρήσει σε μια εφαρμογή δικαιώματα ανάγνωσης και εγγραφής δεδομένων. Ως γνωστόν, τα δεδομένα υγείας είναι συνήθως ευαίσθητα δεδομένα, γεγονός που καθιστά αναγκαία την τήρηση των κανονισμών απορρήτου.

1.8.4 Privacy

Η Apple δίνει έμφαση στο πόσο ενδιαφέρεται για το απόρρητο των πελατών της και η HealthKit αναδεικνύει αυτή τη δέσμευση. Οι εφαρμογές που ενσωματώνονται με τη HealthKit οφείλουν να ζητούν την άδεια του χρήστη για ανάγνωση ή/και εγγραφή των δεδομένων που διαχειρίζεται η βιβλιοθήκη. Ο χρήστης αποφασίζει σε ποια δεδομένα μπορεί ή δεν μπορεί να έχει πρόσβαση μια συγκεκριμένη εφαρμογή. Στο πλαίσιο αυτό, η HealthKit είναι πολύ αυστηρή όσον αφορά τα δικαιώματα. Για παράδειγμα, μια εφαρμογή που χρησιμοποιεί τη βιβλιοθήκη μπορεί να ρωτήσει εάν επιτρέπεται να γράψει δεδομένα συγκεκριμένου τύπου στη βάση, ωστόσο δεν μπορεί να ρωτήσει εάν επιτρέπεται η ανάγνωση δεδομένων συγκεκριμένου τύπου, καθώς η ίδια η ερώτηση εντάσσεται στην κατηγορία δεδομένων ευαίσθητου τύπου. Η απαγόρευση ακόμα και της δήλωσης του τύπου των δεδομένων προς ανάγνωση δείχνει πόσο αυστηρά είναι τα όρια της διασφάλισης των προσωπικών δεδομένων.

1.8.5 Ισχυρό API

Ανάλογα με τον τύπο της εφαρμογής, η συλλογή, οργάνωση και διατήρηση δεδομένων μπορεί να είναι ένα πολύπλοκο εγχείρημα. Η βιβλιοθήκη HealthKit απλοποιεί την υλοποίηση από τη μεριά του προγραμματιστή. Παρόλο που το HealthKit API μπορεί να

είναι αρκετά περίπλοκο στην αρχή, εκθέτει σταθερά endpoints για ανάγνωση, γραφή και παρακολούθηση δεδομένων.

1.8.6 Περιορισμοί

Τα δεδομένα που διαχειρίζεται η HealthKit αποθηκεύονται μόνο τοπικά. Η Apple τονίζει ότι μια εφαρμογή δεν πρέπει να αποθηκεύει δεδομένα υγείας στο iCloud. Οι οδηγίες αναθεώρησης του App Store ορίζουν ρητά ότι οι εφαρμογές που αποθηκεύουν πληροφορίες υγείας που λαμβάνονται μέσω της HealthKit στο iCloud θα απορρίπτονται. Ένας ακόμα περιορισμός είναι πως, η HealthKit δεν υποστηρίζεται από τις συσκευές iPad. Επειδή τα δεδομένα HealthKit αποθηκεύονται μόνο τοπικά, η Apple περιορίζει την υποστήριξη της βιβλιοθήκης στις πιο προσωπικές συσκευές του χρήστη, όπως το τηλέφωνο και το smart watch.

1.9 JavaScript

Ένα άλλος βασικό κομμάτι του έργου της παρούσας διπλωματικής αποτέλεσε η δημιουργία ιστοσελίδας με σκοπό την διαχείριση των χρηστών και των επαγγελματιών καθώς και την προβολή των μετρήσεων των βιοτικών δεδομένων ενός χρήστη. Για τον σκοπό αυτό επιλέχτηκε η ανάπτυξη μια ιστοσελίδας με χρήση της γλώσσα JavaScript και React που ενδείκνυται για frontend development με την βοήθεια του εργαλείου Node.js. Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Είναι γλώσσα που συνδυάζει διαφορετικούς τρόπους προγραμματισμού, υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side). (8)

1.10 React

Η React παρουσιάστηκε ως έργο ανοιχτού κώδικα τον Μάιο του 2013. Ο αρχικός προγραμματιστής ήταν ο Jordan Walke, μηχανικός στο Facebook. Η React μπορεί να χρησιμοποιηθεί ως βάση για την ανάπτυξη εφαρμογών μιας σελίδας ή εφαρμογών κινητού. Ένα από τα βασικά του πλεονεκτήματα είναι ότι μπορεί επίσης να λειτουργήσει από πλευρά του διακομιστή (server) μαζί με την πλευρά του χρήστη (client) και μπορούν να συνεργαστούν διαλειτουργικά.

Χρησιμοποιεί σύνταξη JSX, ένα συνδυασμό JavaScript και HTML. Το JSX απλοποιεί ολόκληρη τη διαδικασία σύνταξης components για τις ιστοσελίδες και η πτυχή HTML επιτρέπει στους προγραμματιστές να εκτελούν λειτουργίες χωρίς να συνδυάζουν συμβολοσειρές. Μία από τις μεγαλύτερες προκλήσεις με τα frameworks της JavaScript είναι ότι δεν είναι καθόλου φιλικά προς τις μηχανές αναζήτησης. Παραδόξως, η React ξεχωρίζει, καθώς καταφέρνει να αποτυπωθεί στο πρόγραμμα περιήγησης ως κανονική ιστοσελίδα. Αυτό που είναι αξιοσημείωτο στη React είναι ότι προσφέρει τη δυνατότητα επαναχρησιμοποίησης των components οποιαδήποτε στιγμή. Αυτό έχει ένα πολύ σημαντικό αποτέλεσμα εξοικονόμησης χρόνου. Τα components στη React είναι απομονωμένα και η αλλαγή σε ένα δεν επηρεάζει τα υπόλοιπα. Αυτό επιτρέπει στους προγραμματιστές να επαναχρησιμοποιήσουν τα components που δεν παράγουν αλλαγές και καθιστούν τον προγραμματισμό πιο ακριβή, εργονομικό και άνετο για αυτούς. (9)

2. Η σχέση της εφαρμογής με τις επιστήμες υγείας

2.1 Εισαγωγή

Όπως γίνεται αντιληπτό ζούμε στην εποχή της πληροφορίας, όπου κάθε άνθρωπος είναι μόνιμα συνδεδεμένος στο διαδίκτυο με μια ή παραπάνω συσκευές και παράγει συνεχώς δεδομένα. Τα δεδομένα αυτά χωρίζονται σε διάφορες κατηγορίες. Καταρχάς, υπάρχουν τα ευαίσθητα δεδομένα, δηλαδή εκείνα που αφορούν την προσωπική ζωή ή επαγγελματικούς λόγους (συνομιλίες, email, web-meetings, κλπ.). Μια άλλη μεγάλη κατηγορία δεδομένων είναι εκείνα τα οποία υπόκεινται σε επεξεργασία για εμπορικούς σκοπούς. Η περιήγηση στα μέσα κοινωνικής δικτύωσης, καθώς και σε online καταστήματα, δημιουργεί δεδομένα σχετικά με τις προτιμήσεις ή τις ανάγκες του χρήστη μέσω των cookies και συνεπώς αξιοποιούνται για την παραγωγή κατάλληλων διαφημιστικών μηνυμάτων και την προώθηση προϊόντων. Τα τελευταία χρόνια, με την πρόοδο στον τομέα του cloud computing και της σχεδόν σύγχρονης επεξεργασίας των δεδομένων λόγω της υψηλής ταχύτητας του διαδικτύου, έχουν γνωρίσει μεγάλη ανάπτυξη οι φορητές συσκευές οι οποίες συλλέγουν και στέλνουν δεδομένα σχετικά με την βιοτική κατάσταση του χρήστη. Παρακάτω παρουσιάζονται οι μετρικές οι οποίες συλλέγονται από την εφαρμογή του ρολογιού και η αξία τους για την εκτίμηση της υγείας του χρήστη μαζί με τις υποδειγματικές τιμές τους και τις παθήσεις με τις οποίες συνδέονται. Στο τέλος του κεφαλαίου γίνεται σύντομη αναφορά για τον στόχο της εφαρμογής και τις καινοτομίες αυτής της διπλωματικής εργασίας.

2.2 Καρδιακοί παλμοί

2.2.1 Γενικά

Ο σφυγμός είναι η κίνηση του τοιχώματος των αρτηριών που συμβαίνει σε κάθε καρδιακό παλμό και μεταδίδεται σαν είδος κύματος (σφυγμικό κύμα). Δημιουργείται όταν, σε κάθε συστολή της αριστερής κοιλίας της καρδιάς, ωθούνται 70 κυβικά εκατοστά αίματος προς την αρχή της αορτής, που είναι ήδη γεμάτη. Τα υγρά είναι ασυμπίεστα οπότε, για να βρεθεί χώρος, διατείνεται το τοίχωμα της αορτής και σε λίγο επανέρχεται στη φυσιολογική του διάσταση. Έτσι γεννάται ένα κύμα, που μεταδίδεται κατά μήκος του ελαστικού τοιχώματος των αρτηριών. (10)

2.2.2 Υποδειγματικές τιμές καρδιακών παλμών

Όπως φαίνεται στον παρακάτω πίνακα οι καρδιακοί παλμοί ποικίλουν. Ύστερα από αρκετές μελέτες έχουν προκύψει κάποιες ενδεικτικές τιμές ανάλογα την ηλικία. Βέβαια, οι τιμές αυτές αναφέρονται σε άνδρες, με τις αντίστοιχες τιμές για γυναίκες να είναι μικρότερες. Πέρα από το φύλλο και την ηλικία υπάρχουν και άλλοι παράγοντες που επηρεάζουν τους καρδιακούς παλμούς ηρεμίας. Τέτοιοι είναι η φυσική κατάσταση του ατόμου, καθώς και πιθανές παθήσεις όπως βραδυκαρδία και ταχυκαρδία. (11)

Καρδιακοί παλμοί ηρεμίας για άντρες (παλμοί το λεπτό)						
Ηλικία	18-25	26-35	36-45	46-55	56-65	65+
Αθλητές	40-52	44-50	47-53	49-54	51-56	52-55
Εξαιρετικοί	56-61	55-61	57-62	58-63	57-61	56-61
Καλοί	62-65	62-65	63-66	64-67	62-67	62-65
Ικανοποιητικοί	66-69	66-70	67-70	68-71	68-71	66-69
Μέτριοι	70-73	71-74	71-75	72-76	72-75	70-73
Κάτω του μετρίου	74-81	75-81	76-82	77-83	76-81	74-79
Κακοί	82+	82+	83+	84+	82+	80+

Πίνακας 1 Αναμενόμενες τιμές καρδιακών παλμών

2.3 Μεταβλητότητα των καρδιακών παλμών (HRV)

2.3.1 Γενικά

Η μεταβλητότητα καρδιακού ρυθμού ή HRV (Heart Rate Variability) αφορά τη μεταβλητότητα στο χρόνο μεταξύ των καρδιακών παλμών. Η καρδιά δε χτυπάει σαν το μετρονόμο. Πρέπει να προσαρμόζεται στο περιβάλλον και στις δισεκατομμύρια διαδικασίες που συμβαίνουν στο σώμα. Στην προσπάθεια να ανταπεξέλθει σε αυτές τις απαιτήσεις, θα επιταχύνει ή θα επιβραδύνει ανάλογα. Ωστόσο, όταν αρχίζει να αντιλαμβάνεται σημαντικούς κινδύνους, ψυχολογικούς και φυσιολογικούς, αρχίζει την αυτορρύθμιση που οδηγεί σε μικρότερη μεταβλητότητα.

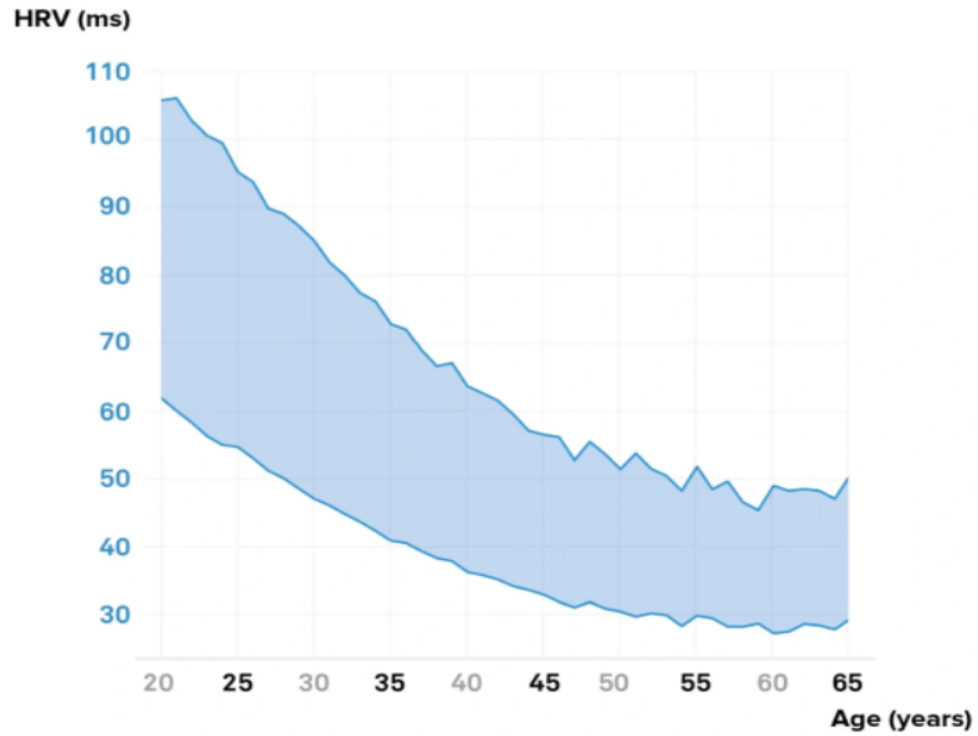
Όταν επικρατεί πανικός, και φυσιολογικά ή ψυχολογικά, η καρδιά κάνει ότι μπορεί για να ρυθμίσει το σύστημα στην προσπάθεια να επιστρέψει στην ομοιόσταση. Το HRV είναι πολύ ευαίσθητο στις αλλαγές του αυτόνομου νευρικού συστήματος και συνδέεται στενά με το στρες, καθώς όταν υπάρχει αυτό, υπάρχει δυσλειτουργία του συμπαθητικού νευρικού συστήματος. (12)

2.3.2 Δείκτης SDNN (SDNNI)

Υπάρχουν πολλοί τρόποι υπολογισμού του HRV. Το HealthKit του Apple Watch Series 7 χρησιμοποιεί μεταβλητότητα καρδιακού ρυθμού SDNN, η οποία βασίζεται στην τυπική απόκλιση των διαστημάτων μεταξύ των παλμών (RR) (συνήθως μετράται σε χιλιοστά του δευτερολέπτου). Το SDNNI είναι ο μέσος όρος των τυπικών αποκλίσεων όλων των διαστημάτων για κάθε τμήμα 5 λεπτών μιας εγγραφής HRV σε συνολικό διάστημα 24 ωρών. Επομένως, αυτή η μέτρηση εκτιμά μόνο τη μεταβλητότητα λόγω των παραγόντων που επηρεάζουν το HRV μέσα σε μια περίοδο 5 λεπτών. Υπολογίζεται διαιρώντας πρώτα την εγγραφή 24 ωρών σε 288 τμήματα των 5 λεπτών και στη συνέχεια υπολογίζοντας την τυπική απόκλιση όλων των διαστημάτων που περιέχονται σε κάθε τμήμα. Το SDNNI είναι ο μέσος όρος αυτών των 288 τιμών και αντανακλά πρωτίστως την αυτόνομη επιρροή στο HRV.

2.3.3 Υποδειγματικές HRV τιμές

Το HRV είναι μια πολύ εξατομικευμένη και συνεχώς μεταβαλλόμενη μέτρηση. Εξαρτάται επίσης από διάφορους παράγοντες όπως η σωματική και ψυχική υγεία, το στρες, η διατροφή, η χρήση αλκοόλ, οι συνήθειες ύπνου, η ηλικία, το φύλο, η γενετική, η συχνότητα και ένταση σωματικής ή ψυχικής άσκησης και πολλά άλλα. Στο παρακάτω διάγραμμα φαίνονται τα ακρότατα όρια για ηλικίες από 20 μέχρι και 65 ετών.



Εικόνα 4 Φάσμα φυσιολογικών τιμών HRV

2.3.4 Παθήσεις συνδεόμενες με κακή λειτουργία της καρδιάς

Η κακή λειτουργία της καρδιάς μπορεί να έχει πολύ σοβαρές επιπλοκές για την υγεία. Μερικές από αυτές παρουσιάζονται παρακάτω (13):

- Λιποθυμία: Εάν η καρδιά χτυπά γρήγορα, η αρτηριακή πίεση μπορεί να πέσει, προκαλώντας λιποθυμία στο άτομο. Αυτό είναι πιο πιθανό σε άτομα με καρδιακό πρόβλημα, όπως συγγενείς καρδιακές παθήσεις, ή ορισμένα προβλήματα βαλβίδων.
- Καρδιακό επεισόδιο: Πιο σπάνια, το αίσθημα παλμών μπορεί να προκληθεί από απειλητικά για τη ζωή προβλήματα καρδιακού παλμού και μπορεί να αναγκάσουν την καρδιά να σταματήσει να χτυπά αποτελεσματικά.
- Εγκεφαλικό: Εάν οι παλμοί καρδιάς οφείλονται σε μια κατάσταση στην οποία οι άνω κοιλότητες της καρδιάς τρέμουν γρήγορα αντί να χτυπούν σωστά (κολπική μαρμαρυγή), το αίμα μπορεί να συσσωρευτεί και να προκαλέσει σχηματισμό θρόμβων. Εάν ένας θρόμβος διασπαστεί, μπορεί να φράξει μια εγκεφαλική αρτηρία, προκαλώντας εγκεφαλικό επεισόδιο.
- Καρδιακή ανεπάρκεια: Ορισμένες αρρυθμίες μπορούν να μειώσουν την ικανότητα άντλησης της καρδιάς. Μερικές φορές, ο έλεγχος του ρυθμού μιας αρρυθμίας που προκαλεί καρδιακή ανεπάρκεια μπορεί και να βελτιώσει την λειτουργία της καρδιάς.

2.4 Πρόσληψη οξυγόνου στο αίμα

2.4.1 Γενικά

Το επίπεδο οξυγόνου στο αίμα (κορεσμός οξυγόνου του αίματος) είναι η ποσότητα οξυγόνου που κυκλοφορεί σε αυτό. Το οξυγόνο είναι απαραίτητο για τη ζωή και το σώμα μας χρειάζεται μια ορισμένη ποσότητα για να λειτουργήσει σωστά. Το οξυγόνο εισέρχεται στο σώμα σας από τη μύτη ή το στόμα κατά την εισπνοή και περνά μέσω των πνευμόνων στην κυκλοφορία του αίματός. Στη συνέχεια πηγαίνει στα κύτταρα σε όλο το σώμα. Όλα τα κύτταρά σας χρειάζονται οξυγόνο για να επιτελέσουν αποτελεσματικά τις λειτουργίες τους. Μόλις τα κύτταρά χρησιμοποιήσουν το οξυγόνο, δημιουργούν διοξείδιο του άνθρακα. Ακολούθως, η κυκλοφορία του αίματός μεταφέρει το διοξείδιο του άνθρακα πίσω στους πνεύμονές και με την εκπνοή φεύγει από αυτό.

Το σώμα ρυθμίζει αυστηρά την ποσότητα κορεσμού οξυγόνου στο αίμα, επειδή τα χαμηλά επίπεδα οξυγόνου (υποξαιμία) μπορεί να οδηγήσουν σε πολλές σοβαρές καταστάσεις και βλάβες σε μεμονωμένα συστήματα οργάνων, ειδικά στον εγκέφαλο και την καρδιά. Τα χαμηλά επίπεδα οξυγόνου στο αίμα υποδηλώνουν ότι οι πνεύμονες και/ή το κυκλοφορικό σύστημα μπορεί να μην λειτουργούν όπως θα έπρεπε. (14)

2.4.2 Υποδειγματικές τιμές

Για τους περισσότερους ανθρώπους, μια κανονική ένδειξη παλμικού οξύμετρου για το επίπεδο κορεσμού οξυγόνου είναι μεταξύ 95% και 100%. Εάν ο ασθενής πάσχει από πνευμονική νόσο όπως ΧΑΠ ή πνευμονία, το φυσιολογικό επίπεδο κορεσμού οξυγόνου μπορεί να είναι χαμηλότερο. Τα επίπεδα κορεσμού οξυγόνου μπορεί επίσης να είναι χαμηλότερα σε περιοχές με υψηλό υψόμετρο. Ενδεικτικά κάποιες υποδειγματικές τιμές για τα επίπεδα οξυγόνου στο αίμα παρουσιάζονται στον παρακάτω πίνακα. (14)

Κορεσμός με οξυγόνο αίματος - Blood Oxygen Saturation (SpO_2)	
Κανονικό	100 – 98 %
Ανεπαρκές – Ανεκτό ο ασθενής δεν παρατηρεί κάποιο πρόβλημα	97 – 95 %
Μειωμένο άμεση παρέμβαση (φαγητό, σωματική άσκηση)	94 – 90 %
Κρίσιμο παραπομπή σε ειδικό	< 90 %
Σοβαρή υποξία νοσηλεία σε νοσοκομείο	< 80 %
Κίνδυνος για την ζωή	< 70 %

Πίνακας 2 Αναμενόμενες τιμές οξυγόνου στο αίμα

2.4.3 Συνδεόμενες παθήσεις

Η μειωμένη ικανότητα του σώματος να προσλαμβάνει οξυγόνο έχει σοβαρές επιπτώσεις στην υγεία ενός ανθρώπου. Μπορεί να οφείλεται στο κάπνισμα, σε αναιμία ή και σε υπνική άπνοια (προσωρινή διακοπή της αναπνοής κατά τη διάρκεια του ύπνου). Ταυτόχρονα, συνδέεται και με παθήσεις των πνευμόνων που περιλαμβάνουν:

- Άσθμα
- Εμφύσημα (βλάβη των αερόσακων στον πνεύμονα)
- Βρογχίτιδα
- Πνευμονία
- Πνευμοθώρακας (διαρροή αέρα στον χώρο μεταξύ του πνεύμονα και του θωρακικού τοιχώματος)
- Σύνδρομο οξείας αναπνευστικής δυσχέρειας (ARDS)
- Πνευμονικό οίδημα (ο πνεύμονας διογκώνεται λόγω συσσώρευσης υγρού)
- Πνευμονική ίνωση (ουλές στους πνεύμονες)
- Διάμεση πνευμονοπάθεια (μια μεγάλη ομάδα πνευμονικών διαταραχών που γενικά προκαλούν προοδευτικές ουλές στους πνεύμονες)
- Ιογενείς λοιμώξεις (π.χ. COVID-19)

2.5 Στόχος της εφαρμογής και προεκτάσεις

Η υγεία παραμένει το υπέρτατο αγαθό. Την σημερινή εποχή, λοιπόν, της ραγδαίας ανάπτυξης της τεχνολογίας και των επιστημών υγείας γίνεται όλο και πιο συστηματική η έρευνα σχετικά με μεθόδους πρόληψης ατυχημάτων, επεισοδίων όπως εμφραγμάτων, θρομβώσεων και άλλων παθήσεων, ώστε να βελτιωθεί το προσδόκιμο ζωής ιδιαίτερα των ανθρώπων που αντιμετωπίζουν προβλήματα υγείας. Επομένως στόχος της παρούσας διπλωματικής ήταν η δημιουργία μιας αυτόνομης εφαρμογής που θα συλλέγει, τα δεδομένα που προκύπτουν από τους αισθητήρες ενός έξυπνου ρολογιού. Στη συνέχεια, θα τα αποθηκεύει στο cloud όπου θα γίνεται η κατάλληλη επεξεργασία με υπολογιστικά μοντέλα ώστε να παρέχουν στον χρήστη και τον επαγγελματία υγείας μια βαθύτερη κατανόηση των δεδομένων.

