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

Αυτόνομη Εφαρμογή IoT για την ευέλικτη Διαχείριση Αισθητήρων Υγείας και την Μέτρηση της Μεταβλητότητας Καρδιακού Ρυθμού (HRV) σε Ρολόι WatchOS

Διπλωματική Εργασία

Κωνσταντίνος Αθανασιάδης

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής ΕΜΠ

Αθήνα, Φεβρουάριος 2023

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

Αυτόνομη Εφαρμογή IoT για την ευέλικτη Διαχείριση Αισθητήρων Υγείας και την Μέτρηση της Μεταβλητότητας Καρδιακού Ρυθμού (HRV) σε Ρολόι WatchOS

Διπλωματική Εργασία

Κωνσταντίνος Αθανασιάδης

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20η Μαρτίου 2023

……..	……..	……..
Π. Τσανάκας	Δ. Σούντρης	Γ. Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2023

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

Περίληψη

Η συγκεκριμένη διπλωματική εργασία αφορά αφενός, την ανάπτυξη μιας εφαρμογής για έξυπνο ρολόι Apple Watch με λειτουργικό σύστημα WatchOS 9 στη γλώσσα Swift της Apple. Αφετέρου, την ανάπτυξη μιας κεντρικής εφαρμογής, η οποία φιλοξενείται στην πλατφόρμα Firebase της Google, όπου περιλαμβάνει μία βάση δεδομένων που προσφέρεται από την πλατφόρμα και δύο ιστότοπους σε γλώσσα JavaScript που επικοινωνούν άμεσα με την εφαρμογή του ρολογιού.

Πιο αναλυτικά, η εφαρμογή που αναπτύχθηκε στο ρολόι λαμβάνει ζωντανές μετρήσεις από διάφορους αισθητήρες του Apple Watch, έτοιμες μετρήσεις που υπάρχουν στη μνήμη του, οι οποίες έχουν γίνει με αλγορίθμους της Apple και επιπλέον έχει τη δυνατότητα μέσω του ηλεκτροκαρδιογραφήματος ενός καναλιού (ECG), που κάνει ο χρήστης στο ρολόι, να υπολογίζει τοπικά και εκείνη ακριβώς τη στιγμή την μεταβλητότητα καρδιακού ρυθμού (HRV). Αυτή η εφαρμογή, μέσω του Internet, στέλνει τις μετρήσεις της στην κεντρική εφαρμογή και στη συγκεκριμένη περίπτωση στη βάση δεδομένων του Firebase, όπου εκεί αποθηκεύονται οι τιμές του κάθε χρήστη.

Οι δύο ιστότοποι οι οποίοι υπάγονται στην κεντρική εφαρμογή ασχολούνται, ο μεν ένας με τον χρήστη και του δείχνει ανά πάσα στιγμή τα δεδομένα του που έχει στείλει στο γιατρό. Ο δε άλλος ιστότοπος αφορά τους γιατρούς, και τους δείχνει τους ασθενείς τους και τα δεδομένα του καθενός και μέσω αυτού έχει τη δυνατότητα να στείλει κάποιο ερωτηματολόγιο αναφορικά με την υγεία του στον κάθε χρήστη, ή ειδοποίηση για να στείλει νέες μετρήσεις, τα οποία και γίνονται μέσω των δυνατοτήτων Remote Config και Cloud Functions του Firebase, αντίστοιχα.

Λέξεις Κλειδιά

Apple Watch, WatchOS, ηλεκτροκαρδιογράφημα, ECG, μεταβλητότητα καρδιακού ρυθμού, HRV, μετρήσεις, Swift, JavaScript, Firebase, κεντρική εφαρμογή

Abstract

This specific thesis concerns, on the one hand, the development of an application for an Apple Watch smart watch with the WatchOS 9 operating system in Apple's Swift language, and On the other hand, the development of a central application, which is hosted on Google's Firebase platform, which includes a database offered by the platform, and two websites in JavaScript language, which communicate directly with the clock application.

In more detail, the application developed on the watch receives live readings from various sensors of the Apple Watch, ready-made readings that exist in its memory, which have been made with Apple algorithms, and in addition, it has the capacity through the single-channel electrocardiogram (ECG), which the user does on the watch, to calculate, locally and at that exact moment, heart rate variability (HRV). This application, via the Internet, sends its measurements to the central application and in this particular case to the Firebase database, where the ratings of each user are being stored.

Regarding the two websites, which are part of the central application, we have the first one dealing with the user to show him at any time the data he has sent to the doctor, whereas the other website is about doctors and shows them their patients and their data , and through that it has the ability to send any given questionnaire regarding their health to each and every user, or a notification to send new measurements, which are applied through Firebase's Remote Config and Cloud Functions features, respectively.

Key words

Apple Watch, WatchOS, electrocardiogram, ECG, heart rate variability, HRV, readings, Swift, JavaScript, Firebase, central application

Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί την επισφράγιση των προπτυχιακών σπουδών μου στο Ε.Μ.Π., στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών. Καθ' όλη τη διάρκεια της φοίτησης μου υπήρξαν συγκεκριμένα άτομα τα οποία με στήριξαν σε όλα τα επίπεδα, τόσο ψυχολογικά, όσο και πνευματικά, αλλά και υλικά – οικονομικά, τα οποία ευχαριστώ μέσα από την ψυχή μου, παρότι ένα ευχαριστώ είναι πολύ μικρό μπροστά σε ότι εκείνα μου προσέφεραν και τα οποία ανήκουν σε μια άλλη ψηλή – ανώτερη σφαίρα. Αρχικά, ευχαριστώ βαθύτατα τον κύριο Π. Τσανάκα, ο οποίος με μεγάλη προθυμία ανέλαβε να με κατευθύνει και να με καθοδηγήσει προκειμένου να καταφέρω να ολοκληρώσω και κατόπιν να συγγράψω τη διπλωματική μου εργασία. Μάλιστα, είχαμε αλλεπάλληλες και σε τακτά χρονικά διαστήματα συναντήσεις, προκειμένου να παρακολουθεί και να παρατηρεί την πορεία της εργασίας μου και να με συμβουλεύει παντοιοτρόπως. Επίσης, θα ήθελα να πω ένα μεγάλο ευχαριστώ στους καθηγητές μου κύριο Δ. Σούντρη και κύριο Δ. Κουτσούρη, για την συμμετοχή τους στην τριμελή επιτροπή. Επιπλέον, αξίζει να ευχαριστήσω όσες φίλες και φίλους απέκτησα τόσο κατά τα γυμνασιακά όσο και στα φοιτητικά μου χρόνια και οι οποίοι με στήριξαν αλλά και μαζί ολοκληρωθήκαμε σαν προσωπικότητες. Θα θεωρούσα παράλειψη μου να μην αποδώσω εγκάρδιες ευχαριστίες στον καθηγητή μου της μουσικής κ. Πάλακα, ο οποίος με την ευγένεια και την ανεκτικότητα του στις υποχρεώσεις μου με στήριξε και αποτελούσε για μένα μια πολύ ευχάριστη διέξοδο. Τέλος, το πιο μεγάλο ευχαριστώ αξίζουν οι γονείς μου, οι οποίοι όλα αυτά τα χρόνια με περιέβαλαν με αγάπη, έδειξαν κατανόηση και ήταν πάντα δίπλα μου στα εύκολα και στα δύσκολα κατά τη διάρκεια όλης αυτής της πορείας, προσφέροντας μου δύναμη, ασφάλεια, αυτοπεποίθηση και ισορροπία σε όλα τα επίπεδα. Σας ευχαριστώ όλους θερμά.

Στους γονείς μου

Περιεχόμενα

Ευρετήριο Πινάκων

Ευρετήριο Εικόνων

Λίστα Αρκτικόλεξων

HRV:	Heart Rate Variability
GB:	Giga-Byte
GHz:	Giga-Hertz
RAM:	Random Access Memory
GPS:	Global Positioning System
GLONASS:	Global Navigation Satellite System
QZSS:	Quasi-Zenith Satellite
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
UI:	User Interface
JSON:	JavaScript Object Notation
SDNN:	Standard Deviation of NN intervals
HTML:	Hypertext Markup Language
CSS:	Cascading Style Sheets
NoSQL:	Non Structured Query Language
API:	Application Programming Interface

Θεωρητικό μέρος – Apple Watch και εργαλεία ανάπτυξης

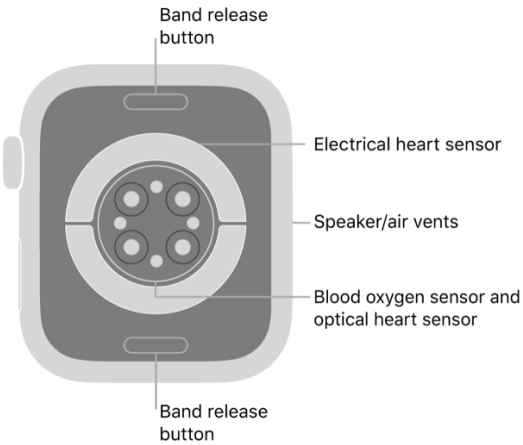
Α.1 Το έξυπνο ρολόι Apple Watch Series 7

Α.1.1 Γενικά

Το Apple Watch είναι ένα έξυπνο ρολόι που το δημιούργησε η Apple και έχουν κυκλοφορήσει πολλές γενιές του. Όταν αυτή η εργασία ξεκίνησε, το νεότερο μοντέλο με τις περισσότερες λειτουργίες και αισθητήρες ήταν το Series 7. Για αυτόν τον λόγο η συγκεκριμένη εργασία είναι δομημένη σε αυτό το μοντέλο, χωρίς να σημαίνει ότι η εφαρμογή δεν είναι συμβατή και με τα προηγούμενα μοντέλα. Επίσης, η εργασία βασίζεται στο WatchOS 9, το οποίο αυτή τη στιγμή είναι το νεότερο λογισμικό της Apple για τα ρολόγια της. Επέλεξα το συγκεκριμένο ηλεκτρονικό ρολόι, διότι μετά από εκτενή έρευνα και συγκρίσεις που έκανα με άλλα έξυπνα ρολόγια, είναι από αυτά που έχουν τη μεγαλύτερη ακρίβεια στις έτοιμες μετρήσεις τους, έχει τους περισσότερους αισθητήρες υγείας (μέτρηση καρδιακών παλμών, οξυγόνου στο αίμα, υψόμετρου κλπ.), έχει τη δυνατότητα λήψης καρδιογραφήματος (ECG) και η εταιρεία προσπαθεί με διαρκείς αναβαθμίσεις να προσθέτει και να βελτιώνει λειτουργίες.

Α.1.2 Αισθητήρες

Το Apple Watch Series 7 περιέχει μια πληθώρα αισθητήρων: GPS, πυξίδα, επιταχυνσιόμετρο, γυροσκόπιο, μαγνητόμετρο, αισθητήρα φωτός, αισθητήρα υψόμετρου και μικρόφωνο. Επιπλέον, έχει αισθητήρες υγείας των οποίων φαίνεται η διάταξη στην παρακάτω εικόνα και είναι οι εξής: αισθητήρας καρδιακών παλμών, αισθητήρας οξυγόνου στο αίμα και αισθητήρας για καρδιογράφημα ενός καναλιού, ο οποίος δε φαίνεται αλλά βρίσκεται πάνω στην κορόνα του ρολογιού. (1)



Εικόνα 1 Οι αισθητήρες του Apple Watch Series 7

Α.1.3 Τεχνικά χαρακτηριστικά

Το συγκεκριμένο ρολόι βασίζεται στον διπύρηνο επεξεργαστή 64-bit Apple S7, ο οποίος είναι χρονισμένος στα 1.8 GHz, έχει 1 GB μνήμης, περιλαμβάνει εξαρτημένη κάρτα γραφικών και 32 GB αποθηκευτικού χώρου. Επιπλέον έχει ενσωματωμένο Bluetooth 5.0, 802.11 b/g/n WiFi σε συχνότητες 2.4 και 5 GHz, LTE για το Cellular μοντέλο και εντοπισμό θέσης μέσω GPS, GLONASS, Galileo, QZSS και BeiDou (για την αγορά της Κίνας). Ακόμη, έχει οθόνη 396 x 484 ή 352 x 430, ανάλογα με το αν το μοντέλο είναι το 45mm ή το 41mm αντίστοιχα, με

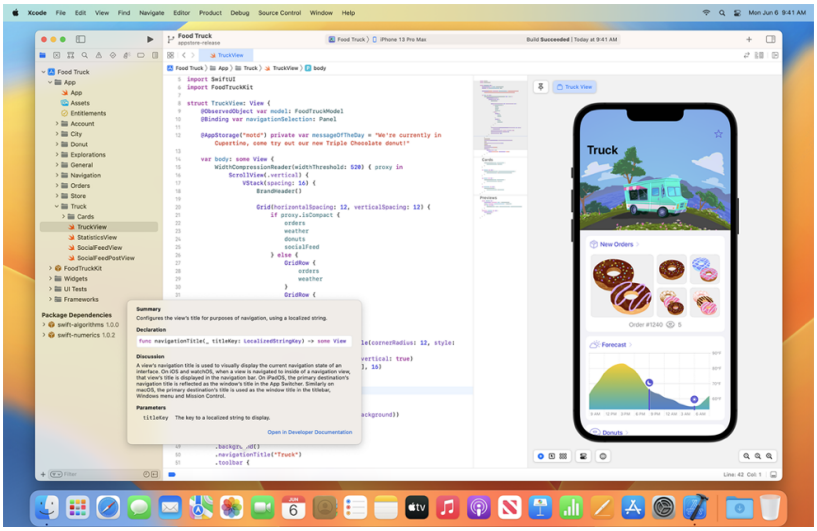
δυνατότητα always-on. Τέλος, έχει πιστοποίηση IP6X για αντοχή σε σκόνη και πιστοποίηση WR50 για αντοχή στο νερό μέχρι 50 μέτρα.

A.2 Swift

Η Swift είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού, ανεπτυγμένη από την Apple και την κοινότητα ανοιχτού κώδικα. Είναι απαραίτητη για ανάπτυξη εφαρμογών στο iOS, το WatchOS, το iPadOS, το macOS και το tvOS και αντίστοιχα στις συσκευές iPhone, Apple Watch, iPad, υπολογιστές Mac και Apple TV, δηλαδή στο οικοσύστημα της Apple. Κυκλοφόρησε για πρώτη φορά το 2014 ως αντικατάσταση της Objective-C, η οποία ήταν ίδια στο μεγαλύτερό της ποσοστό από το 1980, με αποτέλεσμα να μην έχει σημαντικές λειτουργίες που έχουν άλλες μοντέρνες γλώσσες προγραμματισμού. Είναι χτισμένη με τον open-source μεταγλωττιστή LLVM και συμπεριλαμβάνεται στο Xcode από την έκδοση 6. Τέλος, χρησιμοποιεί τη βιβλιοθήκη Objective-C, επιτρέποντας κώδικα γραμμένο σε C, C++, Objective-C και Swift να εκτελείται παράλληλα σε ένα πρόγραμμα. (2)

A.3 Xcode

Το Xcode είναι ένα ολοκληρωμένο περιβάλλον για την ανάπτυξη εφαρμογών στο οικοσύστημα της Apple, το οποίο διατίθεται δωρεάν. Μάλιστα, είναι το μόνο απαραίτητο για την ανάπτυξη εφαρμογών. (3) Διατίθεται μόνο για υπολογιστές Mac και περιλαμβάνει τα απαραίτητα εργαλεία για αυτόν τον σκοπό, όπως simulators, compiler, organizer (για άντληση των logs από τις εφαρμογές) κλπ.. Η τελευταία σταθερή έκδοσή του είναι η 14.1 και κυκλοφόρησε την 1η Νοεμβρίου 2022, στην οποία έχει γίνει build η συγκεκριμένη εργασία, και είναι συμβατή μόνο με macOS Ventura. Υπάρχουν και δοκιμαστικές εκδόσεις του, οι οποίες διατίθενται μόνο στους εγγεγραμμένους Apple developers. Επιπλέον, το Xcode περιλαμβάνει εργαλεία γραμμής εντολών (Command Line Tools), τα οποία δίνουν την δυνατότητα ανάπτυξης εφαρμογών μέσω της εφαρμογής Terminal στο macOS. Στην παρούσα εργασία, το Xcode χρησιμοποιείται για το build και debug της εφαρμογής στο Apple Watch. Δεν γινόταν να εκτελεστεί στον simulator στο macOS, διότι λόγω περιορισμών του στερούσε τις ειδοποιήσεις και κάποια http requests. Στην παρακάτω εικόνα φαίνεται το περιβάλλον του:



Εικόνα 2 Το περιβάλλον του Xcode

A.4 HealthKit

A.4.1 Γενικά

Το HealthKit είναι ένα framework της Apple σχεδιασμένο για την πρόσβαση, διαχείριση και κοινοποίηση δεδομένων υγείας και φυσικής κατάστασης ενός χρήστη, χωρίς να μετριάζεται το απόρρητό του και ο έλεγχος που έχει πάνω σε αυτά, αφού του δίνει πλήρη έλεγχο ποιων δεδομένων θα μοιράζεται και ποιες εφαρμογές θα τα χρησιμοποιούν. Στο Apple Watch κυκλοφόρησε για πρώτη φορά το 2015 στο WatchOS 2. Το μεγαλύτερο ποσοστό εφαρμογών υγείας στο οικοσύστημα της Apple πλέον, χρησιμοποιεί σχεδόν αποκλειστικά το HealthKit για την καταχώρηση και ανάγνωση τέτοιων δεδομένων. (4)

A.4.2 Δυνατότητες του HealthKit

Το HealthKit είναι ένα container το οποίο περιέχει όλα τα δεδομένα υγείας του χρήστη και έχει πάρα πολλές δυνατότητες. Στη συνέχεια, θα αναφέρουμε τις πλέον σημαντικές.
Αρχικά, επιτυγχάνει άριστη κοινή χρήση δεδομένων μεταξύ όλων των συσκευών που έχει στην ιδιοκτησία του ο κάθε χρήστης. Δηλαδή, μπορεί και συλλέγει αυτόματα δεδομένα από το ρολόι και τα στέλνει και στο κινητό. Επίσης, σε όλα αυτά τα δεδομένα έχουν πρόσβαση όσες εφαρμογές το ζητήσουν και μάλιστα μπορούν να γράψουν και καινούργια, τα οποία θα μπορούν αυτομάτως να τα δουν όλες οι υπόλοιπες εφαρμογές και ο χρήστης συγκεντρωτικά στην εφαρμογή Health του iPhone. Επιπλέον, το HealthKit εξασφαλίζει το απόρρητο των δεδομένων του χρήστη. Αυτό επιτυγχάνεται με πολλούς τρόπους, όπως να επιβάλει σε όλες τις εφαρμογές να ζητούν την άδεια από τον χρήστη για την ανάγνωση ή και την εγγραφή των δεδομένων υγείας του. Ο χρήστης αποφασίζει ποια δεδομένα θέλει να μοιραστεί με την εφαρμογή και ποια όχι. Ακόμη, η εφαρμογή είναι υποχρεωμένη να εξηγεί στον χρήστη για ποιον ακριβώς λόγο ζητάει αυτά τα δεδομένα και να τον ενημερώνει για το αν η επεξεργασία είναι τοπική ή στο cloud. Στην περίπτωση που ο χρήστης αρνηθεί να μοιραστεί τα δεδομένα υγείας του, η εφαρμογή απλά βλέπει ότι δεν υπάρχει πρόσφατη μέτρηση, οπότε με αυτόν τον τρόπο αποφεύγονται διαρροές πληροφοριών για το συγκεκριμένο άτομο. Επιπροσθέτως, τα δεδομένα του HealthKit αποθηκεύονται μόνο τοπικά στη συσκευή, όχι σε κάποιο cloud και κρυπτογραφημένα, ώστε να εξασφαλίζεται ότι δεν θα έχει πρόσβαση κάποιος που δεν έχει πάρει την άδεια από τον χρήστη. Τέλος, παρότι γενικά η συλλογή δεδομένων υγείας είναι κάτι δύσκολο, το HealthKit είναι εύκολο στη χρήση για τον προγραμματιστή και ξεκάθαρο, οπότε η ύπαρξή του είναι καθοριστική σε τέτοιου είδους εφαρμογές.

A.4.3 Περιορισμοί

Το HealthKit, παρά τις πάρα πολλές δυνατότητες που προσφέρει, έχει και κάποιους περιορισμούς. Αρχικά, δεν είναι διαθέσιμο στο iPad, το οποίο έχει μεγαλύτερη επεξεργαστική ισχύ και μέσω αυτού θα μπορούσε κανείς να τρέξει κάποιον κώδικα Machine Learning και να βγάλει πολλά παραπάνω συμπεράσματα σχετικά με την υγεία του χρήστη και χωρίς να καταναλώνει υπερβολική ενέργεια από την μπαταρία, όπως με το Apple Watch. Επίσης, το HealthKit δεν δίνει τη δυνατότητα ανάγνωσης ή και επεξεργασίας δεδομένων στο background διότι τα δεδομένα είναι κρυπτογραφημένα και πρέπει η συσκευή να είναι ξεκλείδωτη για να υπάρχει πρόσβαση σε αυτά. Ωστόσο, αυτό δυσχεραίνει την αυτόματη

εξαγωγή δεδομένων, οπότε πρέπει η εφαρμογή να στέλνει ειδοποίηση όταν χρειάζεται δεδομένα και ο χρήστης να μπαίνει στην εφαρμογή και να δίνει την άδειά του χειροκίνητα. Τέλος, η Apple απαγορεύει την αποθήκευση ιατρικών δεδομένων στο iCloud, οπότε σε περίπτωση που η συσκευή του χρήστη είτε χαλάσει είτε χαθεί, χάνονται τα δεδομένα και πρέπει να ξεκινήσουμε τις μετρήσεις από την αρχή.

A.5 Alamofire

A.5.1 Γενικά

Η γλώσσα Swift, δεν έχει έτοιμη βιβλιοθήκη για HTTP/HTTPS requests. Το Alamofire, λοιπόν, είναι μια open-source βιβλιοθήκη βασισμένη στη γλώσσα Swift, η οποία παρέχει την δυνατότητα των HTTP/HTTPS requests, όπως GET, POST κλπ., είναι περιεκτική, πολύ εύκολη στη χρήση της και ο συνολικός κώδικας του προγράμματος γίνεται πιο καθαρός από ότι να χρησιμοποιούσαμε το URLSession της Apple. (5)

A.5.2 Δυνατότητες

Η βιβλιοθήκη Alamofire έχει πολλές δυνατότητες. Αρχικά, υποστηρίζει όλα τα HTTP/HTTPS requests όπως GET, POST, PUT, PATCH, DELETE, COPY κλπ. Επιπλέον, είναι συμβατό με τα iOS 13, macOS 10.15, tvOS 13 και watchOS 6 και όλες τις μεταγενέστερες εκδόσεις τους. Ακόμη, υποστηρίζει σύνδεση με REST APIs και μπορεί να στέλνει παραμέτρους με json, να ανεβάζει όλων των τύπων τα αρχεία και να τα κατεβάζει αντίστοιχα, ενώ ενημερώνει τον χρήστη για το ποσοστό της προόδου. Επίσης, υποστηρίζει αυθεντικοποίηση χρήστη μέσω του URLCredential της Apple, το οποίο είναι σημαντικό για την προστασία των δεδομένων και του απορρήτου των χρηστών. Τέλος, έχει εύκολη διαχείριση σφαλμάτων και η χρήση της είναι εύκολη, κατανοητή και βοηθάει τον προγραμματιστή να είναι παραγωγικός, αφού δεν είναι υποχρεωμένος να γράψει μεγάλο κώδικα για να καταφέρει κάτι απλό, σε αντίθεση με το URLSession της Apple.

A.5.3 Μειονεκτήματα

Τα μειονεκτήματα της βιβλιοθήκης Alamofire είναι τα ίδια με οποιασδήποτε βιβλιοθήκης η οποία είναι ανεπτυγμένη από κάποιον τρίτο. Συγκεκριμένα, υπάρχει ανάγκη για μελλοντική υποστήριξη από τους προγραμματιστές που την έχουν αναπτύξει. Η Swift είναι μια γλώσσα, η οποία ανανεώνεται συνεχώς και στην οποία προστίθενται, αφαιρούνται και μετατρέπονται πολλές από τις λειτουργίες της, οπότε, πολλές φορές θα είναι απαραίτητη η αναμονή επίλυσης της συμβατότητας της βιβλιοθήκης με τη γλώσσα. Παρόλα αυτά, όσα μας προσφέρει η βιβλιοθήκη Alamofire, είναι πολύ πιο σημαντικά από τα προαναφερθέντα μειονεκτήματα και η υποστήριξή της είναι μεγάλη, αφού είναι η πιο γνωστή βιβλιοθήκη στο είδος της.

A.6 UserDefaults

Η UserDefaults είναι μια βάση δεδομένων και μάλιστα η κάθε εφαρμογή έχει τη δική της και την δημιουργεί το λειτουργικό σύστημα. Σε αυτήν μπορεί ο προγραμματιστής να αποθηκεύει

βασικά στοιχεία στα οποία χρειάζεται να έχει πρόσβαση από όλα τα σημεία της εφαρμογής του. Επίσης, έχει τη δυνατότητα να αποθηκεύει σε αυτή, οποιοδήποτε τύπο μεταβλητής θέλει, π.χ. Bool, Float, Double, Int, String, URL, Date, ακόμα και Data μεταβλητές. Όταν αποθηκεύεται κάτι στην UserDefaults, φορτώνεται αυτόματα κατά τη διάρκεια εκκίνησης της εφαρμογής, ώστε να μπορεί να διαβάσει αμέσως η εφαρμογή ό,τι της είναι απαραίτητο. (6) Όμως, για αυτόν τον λόγο σε αυτήν πρέπει να αποθηκεύονται βασικά πράγματα, γιατί διαφορετικά η εφαρμογή θα αργεί πολύ στην εκκίνησή της. Τέλος, ένα παράδειγμα της χρησιμότητας της είναι η δυνατότητα χάρη της οποίας δεν χρειάζεται ο χρήστης κάθε φορά που ανοίγει την εφαρμογή να πληκτρολογεί τα στοιχεία αυθεντικοποίησής του για να εισέλθει στον λογαριασμό του, αλλά η εφαρμογή τα έχει αποθηκεύσει στην UserDefaults και έτσι τα στέλνει κατευθείαν στον authentication server.

