ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σύστημα Διαχείρισης Αθλητικών Δραστηριοτήτων για iOS συσκευές»

Του φοιτητή
Θεοδοσιάδη Παναγιώτη
Αρ. Μητρώου: 154448

Επιβλέπουσα
Ελβίρα-Μαρία Αρβανίτου

10 Σεπτεμβρίου 2023

Τίτλος Δ.Ε. **Σύστημα Διαχείρισης Αθλητικών Δραστηριοτήτων για iOS συσκευές**
Κωδικός Δ.Ε. 23160
Φοιτητής: **Θεοδοσιάδης Παναγιώτης**
Εισηγητής: **Ελβίρα-Μαρία Αρβανίτου**
Ημερομηνία ανάληψης Δ.Ε. **21-03-2023**
Ημερομηνία περάτωσης Δ.Ε. **10-09-2023**

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Η επιλογή της παρούσας διπλωματικής εργασίας ήταν αποτέλεσμα τόσο της επαγγελματικής και ακαδημαϊκής μου σταδιοδρομίας όσο και του προσωπικού μου ενδιαφέροντος για τον αθλητισμό και τη σωματική άσκηση. Έχοντας υπάρξει επαγγελματίας μηχανικός λογισμικού εφαρμογών iOS τα τελευταία τρία έτη, αλλά και ερασιτέχνης αθλητής από τότε που θυμάμαι τον εαυτό μου, το πεδίο της ανάπτυξης μιας εφαρμογής διαχείρισης αθλητικών δραστηριοτήτων και διατροφής μου κέντρισε αμέσως το ενδιαφέρον. Προσωπικός μου στόχος είναι με την περάτωση της διπλωματικής εργασίας να έχω εξελιχθεί στον καθένα από τους παραπάνω τομείς έστω και λίγο καθώς επίσης και να λάβω εφόδια για να συνεχίσω την επιτυχημένη ενασχόληση με αυτούς. Τόσο ο προγραμματισμός όσο και ο αθλητισμός είναι ασχολίες τις οποίες αγαπώ και συνδυαστικά αποτελούν το μεγαλύτερο κομμάτι της καθημερινότητας μου.

Περίληψη

Το θέμα της παρούσας διπλωματικής εργασίας αφορά την υλοποίηση μιας εφαρμογής χρησιμοποιώντας λειτουργικό σύστημα iOS. Η εφαρμογή αυτή αφορά τη διαχείριση αθλητικών δραστηριοτήτων καθώς επίσης και την καταγραφή διατροφικών δεδομένων. Η εφαρμογή που αναπτύχθηκε κατά τη διάρκεια εκπόνησης της διπλωματικής εργασίας, με τις κατάλληλες μελλοντικές υλοποιήσεις και τη συνεχή βελτίωση έχει ως στόχο την κάλυψη ενός μέρους της αγοράς των fitness εφαρμογών, η οποία βρίσκεται σε συνεχή άνοδο τα τελευταία έτη. Ταυτόχρονα με την υλοποίηση της εφαρμογής, βασικός στόχος της διπλωματικής εργασίας είναι να παρουσιάσει ολόκληρη τη διαδικασία ανάπτυξης ενός τέτοιου έργου, από τη διαδικασία μελέτης και σχεδίασης έως και τη διαδικασία ανάπτυξης λογισμικού και επιλογής αρχιτεκτονικής και εργαλείων. Παρέχονται επίσης μια πληθώρα τμημάτων κώδικα σε γλώσσα Swift, τα οποία αφορούν την υλοποίηση της mobile εφαρμογής. Με την επιτυχή μελέτη των παραπάνω, ο αναγνώστης της παρούσας εργασίας θα αποκτήσει γνώση σχετικά με την ανάπτυξη εφαρμογών όπως αυτή γίνεται σε επαγγελματικό επίπεδο και θα μπορεί και ο ίδιος να κάνει τα πρώτα του βήματα στο χώρο αυτό εάν το επιθυμεί. Προσωπικά κρίνω ότι οι παραπάνω στόχοι επιτεύχθηκαν σε ένα ικανοποιητικό βαθμό. Το αποτέλεσμα το οποίο αναλύεται παρακάτω είναι μια εφαρμογή η οποία προσφέρει βασικές λειτουργίες και συνεπώς απαιτεί βελτίωση και εξέλιξη, έχει όμως στηθεί σωστά και αποτελεί μια πολύ καλή βάση για μελλοντική ανάπτυξη.