Αυτό όπως γίνεται αντιληπτό έχει σημαντικές προεκτάσεις, καθώς θα μπορεί μελλοντικά ένας γιατρός, ή ένα αυτοματοποιημένο σύστημα να παρακολουθεί την εξέλιξη της υγείας ενός ασθενή, αφαιρώντας του το βάρος και το άγχος της συνεχούς παρακολούθησης των βιοτικών του στοιχείων. Επίσης, θα είναι εφικτή η ανάλυση όλων αυτών των δεδομένων των ασθενών και η εξαγωγή πορισμάτων σχετικά με μοτίβα στις μετρήσεις ανθρώπων που έχουν παρόμοια προβλήματα. Έτσι, θα δοθεί ώθηση στην έρευνα για την καταπολέμηση παθήσεων, γεγονός που θα γιγαντωθεί και με την προσθήκη νέων αισθητήρων και μετρήσεων, όπως και έχουν στόχο όλες οι εταιρείες να υλοποιήσουν στο εγγύς μέλλον.

2.6 Καινοτομίες

Το πρωτοποριακό στοιχείο της εφαρμογής αυτής, είναι το πώς θα μπορεί να δέχεται εντολές για την λήψη μετρήσεων από τους αισθητήρες από κάποιο άλλον απομακρυσμένο χρήστη. Μέχρι σήμερα, οι συσκευές αυτές λαμβάνουν μετρήσεις από κάθε αισθητήρα μεμονωμένα κατά παραγγελία του χρήστη και βγάζουν γενικές ειδοποιήσεις που είναι χρήσιμες σε ακραίες περιπτώσεις. Αντιθέτως, η εφαρμογή που δημιουργήθηκε και βασίστηκε πάνω στο Apple Watch Series 7, είναι σε θέση να παίρνει εξατομικευμένες μετρήσεις ανάλογα με το προφίλ υγείας του χρήστη. Πιο αναλυτικά, παρέχει την δυνατότητα, εφόσον το ρολόι είναι συνδεδεμένο στο διαδίκτυο, να δεχθεί μήνυμα «Λάβε ένα αριθμό μετρήσεων από τον εξής αισθητήρα». Στη συνέχεια, το ρολόι θα απαντάει στο αίτημα με τα καινούργια δεδομένα που θα στέλνει στο Cloud και συγκεκριμένα στην πλατφόρμα της Google, το Firebase. Επίσης, στη διαμορφωμένη βάση δεδομένων, διατηρείται και το ιατρικό ιστορικό του κάθε χρήστη. Το δεύτερο στοιχείο που αποτελεί καινοτομία, είναι πως οι τιμές των καρδιακών παλμών, της μεταβλητότητάς τους και του οξυγόνου που συλλέγονται υπόκεινται σε επεξεργασία από ένα υπολογιστικό μοντέλο που προβλέπει αν αυτές οι τιμές είναι σε φυσιολογικά πλαίσια λαμβάνοντας υπόψιν πολλές παραμέτρους.

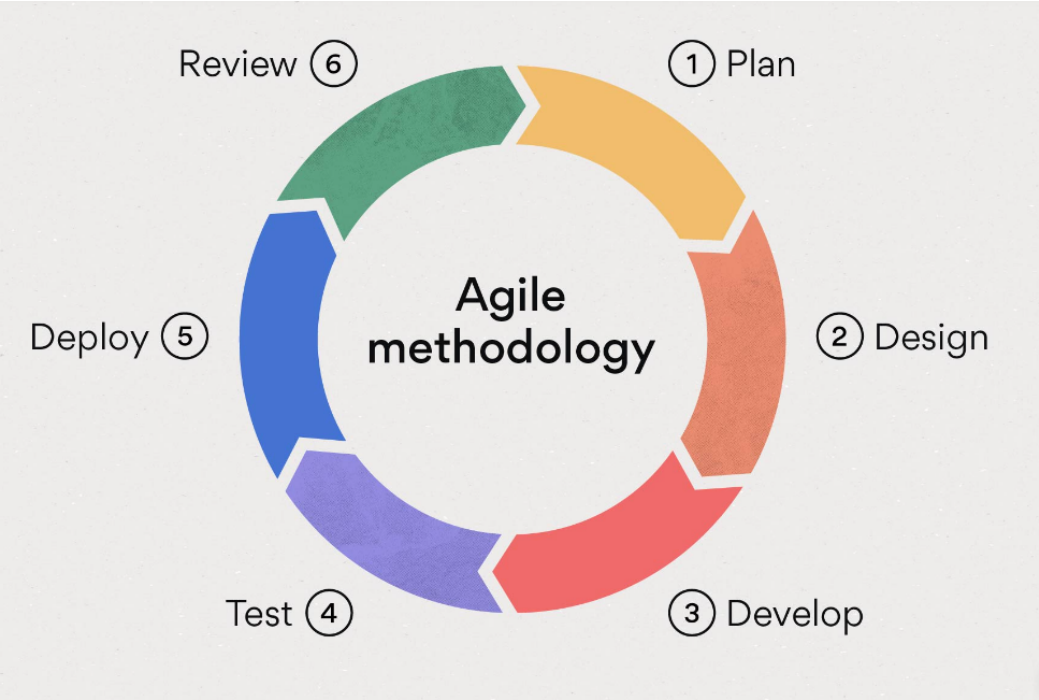
3. Αρχιτεκτονική και μεθοδολογία υλοποίησης

3.1 Εισαγωγή

Στο κεφάλαιο αυτό, γίνεται μια εκτενής αναφορά στα στοιχεία που αποτελούν αυτή την εφαρμογή καθώς και στην αλληλεπίδραση που έχουν μεταξύ τους. Επίσης, παρουσιάζεται η μεθοδολογία με την οποία έγινε η εκπόνηση του έργου και η ανάπτυξη του λογισμικού. Συγκεκριμένα, υιοθετήθηκε η Agile μεθοδολογία η οποία αναλύεται στην επόμενη υποενότητα.

3.2 Agile προσέγγιση

Στο παρόν project επιδιώχθηκε να ακολουθηθεί μια agile προσέγγιση ανάπτυξης για τη διασφάλιση της επεκτασιμότητας, της αποτελεσματικής συνεργασίας και συνεννόησης, καθώς και της διεκπεραίωσης όλων των εφικτών στόχων προς υλοποίηση. Για μια agile ανάπτυξη, απαραίτητη ήταν η δημιουργία μιας αρχιτεκτονικής, δημιουργημένη σε δομές που επιτρέπουν την εφαρμογή των έξι βασικών βημάτων της agile λογικής. (15)



Εικόνα 5 Agile προσέγγιση

1. Στόχοι

Πρώτο και βασικότερο βήμα αποτέλεσε ο ορισμός των επιθυμητών λειτουργιών που ήταν εφικτό να επιτελεστούν. Οι στόχοι έπρεπε να είναι ρεαλιστικοί, βασισμένοι στις ανάγκες

και επιθυμίες πιθανών χρηστών αλλά και καινοτόμοι. Η παροχή ενός συνόλου υπηρεσιών αποτέλεσε προϊόν πολλαπλών συναντήσεων μεταξύ ατόμων της παραγωγής και ατόμων σχετικών με τον επαγγελματικό και επιστημονικό τομέα που πραγματεύεται η εργασία.

2. Σχεδιασμός

Ο σχεδιασμός του έργου έγινε με βάση τους στόχους που έχουν οριστεί και τη διαθεσιμότητα πόρων. Οι συνεχόμενες μεταβολές στις λειτουργίες του έργου επηρέαζαν άμεσα τη σχεδίαση της υλοποίησης. Κάθε νέα προσθήκη λειτουργίας δημιουργούσε την ανάγκη σχεδίασης κατάλληλων δομών για την υποστήριξή της.

3. Ανάπτυξη

Η ανάπτυξη του έργου έγινε με την ανάθεση διαφορετικών μεταξύ τους εργασιών στην παραγωγή. Η λογική ανάπτυξης κώδικα έπρεπε να είναι συγκεκριμένη, έτσι ώστε να υπάρχει δυνατότητα ομαλής συνένωσης για ένα ομοιόμορφο σύνολο. Ιδιαίτερα βοηθητική υπήρξε η χρήση του εργαλείου GitHub, που έκανε εφικτή την ανάκτηση των ξεχωριστών τμημάτων κώδικα εξ' αποστάσεως. Η ανάπτυξη έγινε κυρίως πάνω σε δομές SaaS (Software as a Service), προκειμένου να διατηρούνται καλύτερα πρωτόκολλα ασφάλειας, εύκολη προσβασιμότητα στις δομές και δυνατότητα επέκτασης.

4. Testing και Deployment

Μετά από κάθε υλοποίηση λειτουργίας πραγματοποιήθηκαν πολλαπλές δοκιμές προκειμένου να διασφαλίσουν την ορθότητα των αποτελεσμάτων και τη σωστή λειτουργία των δομών που χρησιμοποιήθηκαν. Συγκεκριμένα για το backend μέρος του project, δοκιμές πραγματοποιήθηκαν και σε τοπικό localhost περιβάλλον. Η διαχείριση και συντήρηση της βάσης δεδομένων γίνεται εξ' ολοκλήρου από τη Google Firebase, επομένως δεν χρειάστηκε η διεξαγωγή testing πάνω στη διαθεσιμότητά της. Το deployment ξεκινούσε μόνο εφόσον κάθε test είχε ολοκληρωθεί με επιτυχία.

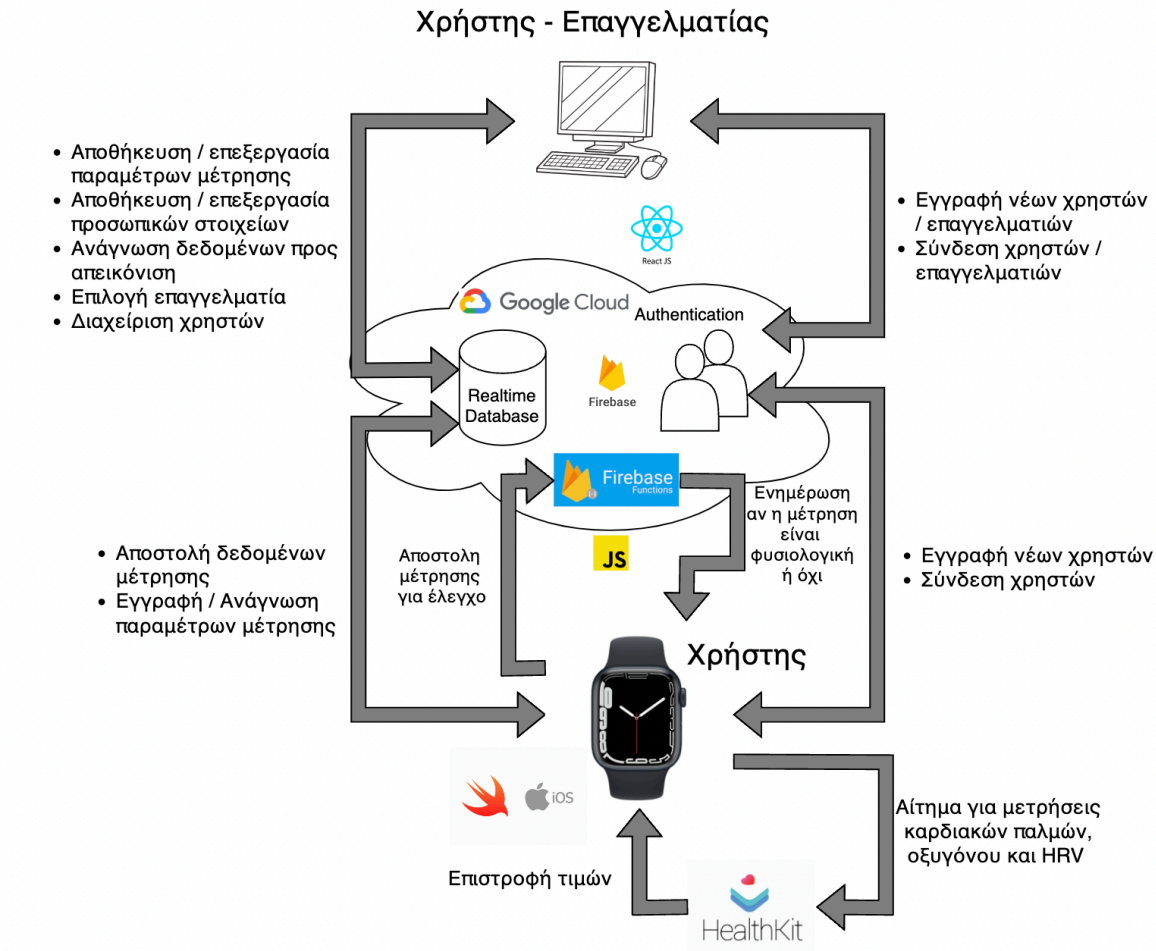
5. Αξιολόγηση

Τελευταίο βήμα της agile μεθοδολογίας αποτέλεσε η αξιολόγηση των αποτελεσμάτων, του τρόπου εργασίας και ανάπτυξης, καθώς και η επισήμανση των δυσκολιών που αντιμετωπίστηκαν στον τρέχοντα κύκλο, ώστε να αποφευχθούν εάν είναι δυνατόν σε μελλοντικούς. Η συνολική εποπτεία των βημάτων υπήρξε ιδιαίτερα σημαντική στην αποφυγή λάνθασμένων προσεγγίσεων και στη λήψη ορθολογικών επιλογών.

Προφανώς, για την εκπόνηση αυτού του έργου χρειάστηκαν αρκετοί κύκλοι εργασιών, καθώς υπήρχε συνεχής προσθήκη καινούργιων στόχων και λειτουργιών, με αποτέλεσμα, να είναι απαραίτητη η εκ νέου οργάνωση, σχεδιασμός ανάπτυξη, δοκιμή και αξιολόγηση των ενεργειών.

3.3 Γενική κάτοψη της αρχιτεκτονικής

Όπως αναφέρθηκε, το έργο της παρούσας διπλωματικής αποτελείται από τρεις βασικούς πυλώνες: την εφαρμογή στο ρολόι, το firebase και την ιστοσελίδα. Η πρώτη είναι υπεύθυνη για την λήψη των μετρήσεων. Το Firebase περιέχει την βάση δεδομένων για την αποθήκευση, το Authentication για την ταυτοποίηση των χρηστών και τα Functions με το μοντέλο επεξεργασίας των δεδομένων. Τέλος, η ιστοσελίδα είναι υπεύθυνη για την αλληλεπίδραση χρήστη και επαγγελματία υγείας, τον καθορισμό των παραμέτρων λήψης των μετρήσεων καθώς και την προβολή αυτών. Η γενική μορφή της αρχιτεκτονικής του έργου παρουσιάζεται στο παρακάτω διάγραμμα.



Εικόνα 6 Αρχιτεκτονική δομή του έργου

3.3 Χρήση Firebase Authentication από Apple Watch και Ιστοσελίδα

Η δημιουργία custom based συστήματος πιστοποίησης χρηστών μπορεί να πάρει μεγάλο χρονικό διάστημα, απαιτεί επιπλέον υπολογιστικούς πόρους, ενδέχεται να έχει προβλήματα με την ασφάλεια και χρειάζεται μια ομάδα μηχανικών για τη συνεχή διατήρησή του. Η χρήση του Firebase Authentication στο συγκεκριμένο project, έδωσε

μια γρήγορη και αποτελεσματική λύση στην διαχείριση νέων και παλαιών εγγραφών, διατηρώντας τα σωστά πρωτόκολλα ασφάλειας, χωρίς να απαιτεί ομάδα διαχείρισης του συστήματος. Η όλη διαδικασία έγινε με την εγκατάσταση των κατάλληλων βιβλιοθηκών στο περιβάλλον του ρολογιού και της ιστοσελίδας. Με αυτόν τον τρόπο, γίνεται εφικτή η χρήση συναρτήσεων που επιτελούν την σύνδεση και την εγγραφή χρηστών κάθε φορά που ανοίγει η εφαρμογή, ή όταν επιλεχθεί η συγκεκριμένη λειτουργία στην ιστοσελίδα.

3.5 Επικοινωνία Apple Watch και Βάσης Δεδομένων

Μεταξύ του Apple Watch και της Realtime βασης δεδομένων της Firebase υπάρχει αμφίδρομη επικοινωνία. Αναλυτικότερα, κάθε μέτρηση που εκτελείται από το χρήστη μέσω της HealthKit εφαρμογής ανεβαίνει απευθείας στη βάση δεδομένων. Δεδομένα που σχετίζονται με τη μέτρηση, όπως η ώρα λήψης, το είδος της δραστηριότητας και η κατηγορία, συνοδεύουν τις τιμές των μετρήσεων.

3.6 Επικοινωνία εφαρμογής και Healthkit

Όπως έχει προαναφερθεί στο κεφάλαιο 1, η HealthKit υπηρεσία παρέχει λειτουργία βάσης δεδομένων. Μέσω της εφαρμογής, δίνεται η δυνατότητα καταμέτρησης των καρδιακών παλμών, της οξυγόνωσης καθώς και του HRV. Για τη μέτρηση των καρδιακών παλμών γίνεται ενεργοποίηση της λειτουργίας των αισθητήρων, με αποτέλεσμα να λαμβάνονται δεδομένα πραγματικού χρόνου. Τα δεδομένα αυτά, συλλέγονται αρχικά στο τοπικό περιβάλλον του Apple Watch και στη συνέχεια αποστέλλονται στη Firebase βάση δεδομένων. Αντιθέτως, για τις μετρήσεις του ποσοστιαίου οξυγόνου και του HRV, αποστέλλεται αίτημα στη βάση δεδομένων της HealthKit, η οποία έχει καταχωρημένες όλες τις παρελθοντικές μετρήσεις του χρήστη. Από αυτές γίνεται επιλογή της πιο πρόσφατης και αμέσως αποστέλλεται στο ρολόι. Τέλος, πραγματοποιείται φόρτωση των δεδομένων στην Firebase.

3.7 Επικοινωνία Ιστοσελίδας και Βάσης Δεδομένων

Η online ιστοσελίδα έχει άμεση επικοινωνία με τη Firebase βάση δεδομένων. Πιο συγκεκριμένα, ο χρήστης έχει πρόσβαση στην αναπαράσταση real time δεδομένων. Οι μετρήσεις καρδιακών παλμών, του οξυγόνου και του HRV απεικονίζονται σε διαγράμματα. Για την απεικόνιση των παραπάνω, η ιστοσελίδα συνδέεται στην realtime βάση της Firebase και εκτελεί queries για την εξαγωγή των κατάλληλων δεδομένων, ανάλογα με τις χρονικές παραμέτρους που έχει επιλέξει ο χρήστης. Εκτός από τα δεδομένα μετρήσεων, η ιστόσελίδα χρησιμοποιεί τη σύνδεση στη βάση και για την απεικόνιση χαρακτηριστικών πληροφοριών του χρήστη, όπως το email, το όνομα, το επώνυμο, καθώς και των ρυθμίσεων λήψης μέτρησης. Παράλληλα, δίνεται η δυνατότητα τροποποίησης αυτών των παραμέτρων όποτε η ιστοσελίδα καλείται όχι μόνο να διαβάσει δεδομένα από την βάση, αλλά και να εγγράψει νέα.

3.8 Σύνδεση Firebase Functions με Εφαρμογή και Ιστοσελίδα

3.8.1 Γενικά

Στο τρέχον project έχει γίνει υλοποίηση δύο βασικών συναρτήσεων, οι οποίες είναι προσβάσιμες μέσω https και είναι αποθηκευμένες στον Cloud χώρο “Functions” της Google Firebase.

Η πρώτη συνάρτηση κατηγοριοποιεί την τελευταία μέτρηση με βάση την ώρα που πραγματοποιήθηκε και τον τύπο της δραστηριότητας. Ύστερα, χρησιμοποιώντας όλες τις παρελθοντικές μετρήσεις που ανήκουν στην ίδια κατηγορία, υπολογίζει με τη βοήθεια στατιστικού μέσου όρου και τυπικής απόκλισης, ένα άνω και ένα κάτω όριο τα οποία και επιστρέφει.