A.7 Firebase - Authentication, Realtime Database και Storage

A.7.1 Γενικά

Η υπηρεσία Firebase παρέχει ένα σύνολο από hosting υπηρεσίες, οι οποίες βοηθούν τον προγραμματιστή στην ανάπτυξη εφαρμογών, χωρίς να χρειάζεται να χάσει χρόνο, ώστε να τις χτίσει τοπικά και χωρίς να πάσχουν από προβλήματα downtime. Είναι διαθέσιμες για iOS, Android, Web (με χρήση JavaScript), Flutter κλπ.. Μια υπηρεσία του Firebase είναι το Authentication. Αυτό βοηθάει στην αυθεντικοποίηση των χρηστών, ώστε να μπορούν τα δεδομένα να διαβαστούν και να επεξεργαστούν με ασφάλεια. Επίσης, διευκολύνει τον κάθε χρήστη να έχει πρόσβαση μόνο στα προσωπικά του δεδομένα. Μια άλλη, επίσης, υπηρεσία του Firebase είναι η Realtime Database. Αυτή είναι μια βάση δεδομένων στο cloud, στην οποία έχουμε την δυνατότητα να αποθηκεύουμε τα δεδομένα των χρηστών σε πραγματικό χρόνο και με ασφάλεια. Μάλιστα, σε περίπτωση απώλειας δικτύου, συγκρίνει τα δεδομένα που έχει λάβει από όλες τις συσκευές εκείνη την χρονική περίοδο και διατηρεί τα πιο καινούργια. Τα δεδομένα στην Realtime Database αποθηκεύονται με τη μορφή του JSON (JavaScript Object Notation). Τέλος, μια ακόμη υπηρεσία του Firebase, είναι το Cloud Storage. Αυτό είναι ένα Google Cloud Storage Bucket, στο οποίο μπορούμε να αποθηκεύουμε και να διαβάζουμε με ασφάλεια οποιουδήποτε είδους αρχεία. Και οι τρεις υπηρεσίες είναι διαθέσιμες και από φορητές συσκευές, από εφαρμογές και υπολογιστή. (7,8,9)

A.7.2 Λειτουργίες

Αρχικά, η Realtime Database και το Storage της Firebase προσφέρουν τα Firebase Security Rules, τα οποία είναι μια γλώσσα κανόνων. Αυτοί οι κανόνες χρησιμοποιούνται, ώστε να υπάρχει έλεγχος του από ποιον και πότε μπορούν να διαβαστούν ή να αποθηκευτούν τα δεδομένα. Αν αυτό συνδυαστεί με την αυθεντικοποίηση του Firebase, δηλαδή την Firebase Authentication, τότε υπάρχει η δυνατότητα να οριστεί ποιοι έχουν πρόσβαση, σε ποια δεδομένα και με ποιον τρόπο. Τέλος, η Realtime Database είναι μία NoSQL βάση δεδομένων, το οποίο σημαίνει ότι είναι σχεδιασμένη για λειτουργίες που μπορούν να γίνουν γρήγορα, οπότε δημιουργείται μία ευχάριστη εμπειρία για τους χρήστες, χωρίς καθυστερήσεις και δίνει τη δυνατότητα εξυπηρέτησης πολλών ατόμων ταυτόχρονα σε σχέση με μία σχεσιακή βάση δεδομένων.

Α.7.3 Πλεονεκτήματα

Τα οφέλη, τα οποία προσφέρουν οι τρεις προαναφερθείσες υπηρεσίες του Firebase ,είναι πολλαπλά. Πρώτον, το Firebase Authentication διευκολύνει την δημιουργία ασφαλών εφαρμογών, αφού υποστηρίζει λογαριασμούς μέσω email και κωδικού πρόσβασης, έλεγχο ταυτότητας τηλεφώνου ακόμα και σύνδεση μέσω Google, Twitter, Facebook και GitHub. Με αυτόν τον τρόπο, ο προγραμματιστής απαλλάσσεται από την δύσκολη εργασία δημιουργίας ενός ασφαλούς authenticator και μπορεί να επικεντρωθεί στην ανάπτυξη των κύριων λειτουργιών της εφαρμογής του. Δεύτερον, η Firebase Realtime Database προσφέρει με απλό τρόπο ανάγνωση και εγγραφή, μέσω του REST API της, αλλά και μέσω των βιβλιοθηκών που προσφέρει για τα υποστηριζόμενα περιβάλλοντά της. Η βάση τα αποθηκεύει με τη μορφή του JSON tree, το οποίο βοηθάει τη βάση να έχει τη μορφή που θέλει ο προγραμματιστής, με αποτέλεσμα να δημιουργεί ευκολότερα αποδοτικούς κώδικες για την ανάγνωση και εγγραφή αυτών. Επίσης, επειδή είναι NoSQL, όλες οι λειτουργίες της γίνονται ακαριαία. Τρίτον, η Firebase Storage έχει το πλεονέκτημα, ότι μπορούμε να αποθηκεύουμε οποιοδήποτε είδος αρχείου θέλουμε και με ασφάλεια, μέσω των κανόνων που ενσωματώνει και του authenticator. Ακόμη, είναι εύκολη η πρόσβαση σε αυτήν, μέσω των βιβλιοθηκών που προσφέρει για τα υποστηριζόμενα περιβάλλοντά της. Τέλος, και οι τρεις αυτές υπηρεσίες έχουν το πλεονέκτημα, ότι ανανεώνονται σε πραγματικό χρόνο και με μεγάλες ταχύτητες, οπότε ο συγχρονισμός μεταξύ των διάφορων εφαρμογών που μπορεί να φιλοξενούνται στην Firebase, γίνεται ακαριαία και ταυτόχρονα.

Α.7.4 Μειονεκτήματα

Η Firebase Realtime Database δεν είναι πάντα βολική, γιατί δεν είναι εύκολο το φιλτράρισμα των δεδομένων όπως με μία κανονική βάση δεδομένων και γενικά δεν έχει δυνατότητες για queries. Όλη η βάση είναι ένα αρχείο JSON στην ουσία, οπότε το format της δεν έχει καμία σχέση με της κλασικής SQL, το οποίο δυσχεραίνει την μοντελοποίηση των δεδομένων. Μία λύση σε αυτό το πρόβλημα, θα ήταν η χρήση της Firestore Database, αλλά αυτή δεν υποστηρίζεται officially στο WatchOS, παρά μόνο με τη χρήση του REST API της. Δυστυχώς, το ίδιο ισχύει και για τη Realtime Database, η υποστήριξή της κατευθείαν από τη βιβλιοθήκη για το WatchOS, με την τελευταία αναβάθμιση του WatchOS στην έκδοση 9, δεν έχει ανανεωθεί μετά από αλλαγές της Apple στο λογισμικό. Αυτό συμβαίνει, επειδή η Google θεωρεί την βιβλιοθήκη της Firebase για το WatchOS δευτερεύουσας σημασίας και την έχει αναθέσει στο community, οπότε η αναβάθμισή της είναι αργή. Αυτό έχει ως αποτέλεσμα και για την Realtime Database να είναι απαραίτητη η χρήση του REST API της προσωρινά, με τη διαφορά, όμως, ότι είναι πολύ πιο εύκολο στη χρήση του από αυτό της Firestore Database.

Α.8 Firebase Cloud Messaging – Push Notifications

Α.8.1 Γενικά

Η υπηρεσία Firebase Cloud Messaging παρέχει τη δυνατότητα αποστολής ειδοποιήσεων στις πλατφόρμες iOS, WatchOS, Android και Web. Συγκεκριμένα, βοηθάει στην αποστολή ειδοποιήσεων χωρίς ο προγραμματιστής να έχει το φορτίο του να φτιάξει έναν Notification server και να πρέπει να τον συμμορφώσει με τις απαιτήσεις του κάθε λογισμικού/γλώσσας. (10)

Α.8.2 Λειτουργίες

Αρχικά, η υπηρεσία Firebase Cloud Messaging μπορεί να στείλει είτε in-app ειδοποιήσεις, δηλαδή ειδοποιήσεις, ενώ χρησιμοποιούμε την εφαρμογή, είτε απλές ειδοποιήσεις, οι οποίες έρχονται με τη μορφή banner στον χρήστη, είτε έχει το κινητό του κλειδωμένο, είτε ξεκλείδωτο και το χρησιμοποιεί. Ακόμη, μας δίνει τη δυνατότητα να στείλουμε ειδοποιήσεις με απλό κείμενο ή και με εικόνες και να επιλέξουμε, αν θέλουμε αυτή η ειδοποίηση να σταλεί σε όλα τα άτομα που χρησιμοποιούν την εφαρμογή μας ή σε μία ομάδα από αυτούς που έχουμε δημιουργήσει. Επίσης, υπάρχει η λειτουργία της προγραμματισμένης ειδοποίησης, αν δεν θέλουμε η ειδοποίηση να πάει αυτή τη στιγμή. Μάλιστα, μπορούμε να προγραμματίσουμε και επαναλαμβανόμενες ειδοποιήσεις, ώστε να μην πρέπει κάθε φορά να την δημιουργούμε από την αρχή. Επιπροσθέτως, προσφέρει την δυνατότητα να συγκεντρώνουμε στατιστικά, που αφορούν τον αριθμό των χρηστών, οι οποίοι είδαν την ειδοποίηση κλπ., ώστε να μπορούμε να τις βελτιώσουμε με τον κατάλληλο τρόπο. Τέλος, μας δίνει τη δυνατότητα να στείλουμε είτε αθόρυβες ειδοποιήσεις, είτε με ήχο και μας αφήνει να επιλέξουμε για πόσο χρονικό διάστημα του επιτρέπουμε να προσπαθεί να στείλει την ειδοποίηση, αν αυτή δεν παραλαμβάνεται από τον χρήστη για οποιονδήποτε λόγο (δηλαδή, αν έχει απενεργοποιημένη τη συσκευή του, χωρίς δίκτυο κλπ.).

Α.8.3 Πλεονεκτήματα – Μειονεκτήματα

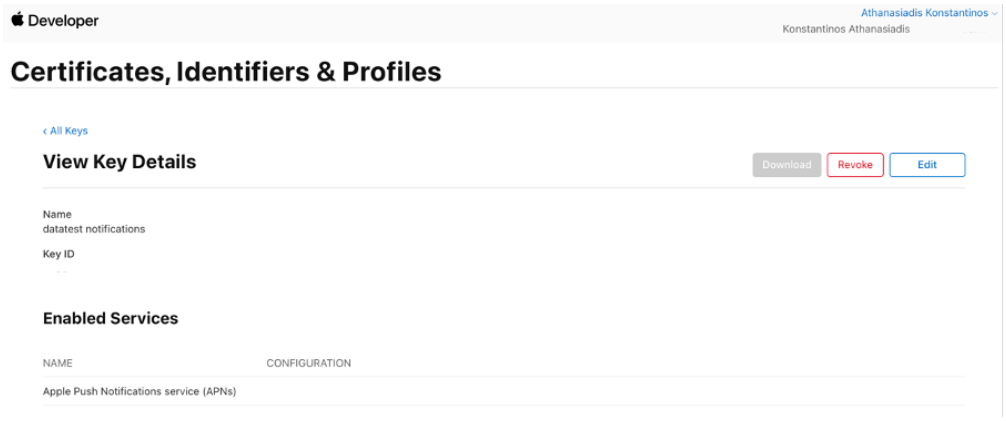
Η υπηρεσία Firebase Cloud Messaging έχει το πλεονέκτημα, ότι περιέχει όλες τις προαναφερθείσες δυνατότητες και ταυτόχρονα δεν χρειάζεται να τις αναπτύξει ο προγραμματιστής, οπότε κερδίζει πολύτιμο χρόνο από την ανάπτυξη της εφαρμογής του. Επίσης, δουλεύει αξιόπιστα χωρίς πολλά outages και downtimes, αφού είναι της Google. Ωστόσο, η υπηρεσία έχει το εξής μειονέκτημα. Συγκεκριμένα, δεν είναι σχεδιασμένη ώστε να στέλνει μεμονωμένες ειδοποιήσεις σε χρήστες, το οποίο είναι απαραίτητο για εφαρμογές που θέλουν εξατομίκευση, ανάλογα με τον καθένα. Στην δική μας περίπτωση αυτό ήταν απαραίτητο. Οπότε, για να επιτευχθεί αυτό, χρησιμοποιήθηκε η υπηρεσία του Firebase, η Cloud Functions για την οποία θα πούμε λεπτομέρειες παρακάτω. Μέσω αυτής, με μία συνάρτηση βρίσκουμε το μοναδικό notification token του κάθε χρήστη και του στέλνουμε μέσω του Cloud Messaging αυτόματα ειδοποιήσεις.

Α.9 Ειδοποιήσεις στο iOS και το WatchOS

Α.9.1 Αρχές ειδοποιήσεων – Τρόπος λειτουργίας

Για να σταλούν ειδοποιήσεις σε μία συσκευή με iOS ή WatchOS, πρέπει, αρχικά, να εισάγουμε στο project μας στο Xcode την δυνατότητα η εφαρμογή να υποστηρίζει ειδοποιήσεις. Το δεύτερο πράγμα που πρέπει να κάνουμε είναι, να δημιουργήσουμε μια backend υπηρεσία (όπως το Firebase Cloud Messaging), η οποία θα στέλνει την ειδοποίηση στο σύστημα διαχείρισης ειδοποιήσεων της Apple το APNS, δηλαδή Apple Push Notification Service. Στη συνέχεια, αφού τη δημιουργήσουμε, εισάγουμε τα απαραίτητα στοιχεία και κατεβάζουμε τα απαραίτητα πιστοποιητικά από την πλατφόρμα του Apple Developer (όπως φαίνεται στην παρακάτω εικόνα) και τα εισάγουμε στο backend, τότε η εφαρμογή θα εκδώσει ένα μοναδικό για κάθε χρήστη κωδικό, το λεγόμενο Push Device Token. Εφόσον το

γνωρίζουμε αυτό, μπορούμε να στέλνουμε ειδοποιήσεις μέσω του backend μας στην συσκευή αυτή.



Εικόνα 3 Η πλατφόρμα από την οποία βγάζουμε το απαραίτητο πιστοποιητικό για τις ειδοποιήσεις

Α.9.2 Περιορισμοί

Ωστόσο, ο μεγάλος περιορισμός που θέτει η Apple είναι το γεγονός ότι για να χρησιμοποιήσει κάποιος το APNS πρέπει να είναι γραμμένος στο επί πληρωμή πρόγραμμα Apple Developer, που διαθέτει η ίδια και δεν γίνεται με την δωρεάν εγγραφή. (11) Αυτό σημαίνει, ότι για να αναπτύξει κάποιος εφαρμογή που έχει push notifications, θα πρέπει να γραφτεί σε αυτό το πρόγραμμα το οποίο κοστίζει 99$ το χρόνο, όπως φαίνεται παρακάτω:

	Sign In with Your Apple ID	Apple Developer Program
Xcode developer tools	●	●
Xcode beta releases	●	●
On-device testing	●	●
Apple Developer Forums	●	●
Bug reporting with Feedback Assistant	●	●
OS beta releases		●
Full access to a comprehensive set of development tools		●
Advanced app capabilities and services		●
Code-level support		●
App distribution on the App Store		●
App management, testing, and analytics with App Store Connect		●
Safari Extensions distribution		●
Software distribution outside the Mac App Store		●
Custom app distribution with Apple Business Manager and Apple School Manager		●
Proprietary app distribution to your employees with Apple Business Manager		●
Ad hoc distribution for testing and internal use		●
Access to members-only developer events or additional event content		●
Cost	Free	99 USD**

Εικόνα 4 Σύγκριση του απλού με το επί πληρωμή πρόγραμμα Apple Developer

A.10 Firebase Cloud Functions

A.10.1 Γενικά

Η υπηρεσία Google Cloud Functions είναι ένα serverless περιβάλλον εκτέλεσης για την ανάπτυξη συναρτήσεων, οι οποίες θα απαντάνε σε εναύσματα από άλλες υπηρεσίες του Firebase ή από κάποιο αίτημα HTTPS, ώστε να εξυπηρετούνται λειτουργίες που σχετίζονται με συμβάντα που παράγονται από την υποδομή και τις υπηρεσίες στο Cloud. Ο κώδικας των συναρτήσεων που είναι γραμμένος, είτε σε JavaScript, είτε σε TypeScript, αποθηκεύεται σε ένα χώρο της Google και εκτελείται σε ένα ελεγχόμενο περιβάλλον, ώστε να μην υπάρχει ανάγκη υποδομής ή διαχείρισης servers. Το Firebase προσφέρει δύο εκδόσεις του Cloud Functions: την αρχική και πρώτη έκδοση v1 και την v2, η οποία είναι η νέα έκδοση και βασίζεται στο Cloud Run και το Eventarc, ώστε να έχει κάποιες παραπάνω δυνατότητες. Για την υλοποίηση των αναγκών της εργασίας, χρησιμοποιήθηκε η πρώτη έκδοση, η οποία είναι και διαθέσιμη κατευθείαν από την κονσόλα του Firebase. (12)

A.10.2 Κατηγορίες συναρτήσεων

Οι Cloud Functions χωρίζονται σε δύο κατηγορίες συναρτήσεων. Η πρώτη εκ των οποίων είναι οι HTTPS συναρτήσεις. Αυτές για να εκτελεστούν πρέπει να κληθούν με HTTPS request. Η μορφή τους φαίνεται παρακάτω. Το req είναι ουσιαστικά το request και το res είναι το αποτέλεσμα που επιστρέφει η συνάρτηση στο HTTPS request.

```
const functions = require('firebase-functions');

exports.webhookNew = functions.https.onRequest((req, res) => {
    res.send("Hello");
});
```

Εικόνα 5 Παράδειγμα HTTPS συνάρτησης

Η δεύτερη κατηγορία είναι οι background συναρτήσεις. Αυτές λειτουργούν στο υπόβαθρο και ενεργοποιούνται με αυτόματο τρόπο από αλλαγές που μπορεί να γίνουν σε κάποιες άλλες υπηρεσίες του Firebase. Οι συγκεκριμένες συναρτήσεις δεν μπορούν να κληθούν με κάποιο HTTPS request. Η μορφή αυτών των συναρτήσεων φαίνεται παρακάτω:

```
// Define a class that implements the BackgroundFunction<T> interface
public class MyBackgroundFunction implements BackgroundFunction<EventDataType> {
  // Implement the accept() method to handle events
  @Override
  public void accept(EventDataType eventData, Context context) {
    // Your code here
    // The eventData argument represents the event data payload
  }
}
```

Εικόνα 6 Παράδειγμα Background συνάρτησης

A.11 Firebase Remote Config

Η υπηρεσία Firebase Remote Config παρέχει τη δυνατότητα να αλλάζουμε πολλές παραμέτρους σε μία εφαρμογή από απόσταση. Συγκεκριμένα, μπορούμε να διαχειριστούμε μεταβλητές, την συμπεριφορά ή και την εμφάνιση της εφαρμογής μας σε πραγματικό χρόνο χωρίς καθυστερήσεις και χωρίς να χρειάζεται να βγάλουμε νέα έκδοση της εφαρμογής. Επίσης, μας δίνει την δυνατότητα να εξατομικεύουμε κάποιο μέρος της εφαρμογής, ανάλογα με ποιον χρήστη τη χρησιμοποιεί. Στην δική μας περίπτωση, χρησιμοποιήθηκε για να εισάγουμε μεταβλητές, οι οποίες θα περιέχουν ένα JSON με ένα ερωτηματολόγιο. (13)

A.12 Firebase Hosting

A.12.1 Γενικά – Δυνατότητες

Η υπηρεσία Firebase Hosting παρέχει τη δυνατότητα ανάπτυξης και φιλοξενίας μίας ή και πολλαπλών Web εφαρμογών. Επίσης, αναλαμβάνει μόνη της να την δημοσιεύσει στο διαδίκτυο, με ασφαλή τρόπο με τη βοήθεια του zero-configuration SSL, χωρίς να χρειάζεται να ασχοληθεί ο προγραμματιστής με αυτό και να χάσει χρόνο από την ανάπτυξη της κύριας εφαρμογής του. Ακόμη, η Firebase Hosting υποστηρίζει από τις κλασσικές στατικές γλώσσες HTML, CSS, JavaScript, μέχρι και express.js για διάφορες υπηρεσίες. Μάλιστα, δίνει τη δυνατότητα διάφορων έτοιμων προτύπων (π.χ. έτοιμη Login/Sign Up σελίδα), έτσι ώστε ο προγραμματιστής να είναι πιο παραγωγικός. Τέλος, για λόγους ανάπτυξης, έχει ενσωματώσει emulator, ο οποίος περιέχει όλες τις απαραίτητες υπηρεσίες του Firebase τοπικά. Έτσι, ο προγραμματιστής δεν χρειάζεται να ανεβάζει συνεχώς την εφαρμογή του στο διαδίκτυο για να διαπιστώσει αν λειτουργεί και δεν δημιουργούνται προβλήματα στην λειτουργικότητα της εφαρμογής, ενώ χρησιμοποιείται από χρήστες. (14)

A.12.2 Πλεονεκτήματα

Η υπηρεσία Firebase Hosting έχει αρκετά πλεονεκτήματα. Αρχικά, είναι πολύ ασφαλής με το περιεχόμενο που μεταφέρει, λόγω του zero-configuration SSL. Ακόμη, μας επιτρέπει να έχουμε πολλαπλές σελίδες, το οποίο είναι πολύ χρήσιμο σε περιπτώσεις όπως, όταν θέλουμε να έχουμε και ένα admin interface ή θέλουμε να έχουμε διάφορες γλώσσες στη σελίδα μας κλπ.. Επίσης, έχει το πλεονέκτημα ότι δεν έχει όριο στον αριθμό των χρηστών, οι οποίοι μπορούν να την επισκέπτονται ανά πάσα στιγμή. Μάλιστα, παρέχει στατιστικά σχετικά με τον αριθμό των χρηστών οι οποίοι την έχουν επισκεφτεί. Επίσης πληροφορεί για το πόσος είναι ο χώρος που καταλαμβάνει η ιστοσελίδα και επιπλέον μας πληροφορεί για τα logs από τα requests που έχουν γίνει σε αυτή. Το τελευταίο βοηθάει στην ευκολότερη επίλυση οποιουδήποτε προβλήματος μας αναφέρουν οι χρήστες της εφαρμογής. Τέλος, ο χρόνος που χρειάζεται για την προετοιμασία του είναι πολύ λιγότερος σε σχέση με άλλες υπηρεσίες της ίδιας κατηγορίας.

A.13 HTML – CSS

Η HTML (HyperText Markup Language, Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, τα στοιχεία της οποίας είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Ο web browser διαβάζει τα αρχεία HTML και τα συνθέτει, ώστε να δημιουργηθεί μία σελίδα που μπορεί να διαβαστεί από τους χρήστες. Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν, επίσης, να ενσωματώνονται σενάρια εντολών σε γλώσσες, όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML και από στατικές τις κάνουν διαδραστικές. Τέλος, οι Web browsers συνήθως αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. (15)

Η CSS (Cascading Style Sheets – διαδοχικά φύλλα ύφους ή επάλληλα φύλλα ύφους) είναι μια γλώσσα υπολογιστή, που ανήκει στην κατηγορία των γλωσσών φύλλων ύφους που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται, δηλαδή, για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα, δηλαδή, να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη. (16)

A.14 JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού, η οποία αποτελεί μία από τις κύριες τεχνολογίες που χρησιμοποιούνται στον Παγκόσμιο Ιστό μαζί με την HTML και τη CSS και δίνει τη δυνατότητα της επικοινωνίας μεταξύ των client scripts και του χρήστη. Επίσης, συμβάλει στο να μπορούν να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου, που εμφανίζεται. Όμως, χρησιμοποιείται και σε πολλές άλλες εφαρμογές, εκτός του Παγκόσμιου Ιστού, όπως σε εξειδικευμένους φυλλομετρητές, σε έγγραφα PDF κλπ.. Τα νεότερα πλαίσια ανάπτυξης για JavaScript, πχ το Node.js ή το express.js, έχουν καταστήσει τη JavaScript πολύ δημοφιλή για την ανάπτυξη εφαρμογών Ιστού από την πλευρά των server. Είναι δυναμική, με ασθενείς τύπους, έχει συναρτήσεις ως αντικείμενα πρώτης τάξης και η σύνταξή της είναι εμπνευσμένη από τη γλώσσα C. Είναι γλώσσα που συνδυάζει αντικειμενοστραφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού. Παρότι έχει πολλά κοινά ονόματα και συμβάσεις με τη Java, δεν έχουν καμία ουσιαστική σχέση μεταξύ τους, αφού έχουν αρκετά διαφορετική σημασιολογία. Τέλος, η JavaScript αποτέλεσε σημαντικό κομμάτι της εργασίας, αφού βοήθησε στην διαμόρφωση της ιστοσελίδας που δείχνει τα δεδομένα σε χρήστες και

ταυτόχρονα σε γιατρούς, ενώ παράλληλα δίνει στους τελευταίους τη δυνατότητα αποστολής ειδοποιήσεων στους χρήστες.

Θεωρητικό Μέρος - Η εφαρμογή και η ανθρώπινη υγεία

Β.1 Εισαγωγή

Στη σημερινή εποχή, η τεχνολογία είναι άμεσα συνδεδεμένη με τη ζωή και την καθημερινότητα, όχι μόνο των επιστημόνων, άλλα όλων των ανθρώπων ανεξαρτήτως επαγγέλματος. Έτσι, οι διάφορες συσκευές, όπως προσωπικοί υπολογιστές, έξυπνα κινητά, έξυπνα ρολόγια, τάμπλετ κλπ., συλλέγουν διαρκώς δεδομένα, με στόχο την διευκόλυνση της καθημερινότητας των χρηστών τους. Για παράδειγμα, η χρήση των μέσων κοινωνικής δικτύωσης ή η αγορά προϊόντων από online καταστήματα μέσω των προαναφερθέντων συσκευών και με τη συμβολή των cookies, βοηθούν στη συλλογή δεδομένων, που σχετίζονται με τις κλίσεις και τα ενδιαφέροντα ή τις ανάγκες του χρήστη, και στη συνέχεια, επεξεργάζονται για την προβολή εξατομικευμένων διαφημιστικών μηνυμάτων, την προώθηση προϊόντων και την εξαγωγή στατιστικών, ανάλογα τον χρήστη. Αξίζει να σημειωθεί, ότι με την τεράστια βελτίωση των ταχυτήτων του διαδικτύου αλλά και την ραγδαία πρόοδο των υπηρεσιών του υπολογιστικού νέφους (cloud computing), οι φορητές συσκευές που έχουν τη δυνατότητα να συλλέγουν και να στέλνουν δεδομένα υγείας είναι πολύ σημαντικές, αφού βοηθούν την παρακολούθηση της ανθρώπινης υγείας σε ένα αρχικό στάδιο, τη μακροζωία και βελτιώνουν την καθημερινότητα των ανθρώπων. Παρακάτω θα αναφερθούν διάφοροι τύποι δεδομένων, που βοηθούν προς αυτήν την κατεύθυνση, τρόποι που μπορούμε να τους αξιοποιήσουμε και να εξάγουμε αποτελέσματα, αναφορικά με την υγεία του κάθε χρήστη.

Β.2 Βήματα – Ταχύτητα βηματισμού

Β.2.1 Γενικά

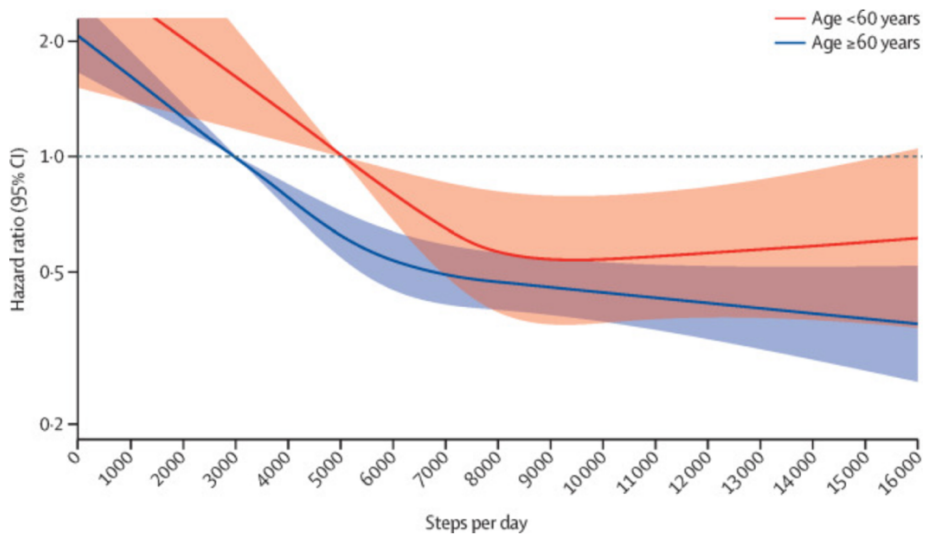
Τα βήματα και γενικά το περπάτημα με οποιονδήποτε ρυθμό, είτε δηλαδή αργά, είτε γρήγορα, παρέχει πάρα πολλά πλεονεκτήματα στον άνθρωπο και την υγεία του. Συγκεκριμένα, βοηθάει:

- στην μείωση του σωματικού λίπους και στην διατήρηση ενός φυσιολογικού βάρους
- στην αποφυγή ή έλεγχο των καρδιακών παθήσεων όπως έμφραγμα, υψηλή πίεση
- στην αποφυγή ή τον έλεγχο του καρκίνου
- στον έλεγχο του διαβήτη τύπου 2
- στις καρδιαγγειακές παθήσεις
- στην ενδυνάμωση των οστών και των μυών
- στα επίπεδα ενέργειας
- στην διάθεση, τη μνήμη και τον ύπνο
- στην μείωση του στρες
- στην ισχυροποίηση του ανοσοποιητικού συστήματος

Στις περισσότερες από τις προαναφερόμενες ασθένειες, με εξαίρεση τα καρδιαγγειακά νοσήματα, ο μεγαλύτερος ρυθμός βηματισμού ενισχύει τα παραπάνω οφέλη. (17)

B.2.2 Υποδειγματικές τιμές βημάτων και ταχύτητας ανά μέρα

Παραθέτω διάγραμμα στο οποίο απεικονίζονται υποδειγματικές τιμές βημάτων ανά μέρα και ανά ηλικία:



Εικόνα 7 Υποδειγματικές τιμές βημάτων ανά ημέρα και ηλικία

(18)
Όπως μπορούμε να δούμε, το ιδανικό για ηλικίες κάτω των 60 είναι τα 9000 βήματα τη μέρα κατά μέσο όρο. Ενώ, για ηλικίες άνω των 60 ετών, φαίνεται πως όσο αυξάνεται ο αριθμός των βημάτων, τόσο μειώνεται ο κίνδυνος για οποιαδήποτε ασθένεια, εξαιρουμένων των καρδιοπαθών.