«Sport Activity Management System for iOS devices»

«Theodosiadis Panagiotis»

Abstract

The subject of this thesis concerns the implementation of an application for the iOS operating system. This application involves the management of sports activities as well as the recording of dietary data. The application developed during the thesis, with appropriate future implementations and continuous improvement, aims to cover a portion of the fitness application market, which has been continuously rising in recent years. Concurrently with the implementation of the application, the main objective of the thesis is to present the entire process of developing such a project, from the study and design process to the software development process and the selection of architecture and tools. A plethora of Swift code segments related to the implementation of the mobile application are also provided. With the successful study of the above, the reader of this thesis will acquire knowledge regarding application development at a professional level and will be able to take their first steps in this field if they wish. Personally, I believe that the above-mentioned objectives have been achieved to a satisfactory extent. The result analyzed below is an application that offers basic functions and therefore requires improvement and evolution, but has been assembled correctly and serves as a very good foundation for future development.

Ευχαριστίες

Πρωτίστως θα ήθελα να ευχαριστήσω το συμφοιτητή, συνάδελφο αλλά και πολύ καλό φίλο Παναγιώτη Τούμπα, ο οποίος υλοποίησε πτυχιακή εργασία με το αντίστοιχο θέμα. Το κομμάτι της εκπόνησης μιας τέτοιου είδους εργασίας είναι αρκετά δύσκολο και επίπονο, μέσα από τη συνεργασία μας όμως ήταν επίσης ευχάριστο και δημιουργικό. Στις επόμενες συνεργασίες μας, επαγγελματικές και μη!

Έπειτα θα ήθελα να ευχαριστήσω το μέντορα μου, Χρήστο Χρήστου, ο οποίος μου πρόσφερε απλόχερα γνώση και ήταν πάντα πρόθυμος να συζητήσουμε τις απορίες μου. Τα εφόδια που μου πρόσφερε σε συνδυασμό με την προσωπική μου ενασχόληση και μελέτη, οδήγησαν στη μετάβαση μου από φοιτητή πληροφορικής σε μηχανικό λογισμικού. Η παρούσα εργασία δε θα ήταν δυνατό να πραγματοποιηθεί χωρίς τις βάσεις τις οποίες έλαβα από αυτόν.

Τέλος, σε όλους τους ανθρώπους που αποτελούν κομμάτι της ζωής μου και συνθέτουν το ποιος είμαι και το πώς θα εξελιχθώ. Είναι πολλοί για να τους αναφέρω ονομαστικά, ξέρουν όμως ποιοι είναι και προσπαθώ να τους ευχαριστώ εμπράκτως καθημερινά.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
MVVM	Model View ViewModel
DIContainer	Dependency Injection Container
JSON	JavaScript Object Notation

Κεφάλαιο 1ο: Εισαγωγή

Οι εφαρμογές κινητών τηλεφώνων έχουν ενσωματωθεί στην καθημερινή μας ζωή, επηρεάζοντας τον τρόπο που επικοινωνούμε, ψυχαγωγούμαστε, αγοράζουμε και διαχειριζόμαστε τις χρηματοοικονομικές μας υποθέσεις [1]. Στα τελευταία χρόνια, παρατηρείται μια αυξανόμενη τάση τέτοιων εφαρμογών που σχεδιάστηκαν ειδικά για να βελτιώσουν την υγεία και τη φυσική κατάσταση των χρηστών τους [2]. Αυτές οι εφαρμογές μπορούν να παρακολουθούν τη σωματική δραστηριότητα, να παρέχουν προσαρμοσμένη καθοδήγηση και ακόμα να βοηθούν στη διαχείριση χρόνιων ασθενειών [3].