Η δεύτερη συνάρτηση χρησιμοποιεί το πόρισμα της πρώτης και έχει ρόλο αξιολογητή, καθώς αξιολογεί ως φυσιολογική η μη την τελευταία μέτρηση του χρήστη. Σε περίπτωση που η τελευταία μέτρηση δεν ανήκει στα όρια που υπολόγισε η πρώτη συνάρτηση, προχωράει στην αποστολή αιτήματος στην εφαρμογή του Apple Watch προκειμένου να ενημερωθεί ο χρήστης από το ρολόι. Εάν η μέτρηση ήταν εντός ορίων καταχωρείται με επιτυχία στη βάση δεδομένων.

3.8.2 Χρήση Cloud Functions από το front end της ιστοσελίδας

Η πρώτη Cloud Function που περιγράφηκε παραπάνω, καλείται από το front end της ιστοσελίδας μέσω https. Η συνάρτηση αυτή, επιστρέφει την τελευταία μέτρηση που καταχωρήθηκε μαζί με τα δύο άκρα που υπολογίστηκαν. Τα αποτελέσματα της χρησιμοποιούνται για την δημιουργία ενός x-y γραφήματος με σκοπό την απεικόνιση της σχετικής θέσης της τελευταίας μέτρησης σε σύγκριση με τα δύο όρια. Σε περίπτωση που ο μέσος όρος των τιμών της μέτρησης είναι εκτός διαστήματος, ο χρήστης ή ο επιβλέπων μπορεί να αποφανθεί για την ανακρίβεια της συλλογής δεδομένων.

3.8.3 Χρήση Cloud Functions από το περιβάλλον του Apple Watch

Αμέσως μετά την ολοκλήρωση μέτρησης και την αποστολή των σχετικών δεδομένων στη βάση, καλείται η δεύτερη συνάρτηση από το περιβάλλον του Apple Watch. Η συνάρτηση επιστρέφει True, εφόσον η μέτρηση ήταν εντός ορίων, ή αντίστοιχα False, εφόσον ήταν ανεπιτυχής. Στην περίπτωση της ανεπιτυχούς μέτρησης το ρολόι ειδοποιεί άμεσα το χρήστη, ενημερώνοντάς τον να επαναλάβει τη διαδικασία.

3.9 Κόστος έργου

3.9.1 Firebase

Ένας σημαντικός παράγοντας για την επιλογή της αρχιτεκτονικής και των συγκεκριμένων τεχνολογικών εργαλείων αποτελεί το κόστος υλοποίησης ενός τέτοιου συστήματος.

Παρόλο που η πλατφόρμα Firebase προσφέρει αρκετές δυνατότητες και υπηρεσίες στους χρήστες της, παραμένει αρκετά οικονομική. Αναλυτικότερα, η Realtime Database και το Authentication έχουν ένα δοκιμαστικό περιβάλλον δωρεάν που περιλαμβάνει αρκετές συνδέσεις / εγγραφές χρηστών και αρκετό αποθηκευτικό χώρο στην βάση. Όσον αφορά το Cloud Functions, αυτό δεν προσφέρει δωρεάν πρόγραμμα, αλλά παραμένει αρκετά οικονομικό ιδιαίτερα για μια εφαρμογή με λίγες συνδέσεις στην μονάδα του χρόνου και μικρή μεταφορά δεδομένων. Παρακάτω παρουσιάζεται ο αναλυτικός πίνακας με τα κόστη για κάθε υπηρεσία. (16)

Products	No-cost Spark Plan Generous limits to get started	Pay as you go Blaze Plan Calculate pricing for apps at scale ✓ No-cost usage from Spark plan included*
Authentication		
Phone Auth - US, Canada, and India	10k/month	$0.01/verification
Phone Auth - All other countries	10k/month	$0.06/verification
Other Authentication services	✓	✓
With Identity Platform		
Monthly active users	50k/month	No-cost up to 50k MAUs Then Google Cloud pricing
Monthly active users - SAML/OIDC	50/month	No-cost up to 50 MAUs Then Google Cloud pricing
Realtime Database		
Simultaneous connections	100	200k/database
GB stored	1 GB	$5/GB
GB downloaded	10 GB/month	$1/GB
Multiple databases per project	✗	✓
Cloud Functions		
Invocations	*Not applicable*	No-cost up to 2M/month Then $0.40/million
GB-seconds		No-cost up to 400K/month Then Google Cloud pricing
CPU-seconds		No-cost up to 200K/month Then Google Cloud pricing
Outbound networking		No-cost up to 5GB/month Then $0.12/GB
Cloud Build minutes		No-cost up to 120min/day Then $0.003/min
Container storage		No-cost up to 500MB of storage Then $0.10/GB/month
		*Pricing varies based on location

Εικόνα 7 Πλάνο κοστολόγησης Firebase

3.9.2 Κόστος ανάπτυξης εφαρμογής σε iOS

Η ανάπτυξη μιας εφαρμογής τόσο για ρολόι (watchOS), όσο και για τις πλατφόρμες του περιβάλλοντος Apple, δηλαδή macOS για υπολογιστές, iOS για κινητά, tvOS για τηλεοράσεις, δεν προϋποθέτουν κάποιο κόστος παρά μόνο την απόκτηση των εν λόγω συσκευών. Πιο συγκεκριμένα, η ανάπτυξη λογισμικού γίνεται στην ειδικά διαμορφωμένη εφαρμογή Xcode που αναλύθηκε προηγουμένως. Αυτή δεν διατίθεται από άλλα λειτουργικά με αποτέλεσμα η απόκτηση υπολογιστή macOS να είναι αναγκαστική. Πέραν από αυτό, εάν ο προγραμματιστής επιθυμεί να δοκιμάσει την εφαρμογή του σε πραγματικό ρολόι και όχι μόνο στους simulators που προσφέρει το Xcode τότε οφείλει όχι μόνο να αποκτήσει το ρολόι αλλά και κάποιο κινητό iOS στο οποίο θα συνδεθεί.

3.9.3 Υπόλοιπα κόστη

Σε αντίθεση με την εφαρμογή, η ανάπτυξη του κώδικα για την ιστοσελίδα δεν προσθέτει κάποιο κόστος. Τα εργαλεία που χρησιμοποιήθηκαν (NodeJS, Visual Studio Code) είναι εντελώς δωρεάν και διατίθενται σε όλα τα λειτουργικά συστήματα. Το front-end εκτελείται σε localhost περιβάλλον κάνοντας τη χρήση του δωρεάν Μελλοντικά, η ενοικίαση κάποιου server όπου θα κατοικεί μόνιμα η ιστοσελίδα και θα είναι προσβάσιμη από όλους ίσως να προϋποθέτει κάποιο κόστος για την αγορά του domain name. Τέλος, το EmailJS που χρησιμοποιείται είναι δίχως κόστος για αρκετές αποστολές email, γεγονός που καλύπτει τις ανάγκες του έργου.

3.10 Εναλλακτικοί τρόποι υλοποίησης

Προφανώς, η Apple δεν αποτελεί την μοναδική επιλογή στην αγορά. Μια άλλη μεγάλη κατηγορία τέτοιων ρολογιών είναι εκείνα που υποστηρίζουν Android λογισμικό. Τεχνολογικοί κολοσσοί όπως η Samsung, Huawei, Xiaomi κυκλοφορούν έξυπνα ρολόγια αντίστοιχα με το Apple Watch Series 7. Όπως είναι λογικό ο προγραμματισμός αντίστοιχης εφαρμογής σε Android λογισμικό είναι διαφορετικός, αλλά ακολουθεί μια παρόμοια προσέγγιση με αυτή που πραγματοποιήθηκε. Στις Android συσκευές η ανάπτυξη εφαρμογών γίνεται μέσω του εργαλείου Android Studio αντί του Xcode, ενώ η γλώσσα προγραμματισμού που υποστηρίζεται είναι η Java που αποτελεί και αυτή μια αντικειμενοστραφή γλώσσα προγραμματισμού με πολλά κοινά στοιχεία με την Swift. Μία ίσως βασική διαφορά των δύο αυτών κόσμων προγραμματισμού που αξίζει να σημειωθεί, είναι το γεγονός πως το Android Studio είναι διαθέσιμο σε όλες τις πλατφόρμες (Windows, Mac, Linux, ChromeOS) σε αντίθεση με το Xcode που είναι αποκλειστικά για Mac. Ο χρήστης έχει στην διάθεση του το Play Store αντί του App Store ώστε να κατεβάζει τις εφαρμογές της αρεσκείας του.

Παράλληλα, υπάρχουν αρκετοί πάροχοι Cloud στην αγορά πέραν της Google. Η Amazon, η Microsoft και η IBM είναι ίσως οι μεγαλύτεροι ανταγωνιστές, προσφέροντας παρόμοιες δυνατότητες με την πλατφόρμα του Firebase, όποτε και κανείς θα μπορούσε να στραφεί σε αντίστοιχες υλοποιήσεις. Όσον αφορά την ιστοσελίδα, η ανάπτυξή της σε React και JavaScript αποτελεί μια από τις πολλές επιλογές που διαθέτουν οι προγραμματιστές.

Χαρακτηριστικό παράδειγμα αποτελεί η διαχρονική γλώσσα ανάπτυξης ιστοσελίδων HTML.

Τέλος, το μοντέλο που αναπτύξαμε έγινε σε γλώσσα JavaScript και τρέχει στα Cloud Functions της Firebase υλοποιώντας έτσι αυτόματα μία μορφή back-end. Μία εναλλακτική, θα μπορούσε να είναι η δημιουργία back-end από την αρχή χρησιμοποιώντας μια γλώσσα όπως Python η οποία να διαχειρίζεται και την επικοινωνία με την Firebase.

4. Παρουσίαση και Ανάλυση Κώδικα

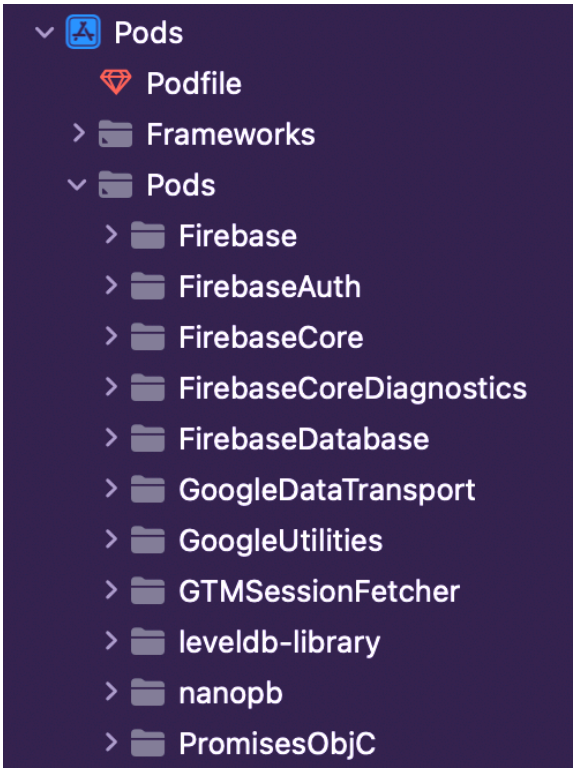
4.1 Εισαγωγή

Μεγάλο μέρος της διπλωματικής αυτής εργασίας αποτελεί η ανάπτυξη κώδικα, ώστε να γίνουν εφικτά όλα τα παραπάνω. Έτσι το παρόν κεφάλαιο, ασχολείται με την ανάλυση του κώδικα που χρησιμοποιήθηκε τόσο για την εφαρμογή, όσο και για την ιστοσελίδα και το μοντέλο που επεξεργάζεται τις μετρήσεις.

4.2 Εφαρμογή

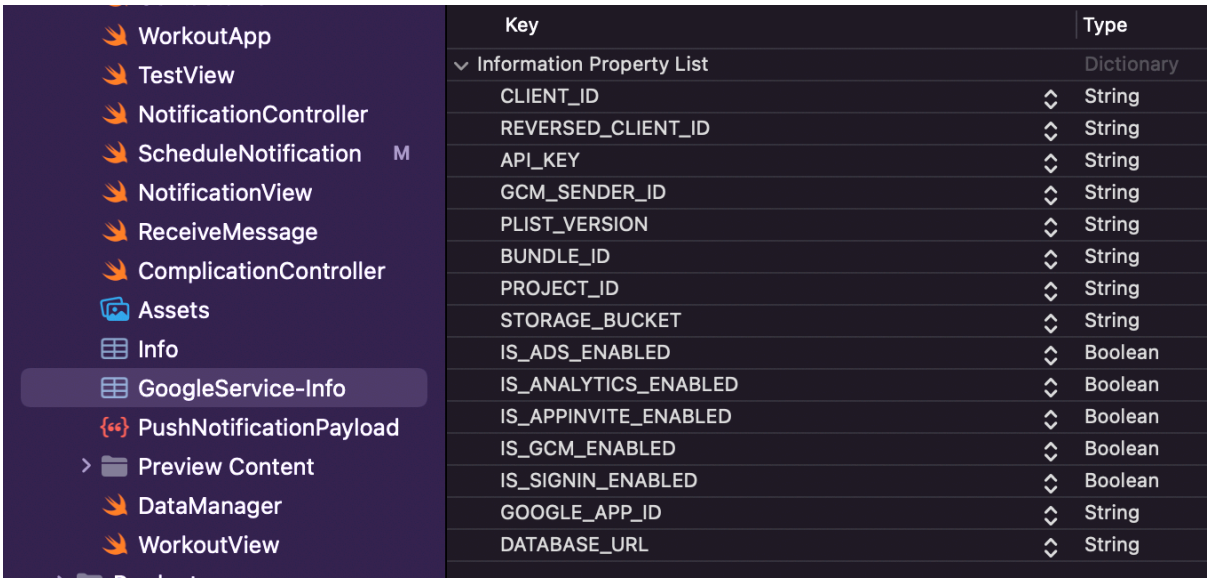
4.2.1 Επικοινωνία με firebase

Για να μπορέσει η εφαρμογή μας να επικοινωνήσει με την πλατφόρμα του Firebase της Google, αρχικά χρειάζεται η εγκατάσταση των απαραίτητων pod files με την βοήθεια του terminal.



Εικόνα 8 Απαραίτητα podfiles

Ακολούθως, το Xcode και το Firebase κάνουν εύκολη την ζωή στον προγραμματιστή, αφού παρέχουν την δυνατότητα εγκατάστασης ενός αρχείου που περιέχει όλες τις απαραίτητες πληροφορίες για να γίνει η σύνδεση μεταξύ του Cloud και του ρολογιού. Πιο συγκεκριμένα, το GoogleService-Info.plist παράγεται στην πλατφόρμα της Firebase, προστίθεται στο project του Xcode και είναι εξιδεικευμένο για την κάθε εφαρμογή. Όπως φαίνεται στην φωτογραφία, περιέχει διάφορες μεταβλητές με τις τιμές τους απαραίτητες για την σύνδεση της εφαρμογής.



Εικόνα 9 Μορφή GoogleService-Info.plist

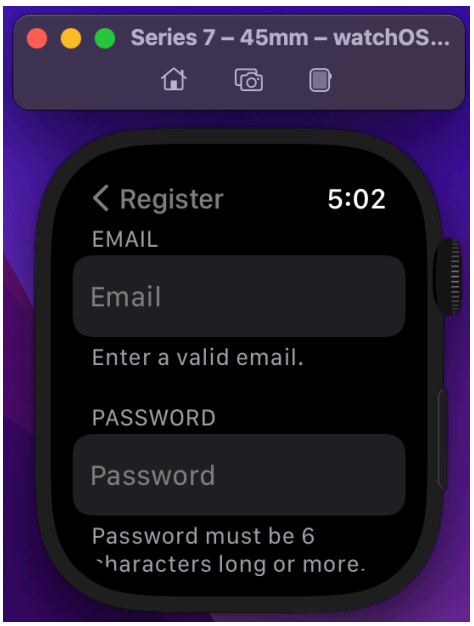
Ύστερα από τα παραπάνω βήματα, ο προγραμματιστής το μόνο που έχει να κάνει είναι να καλέσει την συνάρτηση init() κατά την εκκίνηση της εφαρμογής.

```
17      init() {
18          FirebaseApp.configure()
19      }
```

Εικόνα 10 Initialize Firebase

Πλέον η σύνδεση έχει επιτευχθεί και η κλήση των συναρτήσεων του Firebase είναι εφικτή εφόσον έχει προηγουμένως συμπεριληφθεί η κατάλληλη βιβλιοθήκη. (17)

4.2.2 Εγγραφή νέων χρηστών



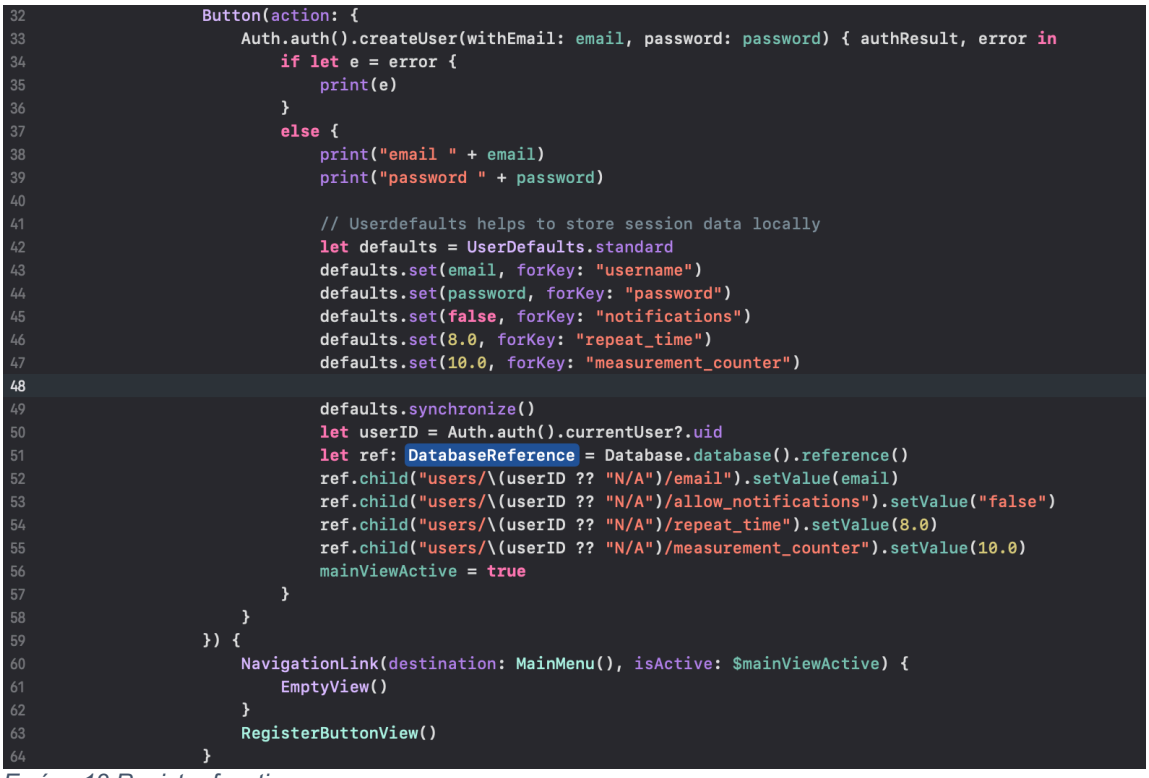
Αρχικά, η οθόνη εγγραφής των νέων χρηστών απεικονίζεται παραπλεύρως. Ακολουθώντας το documentation του Firebase, πραγματοποιήθηκε η εγγραφή νέων χρηστών στην εφαρμογή μόνο με email και password. Για τον σκοπό αυτό, γίνεται import οι κατάλληλες βιβλιοθήκες όπως φαίνονται στην εικόνα.

```
11  import Firebase
12  import FirebaseDatabase
```

Εικόνα 11 Libraries

Στην συνέχεια καλείται η συνάρτηση createUser(withEmail: , password:), δίνοντας φυσικά τα κατάλληλα ορίσματα όπως φαίνεται παρακάτω. (18)

Εικόνα 12 Register page

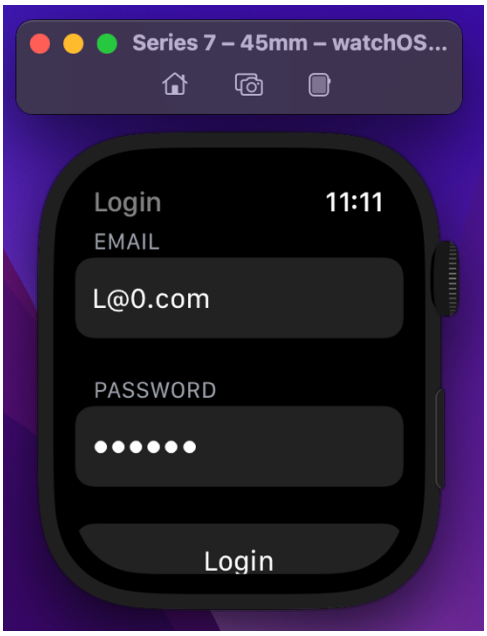


```
32                  Button(action: {
33                      Auth.auth().createUser(withEmail: email, password: password) { authResult, error in
34                          if let e = error {
35                              print(e)
36                          }
37                          else {
38                              print("email " + email)
39                              print("password " + password)
40
41                              // Userdefaults helps to store session data locally
42                              let defaults = UserDefaults.standard
43                              defaults.set(email, forKey: "username")
44                              defaults.set(password, forKey: "password")
45                              defaults.set(false, forKey: "notifications")
46                              defaults.set(8.0, forKey: "repeat_time")
47                              defaults.set(10.0, forKey: "measurement_counter")
48
49                              defaults.synchronize()
50                              let userID = Auth.auth().currentUser?.uid
51                              let ref: DatabaseReference = Database.database().reference()
52                              ref.child("users/\(userID ?? "N/A")/email").setValue(email)
53                              ref.child("users/\(userID ?? "N/A")/allow_notifications").setValue("false")
54                              ref.child("users/\(userID ?? "N/A")/repeat_time").setValue(8.0)
55                              ref.child("users/\(userID ?? "N/A")/measurement_counter").setValue(10.0)
56                              mainViewActive = true
57                          }
58                      }
59                  }) {
60                      NavigationLink(destination: MainMenu(), isActive: $mainViewActive) {
61                          EmptyView()
62                      }
63                      RegisterButtonView()
64                  }
```