Τέλος, παραθέτω έναν πίνακα στον οποίο απεικονίζονται οι μέσες τιμές ταχύτητας ανά φύλο και ηλικία (19):

Πίνακας 1 Μέσες τιμές ταχύτητας βηματισμού ανά φύλο και ηλικία

Age	Sex	Meters/second	Miles/hour
20 to 29	Male	1.36	3.04
	Female	1.34	3.0
30 to 39	Male	1.43	3.2
	Female	1.34	3.0
40 to 49	Male	1.43	3.2
	Female	1.39	3.11
50 to 59	Male	1.43	3.2
	Female	1.31	2.93
60 to 69	Male	1.34	3.0
	Female	1.24	2.77
70 to 79	Male	1.26	2.82
	Female	1.13	2.53
80 to 89	Male	0.97	2.17
	Female	0.94	2.10

Β.3 Ηλεκτροκαρδιογράφημα (ECG), καρδιακοί παλμοί και υπολογισμός της μεταβλητότητας καρδιακών παλμών (HRV)

Β.3.1 Γενικά

Το ηλεκτροκαρδιογράφημα (ECG) είναι μια γρήγορη εξέταση της δραστηριότητας της καρδιάς. Με αυτήν καταγράφεται η ηλεκτρική δραστηριότητα των μυών της παλλόμενης καρδιάς, αποδίδοντας μέσω του ηλεκτροκαρδιογράφου στην οθόνη χαρακτηριστικά γραφήματα που ονομάζονται επάρματα. Στα επάρματα, ο οριζόντιος άξονας αντιστοιχεί στο χρόνο και ο κάθετος στο ηλεκτρικό δυναμικό. Το καρδιογράφημα, στα πλαίσια αυτής της εργασίας, που γίνεται μέσω του ρολογιού, είναι ενός καναλιού (σε αντίθεση με τα πολυκάναλα που κάνουν οι γιατροί), οπότε είναι επόμενο να έχει μειωμένη ακρίβεια. Μέσω αυτού, έχουμε τη δυνατότητα υπολογισμού του HRV. Ο τρόπος θα αναλυθεί παρακάτω. (20)

Ο καρδιακός παλμός 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. Τον καρδιακό παλμό, στην συγκεκριμένη εργασία, τον εξάγουμε με ευκολία με μεγάλη ακρίβεια, είτε από τον αισθητήρα που έχει το ρολόι, είτε από το καρδιογράφημα που έχει κάνει ο χρήστης, σαν μέση τιμή. (21)

Η μεταβλητότητα καρδιακών παλμών (HRV) είναι το φαινόμενο της διακύμανσης του χρονικού διαστήματος μεταξύ των καρδιακών παλμών. Μετριέται από τη διακύμανση του διαστήματος από παλμό σε παλμό. (22)

Β.3.2 Υποδειγματικό ηλεκτροκαρδιογράφημα σε κατάσταση ηρεμίας

Εικόνα 8 Φυσιολογικό ηλεκτροκαρδιογράφημα σε κατάσταση ηρεμίας

Ωστόσο, παρακάτω απεικονίζεται και ένα μη φυσιολογικό ηλεκτροκαρδιογράφημα, το οποίο ανήκει πιθανότατα σε άτομο με κολπική μαρμαρυγή, η οποία είναι ουσιαστικά η αρρυθμία που αναγκάζει την καρδιά να μη λειτουργεί ρυθμικά, με αποτέλεσμα να συστέλλεται και να διαστέλλεται άρρυθμα. (23)

Εικόνα 9 Μη φυσιολογικό καρδιογράφημα σε κατάσταση ηρεμίας

Β.3.3 Υποδειγματικές τιμές καρδιακών παλμών σε κατάσταση ηρεμίας

Ο πίνακας που ακολουθεί με τις τιμές καρδιακών παλμών σε κατάσταση ηρεμίας, αφορά μόνο άντρες (24):

Πίνακας 2 Τιμές καρδιακών παλμών για άντρες

Age (in years)	18-25	26-35	36-45	46-55	56-65	65+
Athlete	40-52	44-50	47-53	49-54	51-56	52-55
Excellent	56-61	55-61	57-62	58-63	57-61	56-61
Good	62-65	62-65	63-66	64-67	62-67	62-65
Above Average	66-69	66-70	67-70	68-71	68-71	66-69
Average	70-73	71-74	71-75	72-76	72-75	70-73
Below Average	74-81	75-81	76-82	77-83	76-81	74-79
Poor	82+	82+	83+	84+	82+	80+

Ο πίνακας που ακολουθεί με τις τιμές καρδιακών παλμών σε κατάσταση ηρεμίας, αφορά μόνο γυναίκες:

Πίνακας 3 Τιμές καρδιακών παλμών για γυναίκες

Age (in years)	18-25	26-35	36-45	46-55	56-65	65+
Athlete	40-48	42-46	45-49	48-54	50-55	52-55
Excellent	61-65	60-64	60-64	61-65	60-64	60-64
Good	66-69	65-68	65-69	66-69	65-68	65-68
Above Average	70-73	69-72	70-73	70-73	69-73	69-72
Average	74-78	73-76	74-78	74-77	74-77	73-76
Below Average	79-84	77-82	79-84	78-83	78-83	77-84
Poor	85+	83+	85+	84+	84+	84+

B.3.4 Υποδειγματικές τιμές HRVSDNN

Στην παρακάτω εικόνα, φαίνονται οι φυσιολογικές τιμές του HRVSDNN, ανάλογα με την ηλικία του κάθε ανθρώπου (25):

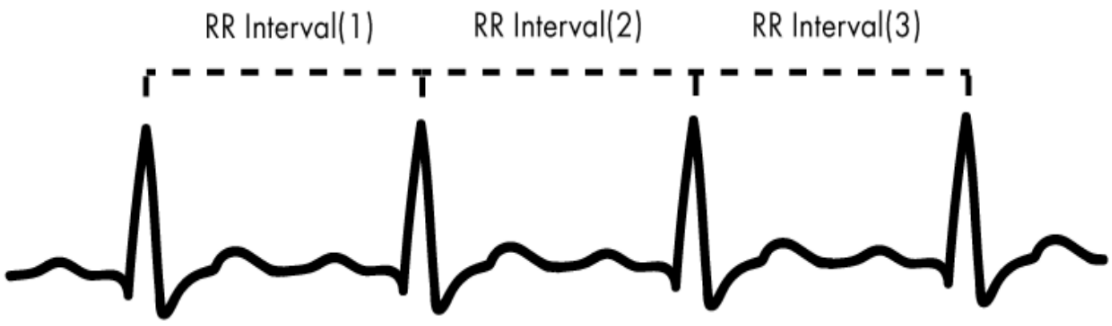
Πίνακας 4 Φυσιολογικές τιμές HRVSDNN σε σύγκριση με την ηλικία

Age (yr)	SDNN Index (ms)
10–19	81 ± 20
20–29	72 ± 22
30–39	64 ± 15
40–49	60 ± 13
50–59	52 ± 15
60–69	42 ± 13
70–79	43 ± 11
80–99	37 ± 12

Β.3.5 Η τιμή HRVSDNN και ο υπολογισμός της

Πέραν των καρδιακών παλμών, μία εξίσου χρήσιμη πληροφορία που μπορούμε να αναλύσουμε, είναι η μεταβλητότητα των καρδιακών παλμών. Γενικά, μία υψηλή τιμή του HRV δείχνει καλή σωματική υγεία, ενώ αν είναι χαμηλή η τιμή του, υπάρχει πιθανότητα να σχετίζεται με κάποια πολύ σοβαρή ασθένεια του ανθρώπου. Ωστόσο, η μεταβλητότητα των καρδιακών παλμών, όπως έχουν δείξει έρευνες ψυχολόγων, αν είναι ψηλή, μπορεί να δείχνει υψηλή αυτοσυγκράτηση, αυτοέλεγχο και αυτοκυριαρχία, κοινωνικές δεξιότητες, καλύτερη αντιμετώπιση του άγχους και ψυχική ευφορία. Από την άλλη, όταν είναι χαμηλή η τιμή του, δείχνει αυξημένο άγχος, ένταση και γενικότερη διέγερση και έλλειψη ηρεμίας, γαλήνης και ισορροπίας στον άνθρωπο. Με άλλα λόγια, οι μηχανισμοί της καρδιάς προσπαθούν να επιφέρουν ισορροπία σε όλα τα επίπεδα στον ανθρώπινο οργανισμό. Μάλιστα, το HRV είναι πολύ ευαίσθητο στις αλλαγές του αυτόνομου νευρικού συστήματος και αφού συνδέεται στενά με το στρες, όταν υπάρχει αυτό, υπάρχει δυσλειτουργία του συμπαθητικού νευρικού συστήματος. (26)

Το HRVSDNN είναι η τιμή του HRV υπολογισμένη από το ηλεκτροκαρδιογράφημα, παίρνοντας την τυπική απόκλιση των διαστημάτων μεταξύ των κορυφών RR, όπως φαίνεται στην εικόνα παρακάτω.



Εικόνα 10 Παράδειγμα διαστημάτων RR

Στη στατιστική, η τυπική απόκλιση είναι ένα μέτρο που χρησιμοποιείται για να υπολογιστεί το ποσό της μεταβολής ή της διασποράς ενός συνόλου τιμών δεδομένων. Ο τύπος της τυπικής απόκλισης σ είναι ο παρακάτω:

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

όπου N το μέγεθος του πληθυσμού, xi η κάθε τιμή από τον πληθυσμό και μ ο μέσος όρος του πληθυσμού. (27)

Β.3.6 Παθήσεις σχετιζόμενες με την καρδιά

Όταν οι παραπάνω τιμές που αναλύσαμε δεν είναι οι φυσιολογικές ή έχουν μεγάλη απόκλιση από αυτές, τότε προκαλούνται στην ανθρώπινη υγεία πολλά και διάφορα προβλήματα. Κάποια από αυτά είναι τα παρακάτω:

- Στεφανιαία νόσος: είναι μία πάθηση που οφείλεται σε ρήξη ευάλωτης αθηρωματικής πλάκας με αποτέλεσμα να αποφράσσεται το αγγείο εν όλω ή εν μέρει

- Βαλβιδοπάθειες: είναι όταν η βαλβίδα της αορτής έχει πρόβλημα σαν ιστός και σαν ανατομία και δημιουργεί στένωση
- Καρδιακή ανεπάρκεια: είναι η καρδιακή πάθηση κατά την οποία δεν μεταφέρεται επαρκής ποσότητα αίματος στα περιφερειακά όργανα και οφείλεται είτε σε αρρύθμιστη αρτηριακή πίεση, είτε σε πρωτοπαθή πάθηση του καρδιακού μυός, είτε σε πάθηση των βαλβίδων της καρδιάς ή σε στεφανιαία νόσο
- Αρρυθμίες: είναι διαταραχές του καρδιακού ρυθμού

Β.4 Οξυγόνο στο αίμα

Β.4.1 Γενικά

Το επίπεδο οξυγόνου στο αίμα 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 με την αναπνοή κατά την εισπνοή και με τη βοήθεια των πνευμόνων εισέρχεται στο αίμα και μεταφέρεται σε όλο το σώμα. Τα κύτταρα του ανθρώπου χρειάζονται οξυγόνο για να ζήσουν και να επιτελέσουν αποτελεσματικά τις λειτουργίες τους. Τα κύτταρα χρησιμοποιούν το οξυγόνο και δημιουργούν διοξείδιο του άνθρακα, το οποίο πρέπει να αποβληθεί από τον οργανισμό. Αυτό γίνεται με τη βοήθεια των πνευμόνων και την εκπνοή.
Το σώμα ρυθμίζει αυστηρά την ποσότητα κορεσμού οξυγόνου στο αίμα, επειδή τα χαμηλά επίπεδα οξυγόνου 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 ή το κυκλοφορικό σύστημα μπορεί να μην λειτουργούν όπως θα έπρεπε. (28)

Β.4.2 Ενδεικτικές τιμές οξυγόνου στο αίμα

Τα φυσιολογικά επίπεδα οξυγόνου κυμαίνονται συνήθως από 95% έως 100%. Όταν το οξυγόνο πέσει από 90%, θεωρείται χαμηλό (υποξαιμία). Παρακάτω φαίνεται ένας πίνακας με τα επίπεδα οξυγόνου και την κατάσταση που προκαλούν στον οργανισμό. (29)

Πίνακας 5 Οξυγόνο στο αίμα

ΕΠΙΠΕΔΑ ΟΞΥΓΟΝΟΥ ΣΤΟ ΑΙΜΑ ΜΕ ΠΑΛΜΙΚΟ ΟΞΥΜΕΤΡΟ	
Κατάσταση	**Εύρος SpO2**
Κανονική	95-100%
Επηρεάζεται ο εγκέφαλος	80-85%
Κυάνωση	Κάτω από 67%

Β.4.3 Συμπτώματα εξαιτίας της έλλειψης οξυγόνου στο αίμα

Η έλλειψη οξυγόνου στο αίμα προκαλεί αρκετές επιπλοκές στην υγεία. (30) Τα κυριότερα συμπτώματα εξαιτίας της ανεπάρκειας του οξυγόνου στο αίμα είναι:

- Αδυναμία
- Ζαλάδα
- Κόπωση
- Ταχυκαρδίες
- Λαχάνιασμα – Δύσπνοια
- Πονοκέφαλος και σύγχυση
- Ταχυαρρυθμίες
- Βήχας
- Κυάνωση

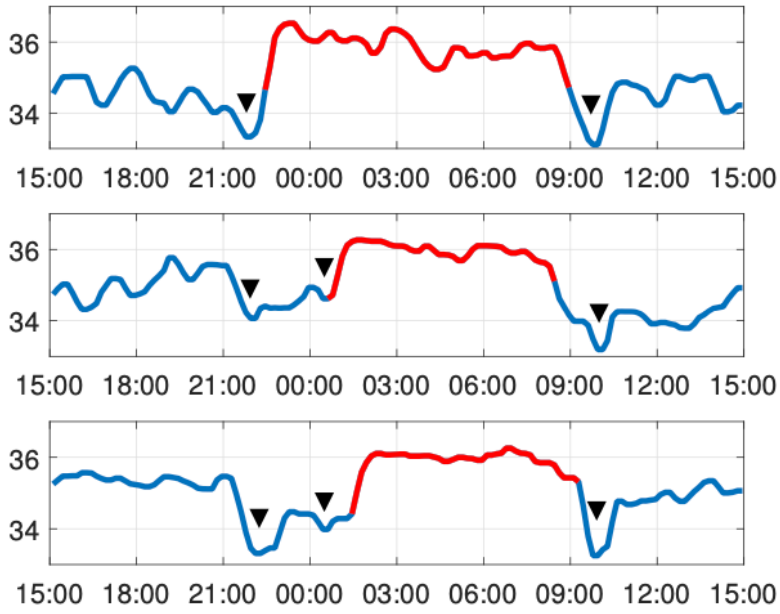
Β.4.4 Ασθένειες που προκαλούν την έλλειψη οξυγόνου στο αίμα

Η ανεπάρκεια οξυγόνου στο αίμα μπορεί να προκληθεί από διάφορες ασθένειες. Ειδικότερα:

- 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)

Β.5 Θερμοκρασία καρπού κατά τον ύπνο

Η θερμοκρασία του καρπού κατά τη διάρκεια του ύπνου μπορεί να μας δώσει πολλές πληροφορίες για τον άνθρωπο. Συγκεκριμένα, μπορούμε να καταλάβουμε πόσες ώρες κοιμόταν. Το είδος της διατροφής που ακολουθεί. Εάν ασκείται ή εάν ασχολείται γενικότερα με τον αθλητισμό. Επίσης, μπορούμε να συμπεράνουμε αν καταναλώνει αλκοόλ ή αν έχει κάποια ασθένεια. Στο επόμενο διάγραμμα φαίνεται ότι υπάρχει μείωση θερμοκρασίας του καρπού, όταν κάποιος καθυστερεί τον ύπνο του. (31, 32)



Εικόνα 11 Μείωση θερμοκρασίας εξαιτίας της αναβολή ύπνου

Β.6 Ηχογράφηση ανάσας

Β.6.1 Γενικά

Η ηχογράφηση ανάσας μπορεί να βοηθήσει στην αναγνώριση κάποιας δυσλειτουργίας στους πνεύμονες. Έρευνες έχουν δείξει, ότι φασματογράμματα ανθρώπων που έχουν ασθένειες, όπως χρόνιο άσθμα ή άσθμα που προκαλείται από άσκηση, έχουν αισθητή διαφορά από αυτά των υγειών ανθρώπων και μάλιστα με πιθανότητα λάθους $p < 0,0001$. Η ύπαρξη άσθματος σε μία ηχογράφηση μπορεί να εντοπιστεί είτε από κάποιο πρόγραμμα που θα κάνει την ανάλυση αυτόματα, είτε, αν η κεντρική εφαρμογή δίνει τη δυνατότητα απευθείας στο γιατρό να ακούσει την ηχογράφηση και να αποφανθεί για το τι συμβαίνει.

Β.6.2 Συμπτώματα

Η ηχογράφηση ανάσας και η έγκαιρη ειδοποίηση του χρήστη από τον γιατρό, εάν υπήρχε ήδη ιστορικό, μπορεί να βοηθήσει τον χρήστη, ώστε να πάρει την αγωγή του νωρίτερα και να μην έχει συμπτώματα όπως:

- Δύσπνοια
- Στηθάγχη
- Βήχας
- Συριγμό
- σοβαρή στένωση των αεραγωγών και ως αποτέλεσμα μία κρίση άσθματος

(33, 34)

Β.7 Ερωτηματολόγιο

Πολύ σπουδαίο ρόλο παίζει η ύπαρξη ενός εύστοχου ερωτηματολογίου και οι απαντήσεις στα ερωτήματα που θέτει. Για παράδειγμα, ένα πλήρες ερωτηματολόγιο μπορεί να βοηθήσει τον γιατρό ώστε να εξάγει τα συμπεράσματά του για την υγεία των ασθενών του και να συντάξει ένα ολοκληρωμένο ιατρικό ιστορικό. Επίσης, βοηθάει τόσο τον γιατρό, όσο και τους ασθενείς, διότι έτσι μπορούν να προλάβουν επικίνδυνες καταστάσεις και περιστατικά. Τέλος, τα ερωτηματολόγια μπορούν να λειτουργήσουν και να εκτιμηθούν συνδυαστικά με τα αποτελέσματα διαφόρων διαγνωστικών εξετάσεων και με αυτό τον τρόπο να εξαχθούν πληρέστερα πορίσματα για την κατάσταση της υγείας του ασθενούς.

Β.8 Στόχος της εφαρμογής

Ο πρωταρχικός στόχος της παρούσας εφαρμογής είναι να βοηθήσει στον τομέα της ανθρώπινης υγείας και γενικά να εξυπηρετήσει τον άνθρωπο και να βελτιώσει την ποιότητα ζωής του με τελικό στόχο τη μακροζωία του και την ψυχοσωματική του υγεία. Σημαντικός αρωγός για αυτόν το σκοπό είναι η τεχνολογία. Συγκεκριμένα, ο στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία δύο εφαρμογών. Η μία θα είναι ανεπτυγμένη για το περιβάλλον του έξυπνου ρολογιού Apple Watch, θα είναι ανεξάρτητη από το κινητό τηλέφωνο του χρήστη και θα συλλέγει δεδομένα που προκύπτουν από τους αισθητήρες υγείας, και όχι μόνο, του συγκεκριμένου έξυπνου ρολογιού. Η άλλη είναι ανεπτυγμένη, ώστε να τρέχει σε όλους τους περιηγητές ιστού και είναι διαθέσιμη και στους απλούς χρήστες αλλά και στους επιστήμονες υγείας. Στους απλούς χρήστες, δείχνει τα δεδομένα τους από όπου και αν βρίσκονται. Ωστόσο, στους επιστήμονες υγείας δείχνει όχι μόνο τα δεδομένα των ασθενών τους, αλλά επιπλέον τους δίνει και τη δυνατότητα να στέλνουν μεμονωμένες και ατομικές ειδοποιήσεις σε όποιον ασθενή το κρίνουν σκόπιμο και απαραίτητο. Τέλος, τους δίνει τη δυνατότητα να στέλνουν, ανά πάσα στιγμή, κάποιο ερωτηματολόγιο στον χρήστη ξεχωριστά, ώστε να εξάγουν δεδομένα υγείας, που αυτοί κρίνουν ότι είναι απαραίτητα.

Με την βοήθεια της συγκεκριμένης εφαρμογής, η οποία αποτελεί αντικείμενο της διπλωματικής μου εργασίας, θα υπάρχει δυνατότητα ανάλυσης όλων των δεδομένων των ασθενών/χρηστών και η εξαγωγή συμπερασμάτων και στατιστικών δεδομένων σχετικών με την υγεία ατόμων με παρόμοια προβλήματα και δυσλειτουργίες. Με αυτόν τον τρόπο θα συμβάλλουμε παραπάνω στην περαιτέρω έρευνα με σκοπό τη θεραπεία πολλών ασθενειών. Μάλιστα, η διαρκής εξέλιξη της τεχνολογίας και η χρησιμοποίηση περισσότερων αισθητήρων, θα συμβάλει ακόμα περισσότερο και θα στηρίξει καθοριστικά όλο τον κλάδο της υγείας και κυρίως την ποιότητα ζωής και υγείας του πληθυσμού.

Β.9 Καινοτομία

Μέχρι σήμερα, το HRV το δίνει η Apple μέσω κάποιου κρυφού αλγόριθμου που τρέχει στο παρασκήνιο, όποτε αυτή το κρίνει σωστό. Αξίζει να αναφερθεί, ότι από τις ήδη υπάρχουσες εφαρμογές, καμία μέχρι σήμερα δεν παρέχει την δυνατότητα εξαγωγής του HRV όποια στιγμή το ζητήσει ο χρήστης. Κατά την εκπόνηση της συγκεκριμένης διπλωματικής εργασίας και εμβαθύνοντας και ερευνώντας όσο το δυνατόν περισσότερο, κατάφερα, μέσω της

δυνατότητας του ηλεκτροκαρδιογραφήματος ενός καναλιού (ECG 1-channel) του Apple Watch, να υπολογίσω το HRVSDNN. Σύμφωνα με τους επιστήμονες, μία σωστή μέτρηση HRV απαιτεί δείγμα διάρκειας 5 λεπτών, ενώ αυτό το έξυπνο ρολόι εξάγει καρδιογραφήματα 30 δευτερολέπτων. Για αυτό λοιπόν τον λόγο λαμβάνουμε υπόψη μας τα τελευταία 10 καρδιογραφήματα και υποδεικνύουμε στο χρήστη-ασθενή να κάνει 10 διαδοχικές μετρήσεις.

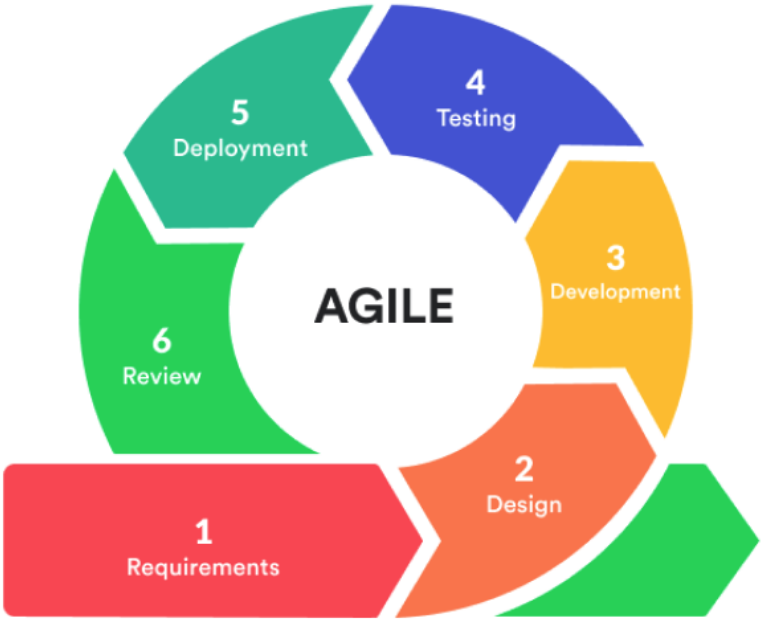
Πρακτικό μέρος – Δομή και μεθοδολογία υλοποίησης της εφαρμογής

Γ.1 Εισαγωγή

Σε αυτήν την ενότητα, γίνεται λεπτομερής περιγραφή των συστατικών μερών της εφαρμογής, της αλληλοεξάρτησής τους και γενικά περιγράφεται ο τρόπος με τον οποίο συνδέονται μεταξύ τους και επηρεάζει το ένα το άλλο. Επιπλέον, η μεθοδολογία που ακολουθήθηκε κατά την εκτέλεση της εργασίας και την ανάπτυξη του λογισμικού, είναι η ευέλικτη μέθοδος ή αλλιώς agile μέθοδος, την οποία θα αναλύσουμε και στην επόμενη υποενότητα. Τέλος, γίνεται αναφορά στα έξοδα που ήταν απαραίτητα για να φτάσουμε σε αυτό το αποτέλεσμα.

Γ.2 Ευέλικτη μεθοδολογία (Agile)

Για την εκπόνηση της διπλωματικής μου εργασίας χρησιμοποιήθηκε η ευέλικτη ανάπτυξη λογισμικού (agile). Η ευέλικτη μεθοδολογία (Agile software development) είναι μια ομάδα μεθόδων ανάπτυξης λογισμικού όπου οι απαιτήσεις και οι λύσεις εξελίσσονται μέσω της συνεργασίας μεταξύ αυτοοργανωμένων, διατμηματικών ομάδων εργασίας. Προωθεί τον προσαρμοστικό σχεδιασμό, την εξελικτική ανάπτυξη, την έγκαιρη παράδοση, τη συνεχή βελτίωση, και ενθαρρύνει την ταχεία και ευέλικτη ανταπόκριση στις αλλαγές. Με πολύ απλά λόγια, αφορά τον γρήγορο και σταδιακό σχεδιασμό, παραγωγή και παράδοση ενός υπό-προϊόντος το οποίο μονίμως παρακολουθείται. Παραδίδεται το συντομότερο δυνατόν στον πελάτη, με στόχο την ανάδρασή του. Έπειτα, το project ξανά σχεδιάζεται με βάση τις νέες απαιτήσεις και συνεχίζεται η διαδικασία έως ότου καλυφθούν όλες οι απαιτήσεις του πελάτη. (35, 36)



Εικόνα 12 Μεθοδολογία Agile

1. Απαιτήσεις (Requirements)

Το πρώτο βήμα στην ευέλικτη μεθοδολογία αποτελεί το να τεθούν οι βασικές λειτουργίες και οι απαιτήσεις, οι οποίες να είναι δυνατό να υλοποιηθούν, να είναι ρεαλιστικές, πρωτοποριακές και βασισμένες στο να εξυπηρετούν τις ανάγκες και απαιτήσεις των χρηστών, χωρίς πλεονασμούς και περιττά σημεία.