Το πεδίο μελέτης αυτή της διπλωματικής εργασίας αφορά την σχεδίαση και την υλοποίηση μιας εφαρμογής κινητών τηλεφώνων και συγκεκριμένα μιας fitness εφαρμογής, το Thesis Fitness App. Στο κεφάλαιο αυτό θα γίνει ιστορική αναδρομή στις iOS εφαρμογές στο σύνολο τους, καθώς και στο ρόλο των αθλητικών εφαρμογών στη σημερινή πραγματικότητα και στο πως αυτό σε συνδυασμό με άλλους παράγοντες αποτέλεσε το κίνητρο για την εκπόνηση της διπλωματικής εργασίας.

1.1 Ιστορικό iOS Εφαρμογών

Από το 2007 μέχρι και σήμερα, οι iOS εφαρμογές αποτελούν σημαντικό κομμάτι της καθημερινότητας ενός μεγάλου ποσοστού του παγκόσμιου πληθυσμού. Ειδικότερα θα μπορούσαμε να χωρίσουμε την ιστορία των iOS εφαρμογών στις παρακάτω χρονικές περιόδους:

- Εισαγωγή του iPhone και του App Store (2007-2008): Η Apple παρουσίασε το πρώτο iPhone το 2007, ανατρέποντας τη βιομηχανία των smartphones. Διέθετε μια οθόνη αφής και ένα ισχυρό λειτουργικό σύστημα. Το 2008, η Apple κυκλοφόρησε το App Store, το οποίο επέτρεψε στους προγραμματιστές να δημιουργήσουν και να διανέμουν εφαρμογές για το iPhone. Αυτό αποτέλεσε μια παράδοξη αλλαγή στον χώρο της φορητής υπολογιστικής.
- Κορυφαίες Εφαρμογές (2008-2012): Οι πρώτες εφαρμογές για το iOS επικεντρώθηκαν σε βασικές λειτουργίες όπως ηλεκτρονικό ταχυδρομείο, μηνύματα και περιήγηση στο διαδίκτυο. Παιχνίδια όπως το "Angry Birds" και εργαλεία όπως το "Shazam" απέκτησαν τεράστια δημοτικότητα. Η εισαγωγή του iPad το 2010 διεύρυνε ακόμα περισσότερο τις δυνατότητες για την ανάπτυξη εφαρμογών, ιδίως στους τομείς της παραγωγικότητας και της δημιουργικότητας.
- Ανάπτυξη του Οικοσυστήματος των Εφαρμογών (2012-2015): Ο App Store συνέχισε να αναπτύσσεται με γοργούς ρυθμούς, με αυξανόμενο αριθμό προγραμματιστών να δημιουργούν εφαρμογές για διάφορους σκοπούς, συμπεριλαμβανομένων των κοινωνικών μέσων, της παραγωγικότητας, της υγείας και της ψυχαγωγίας. Η εισαγωγή της γλώσσας προγραμματισμού Swift το 2014 διευκόλυνε και έκανε πιο αποδοτική τη δημιουργία εφαρμογών για το iOS.
- Κερδοσκοπία και Αγορές Εντός των Εφαρμογών (2012-Παρόν): Οι προγραμματιστές άρχισαν να εξερευνούν διάφορες στρατηγικές κερδοσκοπίας, συμπεριλαμβανομένων των μοντέλων preemium (προσφορά μιας δωρεάν έκδοσης με προαιρετικές αγορές εντός της εφαρμογής) και των υπηρεσιών βασισμένων σε συνδρομή. Αυτό επέτρεψε στους προγραμματιστές να δημιουργούν έσοδα προσφέροντας τη δυνατότητα δωρεάν λήψης.
- Επίδραση στους Κλάδους (2015-Παρόν): Οι εφαρμογές του iOS έχουν μετασχηματίσει πολλούς κλάδους, συμπεριλαμβανομένων των παιχνιδιών, των κοινωνικών μέσων, του ηλεκτρονικού εμπορίου, των μεταφορών, της υγείας, της εκπαίδευσης και άλλων.