Εικόνα 13 Register function

Σε περίπτωση ανεπιτυχούς εγγραφής, που μπορεί να οφείλεται σε αρκετούς λόγους όπως λανθασμένη διεύθυνση email, κωδικός με λιγότερο από 6 χαρακτήρες ή και ακόμα σφάλμα δικτύου, τότε απλώς εκτυπώνονται τα σφάλματα. Σε επιτυχημένη προσπάθεια, ακολουθεί μια σειρά ενεργειών για τον καθορισμό των βασικών παραμέτρων του χρήστη. Όπως φαίνεται στις γραμμές 42-49 αποθηκεύονται μόνιμα στη συσκευή κάποιες μεταβλητές όπως τα credentials του χρήστη. Αυτό γίνεται, διότι η πληκτρολόγηση στο ρολόι είναι δύσκολη και έτσι ο χρήστης θα μπορεί να συνδέεται κατευθείαν κάθε φορά που ανοίγει την εφαρμογή. Για να επιτευχθεί αυτό ορίζονται κάποιες μεταβλητές με τις τιμές τους στο UserDefaults.standard και στη συνέχεια καλείται η εντολή synchronize(). Στις γραμμές 50 και 51 σώζεται σε μεταβλητές ένα reference της βάσης όπου θα αποθηκευτούν τα δεδομένα καθώς και το ID του χρήστη. Έτσι, είναι πλέον εφικτό, με την εντολή ref.child("users/\(userID ?? "N/A")/email").setValue(email) (γραμμές 52-55) να αποθηκευτούν στη βάση πληροφορίες για τον χρήστη σε μια διεύθυνση που περιέχει το ID του που είναι μοναδικό. Το τελευταίο στάδιο είναι να γίνει true η μεταβλητή mainViewActive η οποία ενεργοποιεί αυτόματα τη μετάβαση στο κύριο μενού της εφαρμογής (γραμμή 60). (19)

4.2.3 Σύνδεση χρηστών



Η σύνδεση των χρηστών ακολουθεί παρόμοια υλοποίηση με την εγγραφή. Παρακάτω παρουσιάζεται η οθόνη που εμφανίζεται στον χρήστη. Η βασική διαφοροποίηση είναι η μέθοδος που καλείται που είναι η createUser(withEmail: , password:) δίνοντας τα κατάλληλα ορίσματα. (18)

Εικόνα 14 Login page

```
40                  Auth.auth().signIn(withEmail: email, password: password) { authResult, error in
41                      if let e = error {
42                          print(e)
43                      }
44                      else {
45                          print("email " + email)
46                          print("password " + password)
47                          // Userdefaults helps to store session data locally
48                          let defaults = UserDefaults.standard
49
```

Εικόνα 15 Login function 1

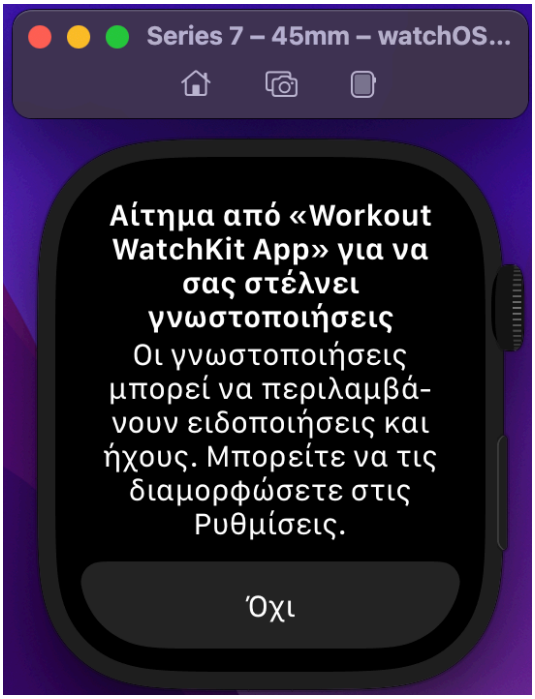
Επίσης, πραγματοποιείται έλεγχος των παραμέτρων του χρήστη βάσει των οποίων γίνονται οι μετρήσεις σε σχέση με αυτά που έχουν αποθηκευτεί στην βάση του Firebase. Σε περίπτωση αλλαγής ανανεώνονται οι τιμές. Αυτό συμβαίνει προκειμένου ο επιβλέποντας, ο οποίος έχει πρόσβαση στην παραμετροποίηση των τιμών, να έχει τη δυνατότητα να λαμβάνει με μεγαλύτερη ή μικρότερη συχνότητα, μετρήσεις από τον ασθενή. Η ανάγνωση των τιμών γίνεται με την εντολή ref.child("/users/\(userID)/repeat_time").getData(completion: { error, snapshot in όπου στην ουσία δίνουμε ένα reference της βάσης, το κατάλληλο path όπου βρίσκονται τα δεδομένα που θέλουμε (μέσα κρύβεται και το user ID) και την εντολή getData. Σε περίπτωση επιτυχίας τα δεδομένα βρίσκονται στη μεταβλητή snapshot η οποία και συγκρίνεται με τις τιμές που έχουν αποθηκεύτει στη συσκευή και εφόσον χρειαστεί, γίνεται ανανέωση (έλεγχος if … else γραμμές 62-68). (19)

```
ref.child("/users/\(userID)/repeat_time").getData(completion:  { error, snapshot in
    guard error == nil else {
      print(error!.localizedDescription)
      return;
    }
    let repeat_time = snapshot.value as? Float ?? 8.0
    print(repeat_time)
    if repeat_time == UserDefaults.standard.float(forKey: "repeat_time"){
        print("saved correctly")
    }
    else {
        print("changed remotely have to set new notification")
        notification.scheduleNotification(repeat_time: repeat_time)
        defaults.set(repeat_time, forKey: "repeat_time")
    }
});
```

Εικόνα 16 Login function 2

4.2.4 Άδεια για ειδοποιήσεις, καταγραφή μετρήσεων

Η Apple για να περιορίσει την ανεξέλεγκτη χρήση των προσωπικών δεδομένων χωρίς τη συναίνεση του χρήστη, επιβάλει στον προγραμματιστή να ζητάει άδεια με σκοπό την είσοδο και επεξεργασία ευαίσθητων δεδομένων. Για το σκοπό αυτό, είναι αναγκαία τα αιτήματα για άδεια τόσο μέτρησης των ζωτικών σημάτων που πραγματεύεται η εφαρμογή (καρδιακοί παλμοί οξυγόνο κλπ), όσο και για την παραγωγή ειδοποιήσεων. Στις πρώτες δύο φωτογραφίες παρουσιάζεται το κομμάτι κώδικα που ζητάει άδεια για δημιουργία ειδοποιήσεων μέσω της κλήσης της συνάρτησης requestAuthorization() στο UNUserNotificationCenter, καθώς και ο τρόπος που φαίνεται στον χρήστη. (20)

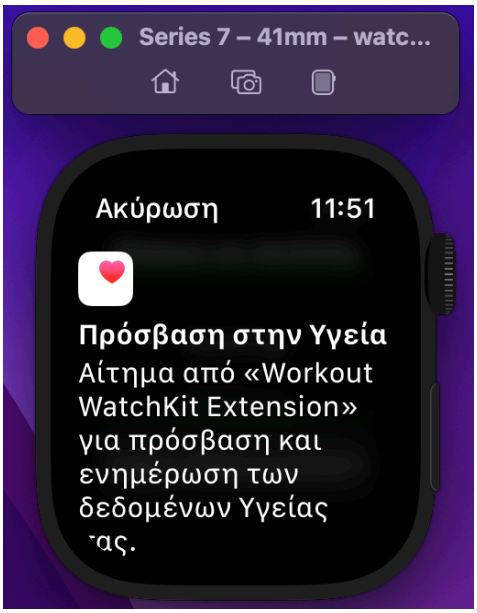


Εικόνα 17 Allow notifications

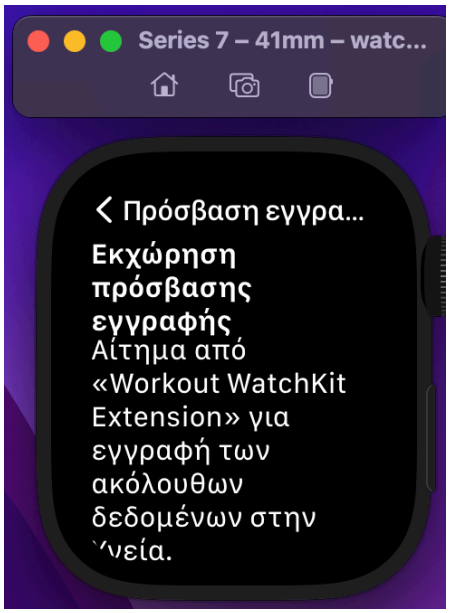
```
func allowNotification() {
    let center = UNUserNotificationCenter.current()

    center.requestAuthorization(options: [.alert, .badge, .sound]) { (granted, error) in
        if granted {
            print("Yay!")
        } else {
            print("D'oh")
        }
    }
}
```

Εικόνα 18 Allow notifacations function

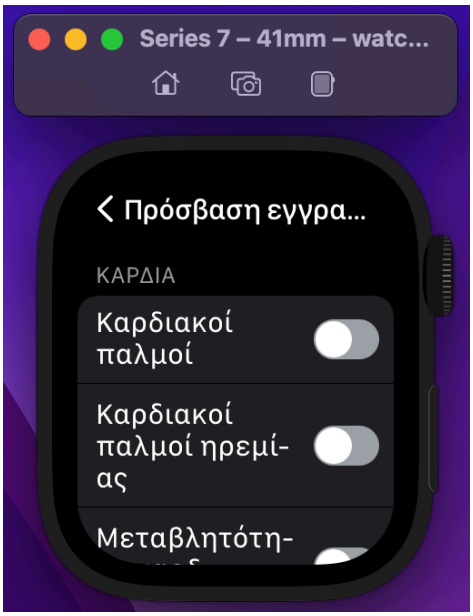
Αντίστοιχα, παρατηρείται το αίτημα για άδεια μέτρησης από την εφαρμογή των ζωτικών λειτουργιών του χρήστη και τον κώδικα που το υλοποιεί. Συγκεκριμένα, επιτυγχάνεται μέσω της κλήσης της συνάρτησης requestAuthorization() αυτή τη φορά στο HKHealthStore για συγκεκριμένα μεγέθη (quantities).



Εικόνα 19 Allow measurements 1



Εικόνα 20 Allow measurements 2



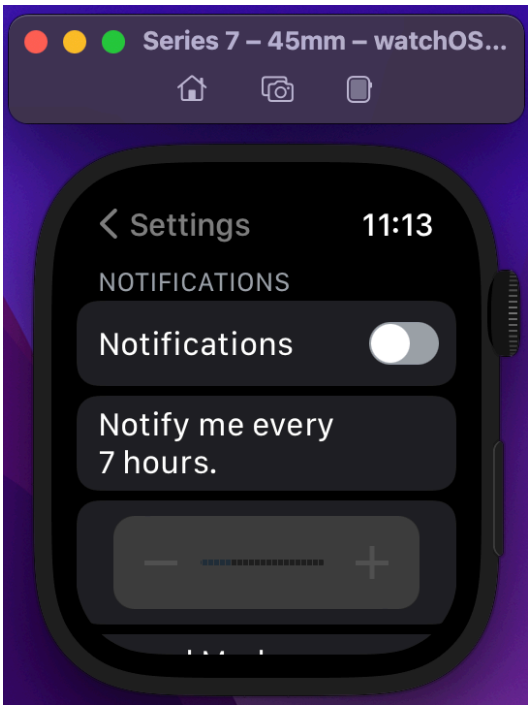
Εικόνα 21 Allow measurements 3

```swift
func start() {
    let sampleTypes: Set<HKSampleType> = [
        .workoutType(),
        .quantityType(forIdentifier: .heartRate)!,
        .quantityType(forIdentifier: .heartRateVariabilitySDNN)!,
        .quantityType(forIdentifier: .restingHeartRate)!,
        .quantityType(forIdentifier: .oxygenSaturation)!,
        .quantityType(forIdentifier: .distanceWalkingRunning)!,
        .quantityType(forIdentifier: .distanceDownhillSnowSports)!,
        .quantityType(forIdentifier: .distanceCycling)!
    ]
    healthStore.requestAuthorization(toShare: sampleTypes, read: sampleTypes) { success, error in
        if success {
            self.beginWorkout()
        }
    }
}
```

Εικόνα 22 Allow measurements function

4.2.5 Δημιουργία ειδοποιήσεων

Στην περίπτωση που παραχωρηθεί το δικαίωμα στην εφαρμογή για αποστολή ειδοποιήσεων, τότε παρέχεται η δυνατότητα στον χρήστη ή ακόμα και στο γιατρό να προγραμματίζει ειδοποιήσεις ανά κάποιο διάστημα ωρών. Έτσι, γίνεται υπενθύμιση στο χρήστη πότε πρέπει να λάβει μέτρηση και παράλληλα έχει τη δυνατότητα παραμετροποίησης της συχνότητας αποστολής ειδοποίησης μέσω των ρυθμίσεων όπως φαίνεται στην διπλανή εικόνα. Για να γίνει αυτό εφικτό, καλείται η συνάρτηση func scheduleNotification(repeat_time: Float) δίνοντας ως παράμετρο το χρονικό διάστημα μεταξύ 2 διαδοχικών ειδοποιήσεων. Εκεί ορίζεται το περιεχόμενο του notification, δηλαδή τον τίτλο, το μήνυμα, τον ήχο κλπ (γραμμές 42-46) και στην συνέχεια δημιουργείται ένα επαναλαμβανόμενο trigger κάθε όσες ώρες λήφθηκε ως παράμετρο (γραμμή 48). Τέλος, προωθείται το αίτημα στο UNUserNotificationCenter του ρολογιού (γραμμές 51-54).

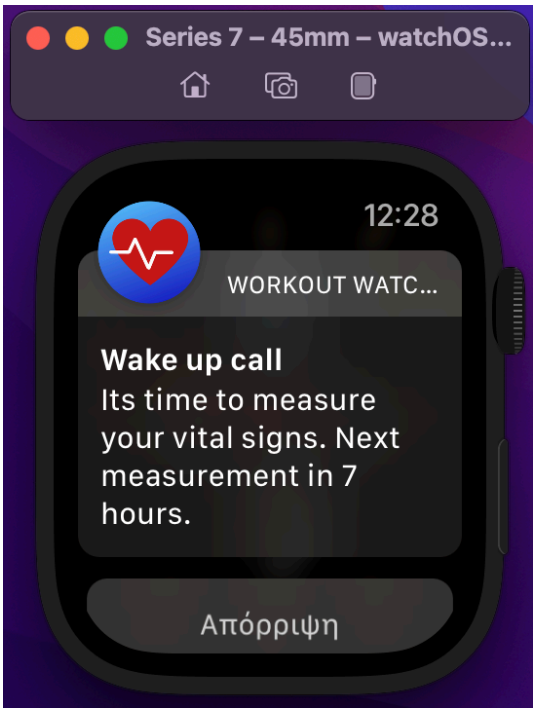


Εικόνα 23 Ενεργοποίηση ειδοποιήσεων

```swift
func scheduleNotification(repeat_time: Float) {
    let hours: Double = Double(repeat_time)
    // Remove old notification if existed
    self.removeNotification()

    // Save new notication id
    let notification_id = UUID().uuidString
    let defaults = UserDefaults.standard
    defaults.set(notification_id, forKey: "notification_id")
    defaults.synchronize()

    //Create new notification
    let center = UNUserNotificationCenter.current()
    let content = UNMutableNotificationContent()

    content.title = "Wake up call"
    content.body = "Its time to measure your vital signs. Next measurement in " + String(format: "%.0f", hours) + " hours."
    content.categoryIdentifier = "alarm"
    content.userInfo = ["customData": "fizzbuzz"]
    content.sound = UNNotificationSound.default

    let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 60 * hours, repeats: true)
//        let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 60, repeats: true)

    let request = UNNotificationRequest(identifier: notification_id,
                                        content: content,
                                        trigger: trigger)
    center.add(request)
}
```

Εικόνα 24 Create notification function

Οι παραπάνω ενέργειες έχουν ως αποτέλεσμα να εμφανιστεί στον χρήστη η παρακάτω ειδοποίηση.

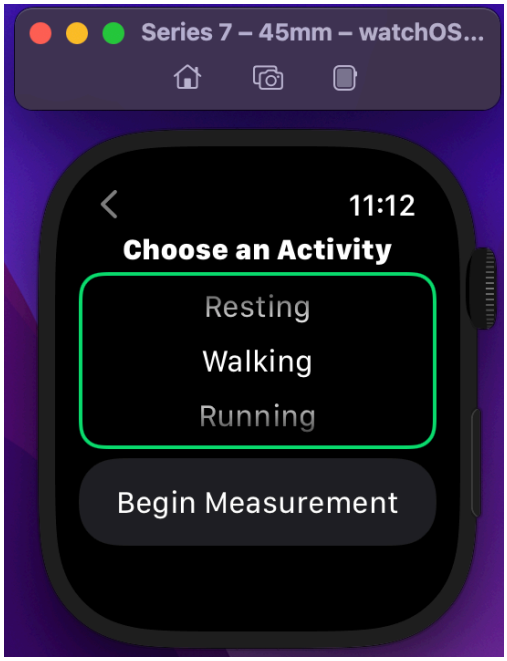


Εικόνα 25 Ειδοποίηση λήψης μέτρησης

4.2.6 Λήψη μετρήσεων

4.2.6.1 Αρχικοποίηση του Workout Session

Έχοντας λάβει από τον χρήστη έγκριση ώστε η εφαρμογή μας να χρησιμοποιεί τους αισθητήρες, ο προγραμματιστής έχει τη δυνατότητα να κάνει αίτηση στο Healthkit για να λάβει τις τιμές των μετρήσεων για τους καρδιακούς παλμούς, το οξυγόνο, το HRV και την απόσταση. Όπως έχει αναφερθεί, η Apple δεν παρέχει άμεση πρόσβαση στους αισθητήρες και κρατά κρυπτογραφημένη την υλοποίηση της. Έτσι, ο μοναδικός τρόπος είναι η προώθηση ενός request για τις ενδιαφερόμενες τιμές και το Healthkit απαντά στο αίτημα, όταν έχει διαθέσιμες τιμές. Για τον σκοπό αυτό, δημιουργείται ένα workout session κατά τη διάρκεια του οποίου λαμβάνονται συνεχώς οι μετρήσεις που είναι επιθυμητές. Πιο συγκεκριμένα, αρχικοποιείται το workout session δίνοντάς του ως παραμέτρους τον τύπο άσκησης που κάνει ο χρήστης αφού τον επιλέξει ο ίδιος από την λίστα δραστηριοτήτων όπως φαίνεται στην εικόνα.



Εικόνα 26 Επιλογή δραστηριότητας

```
28                  Button("Begin Measurement")  {
29                      guard HKHealthStore.isHealthDataAvailable() else { return }
30
31                      dataManager.activity = activities[selectedActivity].type
32                      dataManager.activityString = activities[selectedActivity].name
33                      dataManager.start()
34                  }
```

Εικόνα 27 Αρχικοποίηση Workout Session

4.2.6.2 Έναρξη του Workout Session

Εφόσον ο χρήστης επιλέξει να εκκινήσει την μέτρηση, ξεκινάει ταυτόχρονα και ένας listener που περιμένει από το HealthStore να αποστείλει τιμές για τα αιτήματα που πραγματοποιήθηκαν. Συνεπώς, δουλειά του προγραμματιστή είναι να ακούει τις απαντήσεις, να αναγνωρίζει ποιό μέγεθος αφορά, και στη συνέχεια να αποθηκεύει την τιμή που έλαβε μαζί με τη χρονική στιγμή που έφτασε. Όλες αυτές οι τιμές αποθηκεύονται ανά κατηγορία και περιμένουν να εκπληρωθεί η συνθήκη τερματισμού για να αποθηκευτούν στη βάση δεδομένων του Firebase (γραμμές 146-154). Στο παρακάτω παράδειγμα κώδικα φαίνεται ο διακόπτης switch που περιέχει όλες τις περιπτώσεις μετρήσεων που λαμβάνει το πρόγραμμα από το HealthStore, καθώς και το παράδειγμα

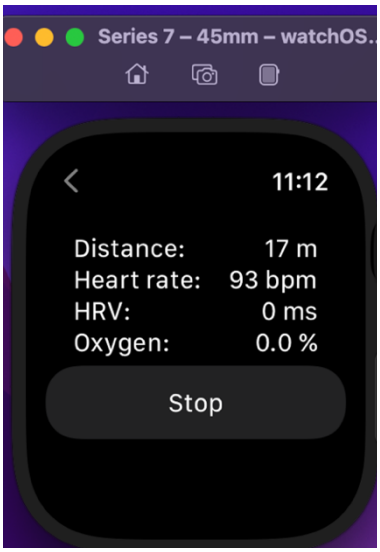
της τιμής για τον καρδιακό παλμό. Παράλληλα, ανάλογα τον τύπο, η τιμή που λήφθηκε υπόκειται και στις ανάλογες μετατροπές μονάδας (σειρές 137-138). Με αντίστοιχο τρόπο γίνεται η συλλογή των δεδομένων και για τις υπόλοιπες μετρήσεις (οξυγόνο, HRV, απόσταση).

```
129         for type in collectedTypes {
130             guard let quantityType = type as? HKQuantityType else { continue }
131             guard let statistics = workoutBuilder.statistics(for: quantityType) else { continue }
132 
133             DispatchQueue.main.async {
134                 switch statistics.quantityType {
135                 case HKQuantityType.quantityType(forIdentifier: .heartRate):
136                     if (self.heartCounter != Int(self.measurement_counter)) {
137                         let heartRateUnit = HKUnit.count().unitDivided(by: .minute())
138                         self.lastHeartRate = statistics.mostRecentQuantity()?.doubleValue(for: heartRateUnit) ?? 0
139 
140                         let today = Date()
141                         self.heartTimesArray.append(today.toString(dateFormat: "yyyy-MM-dd-HH:mm:ss:SSS"))
142                         self.heartRate.append(self.lastHeartRate)
143 
144                         self.heartCounter += 1
145                         if (self.heartCounter == Int(self.measurement_counter)) {
146                             self.sendRequest() { data in
147                                 if let data = data {
148                                     print(data)
149                                     if (data == "true") {
150                                         self.invalidMeasurement = false
151                                         return
152                                     }
153                                 }
154                             }
155                             let lastminute = today.toString(dateFormat: "yyyy-MM-dd-HH:mm")
156                             let even = Dictionary(uniqueKeysWithValues: zip(self.heartTimesArray, self.heartRate))
157                             self.ref.child("users/\(self.userID ??
                                 "N/A")/heartRate/\(self.activityString)/\(lastminute)").setValue(even)
158                             self.ref.child("users/\(self.userID ?? "N/A")/lastType").setValue(self.activityString)
159                             self.ref.child("users/\(self.userID ?? "N/A")/lastTimestamp").setValue(lastminute)
160 //                            self.end()
161                             if (self.oxygenCounter == 1) {
162                                 self.end()
163                             }
```

Εικόνα 28 Συλλογή μετρήσεων

4.2.6.3 Προβολή των μετρήσεων

Κατά την διάρκεια του Workout Session κάθε τιμή που έρχεται από τους αισθητήρες προβάλλεται ταυτόχρονα στην οθόνη του ρολογιού όπως παρουσιάζεται παρακάτω. Ο κώδικας που υλοποιεί το συγκεκριμένο view και ενημερώνει τις αντίστοιχες τιμές φαίνεται στην επόμενη φωτογραφία.