2. Σχεδιασμός (Design)

Ο σχεδιασμός της εφαρμογής έγινε με γνώμονα τις απαιτήσεις που είχαν καθοριστεί στο προηγούμενο στάδιο, καθώς και τη διαθεσιμότητα των πόρων όπως χρήματα, εξοπλισμός/εργαλεία, χρόνος κλπ.. Οποιαδήποτε προσθήκη απαίτησης, επηρέαζε τον σχεδιασμό της εφαρμογής, δημιουργώντας την ανάγκη εύρεσης της κατάλληλης δομής για την υλοποίησή της.

3. Ανάπτυξη (Development)

Η ανάπτυξη της εφαρμογής έγινε με τη διαδοχική ενασχόληση και ολοκλήρωση κάθε φορά ενός ορισμένου τμήματος της εργασίας, το οποίο θα έπρεπε να είναι συγκεκριμένο, ώστε να ενώνεται ομαλά με τα υπόλοιπα και όλα μαζί να έχουν ομοιογένεια. Αναλυτικά, χρησιμοποιήθηκαν δομές SaaS (Software as a Service), για να υπάρχει μεγαλύτερη ασφάλεια και δυνατότητα επέκτασης, εάν θα χρειαστεί, λόγω νέων διαμορφωμένων απαιτήσεων. Τέλος, χρησιμοποιήθηκε το GitHub για την ασφαλή αποθήκευση του κώδικα και για να υπάρχει δυνατότητα ανάκτησης παλιότερων εκδόσεων και ιστορικό προόδου.

4. Χτίσιμο και δοκιμή (Deployment και Testing)

Μετά την ολοκλήρωση ανάπτυξης κάθε λειτουργίας, ακολουθούσε η δοκιμή αρχικά σε τοπικό περιβάλλον και στη συνέχεια δοκιμή κάτω από αληθοφανείς συνθήκες, ώστε να εξασφαλιστεί η ορθότητα της υλοποίησης, των αποτελεσμάτων όσο και η αποδοτικότητά της. Για τις υπηρεσίες που ήταν έτοιμες από το Google Firebase, προφανώς, δεν χρειάστηκε δοκιμή, παρά μόνο για την επιβεβαίωση της σωστής συνεργασίας μεταξύ της εφαρμογής και του Firebase. Τέλος, το deployment γινόταν μόνο μετά από επιτυχημένο testing.

5. Αξιολόγηση (Review)

Το τελευταίο στάδιο της ευέλικτης μεθοδολογίας είναι η αξιολόγηση των αποτελεσμάτων, του τρόπου ανάπτυξης και ο εντοπισμός των δυσκολιών, προκειμένου να μην επαναληφθούν σε μελλοντική ανάπτυξη άλλης λειτουργίας. Ο σχολαστικός έλεγχος ήταν πολύ σπουδαίος για την αποφυγή λαθών. Τέλος, αξίζει να σημειωθεί πως για την ολοκλήρωση αυτής της εργασίας ,απαιτήθηκαν πολλοί επανέλεγχοι, ώστε το τελικό αποτέλεσμα, να είναι σίγουρα σωστό και ακριβές.

Γ.3 Ευρύτερη δομή των μερών της εφαρμογής

Η συγκεκριμένη διπλωματική εργασία απαρτίζεται από τρία βασικά μέρη: την εφαρμογή στο ρολόι Apple Watch, το Google Firebase και την κεντρική εφαρμογή που περιλαμβάνει μία ιστοσελίδα για τους χρήστες/ασθενείς και άλλη μία για τους γιατρούς. Αρχικά, η εφαρμογή για το ρολόι είναι υπεύθυνη για την λήψη των μετρήσεων και την τέλεση των ερωτηματολογίων. Επίσης, είναι υπεύθυνη για την ενημέρωση του χρήστη, αν ο γιατρός του έχει ζητήσει κάποια επιπλέον μέτρηση και για τη γενικότερη αλληλεπίδραση του χρήστη με την κεντρική εφαρμογή. Στη συνέχεια, το Google Firebase παρέχει τη δυνατότητα του authentication, ώστε να γίνεται ασφαλής ταυτοποίηση των χρηστών και των γιατρών. Ακόμη, παρέχει την βάση δεδομένων και τον χώρο για την αποθήκευση των δεδομένων των χρηστών και, επιπλέον, δίνει τη δυνατότητα του Hosting για την κεντρική εφαρμογή. Επιπροσθέτως, δίνει την δυνατότητα της αποστολής των ειδοποιήσεων προς τα Apple Watch των χρηστών μέσω του Cloud Messaging και με τη χρήση των Functions μπορούμε να στέλνουμε ξεχωριστές ειδοποιήσεις στον καθένα. Τέλος, το πρώτο μέρος της κεντρικής εφαρμογής, το οποίο αφορά τους χρήστες, δίνει τη δυνατότητα να βλέπει ο καθένας τα δικά του δεδομένα και να ενημερώνεται σχετικά. Το δεύτερο μέρος της, το οποίο αφορά τους γιατρούς, δίνει τη δυνατότητα της άμεσης αλληλεπίδρασης με τους χρήστες και ενημέρωσης των γιατρών για την κατάσταση υγείας αυτών μέσα από τις τιμές που δίνει το Apple Watch από το HealthKit και του HRV που υπολογίζουμε.

Εικόνα 13 Εσωτερική επικοινωνία εφαρμογής

Γ.4 Υλοποίηση της κεντρικής εφαρμογής με τη βοήθεια του Firebase Hosting

Η δημιουργία μίας προσαροσμένης κεντρικής εφαρμογής, στην οποία θα έχουν πρόσβαση και οι απλοί χρήστες αλλά και οι γιατροί, θα απαιτούσε να βρεθεί ο κατάλληλος πάροχος για το hosting, να αγοραστεί κάποιο κατάλληλο domain name και να γίνουν οι απαραίτητες συνδέσεις των δικτύων. Για τον λόγο αυτό και για να αποφευχθεί όλη αυτή η χρονοβόρα και δαπανηρή διαδικασία, χρησιμοποιήθηκε το Firebase Hosting, το οποίο όλα τα παραπάνω τα κάνει από μόνο του και από εμάς ζητάει μόνο τον κώδικα, ώστε να τρέξει σωστά η ιστοσελίδα μας.

Γ.5 Χρήση Firebase Authentication από το Apple Watch και την κεντρική εφαρμογή

Η δημιουργία ενός προσαρμοσμένου συστήματος αυθεντικοποίησης χρηστών είναι μία δύσκολη και χρονοβόρα διαδικασία, αφού θα πρέπει να βεβαιωθεί τόσο η σωστή λειτουργία του όσο και η εγγυημένη ασφάλειά του. Πέρα από αυτά, η συντήρηση, οι απαραίτητες αναβαθμίσεις και γενικά η παρακολούθηση ενός τέτοιου αυθεντικοποιητή απαιτεί εξειδικευμένη ομάδα ανθρώπων, κάτι το οποίο δεν είναι εφικτό στα πλαίσια μίας διπλωματικής εργασίας. Για αυτόν τον λόγο προτιμήθηκε η χρήση ενός έτοιμου αυθεντικοποιητή του Firebase Authenticator, ο οποίος με τη βοήθεια συναρτήσεων που δίνει η ίδια η Google, συνδέθηκε τόσο με το Apple Watch, όσο και με την κεντρική εφαρμογή.

Γ.6 Επικοινωνία Apple Watch και βάσης δεδομένων Firebase Realtime Database

Πρέπει να αναφερθεί ότι μεταξύ του Apple Watch και της βάσης δεδομένων Firebase Realtime Database υπάρχει αμοιβαία σύνδεση. Συγκεκριμένα, οι μετρήσεις των δεδομένων υγείας και το HRV ανεβαίνουν σε πραγματικό χρόνο στην βάση δεδομένων κατηγοριοποιημένες ανά χρήστη και συνοδεύονται από ημερομηνία και ώρα που έγινε η μέτρηση.

Γ.7 Επικοινωνία Apple Watch και Firebase Remote Config

Για την υλοποίηση των ερωτηματολογίων είναι απαραίτητη η σύνδεση του Apple Watch με το Firebase Remote Config. Μέσω αυτής της λειτουργίας στο Firebase, μπορούμε να αποθηκεύουμε ερωτηματολόγια σε μορφή JSON και οι γιατροί να επιλέγουν κάποιο από αυτά για να πάει σε όποιον ασθενή θέλουν, μέσω μίας μεταβλητής στην Realtime Database, που υπάρχει για τον κάθε χρήστη ξεχωριστά. Τα αποθηκευμένα ερωτηματολόγια μορφής JSON τα κατεβάζει το Apple Watch και τα δείχνει στον χρήστη μέσα από κώδικα που τρέχει στο ίδιο το ρολόι.

Γ.8 Επικοινωνία εφαρμογής στο Apple Watch με το HealthKit

Η υπηρεσία HealthKit προσφέρει μια βάση δεδομένων, όπου η λειτουργία της έχει αναλυθεί σε προηγούμενο κεφάλαιο. Με την εφαρμογή που δημιουργήθηκε στο Apple Watch, παρέχεται η δυνατότητα εξαγωγής μετρήσεων που έχει κάνει το ρολόι, με αλγορίθμους της

Apple, οι οποίες αφορούν τους καρδιακούς παλμούς, HRV, ταχύτητα βαδίσματος, θερμοκρασία καρπού κατά τον ύπνο, οξυγόνο στο αίμα κλπ.. Επίσης, μας δίνει τη δυνατότητα ανάγνωσης ολόκληρου του καρδιογραφήματος, εξαγωγής των μετρήσεων και επεξεργασίας τους, ώστε να βγάλουμε περισσότερα δεδομένα. Όλες αυτές οι τιμές συγκεντρώνονται και αποστέλλονται στη βάση δεδομένων Firebase Realtime Database.

Γ.9 Επικοινωνία κεντρικής εφαρμογής και βάσης δεδομένων Firebase Realtime Database

Η άμεση και σε πραγματικό χρόνο επικοινωνία, τόσο της κεντρικής εφαρμογής που απευθύνεται στους χρήστες/ασθενείς, όσο αυτής που απευθύνεται στους γιατρούς, είναι απολύτως απαραίτητη, ώστε να ενημερώνονται οι χρήστες για τα δεδομένα υγείας τους και οι γιατροί να μπορούν να επιλέξουν έναν από τους ασθενείς τους, προκειμένου να δουν τα δεδομένα τους σε πρώτη φάση και κατόπιν να τους στείλουν, για παράδειγμα, κάποια ειδοποίηση για να ζητήσουν παραπάνω μετρήσεις. Πέρα από τα δεδομένα υγείας, η σύνδεση της κεντρικής εφαρμογής με τη βάση είναι απαραίτητη, ώστε να μπορούμε να απεικονίσουμε όλους τους χρήστες στον γιατρό και να μπορεί να επιλέξει από αυτούς, όποιον κρίνει. Τέλος, αυτή σύνδεση βοηθάει στην εξαγωγή πληροφοριών για τον συνδεδεμένο γιατρό ή χρήστη, ώστε να μπορούμε να τον προσφωνήσουμε πιο εύκολα μέσα στην ιστοσελίδα με την οποία αλληλοεπιδρά.

Γ.10 Επικοινωνία Apple Watch και κεντρικής εφαρμογής με το Firebase Cloud Storage

Η επικοινωνία του Apple Watch και της κεντρικής εφαρμογής, είναι απαραίτητη για δύο λόγους. Αφενός, μας προσφέρει τη δυνατότητα να μεταφέρουμε τις μετρήσεις από το καρδιογράφημα που γίνεται στο ρολόι, οι οποίες είναι εγγεγραμμένες σε ένα αρχείο txt, στον αποθηκευτικό χώρο του Firebase και στη συνέχεια η κεντρική εφαρμογή κατεβάζει από τον χώρο το αρχείο και το αναλύει, ώστε να το δείχνει σε μορφή γραφήματος στον χρήστη και στον γιατρό. Αφετέρου, μας επιτρέπει να στέλνουμε την ηχογράφηση από την ανάσα του ασθενή στον αποθηκευτικό χώρο σε μορφή mp4 και στη συνέχεια η κεντρική εφαρμογή να την κατεβάζει, ώστε, να έχει τη δυνατότητα ο γιατρός να την ακούσει και να την αξιολογήσει.

Γ.11 Επικοινωνία Apple Watch και κεντρικής εφαρμογής με το Firebase Cloud Messaging

Η δημιουργία ενός προσαρμοσμένου συστήματος, το οποίο να στέλνει ειδοποιήσεις, είναι μία πολύ δύσκολη και χρονοβόρα διαδικασία, αφού θα πρέπει να βεβαιωθεί η σωστή λειτουργία του και να προγραμματιστεί σωστά, ώστε να είναι ασφαλές και να πληροί τα κριτήρια της Apple και του APNS. Για αυτόν τον λόγο, προτιμήθηκε η χρήση ενός έτοιμου συστήματος ειδοποιήσεων, του Firebase Cloud Messaging. Αφού έγιναν τα απαραίτητα βήματα, ώστε να συνδεθεί με το Apple Watch, τα οποία έχουν περιγραφεί σε προηγούμενο κεφάλαιο, ο push notification server ήταν έτοιμος προς χρήση. Τέλος, η σύνδεση της κεντρικής εφαρμογής με το Firebase Cloud Messaging, έγινε με τη βοήθεια μιας Fibebase Cloud Function, για την οποία θα μιλήσουμε παρακάτω. Αυτή μέσω του notification token που της δίνεται από τη Realtime Database, στέλνει μεμονωμένες ειδοποιήσεις σε όποιον

χρήστη επιλέξει ο γιατρός. Έτσι, ο γιατρός μπορεί να ειδοποιήσει μεμονωμένα όποιον ασθενή κρίνει ότι είναι απαραίτητο.

Γ.12 Χρήση συναρτήσεων Firebase Cloud Functions στην κεντρική εφαρμογή

Στη συγκεκριμένη διπλωματική εργασία χρειάστηκε μία κατηγορία συναρτήσεων. Αυτές οι συναρτήσεις είναι αποθηκευμένες στον Cloud χώρο “Functions” της Google Firebase και οι συγκεκριμένες είναι προγραμματισμένες, ώστε να βρίσκουν το notification token του χρήστη, το οποίο είναι αποθηκευμένο στην Firebase Realtime Database, ώστε μετά με τη χρήση του Firebase Admin, το οποίο έχει πρόσβαση στο Firebase Cloud Messaging, όταν καλείται η συνάρτηση να μπορεί να στείλει ειδοποίηση, ώστε ο χρήστης να στείλει μία νέα μέτρηση για ένα συγκεκριμένο δεδομένο υγείας. Τέλος, αυτή η Firebase Cloud Function που περιεγράφη παραπάνω, είναι σχεδιασμένη ώστε να τρέχει με http κλήση από το front end της κεντρικής εφαρμογής και συγκεκριμένα από το μέρος που έχουν πρόσβαση οι γιατροί. Έτσι, επιτελεί το στόχο της, ο οποίος είναι να μπορούν οι γιατροί να ειδοποιούν τους χρήστες.

Γ.13 Κόστος διπλωματικής εργασίας

Γ.13.1 Κόστος ανάπτυξης εφαρμογής στο οικοσύστημα της Apple

Η ανάπτυξη μιας εφαρμογής στο οικοσύστημα της Apple, δηλαδή σε watchOS για τα έξυπνα ρολόγια, macOS για τους υπολογιστές, iOS για τα κινητά, tvOS για τα smart boxes τηλεοράσων και iPadOS για τα tablets, προϋποθέτουν πέρα από την απόκτηση της συσκευής για την οποία θέλεις να αναπτύξεις εφαρμογή, τη χρήση υπολογιστή Mac και μάλιστα σχετικά νέου μοντέλου, επειδή το Xcode, που είναι το περιβάλλον ανάπτυξης εφαρμογών της Apple, απαιτεί την τελευταία έκδοση λογισμικού MacOS. Επιπλέον, το Xcode απαιτεί από μόνο του, την τελευταία έκδοση λογισμικού στην αντίστοιχη συσκευή για την οποία αναπτύσσεται η εφαρμογή. Μάλιστα, αν κάποιος θέλει να χτίσει εφαρμογή για έξυπνο ρολόι Apple Watch, είναι αναγκασμένος να αγοράσει και iPhone, το οποίο να έχει τη δυνατότητα να τρέξει την τελευταία έκδοση iOS, ώστε να λειτουργεί σαν κόμβος επικοινωνίας του Xcode που τρέχει στον Mac και του Apple Watch, το οποίο είναι συνδεδεμένο στο iPhone μέσω Bluetooth. Επιπροσθέτως, η Apple μέσω του Xcode, παρότι παρέχει simulators, δεν μπορούν να αντικαταστήσουν καθολικά τις πραγματικές συσκευές, διότι πολλά πράγματα που σχετίζονται με τη συνδεσιμότητα στο διαδίκτυο και τη μεταφορά δεδομένων, δεν δουλεύουν όπως στην πραγματική συσκευή. Τέλος, παρότι η Apple παρέχει και δωρεάν λογαριασμό developer, ώστε να έχει ο καθένας πρόσβαση στο Xcode, για πολλές λειτουργίες (π.χ. ειδοποιήσεις) είναι απαραίτητο το μεγαλύτερο πρόγραμμα, το οποίο έχει συνδρομή 99$ το χρόνο, και φαίνεται παρακάτω:

	Sign In with Your Apple ID	Apple Developer Program
Xcode developer tools	●	●
Xcode beta releases	●	●
On-device testing	●	●
Apple Developer Forums	●	●
Bug reporting with Feedback Assistant	●	●
OS beta releases		●
Full access to a comprehensive set of development tools		●
Advanced app capabilities and services		●
Code-level support		●
App distribution on the App Store		●
App management, testing, and analytics with App Store Connect		●
Safari Extensions distribution		●
Software distribution outside the Mac App Store		●
Custom app distribution with Apple Business Manager and Apple School Manager		●
Proprietary app distribution to your employees with Apple Business Manager		●
Ad hoc distribution for testing and internal use		●
Access to members-only developer events or additional event content		●
Cost	Free	99 USD**

Εικόνα 14 Πρόγραμμα Apple Developer

Γ.13.2 Firebase

Η πλατφόρμα του Firebase παρέχει τις περισσότερες από τις λειτουργίες, που έχουμε περιγράψει στα προηγούμενα κεφάλαια, εντελώς δωρεάν. Η μόνη που ήθελε το πρόγραμμα που είναι επί πληρωμή ήταν η Firebase Cloud Functions. Όμως, ούτε για αυτό υπήρξε χρέωση, επειδή η Google το χει κάνει “Pay as you go”, οπότε μέχρι ένα όριο χρήσης που έχουν θέσει, είναι και αυτό δωρεάν, απλά διατηρούν το δικαίωμα να σε χρεώσουν αυτόματα, αν το ξεπεράσεις. Ο παρακάτω πίνακας δείχνει όλες τις χρεώσεις που θέτει η Google για κάθε υπηρεσία:

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*
A/B Testing	No-cost	
Analytics	No-cost	
App Distribution	No-cost	
App Indexing	No-cost	
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
Cloud Firestore		
Stored data	1 GiB total	No-cost up to 1 GiB total Then Google Cloud pricing
Network egress	10 GiB/month	No-cost up to 10 GiB/month Then Google Cloud pricing
Document writes	20K writes/day	No-cost up to 20K writes/day Then Google Cloud pricing
Document reads	50K reads/day	No-cost up to 50K reads/day Then Google Cloud pricing
Document deletes	20K deletes/day	No-cost up to 20K deletes/day Then Google Cloud pricing

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
Cloud Messaging (FCM)	No-cost	
Crashlytics	No-cost	
Dynamic Links	No-cost	
Hosting		
Storage	10 GB	$0.026/GB
Data transfer	360 MB/day	$0.15/GB
Custom domain & SSL	✓	✓
Multiple sites per project	✓	✓
In-App Messaging	No-cost	
Firebase ML		
Custom Model Deployment	✓	✓
Cloud Vision APIs	✗	$1.50/K (See Cloud Vision pricing)
Performance Monitoring	No-cost	
Realtime Database		
Simultaneous connections	100	200k/database
GB stored	1 GB	$5/GB
GB downloaded	10 GB/month	$1/GB
Multiple databases per project	✗	✓

Remote Config	No-cost	
Cloud Storage		
GB stored	5 GB	$0.026/GB
GB downloaded	1 GB/day	$0.12/GB
Upload operations	20K/day	$0.05/10k
Download operations	50K/day	$0.004/10k
Multiple buckets per project	✗	✓
Test Lab		
Virtual Device Tests	10 tests/day	No-cost up to 60 min/day Then $1/device/hour
Physical Device Tests	5 tests/day	No-cost up to 30 min/day Then $5/device/hour
Google Cloud		
BigQuery	✓	✓
Other IaaS	✗	✓

Εικόνα 15 Πρόγραμμα Blaze του Firebase

Γ.13.3 Λοιπά έξοδα

Ενώ για τη δημιουργία της εφαρμογής στο ρολόι απαιτήθηκαν κάποια έξοδα, για την κεντρική εφαρμογή και όλα τα υπόλοιπα δεν απαιτήθηκε η πληρωμή κάποιου ποσού. Τα εργαλεία που χρησιμοποιήθηκαν (Visual Studio Code κλπ.) ήταν δωρεάν, ανεξαρτήτως λειτουργικού συστήματος και το περιβάλλον της ιστοσελίδας της κεντρικής εφαρμογής, μέσω του Firebase Hosting, ανέβηκε στο διαδίκτυο εντελώς δωρεάν.

Γ.14 Άλλες μέθοδοι υλοποίησης της εργασίας

Στην συγκεκριμένη εργασία, λόγω της εύρεσης του HRV από το ηλεκτροκαρδιογράφημα, είναι απαραίτητη η χρήση έξυπνου ρολογιού το οποίο να περιέχει αισθητήρα για ηλεκτροκαρδιογράφημα. Επίσης, θα ήταν καλό να έχει και κάποια πιστοποίηση από κάποιον αρμόδιο φορέα ότι βγάζει σωστά αποτελέσματα. Αυτή τη στιγμή οι μόνες εταιρείες που βγάζουν έξυπνα ρολόγια με ηλεκτροκαρδιογράφημα που έχει πιστοποίηση είναι η Apple, η Samsung, η Google και η Fitbit. Ο προγραμματισμός των Samsung και των Google, εφόσον τρέχουν Wear OS, γίνεται μέσω του Android Studio και μέσω της γλώσσας προγραμματισμού Java, η οποία είναι και αυτή αντικειμενοστραφής και έχει πολλά κοινά με τη Swift. Το θετικό με το Android Studio είναι ότι τρέχει σε οποιοδήποτε λειτουργικό (Windows, Mac, Linux, ChromeOS), σε αντίθεση με το Xcode που τρέχει μόνο σε Mac. Ο προγραμματισμός των

Fitbit, επειδή τρέχουν το δικό τους Fitbit OS, γίνεται μέσω του Fitbit Studio το οποίο τρέχει σε Windows και Mac και μέσω των γλωσσών προγραμματισμού JavaScript, CSS, και SVG.

Στη συνέχεια, πέρα από τη Google Firebase, υπάρχουν κι άλλοι πάροχοι Cloud, όπως η Microsoft, η IBM, η Amazon κι άλλες, οι οποίες προσφέρουν παρόμοιες δυνατότητες. Τέλος, όσον αφορά την ιστοσελίδα της κεντρικής εφαρμογής, η ανάπτυξή της σε HTML, CSS και JavaScript, αποτελεί μια από τις πολλές επιλογές που δίνονται. Υπάρχουν κι άλλες λύσεις όπως η React κλπ.. (37)

Πρακτικό Μέρος – Απεικόνιση και ερμηνεία κώδικα

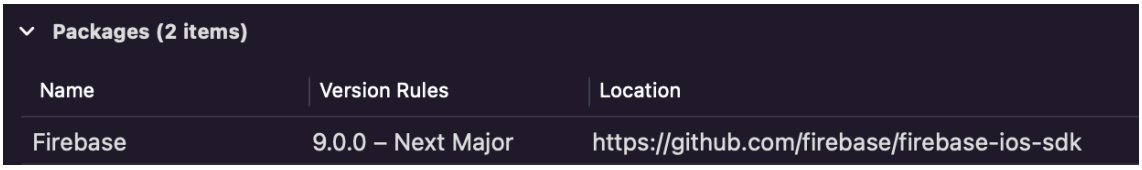
Δ.1 Εισαγωγή

Για να πραγματοποιηθούν όλες οι προαναφερθείσες λειτουργίες και να επέλθουν τα επιθυμητά αποτελέσματα, ήταν απαραίτητη και η ανάπτυξη κώδικα, ο οποίος αποτέλεσε το μεγαλύτερο μέρος της παρούσας διπλωματικής εργασίας. Η παρούσα ενότητα, αναφέρεται στην ερμηνεία του κώδικα της εφαρμογής που τρέχει στο έξυπνο ρολόι Apple Watch, του κώδικα της κεντρικής εφαρμογής/ιστοσελίδας καθώς και στον τρόπο με τον οποίο συνδέονται όλα αυτά με την πλατφόρμα Firebase της Google.

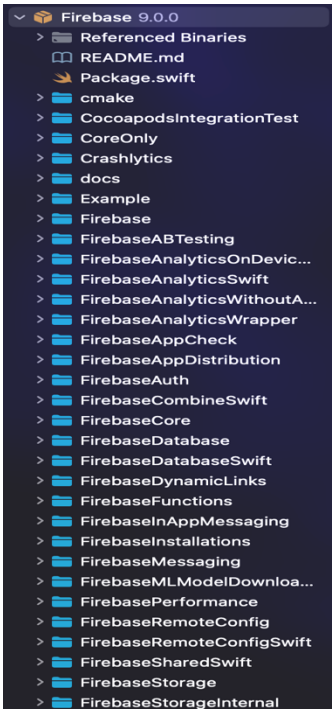
Δ.2 Εφαρμογή στο έξυπνο ρολόι Apple Watch

Δ.2.1 Επικοινωνία με Firebase

Για την επικοινωνία της εφαρμογής με το Firebase, είναι απαραίτητη η εγκατάσταση της έτοιμης βιβλιοθήκης που δίνει η Google, η οποία είναι σε μορφή package. Αρχικά, προσθέτουμε το dependency, όπως φαίνεται παρακάτω, το οποίο όχι μόνο κάνει εγκατάσταση την βιβλιοθήκη, αλλά, με το πάτημα ενός κουμπιού γίνεται και update όποτε χρειαστεί.