  Για παράδειγμα, πλατφόρμες όπως η Uber και η Airbnb έχουν επαναπροσδιορίσει τους κλάδους των μεταφορών και της φιλοξενίας μέσω των εφαρμογών τους για το iOS.

- Το iOS ως Πλατφόρμα για την Ανάπτυξη (2015-Παρόν): Το iOS έχει γίνει μια κυρίαρχη πλατφόρμα για την ανάπτυξη εφαρμογών, γνωστή για τα ανθεκτικά μέτρα ασφαλείας, τη συνεπή εμπειρία χρήστη και τις υψηλής ποιότητας εφαρμογές διαθέσιμες στο App Store.

  Η κοινότητα ανάπτυξης του iOS έχει αυξηθεί, με πολλούς προγραμματιστές να εξειδικεύονται στη δημιουργία εφαρμογών ειδικά για το οικοσύστημα της Apple.

- Επίδραση στην Εμπειρία Χρήστη (2017-Παρόν): Η Apple έχει δώσει έμφαση στην ασφάλεια και την προστασία της ιδιωτικότητας του χρήστη, πράγμα που έχει επηρεάσει τις πρακτικές ανάπτυξης εφαρμογών. Αυτό περιλαμβάνει χαρακτηριστικά όπως η Διαφάνεια Παρακολούθησης των Εφαρμογών, σχεδιασμένη για να δίνει στους χρήστες περισσότερο έλεγχο επί των δεδομένων τους.
- Παγκόσμια Επιρροή και Διαθεσιμότητα (Παρόν): Οι εφαρμογές του iOS έχουν μια παγκόσμια επιρροή, εξυπηρετώντας ένα ποικίλο κοινό σε όλο τον κόσμο. Διατίθενται σε πολλές γλώσσες και υποστηρίζουν διάφορες περιοχές και πολιτισμούς.

Συνολικά, οι εφαρμογές του iOS έχουν γίνει αναπόσπαστο μέρος της σύγχρονης ζωής, διαμορφώνοντας τον τρόπο που εργαζόμαστε, παίζουμε, επικοινωνούμε και περιηγούμαστε στον ψηφιακό κόσμο. Συνεχίζουν να προωθούν την καινοτομία και να έχουν σημαντική επίδραση σε διάφορους κλάδους και κοινωνίες σε όλο τον κόσμο.

1.2 Mobile Fitness Applications

Ο ρόλος των εφαρμογών στην υγεία και τη φυσική κατάσταση έχει επισημανθεί σε πολλές επιστημονικές μελέτες. Για παράδειγμα, μια μελέτη που δημοσιεύτηκε στο περιοδικό "Ψυχολογία της Υγείας" ανέφερε ότι άνθρωποι που χρησιμοποιούσαν μια εφαρμογή φυσικής κατάστασης για 12 εβδομάδες είχαν μεγαλύτερη πιθανότητα να χάσουν βάρος και να βελτιώσουν τα επίπεδα της φυσικής τους δραστηριότητας σε σχέση με αυτούς που δεν χρησιμοποίησαν την εφαρμογή [1]. Αυτό υπογραμμίζει τη θετική επίδραση που μπορούν να έχουν αυτές οι εφαρμογές στην ανάπτυξη υγιεινών συνηθειών.

Μια άλλη μελέτη που δημοσιεύτηκε στο περιοδικό "Προληπτική Ιατρική" αναφέρει ότι οι ίδιες εφαρμογές μπορούν να αυξήσουν την αποτελεσματικότητα στην προώθηση της φυσικής δραστηριότητας και της υγιεινής διατροφής, ιδίως μεταξύ των εφήβων [3]. Η μελέτη ανέφερε ότι οι έφηβοι που χρησιμοποίησαν κάποια τέτοια εφαρμογή για τουλάχιστον 6 μήνες είχαν περισσότερες πιθανότητες να αυξήσουν τα επίπεδα της φυσικής τους δραστηριότητας και να μειώσουν την κατανάλωση θερμίδων σε σύγκριση με αυτούς που δεν έκαναν κάτι αντίστοιχο [4].