Εικόνα 29 Προβολή μετρήσεων

```
func hrvValue() -> String {
    var mutatableSelf = self
    return mutatableSelf.hrv
}
lazy var array:[String] = ["Distance " , distanceValue() , " m"]

var body: some View {
    VStack {
        Group {
            HStack {
                Text("Distance:")
                Spacer()
                Text(distanceValue() + " m")
            }
            HStack {
                Text("Heart rate:")
                Spacer()
                Text(heartRateValue() + " bpm")
            }
```

Εικόνα 30 Συνάρτηση ενημέρωσης και προβολής τιμών

4.2.7 Έλεγχος μέτρησης

Όταν τελειώσει η μέτρηση και πριν αποθηκευτεί στη βάση δεδομένων, ελέγχονται οι τιμές που λήφθηκαν από τους αισθητήρες. Με αυτόν τον τρόπο, υπάρχει η βεβαιότητα ότι έγινε σωστά η μέτρηση και δεν έχουν συμβεί σφάλματα όπως ο χρήστης να φοράει χαλαρά το ρολόι ή να έχουν βραχεί οι αισθητήρες. Ο έλεγχος πραγματοποιείται με τη βοήθεια των Cloud Functions της Firebase που επιτελούν ρόλο backend και αναλύονται στην επόμενη υποενότητα. Στην παρακάτω φωτογραφία του κώδικα φαίνεται ο τρόπος επικοινωνίας με το backend, καθώς καλείται η συνάρτηση που υπολογίζει τα φυσιολογικά όρια μέτρησης και επιστρέφει true αν οι τιμές ανήκουν μέσα σε αυτά ή false αν τα υπερβαίνουν. Στη συνέχεια, σε περίπτωση true αποθηκεύεται η μέτρηση, ενώ σε αντίθετη περίπτωση ειδοποιείται ο χρήστης μέσω alert, για να επαναλάβει την διαδικασία.

```
func sendRequest (completion: @escaping (String?)->()){
    let userIdString: String = String(self.userID ?? "")
    let urlString = "https://us-central1-healthwatchapp-e636f.cloudfunctions.net/validatemeasurement?uid=" +
        userIdString
    if let url = URL(string: urlString) {
        print(url)
        URLSession.shared.dataTask(with: url ) { data, response, error in
            if error == nil {
                if let data = data {
                    if let jsonString = String(data: data, encoding: .utf8) {
                        print(jsonString)
                        return completion(jsonString)
                    }
                }
            }else{
                print(error ?? "Error")
                return completion(nil)
            }
        }.resume()
    }
    else {
```

Εικόνα 31 Έλεγχος μέτρησης από την εφαρμογή

Εικόνα 32 Ειδοποίηση επανάληψης μέτρησης

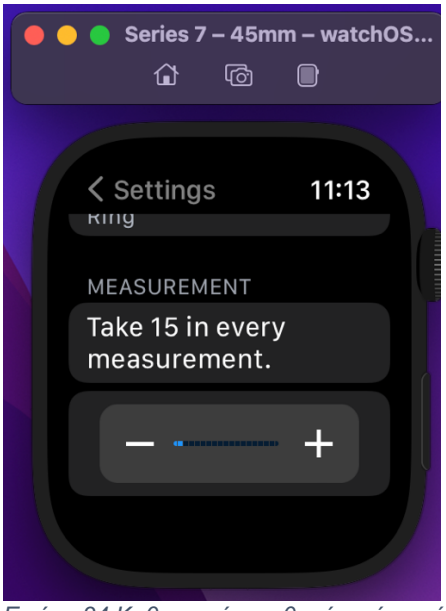
4.2.8 Λήξη της μέτρησης

Κάθε φορά που ο χρήστης εκκινεί μια νέα διαδικασία μέτρησης απαιτείται τόσο να λάβει έγκυρες τιμές, όσο και να συμπληρωθεί ένας αριθμός αυτών για να ολοκληρωθεί η καταγραφή. Συγκεκριμένα, έχει οριστεί μια παράμετρος, ο αριθμός μετρήσεων ανά καταγραφή που αποτελεί και τη συνθήκη τερματισμού. Πιο συγκεκριμένα, αν η μεταβλητή αυτή έχει τιμή 10 και το ρολόι λάβει 10 μετρήσεις για τους καρδιακούς παλμούς από τους αισθητήρες, τότε χρειάζεται να επιβεβαιώσει πως οι τιμές ανήκουν σε φυσιολογικά όρια (γραμμές 146-153). Αν η απάντηση είναι θετική, η διαδικασία θα λάβει και την τελευταία μέτρηση οξυγόνου και HRV από το Healthkit. Ακολούθως, πραγματοποιείται η εγγραφή των τιμών στην βάση δεδομένων στο Firebase (γραμμές 157-159). Τότε και μόνο τότε θα καλέσει την συνάρτηση end() (γραμμή 162) που τερματίζει την διαδικασία.

```
129         for type in collectedTypes {
130             guard let quantityType = type as? HKQuantityType else { continue }
131             guard let statistics = workoutBuilder.statistics(for: quantityType) else { continue }
132 
133             DispatchQueue.main.async {
134                 switch statistics.quantityType {
135                 case HKQuantityType.quantityType(forIdentifier: .heartRate):
136                     if (self.heartCounter != Int(self.measurement_counter)) {
137                         let heartRateUnit = HKUnit.count().unitDivided(by: .minute())
138                         self.lastHeartRate = statistics.mostRecentQuantity()?.doubleValue(for: heartRateUnit) ?? 0
139 
140                         let today = Date()
141                         self.heartTimesArray.append(today.toString(dateFormat: "yyyy-MM-dd-HH:mm:ss:SSS"))
142                         self.heartRate.append(self.lastHeartRate)
143 
144                         self.heartCounter += 1
145                         if (self.heartCounter == Int(self.measurement_counter)) {
146                             self.sendRequest() { data in
147                                 if let data = data {
148                                     print(data)
149                                     if (data == "true") {
150                                         self.invalidMeasurement = false
151                                         return
152                                     }
153                                 }
154                             }
155                             let lastminute = today.toString(dateFormat: "yyyy-MM-dd-HH:mm")
156                             let even = Dictionary(uniqueKeysWithValues: zip(self.heartTimesArray, self.heartRate))
157                             self.ref.child("users/\(self.userID ??
                                 "N/A")/heartRate/\(self.activityString)/\(lastminute)").setValue(even)
158                             self.ref.child("users/\(self.userID ?? "N/A")/lastType").setValue(self.activityString)
159                             self.ref.child("users/\(self.userID ?? "N/A")/lastTimestamp").setValue(lastminute)
160 //                            self.end()
161                             if (self.oxygenCounter == 1) {
162                                 self.end()
163                             }
```

Εικόνα 33 Συνθήκη τερματισμού measurement

Αυτήν τη μεταβλητή αυτή τερματισμού των μετρήσεων μπορεί να καθορίσει τόσο ο χρήστης μέσω των ρυθμίσεων, όπως φαίνεται στην παρακάτω εικόνα, όσο και κάποιος απομακρυσμένος διαχειριστής (γιατρός). Μέσω αλλαγής που συμβαίνει στο Cloud θα πραγματοποιείται αλλαγή και στην εφαρμογή.



Εικόνα 34 Καθορισμός αριθμού τιμών ανά μέτρηση

4.3 Ιστοσελίδα

4.3.1 Σύνδεση με firebase

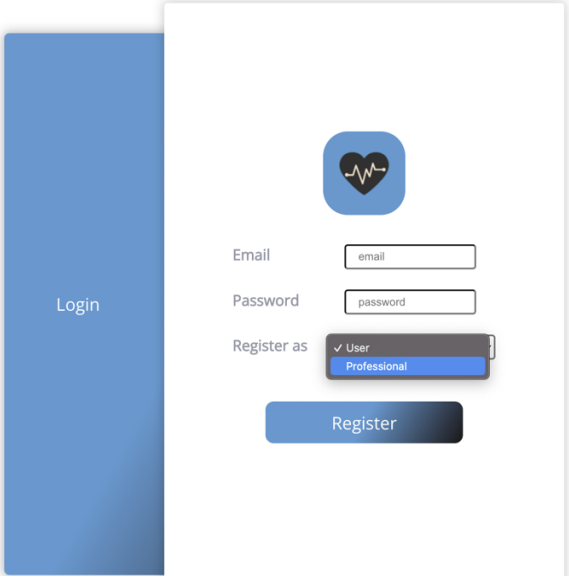
Όπως και στην εφαρμογή, για να πραγματοποιηθεί η σύνδεση με το project της Firebase είναι απαραίτητο ένα αρχείο το οποίο δημιουργείται αυτόματα και περιέχει όλα τα διαπιστευτήρια, ώστε να συνδεθούν επιτυχώς όλα τα services. Ο κώδικας παρουσιάζεται στην παρακάτω φωτογραφία.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";

import { getDatabase } from "firebase/database";

import { getAuth, createUserWithEmailAndPassword, signInWithEmailAndPassword } from "firebase/auth";

// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
    apiKey: "[redacted]",
    authDomain: "[redacted]",
    databaseURL: "[redacted]",
    projectId: "[redacted]",
    storageBucket: "[redacted]",
    messagingSenderId: "[redacted]",
    appId: "[redacted]",
    measurementId: "[redacted]"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

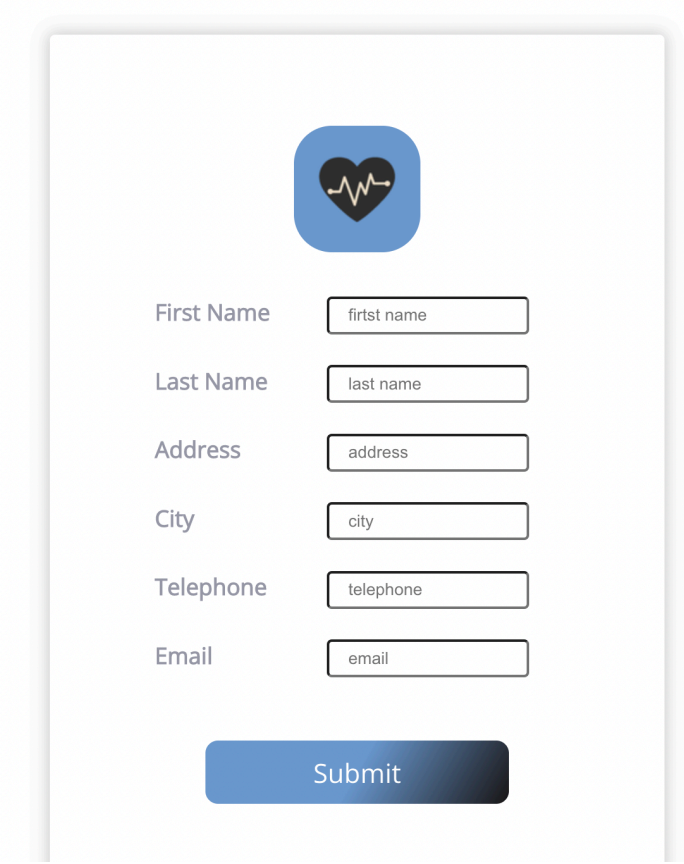
Εικόνα 35 Firebase configuration file JavaScript

4.3.2 Εγγραφή και Σύνδεση χρηστών και επαγγελματιών

Ακολουθεί παρόμοια λογική όπως παρουσιάστηκε και στην εφαρμογή με τη διαφορά πως ο κώδικας είναι γραμμένος σε JavaScript και όχι σε Swift. Όπως φαίνεται και στη φωτογραφία δίνεται η επιλογή είτε σύνδεσης είτε εγγραφής, διαλέγοντας με την βοήθεια του menu. Μια ακόμα προσθήκη στην ιστοσελίδα είναι πως κατόπιν επιλογής της εγγραφής, ο χρήστης καλείται να διαλέξει αν είναι χρήστης ή επαγγελματίας υγείας. Η εφαρμογή μας στο Apple Watch δεν είχε τέτοια δυνατότητα καθώς προορίζεται αποκλειστικά για χρήστες. Στην ιστοσελίδα όμως οι επαγγελματίες μπορούν να ελέγχουν τις μετρήσεις των ασθενών τους και να αλλάζουν την συχνότητα και το πλήθος των μετρήσεων τους.



Εικόνα 36 Σύνδεση-Εγγραφή μέσω της ιστοσελίδας



Εικόνα 37 Επιπλέον πληροφορίες επαγγελματία

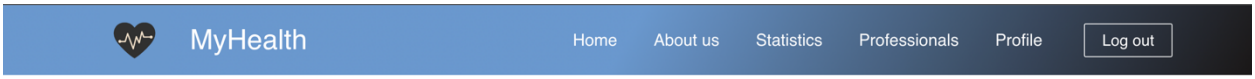
Έτσι, αν επιλεχθεί το “Professional” ο χρήστης θα κληθεί να δώσει μερικές παραπάνω πληροφορίες ώστε να είναι δυνατό να καταχωρηθεί ως επαγγελματίας. Αυτές φαίνονται στην παρακάτω εικόνα και είναι στην ουσία πληροφορίες απαραίτητες για έναν ασθενή ώστε να μπορεί να γνωρίζει τον γιατρό του και να έρχεται σε επικοινωνία με αυτόν. Όταν ολοκληρωθεί η συμπλήρωση της φόρμας και όλων των πεδίων της (σε περίπτωση παράλειψης κάποιου έρχεται ειδοποίηση και δεν συνεχίζεται η εγγραφή μέχρι την υποβολή σωστής δήλωσης) τα στοιχεία εγγράφονται στην βάση δεδομένων στην κατηγορία admin με βάση το id του επαγγελματία.

4.3.3 Navbar

Σε όλες τις σελίδες της εφαρμογής μας υπάρχει μία γραμμή πλοήγησης, η οποία παρέχει τη δυνατότητα στο χρήστη να αλλάζει γρήγορα σελίδα και να πηγαίνει εκεί που επιθυμεί χωρίς να χρειάζεται να επιστρέφει στο βασικό menu. Αυτή η γραμμή έχει 3 μορφές ανάλογα με το αν ο χρήστης είναι συνδεδεμένος στον λογαριασμό του, επαγγελματίας ή όχι όπως φαίνεται παρακάτω.

Εικόνα 38 Navbar μη συνδεδεμένου χρήστη

Εικόνα 39 Navbar συνδεδεμένου επαγγελματία



Εικόνα 40 Navbar συνδεδεμένου χρήστη

Αυτό γίνεται με την βοήθεια της συνάρτησης useEffect() και συγκεκριμένα της onAuthStateChanged() της Firebase, η οποία διαπιστώνει τον τύπο του χρήστη. Στη συνέχεια, καλεί το αντίστοιχο component Navbar που προβάλλεται σαν αποτέλεσμα.

```
247     useEffect(() => {
248       const auth = getAuth();
249       onAuthStateChanged(auth, (user) => {
250         if (user) {
251           setValidated(true);
252           // User is signed in, see docs for a list of available properties
253           // https://firebase.google.com/docs/reference/js/firebase.User
254           const uid = user.uid;
255           const dbRef = ref(getDatabase());
256           get(child(dbRef, `admins/${uid}`)).then((snapshot) => {
257             if (snapshot.exists()) {
258               setAdmin(true);
259             }
260             else {
261               console.log("its a user");
262             }
263           }).catch((error) => {
264             console.error(error);
265           });
266         }
```

Εικόνα 41 Διαπίστωση του τύπου χρήστη

4.3.4 Περιήγηση

```
49   function App() {
50     return (
51       <>
52         <Router>
53         <Navbar />
54         <Routes>
55           <Route path='/loginregister' element =
56           {<LoginRegister/>}/>
57           <Route path='/' element =
58           {<Home/>} />
59           <Route path='/profile_admin' element =
60           {<ProfileAdmin/>} />
61           <Route path='/profile_user' element =
62           {<ProfileUser/>} />
63           <Route path='/additionalinfo' element =
64           {<AdditionalInfo/>} />
65           <Route path='/aboutus' element =
66           {<AboutUs/>} />
67           <Route path='/admins' element =
68           {<ShowAdmins/>} />
69           <Route path='/users' element =
70           {<ShowUsers/>} />
71           <Route path="/user" element =
72           {<User/>} />
73           <Route path="/contactuser" element =
74           {<ContactUser/>} />
75           <Route path="/contactprofessional" element =
76           {<ContactAdmin/>} />
77         </Routes>
```

Εικόνα 42 Δήλωση διευθύνσεων των ιστοσελίδων

Η εναλλαγή μεταξύ των σελίδων της ιστοσελίδας γίνεται με τη βοήθεια της βιβλιοθήκης 'react-router-dom'. Έτσι στο αρχείο App.js δηλώνονται όλες οι διευθύνσεις <Routes> που θα χρειαστούν συσχετίζοντας την καθεμία με το αντίστοιχο στοιχείο που θα κληθεί όταν γίνει η μετάβαση σε εκείνη την σελίδα, όπως φαίνεται παρακάτω.