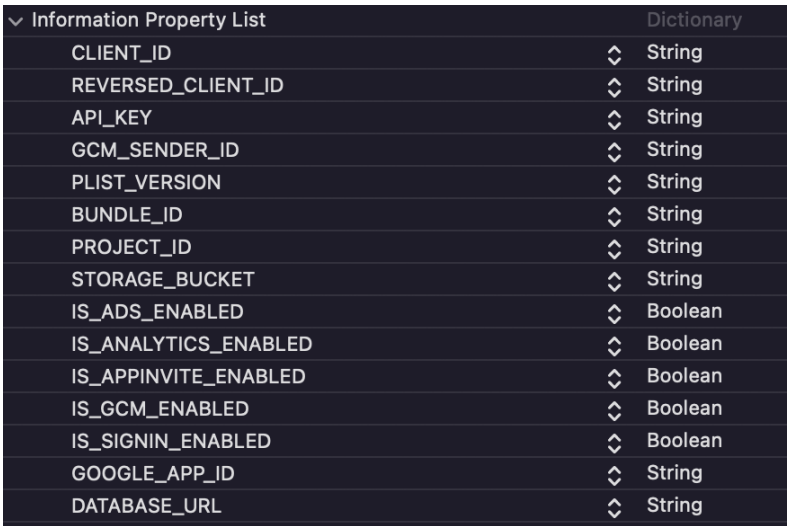


Εικόνα 16 Firebase Package



Εικόνα 17 Περιεχόμενα του πακέτου Firebase

Μόλις γίνει η εγκατάσταση, όπως περιγράφηκε παραπάνω, στα αρχεία της εφαρμογής εμφανίζεται η βιβλιοθήκη του Firebase, με τα απαραίτητα αρχεία της, όπως φαίνεται στη διπλανή εικόνα. Για να πραγματοποιηθεί η σύνδεση, αφού γίνει το setup στην κονσόλα του Firebase, ο οδηγός δίνει ένα αρχείο με την ονομασία GoogleService-Info.plist, το οποίο περιέχει όλες τις απαραίτητες μεταβλητές για να ολοκληρωθεί η σύνδεση μεταξύ εφαρμογής και Firebase. Η μορφή του αρχείου φαίνεται παρακάτω, με κρυμμένα τα values των πεδίων για ευνόητους λόγους:



Εικόνα 18 Αρχείο GoogleService-Info.plist

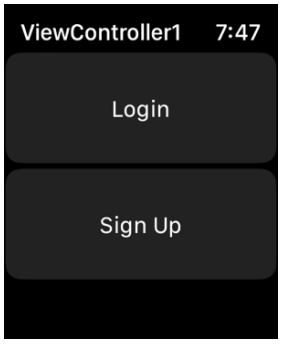
Το τελικό βήμα είναι ο προγραμματισμός της εφαρμογής έτσι ώστε κάθε φορά που ανοίγει, να καλεί την σύνδεση με την Firebase. Αυτό γίνεται στην έτοιμη συνάρτηση που δίνει η Apple, η οποία λέγεται applicationDidFinishLaunching() και καλείται αυτόματα κάθε φορά που η εφαρμογή ανοίγει:

```
func applicationDidFinishLaunching() {
    FirebaseApp.configure()
```

Εικόνα 19 Κλήση της υπηρεσίας του Firebase

Πλέον η σύνδεση έχει επιτευχθεί και η κλήση των συναρτήσεων του Firebase είναι εφικτή, εφόσον έχει προηγουμένως συμπεριληφθεί η κατάλληλη βιβλιοθήκη. (38)

Δ.2.2 Εγγραφή νέου χρήστη

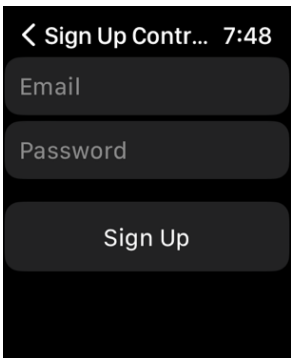


Εικόνα 20 Αρχική οθόνη

Αρχικά, ο χρήστης, μόλις ανοίξει την εφαρμογή, έχει να επιλέξει μεταξύ του να κάνει σύνδεση, αν έχει ήδη κάποιο λογαριασμό ή να κάνει εγγραφή, αν είναι νέος χρήστης.

Για την εγγραφή, η εφαρμογή απαιτεί από τον χρήστη μόνο ένα email και έναν κωδικό, όπως φαίνεται στη διπλανή εικόνα. Για να γίνει αυτό, ακολουθήθηκαν οι οδηγίες στο documentation του Firebase Authentication. Το πρώτο βήμα είναι να γίνουν import οι απαραίτητες βιβλιοθήκες του Firebase.

```
import FirebaseAuth
import Firebase
```



Εικόνα 21 Οθόνη Sign Up

Αμέσως μετά, καλείται η συνάρτηση createUser της βιβλιοθήκης FirebaseAuth, η οποία είναι συνδεδεμένη με το κουμπί του Storyboard “Sign Up”, όπως φαίνεται παρακάτω (39):

```swift
@IBAction func signUpTapped(_ sender: Any) {

    // Validate the fields
    let error = validateFields()

    if error != nil {

        // There's something wrong with the fields, show error message
        showError(error!)
    }
    else {

        // Create cleaned versions of the data
        let email = email.trimmingCharacters(in: .whitespacesAndNewlines)
        let password = password.trimmingCharacters(in: .whitespacesAndNewlines)

        // Create the user
        Auth.auth().createUser(withEmail: email, password: password) { (result, err) in
            // Check for errors
            if err != nil {

                // There was an error creating the user
                self.showError("Error creating user")
            }
            else {
                // Transition to the home screen
                UserDefaults.standard.set(email, forKey: "email")
                UserDefaults.standard.set(password, forKey: "password")
                self.transitionToHome()
            }
        }
    }
}
```

Εικόνα 22 Συνάρτηση του κουμπιού Sign Up

Αν ο χρήστης πληκτρολογήσει στο κουτάκι του email, κάτι το οποίο δεν έχει τη μορφή email ή αν δεν έχει πρόσβαση στο διαδίκτυο ή αν ο κωδικός του είναι κάτω από 6 ψηφία, θα του δείξει με κόκκινα γράμματα ότι υπάρχει σφάλμα. Αν όμως ο χρήστης τα έχει κάνει όλα σωστά, ακολουθεί μια διαδικασία αποθήκευσης των στοιχείων του χρήστη στον σκληρό δίσκο του έξυπνου ρολογιού. Αυτό γίνεται, επειδή η οθόνη του Apple Watch είναι πολύ μικρή και καθιστά πολύ άβολη την πληκτρολόγηση των credentials κάθε φόρα που ανοίγει η εφαρμογή. Για αυτόν τον λόγο χρησιμοποιείται ο χώρος UserDefaults, ο οποίος είναι μοναδικός για κάθε εφαρμογή. Κάθε φορά που ανοίγει η συγκεκριμένη εφαρμογή, φορτώνονται αυτόματα οι μεταβλητές που είναι αποθηκευμένες στη UserDefaults, καλείται ο ViewController1 της παραπάνω εικόνας και καλείται αυτόματα από αυτόν η συνάρτηση signIn της βιβλιοθήκης FirebaseAuth με τα αποθηκευμένα credentials, όπως φαίνεται στην παρακάτω εικόνα:

```swift
class ViewController1: WKInterfaceController {

    @IBOutlet weak var loginButton: WKInterfaceButton!

    @IBOutlet weak var signUpButton: WKInterfaceButton!

    override func willActivate() {
        super.willActivate()
        let email = UserDefaults.standard.string(forKey: "email")
        let password = UserDefaults.standard.string(forKey: "password")
        if email != nil, password != nil {
            Auth.auth().signIn(withEmail: email!, password: password!) { (result, error) in

                if error != nil {
                    // Couldn't sign in
                }
                else {
                    WKInterfaceController.reloadRootPageControllers(withNames: ["InterfaceController"],
                        contexts: nil, orientation: .vertical, pageIndex: 0)
                    MyVariables.yourVariable = email!
                }
            }
        }
    }
}
```

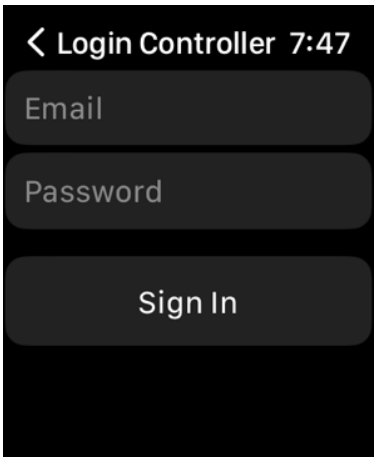
Εικόνα 23 Αποθήκευση των credentials του χρήστη

Τέλος, αφού γίνουν όλα αυτά, καλείται η συνάρτηση transitionToHome(), ώστε να επιτραπεί στην εφαρμογή να προχωρήσει στην αρχική σελίδα της εφαρμογής, όπως φαίνεται στην παρακάτω εικόνα.

```
func transitionToHome() {
    WKInterfaceController.reloadRootPageControllers(withNames: ["InterfaceController"], contexts: nil,
        orientation: .vertical, pageIndex: 0)
}
```

Εικόνα 24 Συνάρτηση επιστροφής στην αρχική σελίδα

Δ.2.3 Σύνδεση χρήστη



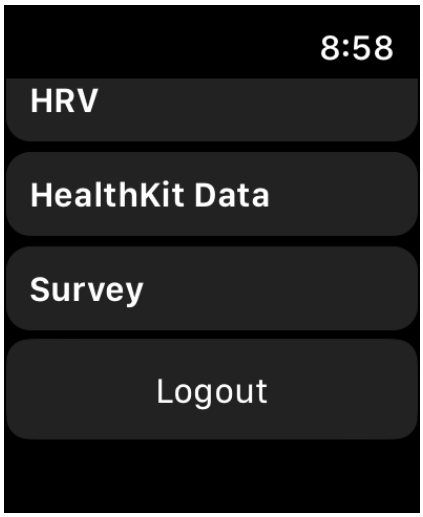
Εικόνα 26 Οθόνη σύνδεσης χρήστη

Σχεδόν όμοια με την εγγραφή χρήστη, που αναλύθηκε παραπάνω, είναι και η σύνδεση, της οποίας το UI φαίνεται στη διπλανή εικόνα. Η διαφορά που υπάρχει στον κώδικα είναι μόνο ότι αντί για createUser, βάζουμε την συνάρτηση signIn της βιβλιοθήκης FirebaseAuth. Τέλος, κάθε φορά που γίνεται login, ανανεώνονται οι μεταβλητές email και password στο UserDefaults, έτσι ώστε, αν ο χρήστης δώσει νέα στοιχεία, να ανανεωθούν, όπως φαίνεται στην παρακάτω εικόνα.

```
@IBAction func loginTapped(_ sender: Any) {

    // TODO: Validate Text Fields

    // Create cleaned versions of the text field
    let email = email.trimmingCharacters(in: .whitespacesAndNewlines)
    let password = password.trimmingCharacters(in: .whitespacesAndNewlines)

    UserDefaults.standard.set(email, forKey: "email")
    UserDefaults.standard.set(password, forKey: "password")

    // Signing in the user
    Auth.auth().signIn(withEmail: email, password: password) { (result, error) in

        if error != nil {
            // Couldn't sign in
            self.errorLabel.setText(error!.localizedDescription)
            self.errorLabel.setAlpha(1)
        }
        else {
            WKInterfaceController.reloadRootPageControllers(withNames: ["InterfaceController"],
                contexts: nil, orientation: .vertical, pageIndex: 0)
            MyVariables.yourVariable = email
        }
    }
}
```

Εικόνα 25 Συνάρτηση κουμπιού Sign In

Δ.2.4 Αποσύνδεση χρήστη



Εικόνα 27 Οθόνη κουμπιού Log Out

Για να μπορεί ο χρήστης είτε να αποσυνδεθεί, είτε να συνδεθεί με νέο χρήστη, στην κεντρική σελίδα της εφαρμογής δημιουργήθηκε ένα κουμπί “logout”. Αυτό το κουμπί, πέρα από το ότι αποσυνδέει τον χρήστη, διαγράφει και από την UserDefaults τα αποθηκευμένα credentials, ώστε ο χρήστης να μπορεί να βάλει τα καινούργια και τον γυρνάει στον ViewController1. Τα παραπάνω είναι φανερά στις εικόνες που έχω παραθέσει.

```
@IBAction func logoutTapped(_ sender: Any) {
    UserDefaults.standard.removeObject(forKey: "email")
    UserDefaults.standard.removeObject(forKey: "password")
    WKInterfaceController.reloadRootPageControllers(withNames: ["ViewController1"], contexts: nil,
        orientation: .vertical, pageIndex: 0)
}
```

Εικόνα 28 Συνάρτηση κουμπιού Log Out

Δ.2.5 Άδεια για ειδοποιήσεις και πρόσβαση στα δεδομένα του HealthKit

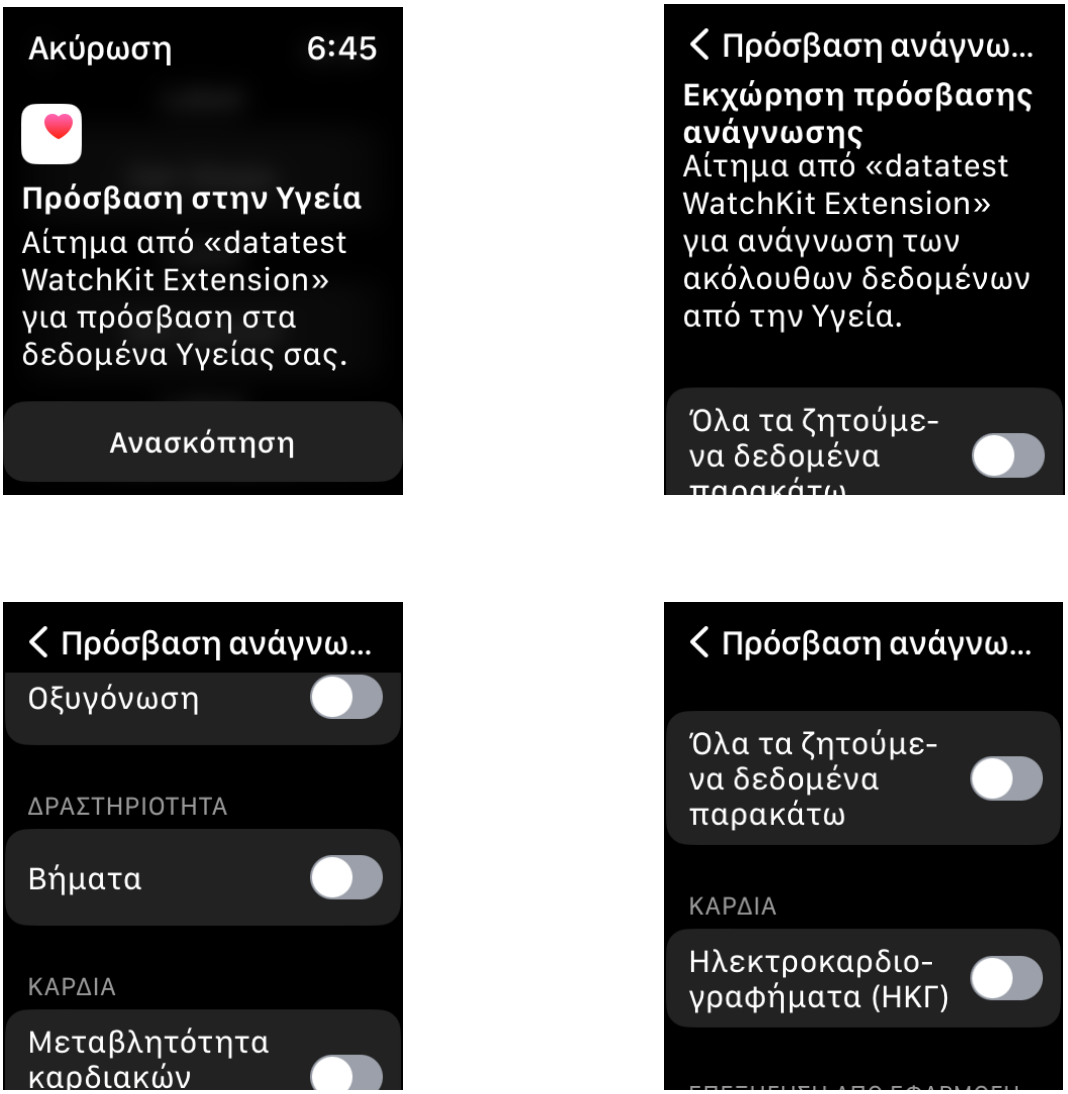
Εικόνα 29 Μήνυμα για άδεια ειδοποιήσεων

Στο οικοσύστημα της Apple, για να μπορέσει ο προγραμματιστής να έχει πρόσβαση σε οποιαδήποτε δεδομένα (είτε ανάγνωση, είτε εγγραφή νέων), εκτός της δικής του εφαρμογής, είναι υποχρεωμένος να ζητάει την άδεια του χρήστη και να εξηγεί πώς θα τα χρησιμοποιήσει. Μάλιστα, το ίδιο ισχύει και για τις συσκευές εισόδου/εξόδου, όπως το μικρόφωνο αλλά και για να στείλει ειδοποιήσεις η εφαρμογή στον χρήστη ενώ τρέχει στο παρασκήνιο. Αυτά συμβαίνουν τόσο για τον περιορισμό της ανεξέλεγκτης χρήσης προσωπικών δεδομένων του χρήστη χωρίς την έγκρισή του, όσο και για να μην υποχρεώνεται να έχει ενοχλητικές ειδοποιήσεις από εφαρμογές που δεν θεωρεί σημαντικές για την καθημερινότητά του. Στην εικόνα στα αριστερά φαίνεται η ειδοποίηση που εμφανίζεται στην οθόνη του Apple Watch για να επιλέξει ο χρήστης, αν θέλει να λαμβάνει ειδοποιήσεις. Στην εικόνα από κάτω φαίνεται το κομμάτι του κώδικα που το καθορίζει αυτό.

```
UNUserNotificationCenter.current().requestAuthorization(options: authOpts) { granted, _ in
    if granted {
        WKExtension.shared().registerForRemoteNotifications()
    }
}
```

Εικόνα 30 Κώδικας για ερώτηση άδειας

Παρακάτω παραθέτω τόσο τα αναδυόμενα μηνύματα που βγάζει το Apple Watch και η εφαρμογή, ώστε να ζητήσει δικαίωμα από τον χρήστη να έχει πρόσβαση στα δεδομένα υγείας, όσο και τον κώδικα, ο οποίος το καθορίζει.



Εικόνα 31 Οθόνες άδειας για δεδομένα του HealthKit

```swift
let healthKitTypes: Set = [
    // access step count
    HKObjectType.quantityType(forIdentifier: HKQuantityTypeIdentifier.stepCount)!,
    HKObjectType.quantityType(forIdentifier: HKQuantityTypeIdentifier.walkingSpeed)!,
    HKObjectType.quantityType(forIdentifier: HKQuantityTypeIdentifier.oxygenSaturation)!,
    HKObjectType.quantityType(forIdentifier: HKQuantityTypeIdentifier.heartRateVariabilitySDNN)!
]
healthStore.requestAuthorization(toShare: nil, read: healthKitTypes) { (success, error) -> Void in
    guard success else {
        print("not allowed")
        return
    }
}
```

Εικόνα 32 Κώδικας για ερώτηση άδειας δεδομένων HealthKit

Δ.2.6 Δημιουργία ειδοποιήσεων και σύνδεση με το Firebase Cloud Messaging

Αφού έχει η εφαρμογή το δικαίωμα να στέλνει ειδοποιήσεις στον χρήστη, τότε είναι ρυθμισμένη να στέλνει ειδοποίηση στον χρήστη κάθε 6 ώρες, με το σκεπτικό ότι έτσι θα παίρνει τρεις μετρήσεις του HRV τη μέρα. Φυσικά, αυτό είναι κάτι που μπορεί να αλλάξει πανεύκολα και να ορίζεται με μία μεταβλητή στη Firebase Realtime Database από τον γιατρό για τον κάθε ασθενή ξεχωριστά. Αυτό καθίσταται δυνατό από την συνάρτηση UNNotificationRequest. Ο κώδικας φαίνεται παρακάτω:

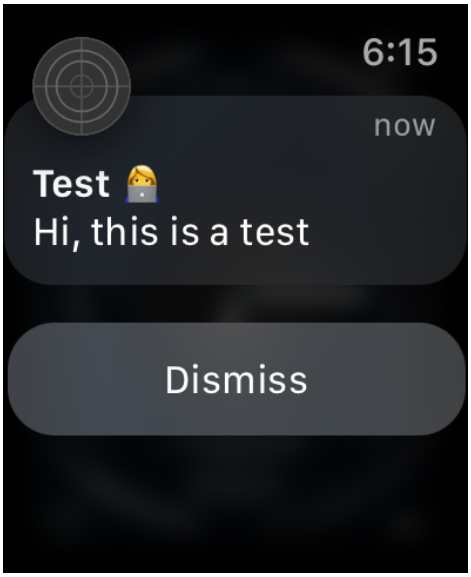
```swift
let center = UNUserNotificationCenter.current()

let content = UNMutableNotificationContent()
content.title = "Don't forget to see your HRV value"
content.body = "Lots of text"
//content.sound = UNNotificationSound.default()
content.categoryIdentifier = "alarm"
content.userInfo = ["example": "information"] // You can retrieve this when displaying notification

// Setup trigger time
//var calendar = Calendar.current
//calendar.timeZone = TimeZone.current
let date = Date()
let triggerDate = date + 21600
let datcomp = Calendar.current.dateComponents([.calendar, .year, .month, .day, .hour, .minute,
    .second], from: triggerDate)
//var currentTime = Date()
//var compareTime = Date().addingTimeInterval(21600)// 6 hours
let trigger = UNCalendarNotificationTrigger(dateMatching: datcomp, repeats: false)

// Create request
let uniqueID = UUID().uuidString // Keep a record of this if necessary
let request = UNNotificationRequest(identifier: uniqueID, content: content, trigger: trigger)
center.add(request) // Add the notification request
```

Εικόνα 33 Κώδικας για προγραμματισμό ειδοποιήσεων

Επίσης, η εφαρμογή διαθέτει δυνατότητα για push notifications, τα οποία μπορεί να τα στείλει ο γιατρός από απόσταση οποιαδήποτε στιγμή το θεωρεί σκόπιμο. Αυτό επιτυγχάνεται μέσω του notification server του Firebase Messaging και της αποθήκευσης του fcmToken στη Realtime Database στο χώρο του χρήστη. Δυστυχώς, κατά τη διάρκεια ανάπτυξης της διπλωματικής αυτής εργασίας, η Apple κατάργησε μία δυνατότητα στα sockets του λειτουργικού και χάλασε η υποστήριξη της αποστολής δεδομένων στην Firebase Realtime Database από την έτοιμη βιβλιοθήκη. Για αυτόν τον λόγο, χρησιμοποιήθηκε το REST API της Firebase σε συνδυασμό με το Alamofire, για να είναι εύκολη η αποστολή των δεδομένων. Ο κώδικας και για τα δύο, όπως και μία ειδοποίηση φαίνονται παρακάτω:

```
extension ExtensionDelegate: UNUserNotificationCenterDelegate {
    func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification:
        UNNotification, withCompletionHandler completionHandler: @escaping
        (UNNotificationPresentationOptions) -> Void) {
        completionHandler([[.badge, .sound]])
    }

    func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive response:
        UNNotificationResponse, withCompletionHandler completionHandler: @escaping () -> Void) {
        completionHandler()
    }
}

extension ExtensionDelegate: MessagingDelegate {
    func messaging(_ messaging: Messaging, didReceiveRegistrationToken fcmToken: String?) {
        print("SUCCESS: Firebase registration token received!")
        print(String(describing: fcmToken))

        UserDefaults.standard.set(fcmToken, forKey: "fcmToken")
    }
}
```

Εικόνα 34 Κώδικας για επικοινωνία με Firebase Messaging



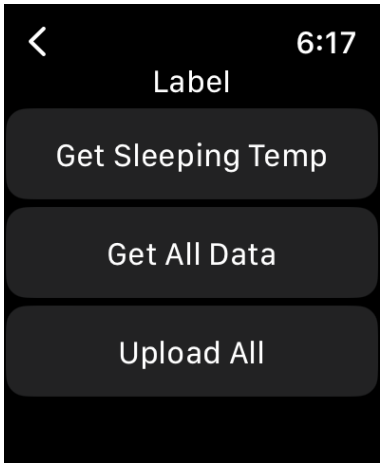
Εικόνα 35 Μια δοκιμαστική ειδοποίηση

```swift
let token = UserDefaults.standard.string(forKey: "fcmToken")
let params: Parameters = [
    "notificationtoken": token!
]
let user = UserDefaults.standard.string(forKey: "email")
let userfree = user!.replacingOccurrences(of: ".", with: "", options: NSString.CompareOptions.literal, range: nil)

AF.request("https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/\(userfree).json", method: .patch,
    parameters: params, encoding: JSONEncoding.default, headers: nil).validate(statusCode: 200 ..< 299).responseData {
    response in
    switch response.result {
        case .success(let data):
            do {
                guard let jsonObject = try JSONSerialization.jsonObject(with: data) as? [String: Any] else {
                    print("Error: Cannot convert data to JSON object")
                    return
                }
                guard let prettyJsonData = try? JSONSerialization.data(withJSONObject: jsonObject, options:
                    .prettyPrinted) else {
                    print("Error: Cannot convert JSON object to Pretty JSON data")
                    return
                }
                guard let prettyPrintedJson = String(data: prettyJsonData, encoding: .utf8) else {
                    print("Error: Could print JSON in String")
                    return
                }

                print(prettyPrintedJson)
            } catch {
                print("Error: Trying to convert JSON data to string")
                return
            }
        case .failure(let error):
            print(error)
    }
}
```

Εικόνα 36 Κώδικας για αποστολή FCM Token στο Firebase Realtime Database

Δ.2.7 Λήψη των μετρήσεων του HealthKit και αποστολή στην Firebase Realtime Database

Δεδομένου ότι η εφαρμογή έχει το δικαίωμα από τον χρήστη της πρόσβασης στη βάση δεδομένων του HealthKit, ο προγραμματιστής έχει τη δυνατότητα να στείλει query σε αυτό και να αντλήσει τιμές, μαζί με το ιστορικό τους, έτοιμων μετρήσεων που κάνει το ρολόι όλο το 24ωρο με αλγορίθμους της Apple. Δυστυχώς, η Apple δεν παρέχει άμεση πρόσβαση στους αισθητήρες εκτός από τον αισθητήρα καρδιακών παλμών. Οπότε, γι' αυτό είμαστε αναγκασμένοι να παίρνουμε έτοιμες τιμές από το HealthKit, το οποίο δεν είναι απαραίτητα κακό, εφόσον δεν βάζουμε τον χρήστη να χρησιμοποιεί την εφαρμογή πολλή ώρα, αλλά μόνο του ζητάμε να πατήσει ένα κουμπί ώστε να σταλθούν τα δεδομένα στη Realtime Database.