Παρά την θετική επίδραση, υπάρχουν και πολλές προκλήσεις σχετικά με τη χρήση τους [2]. Ένα από τα κύρια προβλήματα είναι η δυσκολία πολλών ανθρώπων να τηρούν την καθημερινή τους προπόνηση [1]. Σε αυτή την περίπτωση, οι εφαρμογές αυτές μπορούν να συμβάλουν παρέχοντας ατομική ανατροφοδότηση και κίνητρα [3]. Επιπλέον, μια άλλη πρόκληση αφορά την ποιότητα των διαθέσιμων εφαρμογών, η οποία μπορεί να διαφέρει σημαντικά [4]. Είναι σημαντικό να επιλεχθούν αξιόπιστες εφαρμογές που έχουν αξιολογηθεί από ειδικούς και όχι οποιαδήποτε άλλη εφαρμογή. [2].

Παρά τις προκλήσεις, η χρήση των εφαρμογών για τη φυσική κατάσταση έχει τη δυναμική να βελτιώσει την υγεία και την ευημερία μας [3]. Μπορούν να μας βοηθήσουν να παρακολουθούμε την πρόοδό μας, να διατηρούμε το κίνητρό μας και να μαθαίνουμε νέες ασκήσεις [4]. Επιπλέον, μπορούν να μας παρέχουν πρόσβαση σε προσαρμοσμένη καθοδήγηση και υποστήριξη [1].

1.3 Αθλητικές Εφαρμογές στον COVID-19

Ένα ακόμα στοιχείο που έπαιξε σημαντικό ρόλο στην αύξηση της κινητοποίησης και της ένταξης αυτών των εφαρμογών στην καθημερινότητα μας ήταν ο covid-19. Πιο Συγκεκριμένα:

1. Αύξηση της Συνειδητοποίησης για τη Σημασία της Υγείας:

Σύμφωνα με μια έρευνα που διεξήχθη από το Πανεπιστήμιο του Χάρβαρντ, το 79% των συμμετεχόντων αναφέρει ότι η πανδημία τους έκανε πιο συνειδητοποιημένους γύρω από τη σημασία της υγείας τους. Αυτή η συνειδητοποίηση οδήγησε πολλούς ανθρώπους να αναζητήσουν εργαλεία και εφαρμογές γύρω από την υγεία.

2. Κατανόηση της Σημασίας της Φυσικής Δραστηριότητας:

Μια έρευνα του Κέντρου Ελέγχου και Πρόληψης Νοσημάτων (CDC) ανέφερε ότι περίπου το 51% των ανθρώπων στις Ηνωμένες Πολιτείες αύξησε την ποσότητα της φυσικής τους δραστηριότητας κατά τη διάρκεια της πανδημίας, επειδή συνειδητοποίησαν τη σημασία της φυσικής κατάστασης. Οι εφαρμογές για την υγεία προσέφεραν εύκολη πρόσβαση σε προπονήσεις και άλλες φυσικές δραστηριότητες που μπορούν να γίνουν μέσα στο σπίτι, με βάση την έρευνα.

3. Αποιακρυσμένη Φροντίδα Υγείας:

Σύμφωνα με μια έρευνα που δημοσιεύθηκε στο περιοδικό JAMA Internal Medicine, η χρήση τηλεϊατρικών υπηρεσιών αυξήθηκε κατά 154% την περίοδο της πανδημίας, ορισμένες κινητές εφαρμογές παρέχουν δυνατότητες για τηλεϊατρικές συνεδρίες και παρακολούθηση της υγείας από απόσταση, βοηθώντας τους ανθρώπους να λαμβάνουν φροντίδα χωρίς να εκτίθενται στον κίνδυνο του COVID-19.