Με αυτόν τον τρόπο όταν εκκινεί το front-end, υπάρχει η δυνατότητα μετάβασης στις αντίστοιχες διευθύνσεις με τις εντολές <Link> και window.location.href. Η πρώτη περίπτωση χρησιμοποιείται για τη δήλωση ενός στοιχείου μέσα στην ιστοσελίδα, το οποίο όταν επιλεχθεί θα οδηγήσει το χρήστη σε έναν άλλον σύνδεσμο. Η δήλωση αυτή γίνεται μέσα στο κομμάτι render της κλάσης μας και επίσης υπάρχει η δυνατότητα να σταλούν στοιχεία και πληροφορίες στο element το οποίο θα κληθεί μέσω του αντίστοιχου state. Στο παρακάτω παράδειγμα, όταν επιλεχθεί ένας συγκεκριμένος χρήστης, πραγματοποιείται μετάβαση στο προφίλ του καλώντας την απαραίτητη κλάση, δίνοντάς της και τα στοιχεία του χρήστη για προβολή.

```
render() {
    const backUrl = '/some/other/value'
    const newTo = {
        pathname: "/user",
        param1: "Par1"
    };

    return (
        <div>
            <Link
                to={{pathname: "/user"}}
                state={{
                    user_id: this.state.user_id,
                    email: this.state.email,
                    allow_notifications: this.state.allow_notifications,
                    measurement_counter: this.state.measurement_counter,
                    repeat_time: this.state.repeat_time,
                    }}
            >
```

Εικόνα 43 Χρήση της εντολής <Link>

Η δεύτερη περίπτωση χρησιμοποιείται στην αλλαγή παραθύρων ύστερα από την κλήση κάποιας συνάρτησης. Επί παραδείγματι, στην παρακάτω φωτογραφία γίνεται η προσπάθεια εγγραφής νέου χρήστη στην βάση δεδομένων με τη βοήθεια της συνάρτησης createUserWithEmailAndPassword. Στη συνέχεια, παρατηρείται πως αν ο χρήστης είναι επαγγελματίας τότε απαιτείται να δώσει περαιτέρω πληροφορίες, οπότε και οδηγείται με την εντολή windows.location.href στην σελίδα /additionalinfo, ενώ αν είναι απλός χρήστης τότε οδηγείται αυτόματα στο βασικό μενού.

```
createUserWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
        // Signed in
        const user = userCredential.user;
        Swal.fire({
            title: 'Success',
            text: "Succesful register!",
            icon: 'success',
            customClass: "swal_ok_button",
            confirmButtonColor: "#6a98ce"
        }).then(function () {
            if (role == "admin") {
                window.location.href = '/additionalinfo';
                console.log("new amdin");
            }
            else {
                window.location.href = '/';
                console.log("new user");
            }
        });
    })
```

Εικόνα 44 Χρήση της εντολής windows.location.href

4.3.5 Άντληση και φόρτωση δεδομένων

Παρόμοια με την εφαρμογή, είναι αναγκαία τόσο η απόκτηση των πληροφοριών που βρίσκονται στη βάση, όσο και η εγγραφή νέων. Οι δύο αυτές βασικές διαδικασίες γίνονται με τη χρήση συναρτήσεων της Firebase. Στην περίπτωση ανάγνωσης πληροφοριών από τη βάση, η συνάρτηση get(), επιστρέφει αν επιτύχει ένα snapshot της διεύθυνσης της βάσης που έγινε η αίτηση. Στο παρακάτω παράδειγμα φαίνεται η συνάρτηση, στην οποία παρέχεται ως ορίσματα μια αναφορά στη βάση και τη διεύθυνση (path) στο οποίο βρίσκονται οι επιθυμητές πληροφορίες.

```
get(child(dbRef, `users/${this.state.id_user}/oxygen/`)).then(
  (snapshot) => {
    var oxygenmonthdata = [];
    for (let el in snapshot.val()) {
```

Εικόνα 45 Χρήση της συνάρτησης get() σε JavaScript

Αντίστοιχα, πραγματοποιείται και η εγγραφή πληροφοριών με τη βοήθεια της συνάρτησης set(). Όπως προηγουμένως, παρέχεται στη συνάρτηση αναφορά στην βάση και την διεύθυνση στην οποία έχει επιλεχτεί να αποθηκευτούν τα δεδομένα, ενώ στο τέλος προωθούνται τα δεδομένα σε μορφή json όπως παρουσιάζεται παρακάτω.

```
const uid = user.uid;

set(ref(database, 'admins/' + uid), {
    first_name: first_name,
    last_name: last_name,
    address: address,
    city: city,
    telephone: telephone,
    email: email,
    total_users: 0
})
```

Εικόνα 46 Χρήση της συνάρτησης set() σε JavaScript

Τέλος, υπάρχει η δυνατότητα ανανέωσης των δεδομένων χωρίς νέα εγγραφή που οδηγεί στη διαγραφή όσων πληροφοριών ανήκουν στο ίδιο κλαδί (διεύθυνση αποθήκευσης). Η εντολή αυτή της Firebase είναι η update() και ακολουθεί την ίδια λογική με τις παραπάνω δύο εντολές. Παράδειγμα χρήσης της ακολουθεί παρακάτω. Αναλυτικότερα, καθίσταται σαφές πως σε αυτό το σημείο ανανεώνεται η τιμή της μεταβλητής allow_notifications για ένα συγκεκριμένο user. Αν γινόταν χρήση της συνάρτησης set() σε αυτό το σημείο θα διαγράφονταν όλες οι υπόλοιπες πληροφορίες που υπάρχουν στο χρήστη όπως οι μετρήσεις, προσωπικές πληροφορίες κλπ.

```
update(ref(db, "users/" + uid), {
  allow_notifications: this.state.allow_notifications.toString(),
});
```

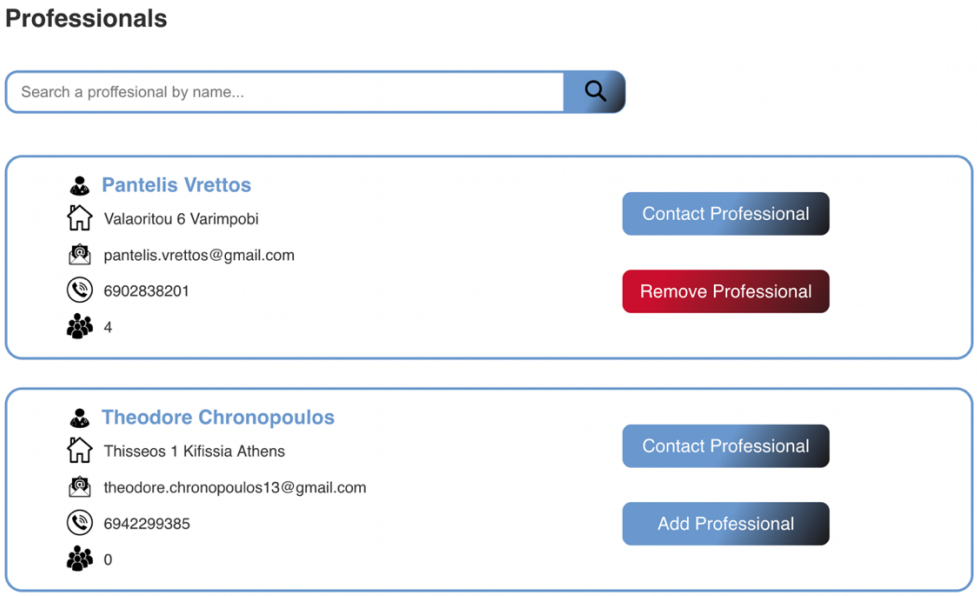
Εικόνα 47 Χρήση της συνάρτησης update() σε JavaScript

Η σύνδεση με την Firebase, η σύνδεση και η εγγραφή νέων χρηστών καθώς και η ανταλλαγή δεδομένων αποτελούν βασικά στοιχεία της ιστοσελίδας, όχι όμως το βασικό σκοπό δημιουργίας της. Η επιλογή επαγγελματία από τον χρήστη, η επικοινωνία χρήστη και επαγγελματία υγείας, η αλλαγή των ρυθμίσεων λήψης μετρήσεων καθώς και η οπτικοποίηση των μετρήσεων που λήφθηκαν από το ρολόι και η εξαγωγή

συμπερασμάτων είναι οι λειτουργίες που τη διαφοροποιούν και την καθιστούν απαραίτητη. Αυτές οι λειτουργίες αναλύονται εκτενώς στη συνέχεια.

4.3.6 Αλληλεπίδραση μεταξύ χρήστη και επαγγελματία υγείας

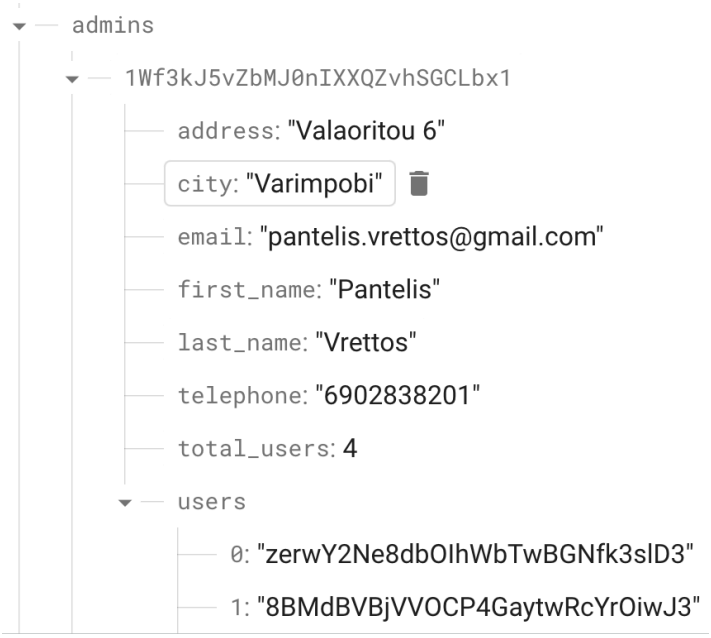
Με πλοήγηση στην καρτέλα “Professionals”, ο χρήστης έχει τη δυνατότητα να δει και να αναζητήσει όλους τους επαγγελματίες υγείας που έχουν συνδεθεί στην εφαρμογή. Για κάθε επαγγελματία παρέχονται διάφορες πληροφορίες επικοινωνίας όπως διεύθυνση εργασίας, τηλέφωνο και email, ενώ αναγράφεται και ο συνολικός αριθμός χρηστών που έχει υπό την επίβλεψή του.



Εικόνα 48 Μενού επιλογής επαγγελματία υγείας

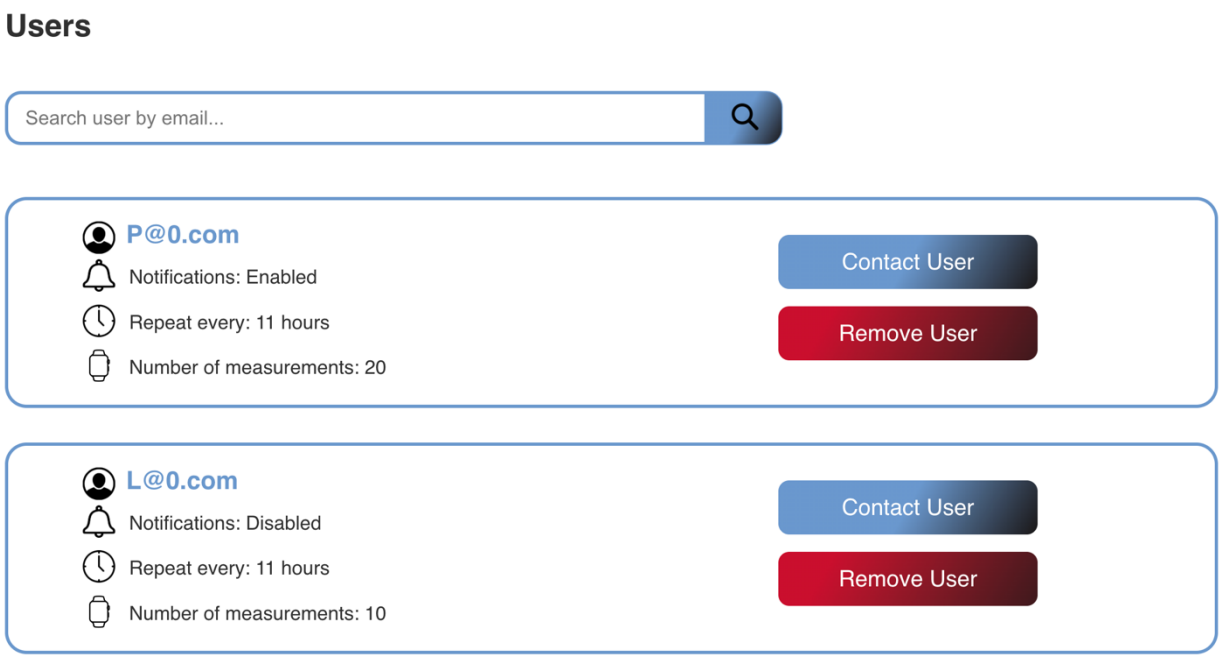
Κάθε επιλογή επαγγελματία διαθέτει δύο κουμπιά. Το πρώτο οδηγεί το χρήστη στη φόρμα επικοινωνίας με το συγκεκριμένο επαγγελματία. Το δεύτερο προσθέτει το γιατρό στη λίστα εκείνων που μπορούν να επιβλέπουν τις μετρήσεις του και να αλλάζουν τις ρυθμίσεις κάτω από τις οποίες λαμβάνονται. Σε περίπτωση επιλογής, το id του επαγγελματία εγγράφεται στο αντίστοιχο σημείο της βάσης όπου αναφέρεται στους επιβλέποντες του τρέχοντος χρήστη. Παράλληλα, το id του χρήστη μπαίνει στη λίστα των ατόμων που παρακολουθεί ο γιατρός στο αντίστοιχο σημείο της βάσης όπως φαίνεται στις παρακάτω φωτογραφίες.

Εικόνα 49 Δέντρο αποθήκευσης πληροφοριών χρήστη



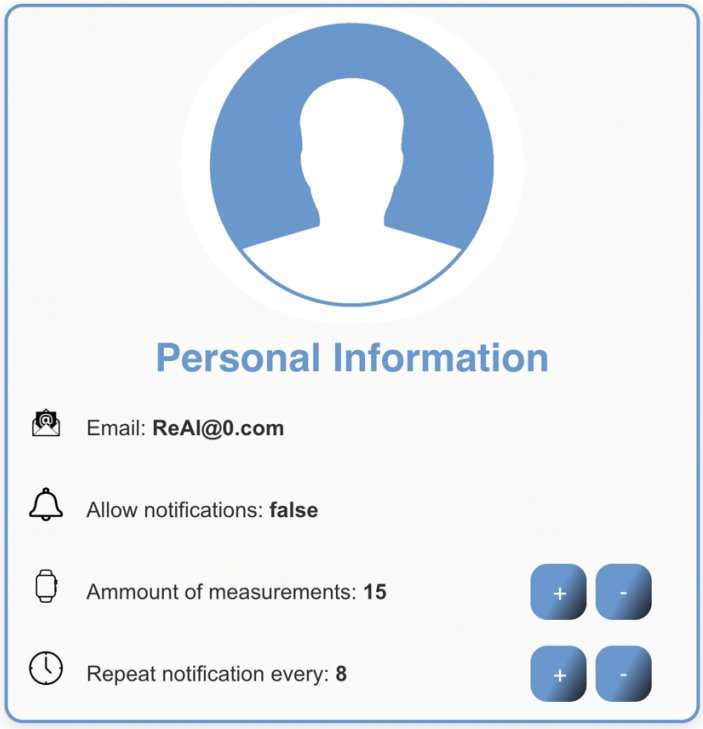
Εικόνα 50 Δέντρο αποθήκευσης πληροφοριών επαγγελματία

Η επιλογή της διαγραφής του επαγγελματία από τη μεριά του χρήστη, ή του χρήστη από την ιστοσελίδα του επαγγελματία οδηγεί στη διαγραφή των ids και από τα δύο σημεία που είχε γινεί εγγραφή. Ακολούθως, παρουσιάζεται το μενού του επαγγελματία όπου βλέπει όλους τους χρήστες που έχει υπό την επίβλεψη του.



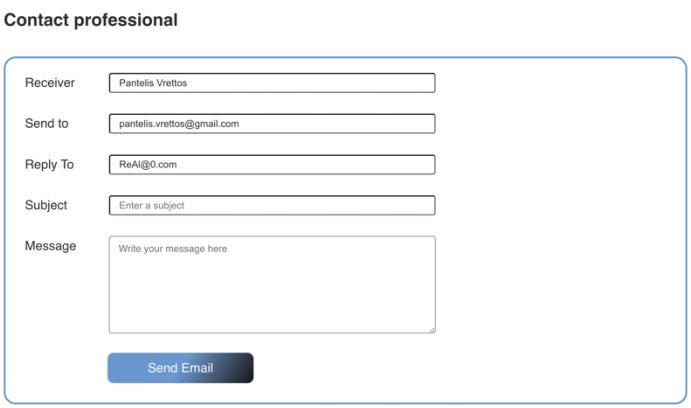
Εικόνα 51 Μενού επιλογής χρήστη

Τέλος, όπως γίνεται αντιληπτό το προφίλ κάθε χρήστη είναι προσβάσιμο μόνο από τον ίδιο ή από τους επιβλέποντες επαγγελματίες υγείας του. Εκεί, δίνεται η δυνατότητα προβολής των μετρήσεων, αλλά και αλλαγής των ρυθμίσεων λήψης της μέτρησης όπως φαίνεται στην φωτογραφία. Συγκεκριμένα, ο χρόνος υπενθύμισης για λήψη μέτρησης καθώς και ο αριθμός των συνολικών μετρήσεων μπορεί να τροποποιηθεί με την βοήθεια των αντίστοιχων κουμπιών. Η αλλαγή αυτή ενημερώνει κατάλληλα την βάση δεδομένων και στη συνέχεια η εφαρμογή του ρολογιού διαβάζει τις αλλαγές κάθε φορά πριν την έναρξη νέας μέτρησης.



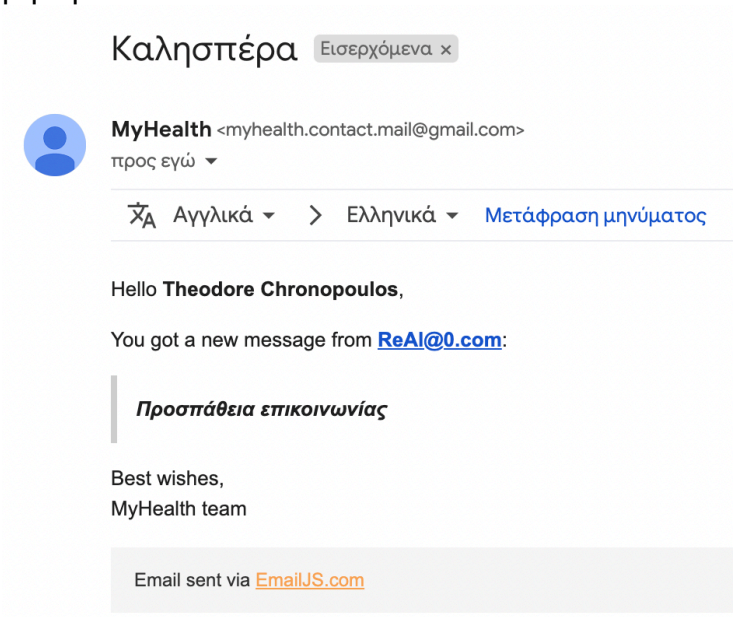
Εικόνα 52 Καρτέλα χρήστη

Όπως αναφέρθηκε προηγουμένως, η δυνατότητα “Contact Professional”, “Contact User” οδηγεί στην συμπλήρωση φόρμας email που παρουσιάζεται. Εκεί ο αποστολέας καλείται να προσθέσει το θέμα και κύριο περιεχόμενο του μηνύματος με τα υπόλοιπα στοιχεία αποστολής να είναι συμπληρωμένα με βάση τα στοιχεία που έχει δηλώσει ο κάθε χρήστης.



Εικόνα 53 Φόρμα επικοινωνίας

Όταν πατηθεί η αποστολή, ο παραλήπτης δέχεται ένα αυτοματοποιημένο email από το email της εφαρμογής (myhealth.contact.mail@gmail.com) που περιέχει το θέμα και το μήνυμα του αποστολέα.



Εικόνα 54 Αυτοματοποιημένο email

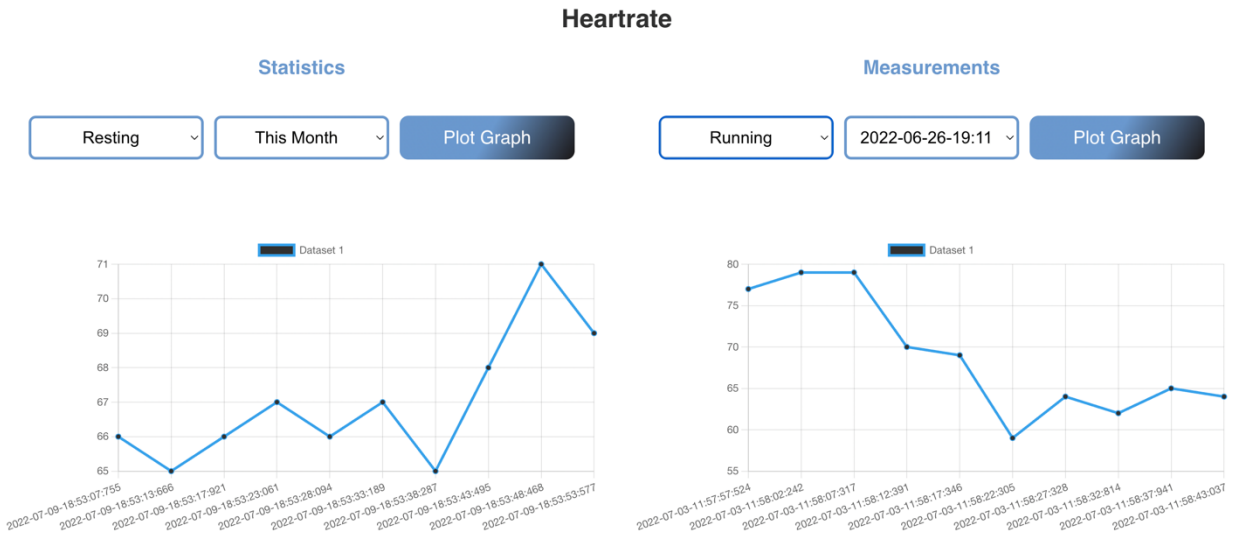
Τόσο η μορφή του αυτοματοποιημένου email, όσο και η αποστολή γίνεται με την βοήθεια ενός email server που συνδέεται με την εφαρμογή του EmailJS.com. Εκεί, δημιουργήθηκε ένας λογαριασμός ο οποίος συνδέθηκε με το email της Google και στη συνέχεια διαμορφώθηκαν οι φόρμες που θα αποστέλλονται κάθε φορά που ένας χρήστης θέλει να επικοινωνήσει. Η σύνδεση με τον server και η αποστολή του email, γίνεται με τη χρήση της συνάρτησης sendForm της βιβλιοθήκης '@emailjs/browser' προσθέτοντας ως ορίσματα τα id του λογαριασμού και της φόρμας που θέλουμε όπως παρακάτω. (21)

```
const serviceID = "[redacted]";
const templateID = "[redacted]";
const publicKey = "[redacted]"
console.log(form.current)

emailjs.sendForm(serviceID, templateID, form.current, publicKey)
    .then((result) => {
        console.log("SUCCESS!", result.status, result.text);
        Swal.fire({
            title: 'Success',
            text: "You have succesfully send a message!",
            icon: 'success',
            customClass: "swal_ok_button",
            confirmButtonColor: "#6a98ce"
        }).then(function () {
            window.location.href = '/';
        });
    }, (error) => {
```