Εικόνα 37 Οθόνη αποστολής δεδομένων HealthKit στην βάση

Για να γίνει η εξαγωγή ενός τύπου δεδομένων από τη βάση του HealthKit, είναι απαραίτητη η εκτέλεση ενός query, στο οποίο ορίζουμε από πότε θέλουμε να ξεκινάνε οι μετρήσεις που μας δίνονται, τον τύπο του δεδομένου υγείας που ζητάμε και με τι τρόπο τα θέλουμε ταξινομημένα, όπως φαίνεται παρακάτω:

```swift
func getWalkingSpeed(completion: @escaping (Result<[HKSample]?, Error>) -> Void) {

    // 1
    let start = Calendar.current.startOfDay(for: Date())

    // 2
    let datePredicate = HKQuery.predicateForSamples(withStart: start, end: Date(), options: [])

    // 3
    let walkSpeedType = HKSampleType.quantityType(forIdentifier: .walkingSpeed)!
    let sortByStartDate = NSSortDescriptor(key: HKSampleSortIdentifierStartDate, ascending: true)

    // 4
    let query = HKSampleQuery(sampleType: walkSpeedType,
                              predicate: datePredicate,
                              limit: HKObjectQueryNoLimit,
                              sortDescriptors: [sortByStartDate]) { (_, samples, error) in
        completion(.success(samples))
    }


    // 5
    healthStore.execute(query)

}
```

Εικόνα 38 Κώδικας για εξαγωγή ταχύτητας βηματισμού από το HealthKit

Παρακάτω φαίνεται η συνάρτηση με την οποία γίνονται όλα τα get από τη βάση του HealthKit ταυτόχρονα, ώστε να μην χρειάζεται ο χρήστης να πατάει πολλά κουμπιά για να σταλθούν τα δεδομένα:

```swift
@IBAction func getAll(_ sender: Any) {

    getWalkingSpeed { (result) in
        print("\(result)")
        DispatchQueue.main.async {
            self.walkingSpeed.setText("\(result)")
            self.walkingspeed = "\(result)"
            //walkingSpeed = result
        }
    }
    getHRV { (result) in
        print("\(result)")
        DispatchQueue.main.async {
            self.hrv.setText("\(result)")
            self.HRV = "\(result)"
        }
    }
    getVo2max { (result) in
        print("\(result)")
        DispatchQueue.main.async {
            self.vo2max.setText("\(result)")
            self.vo2maxx = "\(result)"
        }
    }
    getSleepingWristTemperature { (result) in
        print("\(result)")
        DispatchQueue.main.async {
            self.sleeptemp.setText("\(result)")
            self.SleepingWristTemperature = "\(result)"
        }
    }
}
```

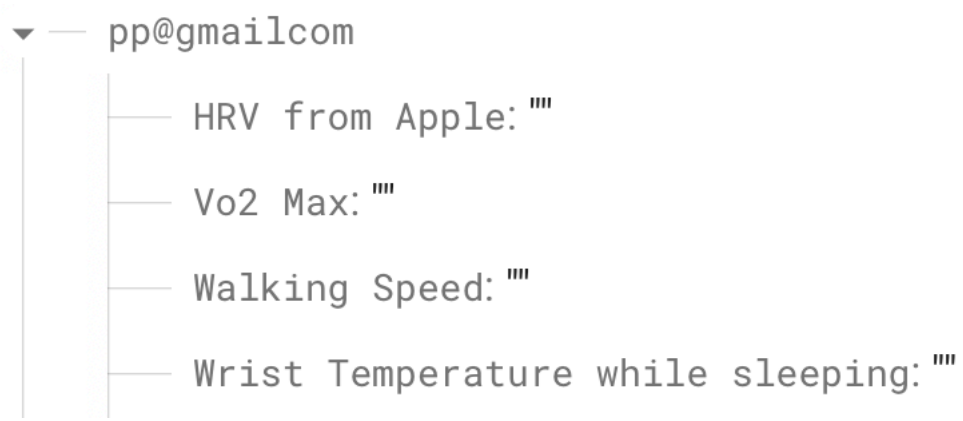
Εικόνα 39 Κώδικας για την λήψη πολλαπλών δεδομένων από το HealthKit

Για την αποστολή των δεδομένων στην Firebase Realtime Database αρχικά μετατράπηκαν σε μορφή String τα δεδομένα που μας έδωσε το HealthKit. Στη συνέχεια, έγινε ξανά χρήση του REST API, αφού η βιβλιοθήκη του Firebase δεν έχει διορθωθεί ακόμη και ορίστηκαν οι κατάλληλες παράμετροι όπως φαίνεται παρακάτω:

```
@IBAction func UploadAll(_ sender: Any) {
    let params: Parameters = [
        "Walking Speed": walkingspeed,
        "HRV from Apple": HRV,
        "Vo2 Max": vo2maxx,
        "Wrist Temperature while sleeping": SleepingWristTemperature
    ]
    let user = MyVariables.yourVariable
    let userfree = user.replacingOccurrences(of: ".", with: "", options: NSString.CompareOptions.literal, range: nil)

    print("https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/\(userfree).json")
    AF.request("https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/\(userfree).json", method: .patch,
        parameters: params, encoding: JSONEncoding.default, headers: nil).validate(statusCode: 200 ..< 299).responseData {
        response in
        switch response.result {
            case .success(let data):
                do {
                    guard let jsonObject = try JSONSerialization.jsonObject(with: data) as? [String: Any] else {
                        print("Error: Cannot convert data to JSON object")
                        return
                    }
                    guard let prettyJsonData = try? JSONSerialization.data(withJSONObject: jsonObject, options:
                        .prettyPrinted) else {
                        print("Error: Cannot convert JSON object to Pretty JSON data")
                        return
                    }
                    guard let prettyPrintedJson = String(data: prettyJsonData, encoding: .utf8) else {
                        print("Error: Could print JSON in String")
                        return
                    }

                    print(prettyPrintedJson)
                } catch {
                    print("Error: Trying to convert JSON data to string")
                    return
                }
            case .failure(let error):
                print(error)
        }
    }
}
```

Εικόνα 40 Κώδικας για αποστολή των δεδομένων του HealthKit στην Firebase Realtime Database

Αφού τα λάβει η Firebase Realtime Database, τα δεδομένα αποθηκεύονται στις θέσεις που φαίνονται στην εικόνα παρακάτω:



Εικόνα 41 Μορφή της Firebase Realtime Database

Δ.2.8 Υπολογισμός HRV από το ηλεκτροκαρδιογράφημα και αποστολή στο Firebase

Αφού η εφαρμογή έχει το δικαίωμα της πρόσβασης στα ηλεκτροκαρδιογραφήματα που είναι αποθηκευμένα στην βάση του HealthKit, ο στόχος είναι να αποθηκευτούν οι τιμές του καρδιογραφήματος σε έναν πίνακα, ώστε να γίνει πιο εύκολη η επεξεργασία τους. Ο κώδικας που βρίσκεται παρακάτω παίρνει όλες τις τιμές και από τα 10 καρδιογραφήματα (όπως έχει εξηγηθεί σε προηγούμενο κεφάλαιο) και τις τοποθετεί με τη σειρά στον πίνακα "hhrvv".

```
let voltageQuery = HKElectrocardiogramQuery(ecgSample) { (query, result) in
    switch(result) {  Switch covers known cases, but 'HKElectrocardiogramQuery.Result' may have additional unkno

    case .measurement(let measurement):
        if let voltageQuantity = measurement.quantity(for: .appleWatchSimilarToLeadI) {
            // Do something with the voltage quantity here.
            DispatchQueue.main.async {
                hhrvv.append(Double((voltageQuantity.doubleValue(for: HKUnit(from: "mcV")))))
            }
            ne += 1
            print("Analyzing voltage measurement: ")
            print(voltageQuantity)
```

Εικόνα 42 Κώδικας για εξαγωγή τιμών καρδιογραφήματος

Αυτές οι τιμές γράφονται σε αρχείο και αποστέλλονται στο Firebase Storage, έτσι ώστε να μπορούμε να τις αναπαραστήσουμε σε γράφημα, το οποίο θα μπορεί να δει ο χρήστης στην κεντρική εφαρμογή. Το πρώτο στάδιο της εγγραφής των τιμών σε αρχείο φαίνεται στην παρακάτω εικόνα:

```
@IBAction func uploadfinalsample () {
    let url = getDocumentsDirectory().appendingPathComponent("test.txt")
    enum FileWriteError: Error {
        case directoryDoesntExist
        case convertToDataIssue
    }
    func write(_ text: String) throws {
        guard let data = text.data(using: .utf8) else {
            throw FileWriteError.convertToDataIssue
        }


        if let fileHandle = FileHandle(forWritingAtPath: url.path) {
            fileHandle.seekToEndOfFile()
            fileHandle.write(data)
        } else {

            try text.write(to: url, atomically: false, encoding: .utf8)
            if let fileHandle = FileHandle(forWritingAtPath: url.path) {
                fileHandle.seekToEndOfFile()
            }
        }
    }
    func getDocumentsDirectory() -> URL {
        // find all possible documents directories for this user
        let paths = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask)

        // just send back the first one, which ought to be the only one
        return paths[0]
    }
        //let url = getDocumentsDirectory().appendingPathComponent("test.txt")
        let numsString = hhrvv.map { String($0) }.joined(separator: ", ")
    do {
        try write(numsString)
    }
    catch {
        print("ne")
    }
        self.uploadText(localFile: url)
        print(hhrvv)

}
```

Εικόνα 43 Κώδικας για εγγραφή τιμών σε αρχείο

Η αποστολή στο Firebase γίνεται μέσω της βιβλιοθήκης FirebaseStorage και συμβαίνει σε αυτό το κομμάτι του κώδικα:

```
func uploadText(localFile: URL) {
    print("llekinezia")
    a = a.replacingOccurrences(of: ".", with: "", options: NSString.CompareOptions.literal, range: nil)
    let c = a + b
    let metadata = StorageMetadata()
        metadata.contentType = "text/csv"
        let riversRef = Storage.storage().reference().child(c).child("test.csv")
        do {
            let audioData = try Data(contentsOf: localFile)
            riversRef.putData(audioData, metadata: metadata){ (data, error) in
                if error == nil{
                    debugPrint("done")
                    riversRef.downloadURL {url, error in
                        guard let downloadURL = url else { return }
                        debugPrint("error url", downloadURL)
                    }
                }
                else {
                    if let error = error?.localizedDescription{
                        debugPrint("error", error)
                    }
                    else {
                        debugPrint("error")
                    }
                }
            }
        } catch {
            debugPrint(error.localizedDescription)
        }
}
```

Εικόνα 44 Κώδικας για αποστολή των τιμών του καρδιογραφήματος στο Firebase Storage

Στη συνέχεια, για να βρεθεί το HRV, πρέπει να βρεθούν οι κορυφές του ηλεκτροκαρδιογραφήματος. Πρώτα φτιάχνουμε έναν κώδικα, που είναι σχεδιασμένος για να βρίσκει κορυφές γενικότερα:

```
func isPeak(arr: [Double], n: Int, num: Double, i: Int, j: Int) -> Bool {

        // If num is smaller than the element
        // on the left (if exists)
        if (i >= 0 && arr[i] > num) {
            return false;
        }

        // If num is smaller than the element
        // on the right (if exists)
        if (j < n && arr[j] > num) {
            return false;
        }
        return true;
}
```

Εικόνα 45 Κώδικας για εύρεση μεγίστων στο καρδιογράφημα

Όμως, ένα καρδιογράφημα έχει πολλά τοπικά μέγιστα, λόγω της μορφής του. Αφού εκτελέστηκαν πολλά καρδιογραφήματα, διαπιστώθηκε, ότι δεν υπάρχουν πολλαπλά μέγιστα μετά την τιμή των 450 mcV και ότι οι μέγιστες τιμές είναι πάντα πολύ υψηλότερες από το

450. Οπότε, αν βρεθούν τα μέγιστα, που είναι πάνω από τα 450 mcV, θα έχουμε σίγουρα τα peaks του ηλεκτροκαρδιογραφήματος και θα μπορούμε να υπολογίσουμε την απόσταση τους, ώστε να βγει το HRV, σύμφωνα πάντα και με την περίοδο δειγματοληψίας. Ο κώδικας για την διαδικασία που περιγράφηκε φαίνεται παρακάτω:

```
while j < hhrvv.count
{

    // If the current element is a peak
    if (isPeak(arr: hhrvv, n: hhrvv.count, num: hhrvv[j], i: j - 1, j: j + 1))
    {
        if (hhrvv[j] > 450) {
            maximas.append(hhrvv[j])
            positions.append(j)
            print(hhrvv[j])
            print(" ")
        }
    }
    j = j+1
}
print("")
print(positions)
var rrintervals : [Double] = []
let o : Int = hhrvv.count
let o1 : Double = (30.0/Double(o))*1000.0
print("o1")
print(o1)
var m : Double = 0
for k in 0...(positions.count - 2) {
    rrintervals.append(Double(positions[k+1])*o1 - Double(positions[k])*o1)
}
for k in 0...(rrintervals.count - 1) {
    m = m + rrintervals[k]
}
let mfinal = m/Double(rrintervals.count)
let N = Double(rrintervals.count)
var n : Double = 0
for k in 0...(rrintervals.count -  1) {
    n = n + (rrintervals[k] - mfinal)*(rrintervals[k] - mfinal)
}
let presdnn = n/N
let sdnn = sqrt(presdnn)
print("hiii again i have the hrvSDNN for you")
print(sdnn)
hrvsdnn = sdnn
```

Εικόνα 46 Κώδικας για υπολογισμό του HRVSDNN

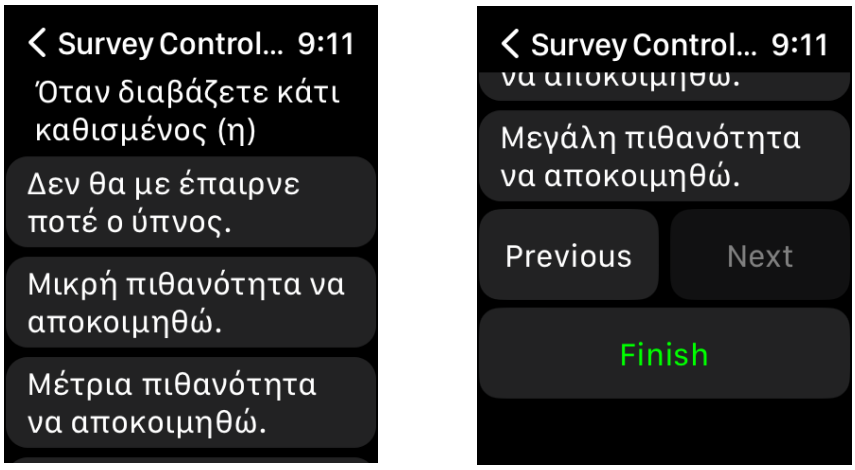
Στο τέλος, στέλνουμε τη μέτρηση του HRV στην Firebase Realtime Database με τον τρόπο ο οποίος έχει αναφερθεί και παραπάνω:

```
let params: Parameters = [
    "hrv": hrvsdnn
]
let user = MyVariables.yourVariable
let userfree = user.replacingOccurrences(of: ".", with: "", options: NSString.CompareOptions.literal, range: nil)

print("https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/\(userfree).json")
AF.request("https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/\(userfree).json", method: .patch, parameters: params,
    encoding: JSONEncoding.default, headers: nil).validate(statusCode: 200 ..< 299).responseData { response in
    switch response.result {
        case .success(let data):
            do {
                guard let jsonObject = try JSONSerialization.jsonObject(with: data) as? [String: Any] else {
                    print("Error: Cannot convert data to JSON object")
                    return
                }
                guard let prettyJsonData = try? JSONSerialization.data(withJSONObject: jsonObject, options: .prettyPrinted) else {
                    print("Error: Cannot convert JSON object to Pretty JSON data")
                    return
                }
                guard let prettyPrintedJson = String(data: prettyJsonData, encoding: .utf8) else {
                    print("Error: Could print JSON in String")
                    return
                }
```

Εικόνα 47 Κώδικας για αποστολή του HRVSDNN στην Firebase Realtime Database

Δ.2.9 Ερωτηματολόγιο

Η εφαρμογή, όπως έχει αναφερθεί και προηγουμένως, έχει την δυνατότητα να εμφανίζει ερωτηματολόγια, τα οποία έχουν ανεβάσει και επιλέξει οι γιατροί στους χρήστες με τη μορφή που φαίνεται στις παρακάτω εικόνες. Αυτά τα ερωτηματολόγια είναι ανεβασμένα στο Firebase Remote Config σε μορφή JSON.



Εικόνα 48 Οθόνες ερωτηματολογίου

Αυτό καθίσταται δυνατό μέσω μίας σειράς από διαδικασίες. Αρχικά, παίρνουμε από την Realtime Database την επιλογή που έχει κάνει ο γιατρός για τον συγκεκριμένο χρήστη. Έπειτα, κατεβάζουμε από το Remote Config, με τη βοήθεια της βιβλιοθήκης FirebaseRemoteConfig, το JSON, όπως φαίνεται παρακάτω:

```
self.remoteConfig.activate() { (changed, error) in
    //try to get survey from
    //let surveyRemoteJsonkey : String = "ne"

    if let remoteSurveyJsonString = self.remoteConfig[surveyRemoteJsonkey].stringValue {
        //if let surveyFromRemote : Survey = Survey.getSurveyFromString(jsonString:
            remoteSurveyJsonString) {

          DispatchQueue.main.async {

              print(remoteSurveyJsonString)
              //presurveyResponse = remoteSurveyJsonString
              //let surveyResponse = remoteSurveyJsonString.toJSON() as? String
              //print(surveyResponse!)
              let genreData = Data(remoteSurveyJsonString.utf8)
              do { try self.ne = JSONDecoder().decode(Survey.self, from: genreData)
              }
              catch {
                  print("ne")
              }
              self.maxQuestions = self.ne.questions[self.currentQuestion].selections.count
              //print(items1)
              self.loadSurvey(currentq: self.currentQuestion)
          }

        // } else {
          //  print("remote survey not loaded")

         //}

    }
}
```

Εικόνα 49 Κώδικας για λήψη αριθμού ερωτηματολογίου για τον συγκεκριμένο χρήστη

Το JSON αποκωδικοποιείται μέσω της JSONDecoder().decode, στην οποία έχει δοθεί η γενική μορφή που θα έχουν τα JSON:

```
struct Survey: Codable {
    let questions: [Questions]
}

struct Questions: Codable {
    let title : String
    let selections : Array<String>
}
extension Questions{
    enum CodingKeys: String, CodingKey {
        case title
        case selections
    }

    init(from decoder: Decoder) throws {
        let values = try decoder.container(keyedBy: CodingKeys.self)
        title = try values.decode(String.self, forKey: .title)
        selections = try values.decode(Array<String>.self, forKey: .selections)
    }
}
```

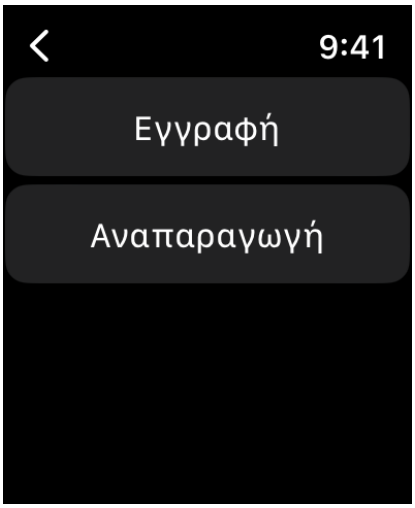
Εικόνα 50 Κώδικας για γενική μορφή JSON ερωτηματολογίου

Τέλος, τα αποτελέσματα γράφονται σε ένα αρχείο txt και στέλνονται στο Firebase Storage:

```
func write(_ text: String) throws {
    enum FileWriteError: Error {
        case directoryDoesntExist
        case convertToDataIssue
    }

    let url = getDocumentsDirectory().appendingPathComponent("survey.txt")

    guard let data = text.data(using: .utf8) else {
        throw FileWriteError.convertToDataIssue
    }


    if let fileHandle = FileHandle(forWritingAtPath: url.path) {
        fileHandle.seekToEndOfFile()
        fileHandle.write(data)
    } else {
        print("kin")
        try text.write(to: url, atomically: false, encoding: .ascii)
        if let fileHandle = FileHandle(forWritingAtPath: url.path) {
            fileHandle.seekToEndOfFile()
        }
    }
}

func writesur() {
    let url = getDocumentsDirectory().appendingPathComponent("survey.txt")

    do {
        try FileManager.default.removeItem(at: url)
        print("tiakanss")
    } catch {
        print("tiakans")
    }
    for i in 0...(ne.questions.count-1) {
        let nee = UserDefaults.standard.integer(forKey: "\(ne.questions[i].title)")
        do {
            try write("\(ne.questions[i].title) -> \(nee)")
        } catch { print("ne")
        }
    }

}
```

Εικόνα 51 Αποστολή αποτελεσμάτων ερωτηματολογίου στο Firebase Storage

Δ.2.10 Καταγραφή ανάσας



Εικόνα 52 Οθόνη καταγραφής ανάσας

Όταν ο χρήστης θέλει να κάνει καταγραφή της ανάσας του, ώστε να τη στείλει στο γιατρό, πατώντας το κουμπί της εγγραφής, δημιουργείται αυτόματα ένα αρχείο mp4, ανοίγει το μικρόφωνο και αρχίζει η καταγραφή. Αφού τελειώσει και πατήσει αποθήκευση, στέλνεται αυτόματα στην Firebase Storage και στην κεντρική εφαρμογή, όπου από εκεί έχει πρόσβαση ο γιατρός για να ακούσει την ανάσα του ασθενή. Επίσης, ο χρήστης έχει τη δυνατότητα να ακούσει αυτό που κατέγραψε και, αν δεν τον ικανοποιεί, να στείλει νέο δείγμα στον γιατρό, αντικαθιστώντας αυτόματα το προηγούμενο. Ο κώδικας για αυτά είναι απλός και φαίνεται στην παρακάτω εικόνα . Το UI αυτού του μέρους της εφαρμογής φαίνεται στα αριστερά.

```
func recFileURL() -> URL? {

    let fileManager = FileManager.default
    let container = fileManager.containerURL(forSecurityApplicationGroupIdentifier: "group.med")
    if let container = container {
        return container.appendingPathComponent("recording.mp4")
    }
    else {
        return nil
    }
}


// ================================================================================
// MARK: - Actions

@IBAction func recBtnTapped() {

    if let recFileUrl = recFileURL() {

        presentAudioRecorderController(
            withOutputURL: recFileUrl,
            preset: WKAudioRecorderPreset.highQualityAudio,
            options: nil,
            completion:
            { (didSave, error) -> Void in
                print("error:\(String(describing: error))\n")
            }
        )
        uploadSound(localFile: recFileUrl)
    }
}

@IBAction func playBtnTapped() {

    if let recFileUrl = recFileURL() {

        presentMediaPlayerController(
            with: recFileUrl,
            options: nil) { (didPlayToEnd, endTime, error) -> Void in
            }
    }
}
```

Εικόνα 53 Κώδικας για καταγραφή ήχου/ανάσας

Ο τρόπος με τον οποίο στέλνεται το αρχείο mp4 στο Firebase Storage είναι ακριβώς ο ίδιος με το txt του ηλεκτροκαρδιογραφήματος, για αυτόν τον λόγο, δεν θα παρουσιάσω τον κώδικά του.

Δ.2.11 Άμεση πρόσβαση στο βηματομετρητή, στο επιταχυνσιόμετρο και στον αισθητήρα μαγνητισμού

Μία από τις λειτουργίες της εφαρμογής είναι η πρόσβαση στα πραγματικά δεδομένα διάφορων αισθητήρων του ρολογιού. Ας σημειωθεί ότι, προς το παρόν οι συγκεκριμένες λειτουργίες δεν έχουν κάποια καθαρή χρήση στην εφαρμογή, αλλά ίσως στο μέλλον θα μπορούσαν να έχουν μετά από την ανάλογη έρευνα.

Ο πρώτος αισθητήρας είναι ο αισθητήρας βημάτων. Η Apple μας δίνει την πρόσβαση σε αυτόν μέσω της βιβλιοθήκης CoreMotion και του CMPedometer. Όπως φαίνεται και στον κώδικα παρακάτω, αυτή η συνάρτηση μας δίνει, χωρίς τη χρήση GPS, αριθμό βημάτων, απόσταση και τον αριθμό των σκαλοπατιών που ανεβήκαμε ή κατεβήκαμε.

```
@IBOutlet weak var labelSteps: WKInterfaceLabel!
@IBOutlet weak var labelDistance: WKInterfaceLabel!
@IBOutlet weak var labelAscended: WKInterfaceLabel!
@IBOutlet weak var labelDescended: WKInterfaceLabel!

let pedometer = CMPedometer()

//private let database = Database.database().reference()

override func awake(withContext context: Any?) {
    super.awake(withContext: context)
}

override func willActivate() {
    super.willActivate()

    if CMPedometer.isPaceAvailable() {
        pedometer.startUpdates(from: Date()) { (pedometerData, error) -> Void in

            if let pedometerData = pedometerData {

                let steps = pedometerData.numberOfSteps.uintValue
                self.labelSteps.setText(String(format: "%lu", steps))

                if let distance = pedometerData.distance {
                    self.labelDistance.setText(String(format: "%lu", distance.uintValue))
                   /* let object: [String: Any] = [
                        "distance": distance
                    ]*/
                }

                if let floorsAscended = pedometerData.floorsAscended {
                    self.labelAscended.setText(String(format: "%lu", floorsAscended.uintValue))
                    /*let object: [String: Any] = [
                        "ascended": floorsAscended
                    ]*/
                    //self.database.child("pedometer").setValue(object)
                }

                if let floorsDescended = pedometerData.floorsDescended {
                    self.labelDescended.setText(String(format: "%lu", floorsDescended.uintValue))
                    /*let object: [String: Any] = [
                        "descended": floorsDescended
                    ]*/
                    //self.database.child("pedometer").setValue(object)
                }
```

Εικόνα 54 Κώδικας για την άντληση δεδομένων μέσω του CMPedometer

Ο δεύτερος αισθητήρας είναι το επιταχυνσιόμετρο. Η Apple μας δίνει την πρόσβαση σε αυτόν ξανά μέσω της βιβλιοθήκης CoreMotion, αλλά του CMMotionManager αυτή τη φορά. Όπως φαίνεται και στον κώδικα παρακάτω, αυτή η συνάρτηση μας δίνει την επιτάχυνση στους 3 άξονες x,y και z και μας παρέχει τη δυνατότητα να ορίσουμε τον ρυθμό ανανέωσης με τον οποίο θέλουμε να μας έρχονται τα δεδομένα.

```
@IBOutlet weak var labelX: WKInterfaceLabel!
@IBOutlet weak var labelY: WKInterfaceLabel!
@IBOutlet weak var labelZ: WKInterfaceLabel!

let motionManager = CMMotionManager()

//private let database = Database.database().reference()

override func awake(withContext context: Any?) {
    super.awake(withContext: context)

    motionManager.accelerometerUpdateInterval = 0.5
}

override func willActivate() {
    super.willActivate()
    //a = a.replacingOccurrences(of: ".", with: "", options: NSString.CompareOptions.literal, range: nil)
    //let c = a + b
    if motionManager.isAccelerometerAvailable {
        let handler:CMAccelerometerHandler = {(data: CMAccelerometerData?, error: Error?) -> Void in
            self.labelX.setText(String(format: "%.2f", data!.acceleration.x))
            self.labelY.setText(String(format: "%.2f", data!.acceleration.y))
            self.labelZ.setText(String(format: "%.2f", data!.acceleration.z))
            let object: Parameters = [
                "acceleration x": data!.acceleration.x,
                "acceleration y": data!.acceleration.y,
                "acceleration z": data!.acceleration.z
            ]
            //self.database.child(c).setValue(object)
            let user = MyVariables.yourVariable
            let userfree = user.replacingOccurrences(of: ".", with: "", options: NSString.CompareOptions.literal, range: nil)

            print("https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/\(userfree).json")
            AF.request("https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/\(userfree).json", method:
                .patch, parameters: object, encoding: JSONEncoding.default, headers: nil).validate(statusCode: 200 ..<
                299).responseData { response in
```

Εικόνα 55 Κώδικας για άντληση δεδομένων του επιτααχυνσιόμετρου

Τέλος, ο άλλος αισθητήρας είναι ο αισθητήρας μαγνητισμού. Η Apple μας δίνει την πρόσβαση και σε αυτόν μέσω της βιβλιοθήκης CoreMotion και του CMMotionManager. Όπως φαίνεται και στον κώδικα παρακάτω, αυτή η συνάρτηση μας δίνει το μαγνητικό πεδίο και στους 3 άξονες και την ακρίβεια της μέτρησης. Μάλιστα, μας επιτρέπει και σε αυτήν την περίπτωση να ορίσουμε τον ρυθμό ανανέωσης με τον οποίο προτιμούμε να μας έρχονται τα δεδομένα.

```
@IBOutlet weak var labelX: WKInterfaceLabel!
@IBOutlet weak var labelY: WKInterfaceLabel!
@IBOutlet weak var labelZ: WKInterfaceLabel!
@IBOutlet weak var labelE: WKInterfaceLabel!
@IBOutlet weak var labelW: WKInterfaceLabel!

let motionManager = CMMotionManager()

//private let database = Database.database().reference()

override func awake(withContext context: Any?) {
    super.awake(withContext: context)
    motionManager.accelerometerUpdateInterval = 1
    motionManager.gyroUpdateInterval = 1
    motionManager.magnetometerUpdateInterval = 1
    motionManager.deviceMotionUpdateInterval = 1
}

override func willActivate() {
    super.willActivate()
    //a = a.replacingOccurrences(of: ".", with: "", options: NSString.CompareOptions.literal, range: nil)
    //let c = a + b
    //print(c)
    motionManager.startDeviceMotionUpdates(to: OperationQueue.current!) { (data, error) in
                guard error == nil else { return }
        self.labelX.setText(String(format: "%.2f", data!.magneticField.field.x))
        self.labelY.setText(String(format: "%.2f", data!.magneticField.field.y))
        self.labelZ.setText(String(format: "%.2f", data!.magneticField.field.z))
        self.labelE.setText(String(format: "%.2f", data!.magneticField.accuracy.rawValue))
```

Εικόνα 56 Κώδικας για άντληση δεδομένων από τον αισθητήρα μαγνητισμού

Δ.3 Κεντρική εφαρμογή – Ιστοσελίδα για χρήστες και ιατρούς

Δ.3.1 Σύνδεση με Firebase

Προκειμένου να πραγματοποιηθεί η σύνδεση της κεντρικής εφαρμογής με την πλατφόρμα Firebase, ακολουθείται μία παρόμοια διαδικασία, σαν αυτή στο έξυπνο ρολόι Apple Watch και στο Xcode. Σε αυτήν την περίπτωση, είναι υποχρεωτική η εισαγωγή ενός κομματιού κώδικα στο αρχείο index.js, το οποίο μας δίνει ο οδηγός του Firebase Hosting κατά τη διάρκεια του set up του. Ο κώδικας φαίνεται στην παρακάτω φωτογραφία.

```
import { initializeApp } from 'firebase/app';
import {
  getAuth,
  onAuthStateChanged,
  signOut,
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  connectAuthEmulator
} from 'firebase/auth';

const firebaseApp = initializeApp({
    apiKey: "AIzaSyCsbm1FC2qnNN0xroAKIBKIAolM23lG880",
    authDomain: "datatest-d81a6.firebaseapp.com",
    databaseURL: "https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app",
    projectId: "datatest-d81a6",
    storageBucket: "datatest-d81a6.appspot.com",
    messagingSenderId: "689218183663",
    appId: "1:689218183663:web:e26d4ca4788460d1c0161a",
    measurementId: "G-4P4K1PN6YE"
});

import { getDatabase, ref, child, get} from "firebase/database";
```

Εικόνα 57 Κώδικας για σύνδεση της κεντρικής εφαρμογής με το Firebase

Επίσης, προκειμένου να υπάρχει η δυνατότητα προσομοίωσης των υποστάσεων του Firebase χωρίς να είναι ανάγκη κάθε φορά να γίνεται deploy, μέσω του npm έγινε εγκατάσταση των firebase-tools. Αυτό εγκατέστησε τους προσομοιωτές τοπικά στο localhost και δημιούργησε μέσα στο project ένα αρχείο με ονομασία “firebase.json”, το οποίο μας δίνει τη δυνατότητα να μπορούμε να κάνουμε configure τον προσομοιωτή, όπως ακριβώς θέλουμε εμείς. Η μορφή του ακολουθεί αμέσως παρακάτω:

```
{
  "database": {
    "rules": "database.rules.json"
  },
  "emulators": {
    "hosting": {
      "host": "localhost",
      "port": "5005"
    },
    "auth": {
      "port": 9099
    },
    "database": {
      "port": 9001
    },
    "ui": {
      "enabled": true
    }
  },
  "functions": [
    {
      "source": "functions",
      "codebase": "default",
      "ignore": [
        "node_modules",
        ".git",
        "firebase-debug.log",
        "firebase-debug.*.log"
      ]
    }
  ],
  "hosting": [
    {
      "target": "user",
      "public": "dist/user",
      "ignore": [
        "firebase.json",
        "**/.*",
        "**/node_modules/**"
      ]
    },
    {
      "target": "admin",
      "public": "dist/admin",
      "ignore": [
        "firebase.json",
        "**/.*",
        "**/node_modules/**"
      ]
    }
  ]
}
```

Εικόνα 58 Κώδικας για το αρχείο firebase.json

Τέλος, επειδή η κεντρική εφαρμογή αποτελείται και από εφαρμογή για τους χρήστες αλλά και για τους γιατρούς, χρειάστηκαν 2 ιστοσελίδες, οπότε και 2 domains. Αυτό διακρίνεται στο παρακάτω αρχείο:

```
{
  "projects": {
    "default": "datatest-d81a6"
  },
  "targets": {
    "datatest-d81a6": {
      "hosting": {
        "user": [
          "datatest-d81a6"
        ],
        "admin": [
          "datatestadmin"
        ]
      }
    }
  },
  "etags": {}
}
```

Εικόνα 59 Κώδικας για να έχουμε δύο σελίδες

Δ.3.2 Εγγραφή και σύνδεση χρηστών και γιατρών

```
module.exports = {
  mode: 'development',
  devtool: 'eval-source-map',
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist/admin'),
    filename: 'bundle.js'
  },
  watch: true,
  module: {
    rules: [
      {
        test: /\.css$/,
        use: [
          'style-loader',
          'css-loader'
        ]
      }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: './src/index.html'
    })
  ],
}
```

Εικόνα 60 Κώδικας για το configuration του webpack

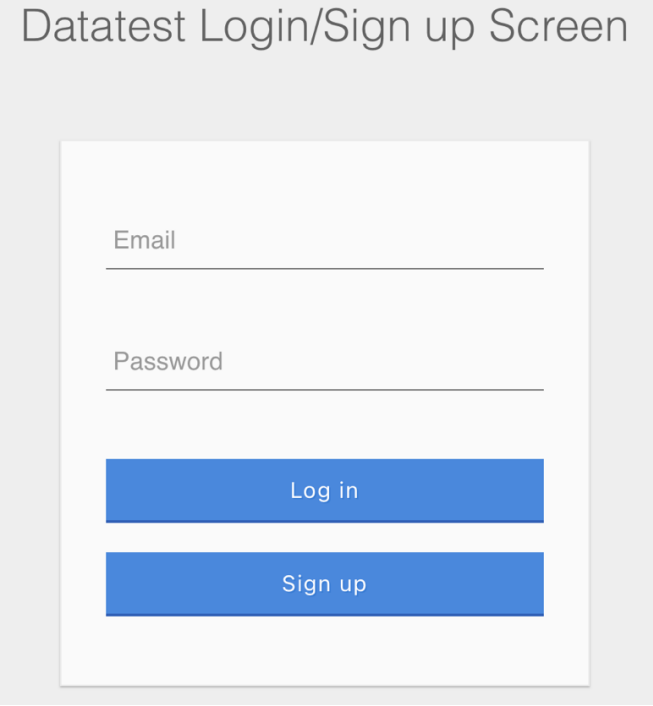
Για την εγγραφή και σύνδεση χρηστών και γιατρών, η γλώσσα που ακολουθήθηκε για την ανάπτυξη του κώδικα είναι η JavaScript, εμφωλευμένη στον κώδικα της HTML. Για την ανάπτυξη της σελίδας αυθεντικοποίησης, η ίδια η Google έδινε έτοιμο UI το οποίο και είναι πιο αξιόπιστο και βοήθησε στον χρόνο ανάπτυξης όλης της διπλωματικής εργασίας. Αυτό εισήχθη στην κεντρική εφαρμογή με τη βοήθεια του webpack, το οποίο εγκαταστάθηκε μέσω του npm και το configuration του φαίνεται στην εικόνα αριστερά.

Επειδή, πρέπει να υπάρχει υποστήριξη και για τους απλούς χρήστες αλλά και για τους γιατρούς, υπάρχουν δύο ιστοσελίδες. Ωστόσο και στις δύο περιπτώσεις, η σελίδα αυθεντικοποίησης είναι ακριβώς η ίδια, απλά άλλο το domain. Οπότε, για αυτόν τον λόγο θα παρουσιαστεί μόνο μία σελίδα αυθεντικοποίησης από τις δύο.

Η σελίδα αυθεντικοποίησης είναι σχεδιασμένη απλά και ζητάει μόνο email και κωδικό μέχρι και στην εγγραφή, λόγω του ότι θέλουμε πρώτον, να μην είναι περίπλοκη για τους ανθρώπους μεγαλύτερης ηλικίας και δεύτερον, επειδή θέλουμε η διαδικασία της εγγραφής να είναι εφικτή και μέσω του Apple Watch, το οποίο έχει μικρή οθόνη/πληκτρολόγιο και δεν θα μας δίνει εύκολα τη δυνατότητα να ζητάμε από το χρήστη παραπάνω προσωπικά στοιχεία, όπως τηλέφωνο, ονοματεπώνυμο, διεύθυνση κλπ.. Τόσο ο κώδικας, όσο και το UI φαίνονται παρακάτω:

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Datatest Login/Sign up Screen</title>
  <link rel="stylesheet" href="style.css">
<script defer src="bundle.js"></script><script defer src="bundle.js"></script><script defer src="bundle.js"></script></head>

<body>

  <div id="login">
    <script src = "https://www.gstatic.com/firebasejs/7.15.4/firebase.js"></script>
    <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-firestore.js"></script>
    <div class="header">
      <h1>Datatest Login/Sign up Screen</h1>
    </div>
    <form>
      <div class="group">
        <input id="txtEmail" type="email">
        <label>Email</label>
      </div>
      <div class="group">
        <input id="txtPassword" type="password">
        <label>Password</label>
      </div>
      <div id="divLoginError" class="group">
        <div id="lblLoginErrorMessage" class="errorlabel">Error message</div>
      </div>
      <button id="btnLogin" type="button" class="button buttonBlue">Log in</button>
      <button id="btnSignup" type="button" class="button buttonBlue">Sign up</button>
    </form>
  </div>
```

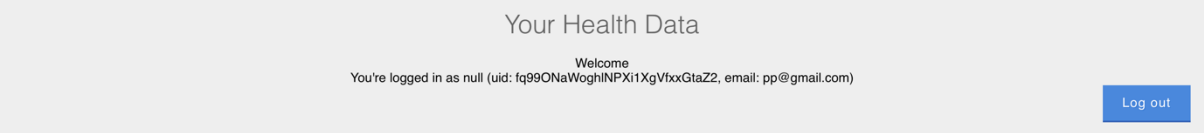
Εικόνα 61 Κώδικας για Sign In



Εικόνα 62 UI κεντρικής εφαρμογής για Sign In/Up

Δ.3.3 Πάνω μέρος κεντρικής εφαρμογής και καλωσόρισμα χρήστη – ιατρού

Στην κεντρική εφαρμογή, που έχουν πρόσβαση τόσο οι χρήστες, όσο και οι γιατροί, υπάρχει πάντα πάνω μια μπάρα, η οποία δείχνει στον χρήστη με ποιο λογαριασμό είναι συνδεδεμένος και του δίνει επιπλέον τη δυνατότητα της αποσύνδεσης. Μάλιστα, δείχνει και το uid, το οποίο είναι χρήσιμο, αν ο χρήστης χρειαστεί κάποια βοήθεια και απευθυνθεί στους διαχειριστές. Τα προαναφερθέντα φαίνονται στην παρακάτω εικόνα:



Εικόνα 63 UI πάνω μέρους

Αυτό επιτυγχάνεται με τη βοήθεια του “lblAuthState” και του “querySelector”, όπως φαίνεται και παρακάτω:

```
export const divAuthState = document.querySelector('#divAuthState')
export const lblAuthState = document.querySelector('#lblAuthState')
```

Εικόνα 64 Κώδικας JavaScript για να βρίσκουμε τον συνδεδεμένο χρήστη

```
<div id="app">
  <div class="header">
    <h1>
      Doctor Administrator Platform
    </h1>
    <div>Patients Health Status</div>
    <div>
      <div>Welcome</div>
      <div id="lblAuthState" class="authlabel">
      </div>
      <buttonout id="btnLogout" type="buttonout" class="buttonout buttonBlue">Log out</buttonout>
    </div>
  </div>
```

Εικόνα 65 Κώδικας HTML για να εμφανίζουμε τον χρήστη

Δ.3.4 Εξαγωγή και παρουσίαση δεδομένων

Η επικοινωνία με τη Firebase Realtime Database είναι απαραίτητη τόσο για την εξαγωγή των δεδομένων των χρηστών, όσο και για την εγγραφή των επιλογών που κάνουν οι γιατροί για κάθε χρήστη. Για να γίνει ανάγνωση δεδομένων, αρχικά πρέπει να πάρουμε το reference object από τη Realtime Database και μετά να παρατάξουμε τα στοιχεία που χρειαζόμαστε σε μορφή πίνακα, με τη βοήθεια της HTML. Το UI και ο κώδικας για τα παραπάνω φαίνονται παρακάτω:

Name	Data
Acceleration x	0.120513916015625
Acceleration y	0.048858642578125
Acceleration z	-0.99395751953125
Gyroscope x	-0.02676706574857235
Gyroscope y	-0.013208694756031036
Gyroscope z	0.05518176034092903
Average Heart Rate in HRV	undefined
HRVSDNN 3 minutes	undefined
HRVSDNN Apple Algorithm	
Vo2 Maximum	
Wrist Temperature while sleeping	

Εικόνα 66 UI εμφάνισης δεδομένων

```
const dbRefObject = firebase.database().ref(uid);
dbRefObject.on('value', snap => {
  console.log(snap.val());
  const myObj = JSON.parse(JSON.stringify(snap.val(), null, 3));
  console.log(myObj['acceleration x']);
  let text = "<p><table border='1'>"
  text += "<tr><td>" + "Name" + "</td><td>" + "Data" + "</td></tr><tr><td>" +
    "Acceleration x" + "</td><td>" + myObj['acceleration x'] + "</td></tr><tr><td>"+
    "Acceleration y" + "</td><td>" + myObj['acceleration y'] + "</td></tr><tr><td>" +
    "Acceleration z" + "</td><td>" + myObj['acceleration z'] + "</td></tr><tr><td>"+
    "Gyroscope x" + "</td><td>" + myObj['gyro x'] + "</td></tr><tr><td>"+
    "Gyroscope y" + "</td><td>" + myObj['gyro y'] + "</td></tr><tr><td>"+
    "Gyroscope z" + "</td><td>" + myObj['gyro z'] + "</td></tr><tr><td>"+
    "Average Heart Rate in HRV" + "</td><td>" + myObj['heart rate'] + "</td></tr><tr><td>"+
    "HRVSDNN 5 minutes" + "</td><td>" + myObj['hrv'] + "</td></tr><tr><td>"+
    "HRVSDNN Apple Algorithm" + "</td><td>" + myObj['HRV from Apple'] + "</td></tr><tr><td>"+
    "Vo2 Maximum" + "</td><td>" + myObj['Vo2 Max'] + "</td></tr><tr><td>"+
    "Wrist Temperature while sleeping" + "</td><td>" + myObj['Wrist Temperature while sleeping'] + "</td></tr>";

  text += "</table></p>"
  sTag.innerHTML = text;
```

Εικόνα 67 Κώδικας για την εμφάνιση δεδομένων του χρήστη

Στη συνέχεια, για τη δυνατότητα ανανέωσης των δεδομένων που βρίσκονται στην Firebase Realtime Database, χρησιμοποιούμε το REST API που μας δίνεται, επειδή μέσω της βιβλιοθήκης και των διαθέσιμων συναρτήσεων, σβηνόντουσαν τα προηγούμενα δεδομένα της βάσης με την εισαγωγή ενός νέου. Με την αποστολή ενός PATCH request, όπως ακριβώς και στο έξυπνο ρολόι, στέλνουμε όποια δεδομένα θέλουμε με ευκολία, όπως φαίνεται και στον κώδικα παρακάτω:

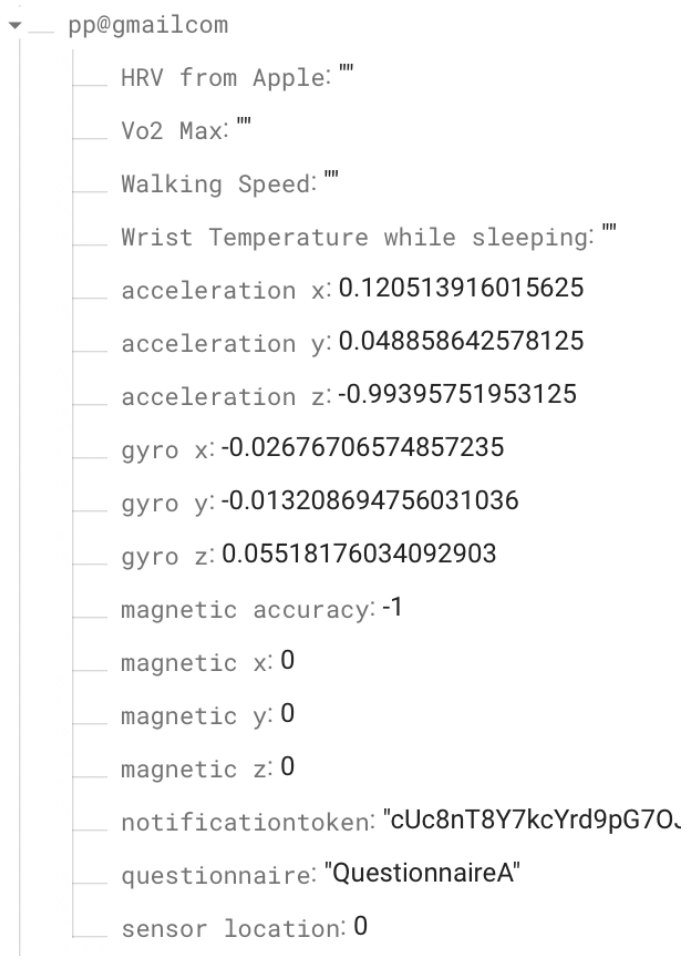
```
function sendSurvey() {
  console.log(surveyy);
  console.log(usersur);
  fetch(`https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/${usersur}.json`, {
    method: 'PATCH',
    body: JSON.stringify({"questionnaire": surveyy}),
    headers: {
    "Content-type": "application/json; charset=UTF-8"
    }
    })
    .then(response => response.json())
    .then(json => console.log(json))
}
```

Εικόνα 68 Κώδικας για αποστολή δεδομένων στην FIrebase Realtime Database

Τέλος, τα δεδομένα στην Firebase Realtime Database έχουν την παρακάτω μορφή:

Εικόνα 69 Μορφή εμφάνισης χρηστών στην Firebase Realtime Database

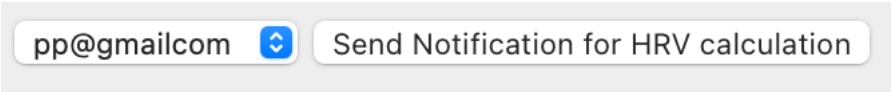
Μάλιστα, μέσα στον χρήστη είναι ως εξής:



Εικόνα 70 Μορφή δεδομένων του κάθε χρήστη στην Firebase Realtime Database

Δ.3.5 Αλληλεπίδραση μεταξύ ιατρού και χρήστη

Αρχικά, η κεντρική εφαρμογή προσφέρει στους γιατρούς τη δυνατότητα να μπορούν να δουν όλα τα δεδομένα ενός χρήστη, όπως φαίνεται και στο προηγούμενο κεφάλαιο. Όμως, έχοντας οι γιατροί δει, για παράδειγμα το HRV, μπορεί να θέλουν να ζητήσουν από τον χρήστη μία νέα μέτρηση, θεωρώντας ότι κάτι δεν πήγε καλά στην συγκεκριμένη. Αυτό γίνεται με το κουμπί “Send Notification for HRV calculation”, το οποίο φαίνεται στην επόμενη εικόνα. Πρώτα δίνεται η δυνατότητα στο γιατρό να επιλέξει ποιον ασθενή θέλει και μετά να του στείλει ειδοποίηση:



Εικόνα 71 UI αποστολής ειδοποιήσεων

Για να μπορούμε να κάνουμε fetch τους χρήστες και να τους βάλουμε μέσα στις επιλογές, αναπτύχθηκε ο εξής κώδικας:

```
var selectTag = document.createElement("SELECT");
selectTag.setAttribute("id","nee");
select.appendChild(selectTag);
var spaceTag = document.createElement("text");
spaceTag.textContent = '  ';
select.appendChild(spaceTag);
var buttonTag = document.createElement("button");
buttonTag.setAttribute("onclick", "sendPush()");
buttonTag.textContent = 'Send Notification for HRV calculation';
select.appendChild(buttonTag);

firebase.database().ref().on('value', function(snapshot) {
        const id = Object.keys(snapshot.val());
        id.forEach(doc => {
          console.log(`${doc}`);
          var el = document.createElement("option");
          el.textContent = doc;
          el.value = doc;
          selectTag.add(el);
      });
});
```

Εικόνα 72 Κώδικας για fetch των χρηστών για αποστολή ειδοποιήσεων

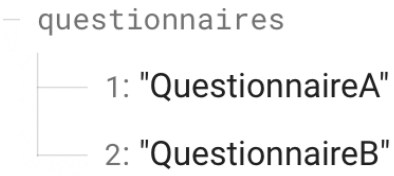
Επίσης, για την αποστολή των ειδοποιήσεων, από τη μεριά της κεντρικής εφαρμογής, τρέχει η παρακάτω συνάρτηση, η οποία καλεί μία Firebase Cloud Function:

```
function sendPush() {
  var xhr = new XMLHttpRequest();
  console.log(nottokk);
  xhr.open('get', `https://us-central1-datatest-d81a6.cloudfunctions.net/sendNotification?uid=${nottokk}`, true);
  xhr.send();
}
```

Εικόνα 73 Κώδικας για αποστολή ειδοποίησης

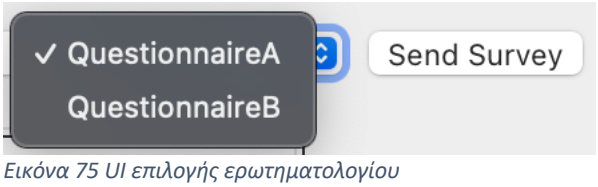
Δ.3.6 Επιλογή ερωτηματολογίου από τους επιστήμονες υγείας

Ακόμη, μία λειτουργία που έχει η κεντρική εφαρμογή είναι η επιλογή ερωτηματολογίου από τον γιατρό, αφού επιλεγεί ο χρήστης πρώτα. Για αρχή, εισάγουμε τους τίτλους των ερωτηματολογίων στην Firebase Realtime Database, όπως φαίνεται παρακάτω, τα οποία πρώτα υπάρχουν στο Firebase Remote Config.



Εικόνα 74 Μορφή των τίτλων των ερωτηματολογίων στην Firebase Realtime Database

Στην κεντρική εφαρμογή η επιλογή ερωτηματολογίων στους γιατρούς φαίνεται στην παρακάτω εικόνα:



Εικόνα 75 UI επιλογής ερωτηματολογίου

Για να μπορούμε να κάνουμε fetch τα ερωτηματολόγια και να τα βάλουμε μέσα στις επιλογές, αναπτύχθηκε ο εξής κώδικας:

```
var selectsurTag = document.createElement("SELECT");
selectsurTag.setAttribute("id","neee");
selectsur.appendChild(selectsurTag);
var spaceTag = document.createElement("text");
spaceTag.textContent = '  ';
select.appendChild(spaceTag);
var buttonsurTag = document.createElement("button");
buttonsurTag.setAttribute("onclick", "sendSurvey()");
buttonsurTag.textContent = 'Send Survey';
selectsur.appendChild(buttonsurTag);
firebase.database().ref('questionnaires').on('value', function(snapshot) {
      const id = Object.values(snapshot.val());
      id.forEach(doc => {
        console.log(`${doc}`);
        var elsur = document.createElement("option");
        elsur.textContent = doc;
        elsur.value = doc;
        selectsurTag.add(elsur);
    });
});
var esur = document.getElementById("neee");
function onChangesur() {
  var uidd = esur.value;
  surveyy = uidd;
  console.log(uidd);
}
```

Εικόνα 76 Κώδικας για το dropdown menu με τα ερωτηματολόγια

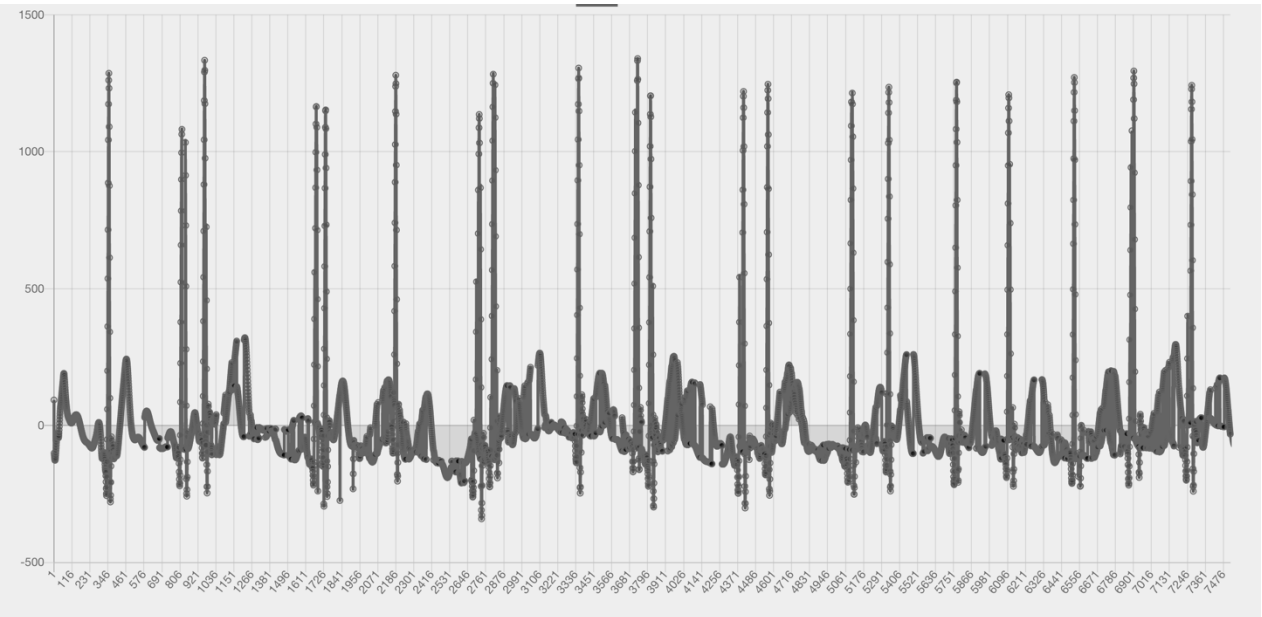
Επίσης, για την καταχώρηση της επιλογής ερωτηματολογίου, από τη μεριά της κεντρικής εφαρμογής, τρέχει ο παρακάτω κώδικας:

```
function sendSurvey() {
  console.log(surveyy);
  console.log(usersur);
  fetch(`https://datatest-d81a6-default-rtdb.europe-west1.firebasedatabase.app/${usersur}.json`, {
    method: 'PATCH',
    body: JSON.stringify({"questionnaire": surveyy}),
    headers: {
    "Content-type": "application/json; charset=UTF-8"
    }
    })
    .then(response => response.json())
    .then(json => console.log(json))
}
```

Εικόνα 77 Κώδικας για καταχώριση της επιλογής ερωτηματολογίου

Δ.3.7 Απεικόνιση των δεδομένων του ηλεκτροκαρδιογραφήματος σε γράφημα

Οι χρήστες έχουν τη δυνατότητα να βλέπουν μέρος του καρδιογραφήματος που έκαναν μέσω του ρολογιού σε μορφή γραφήματος. Δεν απεικονίζεται ολόκληρο το καρδιογράφημα, επειδή δεν χωράει στην ιστοσελίδα. Η απεικόνιση φαίνεται παρακάτω:



Εικόνα 78 UI για την απεικόνιση του καρδιογραφήματος του χρήστη

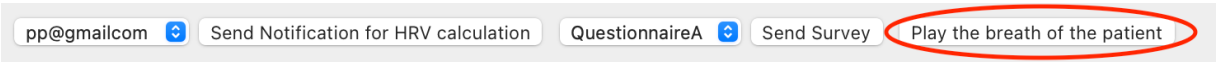
Για την εμφάνιση του παραπάνω γραφήματος, στην πλευρά της εφαρμογής εκτελείται κάθε φορά ο παρακάτω κώδικας, ο οποίος παίρνει ένα csv αρχείο από την Firebase Storage και με τη βοήθεια του ChartJS κάνει τις απαραίτητες ενέργειες:

```
const storage = firebase.storage();
var array = [];
function httpGet(kksk, callback) {

    var xmlhttp=new XMLHttpRequest();
    xmlhttp.onreadystatechange=function() {
        if (xmlhttp.readyState==4 && xmlhttp.status==200) {
            callback(xmlhttp.responseText);
        }
    }
    xmlhttp.open("GET", kksk, false );
    xmlhttp.send();
}
const kksk = storage.ref(`pp@gmailcom/ecg/test.csv`).getDownloadURL().then(function(url) {
  console.log(url);
  httpGet(url, function(textFile){
    array = textFile.slice(1, -1).split(", ");
 });
  var xval = [];
  var ne = 0;
  for (let i=0; i<array.length; i++) {
    ne = ne + 1;
    xval.push(ne);
  }
  let options = {
    type: "line",
    data: {
      labels: xval,
      datasets: [{
        data: array,
        borderColor: 'rgb(100,100,100)'
      }]
    }
  };
  let chart = new Chart("myChart", options);
}).catch(function(error) {
  console.error(error);
});
```

Εικόνα 79 Κώδικας για την εμφάνιση του καρδιογραφήματος του χρήστη

Δ.3.8 Αναπαραγωγή ανάσας

Μέσα από το περιβάλλον της εφαρμογής που έχουν οι επιστήμονες υγείας υπάρχει η δυνατότητα αναπαραγωγής της ανάσας ενός ασθενούς από ηχογράφηση που έχει πάρει ο ίδιος από το έξυπνο ρολόι Apple Watch. Αυτό γίνεται, αφού επιλεγεί ο χρήστης από το dropdown menu, με το πάτημα ενός κουμπιού που φαίνεται στην παρακάτω εικόνα:



Εικόνα 80 UI για την αναπαραγωγή ανάσας

Στην πλευρά της εφαρμογής εκτελείται κάθε φορά ο παρακάτω κώδικας, ο οποίος παίρνει ένα mp4 αρχείο από την Firebase Storage και με τη βοήθεια της λειτουργίας audio της JavaScript γίνεται η αναπαραγωγή της ηχογράφησης στιγμιαία:

```
function playSound() {
  const storage = firebase.storage();
  const audio = new Audio(storage.ref(`${usersur}/message_voice/recording.mp4`).getDownloadURL());
  audio.play();
}
```

Εικόνα 81 Κώδικας για την αναπαραγωγή ανάσας

Το “usersur” κάθε φορά μας επιστρέφει τον χρήστη που είναι επιλεγμένος εκείνη τη στιγμή, ώστε να μπορούμε να επιστρέφουμε τα δεδομένα του συγκεκριμένου χρήστη.

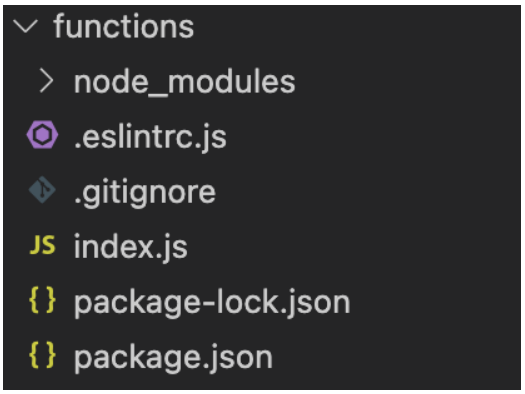
Δ.4 Συναρτήσεις Firebase Cloud Functions για αποστολή ειδοποιήσεων

Δ.4.1 Εισαγωγή

Μέσω της εφαρμογής παρέχεται η δυνατότητα αποστολής ειδοποιήσεων σε κάθε χρήστη ξεχωριστά. Η αποστολή ειδοποιήσεων έγινε μέσω της δημιουργίας ενός backend, το οποίο βασίστηκε στα Firebase Cloud Functions. Αυτό ήταν απαραίτητο, γιατί η Google δεν έδινε αυτή τη δυνατότητα μέσω του Firebase Cloud Messaging και, προφανώς, σε μία εφαρμογή υγείας η άμεση αλληλεπίδραση μεμονωμένων χρηστών και ιατρών είναι το πιο σημαντικό. Η προσέγγιση μέσω των Cloud Functions προσφέρει ασφάλεια στην εφαρμογή, αφού το token του κάθε χρήστη δεν κάνει πολλές διαδρομές μεταξύ των servers και των συσκευών, με αποτέλεσμα να βρίσκεται συνέχεια μέσα στην εφαρμογή. Αναλυτικά, στην παρούσα διπλωματική εργασία χρησιμοποιήθηκε μία Cloud Function, η οποία τρέχει μέσω ενός HTTPS request, καλείται από την κεντρική εφαρμογή που έχουν πρόσβαση οι επιστήμονες υγείας και στέλνει συγκεκριμένο μήνυμα ειδοποίησης. Όμως, η συγκεκριμένη συνάρτηση είναι φτιαγμένη έχοντας την επεκτασιμότητα σαν κυρίαρχο μέλημα. Μπορεί να επεκταθεί είτε μέσω ενός παραπάνω header ο οποίος θα δέχεται σαν όρισμα το κείμενο της ειδοποίησης που θέλουμε να στείλουμε, είτε με τη δημιουργία μιας πανομοιότυπης Cloud Function, η οποία θα έχει απλά διαφορετικό κείμενο ειδοποίησης. Στα κεφάλαια που ακολουθούν, θα

παρουσιαστεί τόσο η εγκατάσταση του Firebase Cloud Functions, όσο και ο κώδικας που γράφτηκε για να λειτουργήσουν τα παραπάνω και η ανάλυση του.
Εγκατάσταση των Firebase Cloud Functions

Η εγκατάσταση των Firebase Cloud Functions γίνεται, είτε στην αρχή που φτιάχνουμε το νέο project με το “firebase init firestore”, επιλέγοντας ότι θα χρησιμοποιήσουμε και Cloud Functions, είτε αφού έχουμε φτιάξει το project με την εντολή “firebase init functions”. Αυτές οι εντολές δημιουργούν ένα directory με το όνομα functions, στο οποίο μπορούμε να γράψουμε τις συναρτήσεις μας, όπως φαίνεται παρακάτω:



Εικόνα 82 Το directory που δημιουργεί το setup του Firebase Cloud Functions

Ο φάκελος αυτός περιέχει οτιδήποτε είναι απαραίτητο για την ανάπτυξη και την δημοσίευση μιας Cloud Function. Αυτές οι συναρτήσεις γράφονται σε γλώσσα JavaScript ή TypeScript. Στην παρούσα διπλωματική αναπτύχθηκαν με τη βοήθεια της JavaScript.

```
"emulators": {
  "hosting": {
    "host": "localhost",
    "port": "5005"
  },
  "auth": {
    "port": 9099
  },
  "database": {
    "port": 9001
  },
  "ui": {
    "enabled": true
  },
  "functions": {
    "port": 9090
  }
},
```

Εικόνα 83 Κώδικας για το configuration των emulators

Αφού έχουμε αναπτύξει τις συναρτήσεις που χρειαζόμαστε, μας δίνεται η δυνατότητα να τις δοκιμάσουμε τοπικά μέσω του emulator του Firebase, ο οποίος τρέχει στο localhost και όπως φαίνεται δίπλα, ο emulator των Cloud Functions τρέχει στη θύρα 9090.

Με αυτή τη δυνατότητα που μας παρέχεται, μπορούμε να δοκιμάσουμε οτιδήποτε θέλουμε πολλές φορές χωρίς να αναγκαζόμαστε να κάνουμε συνέχεια deploy, το οποίο πιάνει χώρο στον server, είναι χρονοβόρο και έχει κάποιο κόστος, εφόσον η συγκεκριμένη υπηρεσία βρίσκεται στο πακέτο που είναι επί πληρωμή.

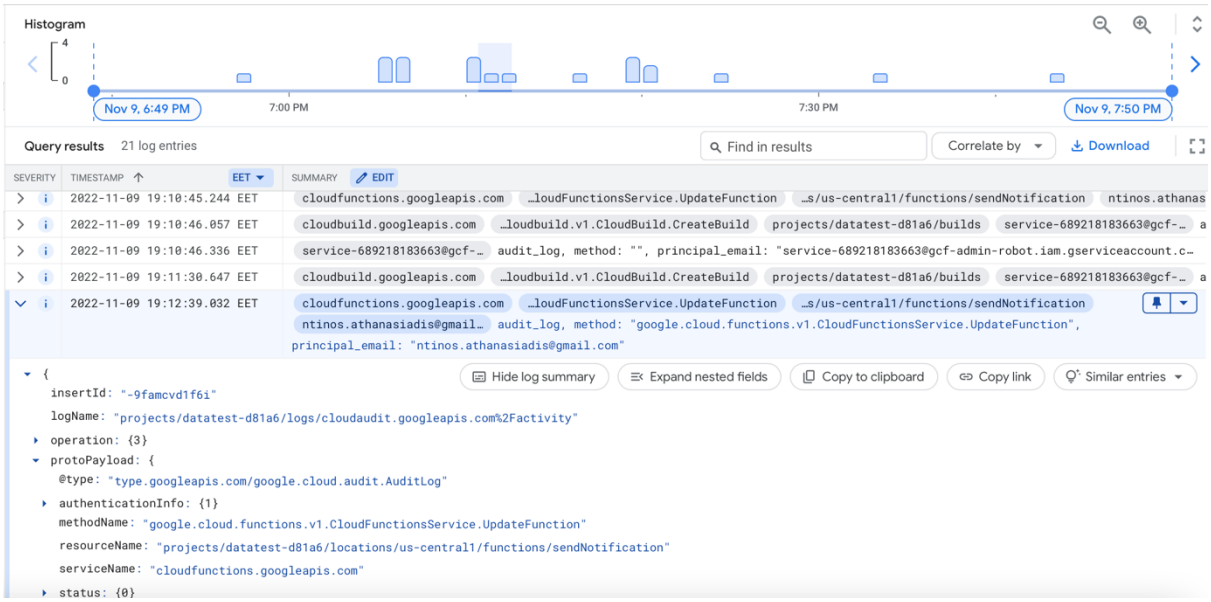
Αφού ολοκληρωθούν οι απαραίτητες δοκιμές, μπορούμε να κάνουμε deploy τις συναρτήσεις μας. Αυτό γίνεται με δύο τρόπους, είτε με το σκέτο “firebase deploy”, το οποίο κάνει deploy και τα cloud functions και το hosting, είτε με το πιο εξειδικευμένο “firebase deploy –only functions”, το οποίο κάνει deploy μόνο τον φάκελο με τις συναρτήσεις μας. Αφού γίνει το

deploy, το terminal μας δίνει τον σύνδεσμο https με τον οποίο μπορούμε να τρέξουμε την συγκεκριμένη συνάρτηση, το οποίο και παραθέτω:

```
β  functions: functions folder uploaded successfully
i  functions: updating Node.js 16 function sendNotification(us-central1)...
β  functions[sendNotification(us-central1)] Successful update operation.
Function URL (sendNotification(us-central1)): https://us-central1-datatest-d81a6
.cloudfunctions.net/sendNotification
i  functions: cleaning up build files...
```

Εικόνα 84 Σύνδεσμος της συνάρτησης ειδοποιήσεων

Ακόμη, μπορούμε να έχουμε τα logs των συναρτήσεων μας μέσω του Google Cloud Console, ώστε να κάνουμε εύκολο debugging και να ξέρουμε τι δεν είναι σωστό, όπως φαίνεται παρακάτω:



Εικόνα 85 Περιβάλλον Google Cloud Console

Δ.4.2 Συνάρτηση ειδοποιήσεων

Αρχικά, γίνεται κλήση της Cloud Function από την κεντρική εφαρμογή, όταν ο γιατρός πατήσει την αποστολή ειδοποίησης, αφού πρώτα έχει επιλέξει τον χρήστη στον οποίο αναφέρεται. Η κλήση της συνάρτησης γίνεται με ένα GET request. Ο κώδικας για το παραπάνω, ο οποίος τρέχει σαν script γλώσσας JavaScript μέσα στην HTML, φαίνεται στην πιο κάτω εικόνα:

```
var buttonTag = document.createElement("button");
buttonTag.setAttribute("onclick", "sendPush()");
buttonTag.textContent = 'Send Notification for HRV calculation';
select.appendChild(buttonTag);
```

Εικόνα 86 Κώδικας για δημιουργία κουμπιού αποστολής ειδοποίησης

```
function sendPush() {
  var xhr = new XMLHttpRequest();
  console.log(nottokk);
  xhr.open('get', `https://us-central1-datatest-d81a6.cloudfunctions.net/sendNotification?uid=${nottokk}`, true);
  xhr.send();
}
```

Εικόνα 87 Κώδικας για αποστολή ειδοποίησης από την κεντρική εφαρμογή

Παραθέτω τον κώδικα της Cloud Function:

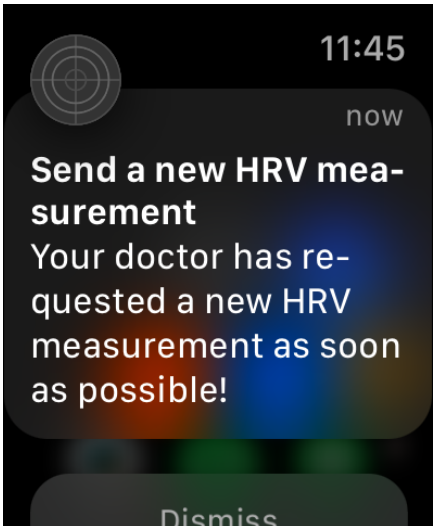
```
exports.sendNotification = functions.https.onRequest(async (req, res) => {

    const queryString = req.url;
    var resq = queryString.slice(1);
    console.log(resq);
    const urlParams = new URLSearchParams(resq);
    const uid = urlParams.get('uid');
    console.log(uid);

    const payload = {
      notification: {
        title: 'Send a new HRV measurement',
        body: `Your doctor has requested a new HRV measurement as soon as possible!`,
      }
    };

    tokens = uid;
    const response = await admin.messaging().sendToDevice(tokens, payload);
    const tokensToRemove = [];
    response.results.forEach((result, index) => {
      const error = result.error;
      if (error) {
        functions.logger.error(
          'Failure sending notification to',
          tokens[index],
          error
        );
        if (error.code === 'messaging/invalid-registration-token' ||
            error.code === 'messaging/registration-token-not-registered') {
          tokensToRemove.push(tokensSnapshot.ref.child(tokens[index]).remove());
        }
      }
    });
    return Promise.all(tokensToRemove);
  });
```

Εικόνα 88 Κώδικας Cloud Function για τη διαχείριση της αποστολής ειδοποίησης

Σε κάθε κλήση της συνάρτησης, προφανώς, απαιτείται το notification token του χρήστη στον οποίο απευθύνεται η ειδοποίηση. Αυτό το token η συνάρτηση το παίρνει ως παράμετρο, όπως φαίνεται και στο πάνω μέρος του κώδικα. Για να του δώσει αυτήν την παράμετρο όμως, η κεντρική εφαρμογή πρέπει να το έχει κάπου αποθηκευμένο. Αυτό το token το αποθηκεύει κάθε φορά η εφαρμογή του έξυπνου ρολογιού στο Firebase Realtime Database και από εκεί το προωθεί στην Cloud Function. Παραθέτω την δομή στην Realtime Database, όπου φαίνεται και το notification token:

Εικόνα 89 Μορφή notification token στην Firebase Realtime Database

Στη συνέχεια, δίνουμε στο Firebase Admin το κείμενο της ειδοποίησης και το token και αυτό επικοινωνεί με το Firebase Cloud Messaging και στέλνει στο ρολόι μία ειδοποίηση σε πραγματικό χρόνο, η οποία φαίνεται παρακάτω:



Εικόνα 90 UI της ειδοποίησης που στείλαμε στο Apple Watch

Τέλος, αν το token δεν υπάρχει πια ή δεν έχει γίνει επαλήθευσή του με το Cloud Messaging, τότε επιστρέφει σφάλμα.

Δ.5 Συμπεράσματα και ιδέες για πιθανές επεκτάσεις και αναβαθμίσεις

Δ.5.1 Εισαγωγή

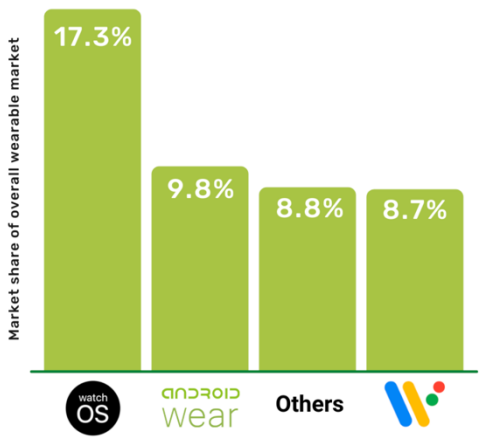
Κατά τη διάρκεια αυτής της διπλωματικής εργασίας, πρωταρχικός σκοπός ήταν η ανάπτυξη μιας όσο το δυνατόν πληρέστερης εφαρμογής, που αφορούσε την συγκέντρωση δεδομένων υγείας διαφόρων χρηστών, την στιγμιαία μέτρηση του HRV και την άμεση επικοινωνία και αλληλεπίδρασή του καθενός ξεχωριστά με τους επιστήμονες υγείας, έτσι ώστε να διευκολυνθεί η καθημερινότητα ανθρώπων με προβλήματα υγείας. Μάλιστα, η παρούσα εφαρμογή θα μπορούσε να επεκταθεί και να εμπλουτιστεί με παραπάνω λειτουργίες, οι οποίες ξεπερνούν τα όρια μιας διπλωματικής εργασίας, όμως θα ήταν πολύ χρήσιμες για την καθημερινότητα των ανθρώπων. Για παράδειγμα, θα μπορούσε να προστεθεί μηχανική μάθηση, να επεκταθεί σε άλλες πλατφόρμες πέραν του WatchOS, αλλά και να προστίθενται συνεχώς νέοι αισθητήρες.

Δ.5.2 Προσθήκη μηχανικής μάθησης

Αρχικά, μια επιπλέον λειτουργία η οποία θα μπορούσε να προστεθεί θα ήταν ο έλεγχος των αποτελεσμάτων από μία Cloud Function. Αυτός ο έλεγχος θα βελτιωνόταν συνεχώς μέσω μηχανικής μάθησης και θα μπορούσε να ενημερώνει τους ασθενείς, αν οι μετρήσεις που πήραν ήταν μέσα στα επιθυμητά πλαίσια, αν ήταν λανθασμένες και έπρεπε να τις ξαναπάρουν ή αν έπρεπε να επικοινωνήσουν με τον γιατρό τους άμεσα. Επιπλέον, θα μπορούσαν μέσω της μηχανικής μάθησης να εντοπιστούν διάφορες πιθανές ασθένειες από τις οποίες μπορεί να έχουν προσβληθεί οι χρήστες και να ειδοποιείται αυτόματα ο γιατρός τους. Τέλος, ένα μοντέλο μηχανικής μάθησης θα μπορούσε να μαθαίνει κάθε πότε ο ασθενής είναι πιο χαλαρός και να θέτει ειδοποιήσεις εξατομικευμένες, έτσι ώστε να υπάρχουν περισσότερες πιθανότητες οι χρήστες να λαμβάνουν μετρήσεις και να μην αδιαφορούν λόγω έλλειψης χρόνου εκείνη τη στιγμή. Αυτή η λειτουργία θα αποτελούσε, με άλλα λόγια, μια παραπάνω φροντίδα για την υγεία των χρηστών.

Δ.5.3 Επέκταση σε άλλες πλατφόρμες

Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο της παρούσας διπλωματικής, για τις ίδιες λειτουργίες θα μπορούσαμε να χρησιμοποιήσουμε και κάποια ακόμα ρολόγια με διαφορετικά λογισμικά. Συγκεκριμένα, το μεγαλύτερο μερίδιο της αγοράς, μετά το WatchOS της Apple, το έχει καταλάβει το Android σε συνδυασμό με το WearOS, τα οποία είναι κυρίαρχα τα τελευταία χρόνια. Αν, λοιπόν, επεκτεινόταν η εφαρμογή του έξυπνου ρολογιού και σε αυτά τα λογισμικά, η απήχησή της θα ανέβαινε και θα κάλυπτε ακόμα μεγαλύτερη μερίδα χρηστών. Στην διπλανή εικόνα φαίνεται η δημοφιλία του κάθε λογισμικού για έξυπνο ρολόι. (40)



Εικόνα 91 Market share των λογισμικών για wearables

Δ.5.4 Προσθήκη νέων αισθητήρων

Μια σοβαρή αναβάθμιση της προαναφερθείσας εφαρμογής θα μπορούσε να είναι η συνεχής προσθήκη αισθητήρων, οι οποίοι θα λαμβάνουν μετρήσεις για ακόμα περισσότερα ιατρικά δεδομένα των χρηστών. Εξάλλου, οι εταιρείες κυκλοφορούν συνεχώς νέα μοντέλα με όλο και περισσότερους αισθητήρες υγείας και αυτό μπορεί να αποτελέσει μία σημαντική αναβάθμιση των λειτουργιών της εφαρμογής και να την φτάσει σε σημείο ώστε να μπορεί να παρακολουθεί την υγεία των χρηστών με μεγάλη ακρίβεια και σε μεγάλη έκταση.

Βιβλιογραφία

1. Apple. Apple Watch Series 7 - Technical Specifications. [Online] 2022. https://support.apple.com/kb/SP860?locale=en_GB.
2. Apple. Swift. [Online] https://developer.apple.com/swift/.
3. Apple. Xcode. [Online] https://developer.apple.com/xcode/.
4. Apple. HealthKit. [Online] https://developer.apple.com/documentation/healthkit.
5. GitHub. Alamofire. [Online] https://github.com/Alamofire/Alamofire.
6. Apple. UserDefaults. [Online] https://developer.apple.com/documentation/foundation/userdefaults.
7. Google. Firebase Auth. [Online] https://firebase.google.com/docs/auth.
8. Google. Firebase Realtime Database. [Online] https://firebase.google.com/docs/database.
9. Google. Firebase Cloud Storage. [Online] https://firebase.google.com/docs/storage.
10. Google. Firebase Cloud Messaging. [Online] https://firebase.google.com/docs/cloud-messaging.
11. Apple. Choosing a Membership. [Online] https://developer.apple.com/support/compare-memberships/.
12. Google. Firebase Cloud Functions. [Online] https://cloud.google.com/functions.
13. Google. Firebase Remote Config. [Online] https://firebase.google.com/docs/remote-config.
14. Google. Firebase Hosting. [Online] https://firebase.google.com/docs/hosting.
15. Wikipedia. HTML. [Online] https://el.wikipedia.org/wiki/HTML.
16. Wikipedia. CSS. [Online] https://el.wikipedia.org/wiki/CSS.
17. Mayoclinic. Walking: Trim your waistline, improve your health. [Online] https://www.mayoclinic.org/healthy-lifestyle/fitness/in-depth/walking/art-20046261.
18. The Lancet. Daily steps and all-cause mortality: a meta-analysis of 15 international cohorts. [Online] https://www.thelancet.com/journals/lanpub/article/PIIS2468-2667(21)00302-9/fulltext.
19. Healthline. What Is the Average Walking Speed of an Adult?. [Online] https://www.healthline.com/health/exercise-fitness/average-walking-speed#average-speed-by-sex.
20. Wikipedia. Ηλεκτροκαρδιογράφημα. [Online] https://el.wikipedia.org/wiki/Ηλεκτροκαρδιογράφημα.
21. Wikipedia. Σφυγμός. [Online] https://el.wikipedia.org/wiki/Σφυγμός.
22. Wikipedia. Heart Rate Variability. [Online] https://en.wikipedia.org/wiki/Heart_rate_variability.
23. 9to5mac. Apple Watch ECG Saves Life. [Online] https://9to5mac.com/2018/12/07/apple-watch-ecg-saves-life/.
24. Medicinenet. What is a Good Resting Heart Rate by Age. [Online] https://www.medicinenet.com/what_is_a_good_resting_heart_rate_by_age/article.htm.
25. ScienceDirect. Twenty-Four Hour Time Domain Heart Rate Variability and Heart Rate: Relations to Age and Gender Over Nine Decades. [Online] https://www.sciencedirect.com/science/article/pii/S0735109797005548.
26. 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. [Online] https://imotions.com/blog/learning/best-practice/heart-rate-variability/.

27. Wikipedia. Τυπική απόκλιση. [Online] https://el.wikipedia.org/wiki/Τυπική_απόκλιση.
28. National Library of Medicine. Oxygen Saturation. [Online] https://www.ncbi.nlm.nih.gov/books/NBK525974/.
29. ProNews. Δείτε πώς αλλάζουν τα επίπεδα οξυγόνου στο αίμα – Τι δείχνει το παλμικό οξύμετρο. [Online] https://www.pronews.gr/ygeia/1055493_deite-pos-allazoyn-ta-epipeda-oxygonoy-sto-aima-ti-deihnei-palmiko-oxymetro/.
30. DrKyriakakis. ΕΛΛΕΙΨΗ ΟΞΥΓΟΝΟΥ ΣΤΟ ΑΙΜΑ-ΣΥΜΠΤΩΜΑΤΑ ΠΟΥ ΤΗΝ ΔΗΛΩΝΟΥΝ. [Online] https://www.drkyriakakis.gr/ελλειψη-οξυγονου-στο-αιμα/.
31. ResearchGate. What Wrist Temperature Tells Us When We Sleep Late A New Perspective of Sleep Health. [Online] https://www.researchgate.net/publication/329477828_What_Wrist_Temperature_Tells_Us_When_We_Sleep_Late_A_New_Perspective_of_Sleep_Health/link/5c1919a592851c22a3342098.
32. Apple. Track your nightly wrist temperature changes with Apple Watch. [Online] https://support.apple.com/en-us/HT213275.
33. National Library of Medicine. Breath sounds, asthma, and the mobile phone. [Online] https://pubmed.ncbi.nlm.nih.gov/11684220/.
34. European Lung. Ασθμα ενηλικων. [Online] https://europeanlung.org/el/information-hub/lung-conditions/άσθμα-ενηλίκων/.
35. Wikipedia. Ευέλικτη Μεθοδολογία. [Online] https://el.wikipedia.org/wiki/Ευέλικτη_μεθοδολογία.
36. Indevlab. What is Agile Development. [Online] https://indevlab.com/blog/what-is-agile-development/.
37. Pocketnow. Best Smartwatches for ECG. [Online] https://pocketnow.com/best-smartwatches-for-ecg/.
38. Google. Firebase Setup. [Online] https://firebase.google.com/docs/ios/setup.
39. Google. Firebase Authentication Setup. [Online] https://firebase.google.com/docs/auth/ios/start.
40. CrankSoftware. What is the future of wearable app development. [Online] https://blog.cranksoftware.com/what-is-the-future-of-wearable-app-development.