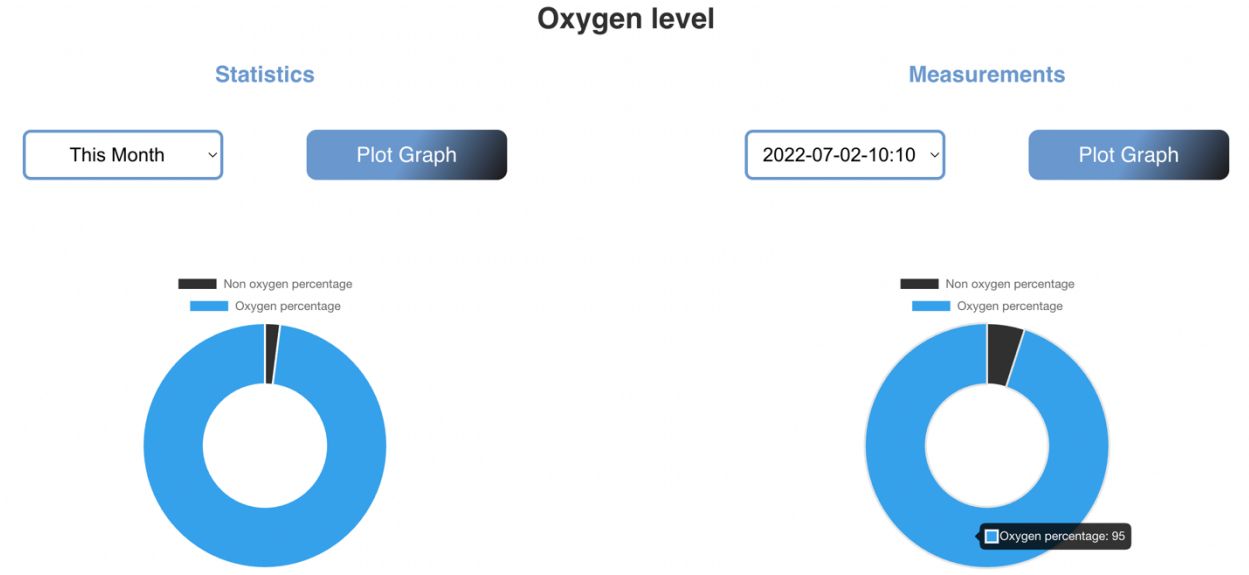
Εικόνα 55 Σύνδεση με EmailJS και αποστολή email

4.3.8 Προβολή μετρήσεων

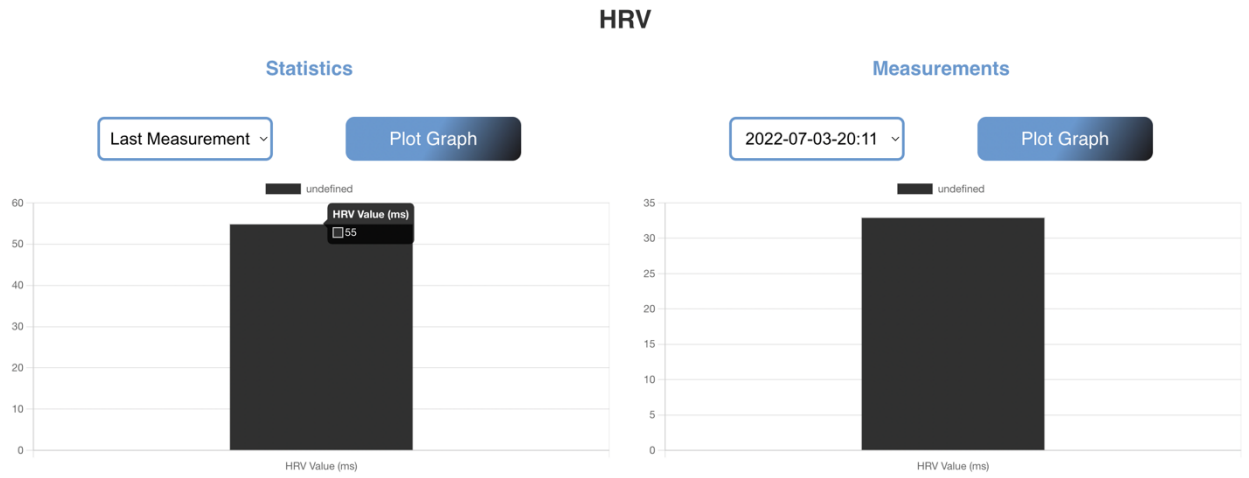
Πρωταρχικός σκοπός της δημιουργίας της ιστοσελίδας αποτελεί η προβολή των μετρήσεων ενός χρήστη. Έτσι στο προφίλ κάθε user, κάτω από τις γενικές πληροφορίες υπάρχουν τα διαγράμματα για την κάθε μετρική που λαμβάνεται από το ρολόι. Σε όλες τις μετρήσεις υπάρχει η δυνατότητα να προβληθεί η τελευταία μέτρηση είτε συγκεντρωτικά αποτελέσματα για τον τελευταίο μήνα όπου υπολογίζεται ο μέσος όρος των μετρήσεων. Παράλληλα, μπορούν να προβληθούν και μεμονωμένες μετρήσεις γεγονός ιδιαίτερα σημαντικό για την παρακολούθηση ενός ασθενή. Επιπρόσθετα, στους καρδιακούς παλμούς είναι αναγκαίο να επιλεχθεί και ο τύπος της άσκησης.



Εικόνα 56 Προβολή μετρήσεων καρδιακών παλμών



Εικόνα 57 Προβολή μετρήσεων οξυγόνου



Εικόνα 58 Προβολή μετρήσεων HRV

Για να γίνει εφικτή η προβολή, χρησιμοποιούνται τρία διαφορετικά components δίνοντάς τους τις τιμές που αντλούμε από τη βάση δεδομένων μας. Συγκεκριμένα, για τους καρδιακούς παλμούς χρησιμοποιείται το στοιχείο “Line”, για το οξυγόνο το “Doughnut“ και για το HRV το “Bar”, το οποία όλα προέρχονται από την βιβλιοθήκη "react-chartjs-2". Με αυτόν τρόπο, όταν ο χρήστης επιθυμεί να δει τις μετρήσεις του, αφού καθορίσει τις χρονικές παραμέτρους της μέτρησης που θέλει, αντλούνται τα δεδομένα από τη βάση, και όταν πατηθεί το κουμπί “Plot graph” καλείται το αντίστοιχο component. Παρακάτω βλέπουμε τις τρεις διαφορετικές συναρτήσεις που καλούνται.

```
export default function BarChart ({hrv}){
    const options = {
      responsive: true,
      plugins: {
        legend: {
          position: "top",
        },
        title: {
          display: true,
          text: 'dog',
        },
      },
    };

    const data = {
      labels: ['HRV Value (ms)'],
      datasets: [
        {
          data: [hrv],
          backgroundColor: ["#303030", "#36A2EB", "#FFCE56"],
          hoverBackgroundColor: ["#303030", "#36A2EB", "#FFCE56"],
          borderWidth: 1,
          barPercentage: 0.4,
        }
      ],
    };
      return <Bar data={data} />;
    }
```

Εικόνα 59 BarChart component

```
export default function DoughnutChart ({oxygen}){

const data = {
  labels: ['Non oxygen percentage', "Oxygen percentage"],
  datasets: [
    {
      data: [100-oxygen, oxygen],
      // backgroundColor: ["#FF6384", "#36A2EB", "#FFCE56"],
      backgroundColor: ["#303030", "#36A2EB", "#FFCE56"],
      hoverBackgroundColor: ["#303030", "#36A2EB", "#FFCE56"],
      borderWidth: 2
    }
  ]
};

  return <Doughnut data={data} />;
}
```

Εικόνα 60 Doughnut component

```
export default function LineChart({ heartrate, labels, timestamp }) {
  const options = {
    responsive: true,
    plugins: {
      legend: {
        position: "top",
      },
      title: {
        display: false,
        text: timestamp,
      },
    },
  };
  const data = {
    labels: [...labels],
    datasets: [
      {
        label: "Dataset 1",
        data: [...heartrate],
        borderColor: "#36A2EB",
        backgroundColor: "rgb(48,48,48)",
      },
    ],
  };
  return <Line options={options} data={data} />;
}
```

Εικόνα 61 Line component

4.3.9 Αξιολόγηση μετρήσεων

Τέλος, στο profile section κάθε χρήστη, υπάρχει ένα διάγραμμα που απεικονίζει τρείς γραφικές παραστάσεις όπως φαίνεται στην παρακάτω φωτογραφία. Η γραφική παράσταση, με όνομα “Last Measurement” και χρώμα μπλέ, είναι τα δεδομένα της τελευταίας μέτρησης που έχει καταχωρήσει ο χρήστης. Οι δύο οριζόντιες ευθείες γραμμές, με χρώμα κόκκινο και ονόματα “Lower Limit” και “Upper Limit” αναπαριστούν τα ακρότατα αποδεκτά όρια μέσα στα οποία πρέπει να είναι ο μέσος όρος της τελευταίας μέτρησης, προκειμένου να θεωρηθεί επιτυχής. Έτσι, όταν ο χρήστης πατήσει το κουμπί “Validate” βλέπει αν η τελευταία μέτρηση που πραγματοποίησε ανήκει στα επιτρεπτά όρια του συστήματος καθώς και ποια είναι αυτά.

Εικόνα 62 Διάγραμμα αξιολόγησης μέτρησης

Η υλοποίηση του διαγράμματος έγινε με τη χρήση της βιβλιοθήκης “Chart.js” της Javascript. Η δομή που δημιουργήθηκε για την αναπαράσταση τριών γραφημάτων φαίνεται παρακάτω. Σε σύγκριση με το “Line Chart” που χρησιμοποιήθηκε για την αναπαράσταση των προηγούμενων γραφημάτων, το “Multiple Chart” περιέχει δύο Datasets για την αναπαράσταση των επιπλέον δύο συναρτήσεων.

```
export default function MultipleLineChart({heartrate,lower,upper,labels }) {
    const options = {
      responsive: true,
      plugins: {
        legend: {
          position: "top",
        },
        title: {
          display: false,
        },
      },
    };
    const lower_array = [];
    const upper_arrray = [];
    for(let i=0; i<heartrate.length; i++){
      lower_array.push(lower);
      upper_arrray.push(upper);
    }
    const data = {
      labels: [...labels],
      datasets: [
        {
          label: "Last Measurement",
          data: [...heartrate],
          borderColor: "#36A2EB",
          backgroundColor: "rgb(48,48,48)",
        },
        {
          label: "Lower Limit",
          data: [...lower_array],
          borderColor: "rgb(255,0,0)",
          backgroundColor: "rgb(255,0,0)",
        },
        {
          label: "Upper Limit",
          data: [...upper_arrray],
          borderColor: "rgb(255,0,0)",
          backgroundColor: "rgb(255,0,0)",
        },
      ],
    };
    return <Line options={options} data={data} />;
  }
```

Εικόνα 63 MultipleLineChart component

Τα δεδομένα αυτά λαμβάνονται ως παράμετροι όταν καλείται η συνάρτηση.

```
<MultipleLineChart
heartrate={this.state.last_meas_data}
lower = {this.state.last_meas_lower}
upper = {this.state.last_meas_upper}
labels={this.state.last_meas_labels}
/>
```

Εικόνα 64 Παράμετροι MultipleLineChart

4.4 Μοντέλο επεξεργασίας μετρήσεων

4.4.1 Εισαγωγή

Το μοντέλο επεξεργασίας μετρήσεων βασίστηκε στη δημιουργία ενός backend, μέσω της χρήσης των Cloud Functions της Google. Η δομή αυτή έδωσε αυτονομία και οργάνωση στη διαδικασία ελέγχου των μετρήσεων, ενώ παράλληλα εξασφάλισε τα απαραίτητα πρωτόκολλα ασφάλειας. Εκτός από τα παραπάνω, αποτέλεσε σημαντικό χαρακτηριστικό επεκτασιμότητας, αφού παρέχει τη δυνατότητα δημιουργίας νέων ανεξάρτητων λειτουργιών.

Στο παρόν project χρησιμοποιήθηκαν δύο HTTPS Cloud συναρτήσεις για την επεξεργασία και αξιολόγηση της τελευταίας μέτρησης που έχει επιτελέσει ο χρήστης. Η πρώτη συνάρτηση χρησιμοποιείται από τη μεριά της ιστοσελίδας προκειμένου να απεικονίσει τα δεδομένα της μέτρησης προς εξέταση καθώς και τις ανώτατες και κατώτατες αποδεκτές τιμές. Η δεύτερη συνάρτηση χρησιμοποιείται από τη μεριά του Apple Watch με σκοπό την ενημέρωση του χρήστη εφόσον οι τιμές είναι εκτός των αναμενόμενων ορίων.

Πέραν όμως της ανάλυσης του κώδικα των συναρτήσεων που γίνεται στο τέλος της υποενότητας αυτής, ιδιαίτερα σημαντικό ρόλο έχουν και τα δεδομένα που χρησιμοποιήθηκαν. Για τον λόγο αυτό, αρχικά, παρουσιάζεται η προέλευση όλων διαφορετικών δεδομένων που χρησιμοποιήθηκαν.

4.4.2 Τα δεδομένα προσομοιώσεων

Τα δεδομένα που χρησιμοποιήθηκαν για τις προσομοιώσεις, τις γραφικές παραστάσεις και τη διεξαγωγή testing σε τμήματα κώδικα, προήλθαν από δύο διαφορετικές μεθόδους. Η πρώτη μέθοδος αφορά την παραγωγή real time δεδομένων καρδιακών παλμών, HRV και ποσοστιαίου οξυγόνου από το Apple Watch. Η δεύτερη μέθοδος αφόρα δημόσια δεδομένα καρδιακών παλμών μέσω της πλατφόρμας Kaggle.

4.4.2.1 Kaggle

Το Kaggle είναι μια δημόσια πλατφόρμα δεδομένων και τεχνητής νοημοσύνης. Βασική λειτουργία της πλατφόρμας είναι η δημοσίευση διαγωνισμών από εταιρείες και οργανισμούς. Εκτός από τους διαγωνισμούς, οι χρήστες έχουν τη δυνατότητα να ανεβάζουν σύνολα δεδομένων, καθώς και να έχουν πρόσβαση σε αυτά που μοιράζονται άλλοι. Ακόμα, παρέχεται η δυνατότητα δημοσίευσης τμημάτων κώδικα χρησιμοποιώντας τα συγκεκριμένα dataset, καθώς και η επικοινωνία μεταξύ των χρηστών πάνω στην ενότητα συζήτησης (22). Στα πλαίσια της διπλωματικής, έγινε χρήση δύο δημόσιων συνόλων δεδομένων.

1. Running and Heart Rate Data

Το συγκεκριμένο dataset αναφέρεται σε δραστηριότητα τρεξίματος ενός άντρα. Εκτός από την τιμή του καρδιακού παλμού και τη χρονική στιγμή της μέτρησης, κάθε γραμμή του CSV αρχείου περιέχει πληροφορίες σχετικές με τη διάρκεια και το περιβάλλον της άθλησης. Από το σύνολο των δεδομένων χρησιμοποιήθηκαν οι καρδιακοί παλμοί, η χρονική στιγμή της καταγραφής και ο τύπος της δραστηριότητας. (23)

timestamp	heart_rate	time_elapsed	total_running_time	altitude	distance	speed	cadence	start_time	start_position_lat	start_position_long	end_position_lat
2017-09-21T2	143	486	436	182.4	1.61317	13.2336	80	##########	41.96782172	-87.65471038	41.96577301
2017-09-21T2	154	883	833	175.8	3.21954	14.3424	81	##########	41.96573907	-87.64633317	41.96506592
2017-09-21T2	156	1317	1267	199	4.83101	12.9996	81	##########	41.96502862	-87.64276626	41.96816923
2017-09-22T1	143	460	448	178.2	1.61227	14.58	82	##########	41.96768191	-87.65460486	41.97702345
2017-09-22T1	155	891	879	187	3.22219	13.2012	82	##########	41.97705598	-87.64969868	41.98369428
2017-09-22T1	158	1335	1323	186.4	4.82929	12.8304	81	##########	41.98369428	-87.65132418	41.97166113
2017-09-23T1	139	424	424	172	1.4448	13.5036	83	##########	41.96800075	-87.65462196	41.96468244
2017-09-23T1	151	689	572	180.6	2.08591	14.544	84	##########	41.96466769	-87.64364828	41.96454305
2017-09-23T1	154	839	720	175.8	2.69909	14.544	81	##########	41.96453081	-87.64347495	41.96452998
2017-09-23T1	156	997	878	176	3.33959	14.544	82	##########	41.96456627	-87.64371148	41.96452864
2017-09-23T1	158	1160	1041	175.4	4.00169	13.9392	82	##########	41.96455445	-87.64373411	41.96452059
2017-09-23T1	160	1319	1200	178.8	4.63745	14.1084	82	##########	41.96456912	-87.64370587	41.96452646
2017-09-23T1	160	1478	1359	173.6	5.2621	14.04	82	##########	41.96455353	-87.64371735	41.96453123
2017-09-23T1	163	1628	1509	169.6	5.86814	14.1408	82	##########	41.96455764	-87.64372649	41.9645304
2017-09-23T1	163	1787	1668	172.2	6.49674	13.9752	82	##########	41.96455881	-87.64380385	41.96458781
2017-09-23T1	163	1948	1829	171.8	7.12494	13.806	82	##########	41.96458781	-87.64379212	41.96453442
2017-09-23T1	164	2101	1982	179.6	7.72906	15.3504	82	##########	41.96458966	-87.64377259	41.96457105
2017-09-23T1	95	2816	1985	170.4	7.73612	0	0	##########	41.96457105	-87.64380142	41.96458572
2017-09-23T1	154	2949	2118	174.6	8.24322	13.2336	83	##########	41.96458354	-87.64375968	41.96462142
2017-09-23T1	158	3092	2261	167.6	8.78477	13.3344	82	##########	41.96459142	-87.64380444	41.96461631

Εικόνα 65 s1_laps_summary.csv Dataset

2. MiBand-2-pulse

Το συγκεκριμένο dataset αφορά τους καρδιακούς παλμούς ενός ατόμου κατά τη διάρκεια μιας ολόκληρης ημέρας. Οι μετρήσεις αυτές δεν παρουσιάζουν ιδιαίτερα μεγάλες τιμές ούτε έντονες αυξομειώσεις. Θεωρήθηκε ότι το άτομα βρισκόταν σε κατάσταση ηρεμίας. Από τα διαθέσιμα δεδομένα χρησιμοποιήθηκαν οι τιμές των καρδιακών παλμών και οι χρονικές στιγμές καταγραφής τους. (24)

dateTime	rate	rateZone
11.08.2018 0	68	35%
11.08.2018 0	48	25%
11.08.2018 0	74	38%
11.08.2018 0	73	38%
11.08.2018 0	73	38%
11.08.2018 0	84	44%
11.08.2018 0	60	31%
11.08.2018 0	68	35%
11.08.2018 0	67	35%
11.08.2018 0	67	35%
11.08.2018 0	62	32%
11.08.2018 0	71	37%
11.08.2018 0	64	33%
11.08.2018 0	81	42%
11.08.2018 0	64	33%
11.08.2018 0	74	38%

Εικόνα 66 Export-2018-08-11-08-14-12.csv Dataset

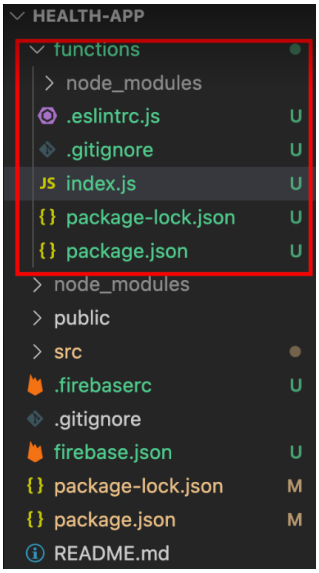
4.4.2.2 Apple Watch

Εκτός από τις δύο παραπάνω πηγές, δεδομένα συγκεντρώθηκαν και από το Apple Watch Series 7 από πραγματικούς χρήστες. Αναλυτικότερα, μετρήσεις καρδιακών παλμών σε καταστάσεις ηρεμίας και τρεξίματος. Επίσης, δεδομένα οξυγόνου και HRV μετρήσεις έγιναν μόνο σε κατάσταση ηρεμίας. Οι χρήστες αυτοί εγκατέστησαν την εφαρμογή με τη βοήθεια του Xcode, καθώς δεν έχει δημοσιευτεί στο App Store. Στη συνέχεια για ένα χρονικό διάστημα περίπου 2 μηνών, λάμβαναν μετρήσεις ανά τακτά χρονικά διαστήματα μέσα στην μέρα. Το ηλικιακό εύρος των ατόμων ήταν από 21 εώς 28 ετών και των δύο φύλλων.

4.4.3 Εγκατάσταση και προσαρμογή των Cloud Functions

Για την εγκατάσταση των Cloud Functions και την προσαρμογή τους στο τρέχον Firebase project πραγματοποιήθηκαν τα παρακάτω βήματα: (25)

Εγκατάσταση Firebase CLI και του SDK των Cloud Functions



Σε αυτό το βήμα δημιουργήθηκε ένα Cloud Functions project το οποίο συγχωνεύτηκε στο αρχικό Firebase project health-app. Στο παρακάτω screenshot, το κόκκινο πλαίσιο συμπεριλαμβάνει το φάκελο “functions” ο οποίος δημιουργήθηκε αυτομάτως από την εγκατάσταση. Ο φάκελος αυτός περιέχει τα κατάλληλα node_modules για το deployment των συναρτήσεων.

Εικόνα 67 Cloud functions node modules

Εγγραφή και testing συναρτήσεων

Κατόπιν της εγκατάσης ακολούθησε η εγγραφή των συναρτήσεων στο αρχείο “index.js” του φακέλου “functions”. Στο συγκεκριμένο project, η εγγραφή έγινε σε γλώσσα Javascript. Για τη δοκιμή του κώδικα δημιουργήθηκε localhost περιβάλλον σε port 8080. (26)

Deployment και διαχείριση

Εφόσον οι διαδικασίες της εγγραφής και του testing ολοκληρώθηκαν με επιτυχία, οι συναρτήσεις έγιναν deploy στο Cloud της Google μέσω της εντολής που φαίνεται στο παρακάτω screenshot.

```
michalispatsakis@dhcp-9-246-205-168 functions % firebase deploy --only functions
```

Εικόνα 68 Cloud functions deployment

Να σημειωθεί πως γίνεται deployment μόνο του φακέλου functions, που εμπεριέχει τα κατάλληλα node modules, καθώς και το αρχείο index.js με τις backend συναρτήσεις. Η Google Cloud παρέχει και την υπηρεσία Google Cloud Console μέσω της οποίας καθίσταται εφικτή η αναζήτη και εποπτεία των logs που παράγονται από την εκτέλεση των συναρτήσεων.

4.4.4 Συνάρτηση υπολογισμού ορίων

Αρχικά, γίνεται κλήση συνάρτησης από την ιστοσελίδα όπως παρουσιάζεται παρακάτω. Καθίσταται σαφές πως το URL της συνάρτησης αποτελείται από το γεωγραφικό τόπο του Data Center στην περίπτωση αυτού του έργου, us-central, το όνομά του, healthwatchapp, το cloudfunctions που δηλώνει την υπηρεσία του Firebase και τέλος στο όνομα της συνάρτησης helloWorld μαζί με όλες τις απαραίτητες παραμέτρους. Τα αποτελέσματα της συνάρτησης γίνονται με GET request. Στο URL προστίθεται το user id του εκάστοτε χρήστη ως παράμετρος.

```
fetch("https://us-central1-healthwatchapp-e636f.cloudfunctions.net/helloWorld?uid=" + this.state.id_user)
.then((res) => res.json())
.then((data) => {
```

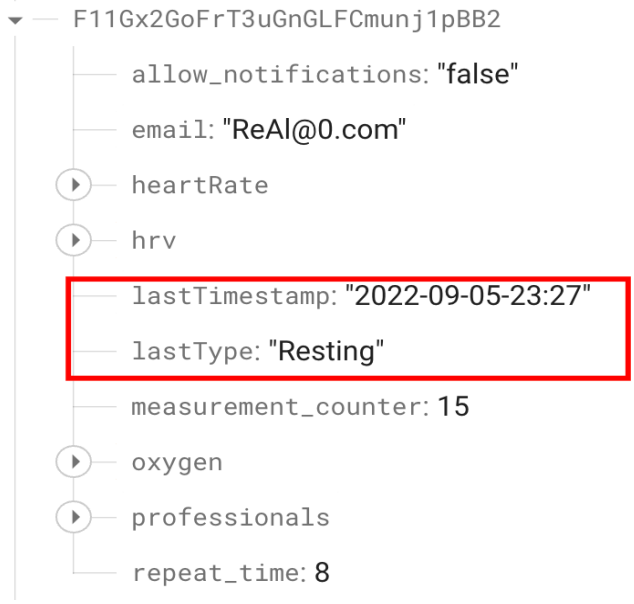
Εικόνα 69 Κλήση συνάρτησης υπολογισμού φυσιολογικών ορίων

Στην επόμενη εικόνα φαίνεται αναλυτικά ο κώδικας συνάρτησης.

```
exports.helloWorld = functions.https.onRequest((request, response) => {
  const uid = request.query.uid;
  const database = admin.database();
  const usersRef = database.ref("users/" + uid);
  usersRef.on("value", (snapshot) => {
    const type = snapshot.val().lastType;
    const stamp = snapshot.val().lastTimestamp;
    const usersRef2 = database.ref("users/"+uid+"/heartRate/"+type+"/"+stamp
    usersRef2.on("value", (snapshot) =>{
      const plotdata = Object.values(snapshot.val());
      const plotlabels = Object.keys(snapshot.val());
      const usersRef3 = database.ref("users/"+uid+"/heartRate/"+type);
      usersRef3.on("value", (snapshot) =>{
        const measurements = Object.values(snapshot.val());
        const dataarray = [];
        for (let i = 0; i < (measurements.length - 1); i ++) {
          const data = Object.values(measurements[i]);
          for (let j=0; j<data.length; j++) {
            dataarray.push(data[j]);
          }
        }
```

Εικόνα 70 Κώδικας συνάρτησης υπολογισμού ορίων

Από το body του request γίνεται ανάκτηση του user id, δημιουργείται ένα reference στη βάση δεδομένων και στη συνέχεια, επικοινωνεί με τη βάση για να αντλήσει τα κατάλληλα δεδομένα. Η πρώτη κλήση γίνεται για την ανάκτηση των μεταβλητών “lastType” και “lastTimestamp” που αποτελούν το είδος και την ημερομηνία της τελευταίας ολοκληρωμένης μέτρσης.



Εικόνα 71 Μεταβλητές τελευταίας μέτρησης

Μέσω των παραπάνω μεταβλητών γίνεται η δεύτερη κλήση στη βάση για την αποθήκευση των δεδομένων της τελευταίας μέτρησης. Η τρίτη και τελευταία κλήση πραγματοποιείται για τη συγκέντρωση των δεδομένων όλων των μετρήσεων του ίδιου είδους. Τα δεδομένα αυτά αποθηκεύονται στον πίνακα “measurements“.

```
 * Solves equations of the form a * x = b
 *@param {Array} t
* @return {Number} Returns the value of x for the equation.
*/
function getStandardDeviation(t) {
  const n = t.length;
  const m = t.reduce((a, b) => a + b) / n;
  return Math.sqrt(t.map((x)=>Math.pow(x-m, 2)).reduce((a, b)=>a+ b)/n);
}
const average = (array) => array.reduce((a, b) => a + b) / array.length;
console.log(average(dataarray));
console.log(getStandardDeviation(dataarray));
const ave = average(dataarray);
const std = getStandardDeviation(dataarray);
const lower = ave - std;
const upper = ave + std;
const dictionary = {
  ave, lower, upper, plotdata, plotlabels,
};
```

Εικόνα 72 Υπολογισμός ορίων

Για τον πίνακα αυτό, γίνεται υπολογισμός του μέσου όρου καθώς και της τυπικής απόκλισης. Έπειτα γίνεται ανάθεση του άνω και κάτω ορίου, ως το άθροισμα ή τη διαφορά του μέσου όρου και της τυπικής απόκλισης αντιστοίχως. Τα παραπάνω όρια μαζί με τα δεδομένα της τελευταίας μέτρησης αποθηκεύονται σε ένα object.

```
        response.set("Access-Control-Allow-Origin", "*");
        if (response.method === "OPTIONS") {
          // Send response to OPTIONS requests
          response.set("Access-Control-Allow-Methods", "GET");
          response.set("Access-Control-Allow-Headers", "Content-Type");
          response.set("Access-Control-Max-Age", "3600");
          response.status(204).send(" ");
        } else {
          response.send(dictionary);
        }
      });
```

Εικόνα 73 Προσθήκη CORS Headers

Πριν την αποστολή των αποτελεσμάτων, γίνεται συγχώνευση CORS Headers στο response προκειμένου να γίνεται επιτυχής ανάκτηση από τη μεριά της ιστοσελίδας.

4.4.5 Συνάρτηση αξιολόγησης μέτρησης

Η δεύτερη συνάρτηση ακολουθεί παρόμοια δομή και λογική, με την προσθήκη του υπολογισμού του μέσου όρου των τιμών της τελευταίας μέτρησης. Όπως παρουσιάζεται στη φωτογραφία του κώδικα, η μεταβλητή “valid” καθορίζεται από το ‘if statement’. Σε περίπτωση ανάθεσης και αποστολής της τιμής false, η συνάρτηση επιστρέφει το response, εφόσον έχουν προστεθεί τα απαιτούμενα CORS Headers.

```
exports.validatemeasurement = functions.https.onRequest((request, response) => {
  const uid = request.query.uid;
  console.log(uid);
  const database = admin.database();
  const usersRef = database.ref("users/" + uid);
  usersRef.on("value", (snapshot) => {
    const type = snapshot.val().lastType;
    const stamp = snapshot.val().lastTimestamp;
    const usersRef2 = database.ref("users/"+uid+"/heartRate/"+type+"/"+stamp);
    usersRef2.on("value", (snapshot) =>{
      const plotdata = Object.values(snapshot.val());
      const average = (array) => array.reduce((a, b) => a + b) / array.length;
      const avelastmeas = average(plotdata);
      const usersRef3 = database.ref("users/"+uid+"/heartRate/"+type);
      usersRef3.on("value", (snapshot) =>{
        const measurements = Object.values(snapshot.val());
        const dataarray = [];
        for (let i = 0; i < (measurements.length - 1); i ++) {
          const data = Object.values(measurements[i]);
          for (let j=0; j<data.length; j++) {
            dataarray.push(data[j]);
          }
        }
```

Εικόνα 74 Κώδικας συνάρτησης αξιολόγησης μέτρησης 1

```
        let valid = true;
        if (avelastmeas > upper || avelastmeas < lower) {
          valid = false;
        }
        response.set("Access-Control-Allow-Origin", "*");
        if (response.method === "OPTIONS") {
          // Send response to OPTIONS requests
          response.set("Access-Control-Allow-Methods", "GET");
          response.set("Access-Control-Allow-Headers", "Content-Type");
          response.set("Access-Control-Max-Age", "3600");
          response.status(204).send(" ");
        } else {
          response.send(valid);
        }
      });
    });
  }, (errorObject) => {
    console.log("The read failed: " + errorObject.name);
  });
});
```

Εικόνα 75 Κώδικας συνάρτησης αξιολόγησης μέτρησης 2

5. Συμπεράσματα και ιδέες για μελλοντικές επεκτάσεις

5.1 Εισαγωγή

Προσπάθεια των εμπλεκόμενων, ήταν η παρούσα διπλωματική εργασία να αποτελέσει ένα ολοκληρωμένο έργο που ξεκινάει από τη συλλογή των βιοτικών δεδομένων του χρήστη, την παροχή άμεσης πρόσβασης σε αυτά από ειδικούς στο χώρο της υγείας, και να ολοκληρώνεται με την επεξεργασία των τιμών από αυτοματοποιημένα μοντέλα, προσφέροντας ταυτόχρονα στους χρήστες ένα εύχρηστο και απλό UI. Παρόλα αυτά, υπάρχουν αρκετές επεκτάσεις του εγχειρήματος αυτού που ενδεχομένως να ξεπερνούν τα όρια της διπλωματικής, αλλά αξίζουν να σημειωθούν.

5.2 Δημιουργία εφαρμογής για κινητά iOS

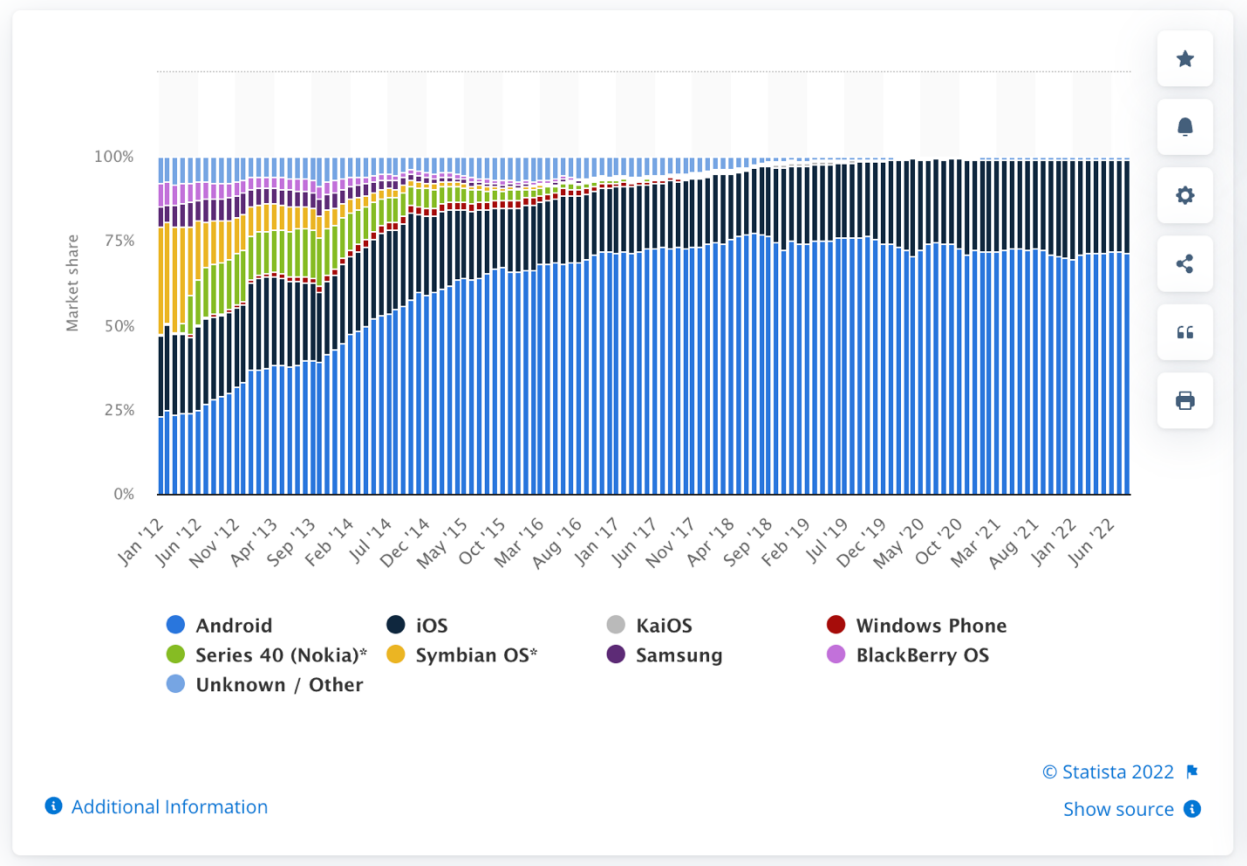
Αρχικά, η δημιουργία μιας αδελφικής εφαρμογής για κινητά iOS, θα μπορούσε να απαλλάξει την εφαρμογή του ρολογιού από την αναγκαιότητα να είναι συνδεδεμένη στο Wi-Fi για να είναι λειτουργική. Με αυτόν τον τρόπο, τα δεδομένα θα στέλνονται μέσω Bluetooth από το ρολόι στο κινητό και ύστερα είτε με τη βοήθεια 4G/5G δικτύου θα μπορούν να αποθηκεύονται στη βάση δεδομένων. Ταυτόχρονα, η εφαρμογή για κινητά θα είναι σε θέση να επιτελεί πλήρως τις λειτουργίες της ιστοσελίδας, όπως προβολή των διαγραμμάτων και των μετρήσεων, επιλογή και επικοινωνία με τον επαγγελματία υγείας καθώς και αλλαγή των ρυθμίσεων λήψης μέτρησης.

5.3 Εφαρμογή περισσότερων υπολογιστικών μοντέλων

Η συλλογή των μετρήσεων από ένα σημαντικό αριθμό χρηστών διαφορετικών ηλικιών και κατάστασης υγείας, προσφέρει τη δυνατότητα μελέτης σε βάθος της σύνδεσης προβλημάτων υγείας με τις βασικές μετρικές των καρδιακών παλμών, του οξυγόνου και του HRV. Ο όγκος των δεδομένων και η συνεχής λήψη μετρήσεων προσφέρουν τη δυνατότητα σε επαγγελματίες υγείας κατανόησης των αίτιων που προκαλούν προβλήματα υγείας σε ασθενείς. Αναλυτικότερα, η εξέταση των τιμών αυτών και η εφαρμογή ακόμα περισσότερων υπολογιστικών μοντέλων, ίσως να οδηγήσει σε δημιουργία μοτίβων που να δείχνουν πιθανή προσβολή ενός χρήστη από κάποια ασθένεια. Το γεγονός αυτό θα δώσει μεγάλη ώθηση στην καλύτερη αντιμετώπιση παθήσεων και στην έγκυρη κινητοποίηση του ασθενή, και ενδεχομένως να σώσει ζωές.

5.4 Επέκταση και σε άλλα λογισμικά

Ιδιαίτερα σημαντικό βήμα που θα διευρύνει το ενδιαφερόμενο κοινό, αποτελεί η ανάπτυξη παρόμοιας εφαρμογής σε λειτουργικό σύστημα πέραν του iOS. Το Android λογισμικό είναι κυρίαρχο σε μερίδιο αγοράς, όπως φαίνεται αναλυτικά στην επόμενη φωτογραφία που παρουσιάζει την αγορά την τελευταία δεκαετία. (27)



Εικόνα 76 Market share of mobile operating systems

Συνεπώς, η ανάπτυξη μιας εφαρμογής Android που να συλλέγει τις μετρήσεις των αισθητήρων από ένα έξυπνο ρολόι του αντίστοιχου περιβάλλοντος, θα βοηθήσει στην ραγδαία αύξηση των χρηστών και των αντίστοιχων δεδομένων προς μελέτη και έρευνα.

5.5 Προσθήκη νέων αισθητήρων

Η συνεχής πρόοδος της τεχνολογίας και η έκδοση νέων μοντέλων έξυπνων ρολογιών και συσκευών δεν αποκλείει την προσθήκη νέων αισθητήρων για την μελέτη περισσότερων ζωτικών ενδείξεων στο άμεσο μέλλον. Ήδη, η Apple Inc. έχει ανακοινώσει την επιθυμία της να εντάξει αισθητήρες που θα μετρούν την συγκεντρωμένη γλυκόζη στο αίμα στο επόμενο Apple Watch, βήμα ιδιαίτερα σημαντικό τόσο για τους ανθρώπους που πάσχουν από διαβήτη, αλλά και για όλους τους χρήστες γενικότερα. (28) Έτσι, είναι πιθανό σε ένα εύλογο χρονικό διάστημα μια παρόμοια εφαρμογή να έχει στην διάθεση της τόσες μετρικές που να μπορεί να προσδιορίσει και να παρακολουθήσει την υγεία ενός χρήστη στο έπακρο γνωρίζοντας κάθε χρονική στιγμή την κατάσταση στην οποία βρίσκεται.

Βιβλιογραφία

1. **Apple.** Apple Watch Series 7 Overview. [Online] https://www.apple.com/apple-watch-series-7/.
2. **Apple**. *Apple Watch Series 7 - Technical Specifications.* [Online] 2022. https://support.apple.com/kb/SP860?viewlocale=en_US&locale=el_GR.
3. **Timmer John.** Swift Overview. [Online] 2014. https://arstechnica.com/gadgets/2014/06/a-fast-look-at-swift-apples-new-programming-language/.
4. **Google.** Cloud Functions. [Online] 2022. https://cloud.google.com/functions.
5. **Google.** Call functions via HTTP requests. [Online] 2022. https://firebase.google.com/docs/functions/http-events.
6. **Google**. Cloud Firestore triggers. [Online] 2022. https://firebase.google.com/docs/functions/firestore-events.
7. **Apple.** HealthKit. [Online] https://developer.apple.com/documentation/healthkit.
8. **Flanagan, .** *JavaScript: The Definitive Guide.* 2006.
9. **Krill Paul.** React: Making faster, smoother UIs for data-driven Web apps. [Online] 2014. https://www.infoworld.com/article/2608181/react--making-faster--smoother-uis-for-data-driven-web-apps.html.
10. **Ασπιώτης Νικόλαος, .** *Φυσιολογία ανθρώπου και λοιπών θηλαστικών.* ΘΕΣΣΑΛΟΝΙΚΗ : s.n., 1976.
11. **Charles Patrick Davis, MD, PhD.** What Is a Healthy and Normal Heart Rate for My Age? . [Online] https://www.onhealth.com/content/1/normal_healthy_heart_rate_charts_health_heart.
12. **Hye-Geum Kim, Eun-Jin Cheon, Dai-Seg Bai, Young Hwan Lee, Bon-Hoon Koo.** *Stress and Heart Rate Variability: A Meta-Analysis and Review of the Literature.* 2018.
13. **Φούσας Στέφανος Γρ.** Αυτές είναι οι έξι συχνότερες παθήσεις της καρδιάς. [Online] 28 9 2019. https://www.ygeiamou.gr/blogs/aftes-ine-i-exi-sichnoteres-pathisis-tis-kardias/.
14. **Hafen BB, Sharma S.** Oxygen Saturation. [Online] StatPearls, 2021. https://www.ncbi.nlm.nih.gov/books/NBK525974/.
15. **Laoyan Sarah.** What is Agile methodology? (A beginner's guide). [Online] 14 7 2022. https://asana.com/id/resources/agile-methodology.
16. **Google.** Pricing Plans. [Online] 2022. https://firebase.google.com/pricing.
17. **Google**. *Add Firebase to your Apple project.* [Online] 2022. https://firebase.google.com/docs/ios/setup.
18. **Google**. *Get Started with Firebase Authentication on Apple Platforms.* [Online] 2022. https://firebase.google.com/docs/auth/ios/start.
19. **Google**. Read and Write Data on Apple platforms. [Online] 2022. https://firebase.google.com/docs/database/ios/read-and-write.
20. **Apple.** userNotificationCenter. [Online] 2022. https://developer.apple.com/documentation/usernotifications/unusernotificationcenterdelegate/1649518-usernotificationcenter.
21. **EmailJS.com.** Creating Contact Form. [Online] https://www.emailjs.com/docs/tutorial/creating-contact-form/.
22. **Kaggle.** What is Kaggle? [Online] 5 2022. https://www.datacamp.com/blog/what-is-kaggle.
23. **Candocia Max.** Running and Heart Rate Data. [Online] 2021. https://www.kaggle.com/datasets/mcandocia/running-heart-rate-recovery.
24. **Oscyp.** MiBand-2-pulse. [Online] 2019. https://www.kaggle.com/datasets/oscypx/miband2pulse.
25. **Google.** Cloud Functions for Firebase. [Online] 2022. https://firebase.google.com/docs/functions#implementation_path.
26. **Google**. Run functions locally. [Online] 2022. https://firebase.google.com/docs/functions/local-emulator.

27. **Laricchia Federica.** Mobile operating systems' market share worldwide from January 2012 to August 2022 . [Online] 30 9 2022. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.
28. **Mark Gurman, Debby Wu.** Apple Plans Faster Watch, Future Temperature and Glucose Sensors. *Bloomberg.* [Online] 14 6 2021. https://www.bloomberg.com/news/articles/2021-06-14/apple-plans-faster-watch-future-temperature-and-glucose-sensors#xj4y7vzkg.