4. Ενθάρρυνση από τις Αρχές και τους Οργανισμούς Υγείας:

Πολλές χώρες και οργανισμοί υγείας ενθάρρυναν τη χρήση εφαρμογών για την υγεία και την παρακολούθηση της φυσικής κατάστασης. Για παράδειγμα, το Κέντρο Ελέγχου και Πρόληψης Νοσημάτων (CDC) των Ηνωμένων Πολιτειών προώθησε ενεργά τη χρήση εφαρμογών για την υγεία και την παρακολούθηση της υγείας του πληθυσμού.

5. Ανάπτυξη Νέων Εφαρμογών:

Η αυξημένη ζήτηση για τις σχετικές με την υγεία εφαρμογές οδήγησε στην ανάπτυξη νέων εφαρμογών. Σύμφωνα με τα στοιχεία της εταιρείας διαφήμισης και έρευνας App Annie, τον Μάιο του 2020, υπήρξε αύξηση κατά 20% των λήψεων εφαρμογών για την υγεία σε παγκόσμιο επίπεδο

1.4 Κίνητρο Διπλωματικής Εργασίας

Με βάση τα παραπάνω είναι εύκολο να αναγνωριστεί ο μείζον ρόλος και η σημασία που έχουν οι fitness εφαρμογές στην καθημερινότητα, γεγονός που αποτελεί κινητροδότηση για την δημιουργία μιας τέτοιου είδους εφαρμογής. Ταυτόχρονα, με την επιτυχή ολοκλήρωση του Thesis Fitness App καλύπτονται και οι δικές μου ανάγκες ως ερασιτέχνης αθλητής, πράγμα που προσδίδει έξτρα κίνητρο αυξάνοντας την καθημερινή ποιότητα ζωής μου στο κομμάτι της άθλησης και της διατροφής. Το Thesis Fitness App, με την κατάλληλη προώθηση και τη συνεχή υποστήριξη μέσω αναβαθμίσεων και βελτιώσεων, θα αποτελεί μια εφαρμογή η οποία θα μπορούσε να εδραιωθεί στο χώρο, συνεπώς μπορεί εν δυνάμει να συμβάλει ως ένα επίτευγμα και στον επαγγελματικό τομέα.

Πέρα όμως από τα παραπάνω, πολύ σημαντικός παράγοντας στην εκπόνηση της εν λόγω διπλωματικής είναι η πολυετής εμπειρία μου ως επαγγελματίας μηχανικός λογισμικού κινητών εφαρμογών iOS. Το

Thesis Fitness App αποτελεί μια ευκαιρία να επαληθεύσω και να εμπλουτίσω τις γνώσεις τις οποίες έχω αποκτήσει και εφαρμόζω καθημερινά στο εργασιακό μου περιβάλλον. Ταυτόχρονα, μέσα από τη διπλωματική αυτή και τον επισυναπτόμενο κώδικα της εφαρμογής, μπορώ να μοιραστώ αυτές τις γνώσεις με συναδέλφους και συμφοιτητές οι οποίοι μπορεί να ενδιαφέρονται να ασχοληθούν με την ανάπτυξη iOS εφαρμογών, είτε επαγγελματικά είτε σε προσωπικό επίπεδο. Θα με χαροποιούσε ιδιαίτερα αν η εν λόγω διπλωματική εργασία χρησιμοποιηθεί ως οδηγός από κάποιον συνάδελφο, μιας και στο σύνολο της αποτελεί ένα επαρκές δείγμα σχετικά με τη μεθοδολογία ανάπτυξης κινητών εφαρμογών σε επαγγελματικό πλαίσιο, όπως απαιτείται δηλαδή και σε πραγματικές συνθήκες εργασίας.

Τέλος, με το πέρας της διπλωματικής εργασίας έρχεται και η ολοκλήρωση των προπτυχιακών σπουδών μου που αποτελεί επίτευξη ενός πολύ σημαντικού προσωπικού στόχου. Μέσα από αυτό τον κύκλο δημιουργικών χρόνων, αποκόμισα εφόδια και γνώσεις απαραίτητες για την επαγγελματική μου σταδιοδρομία αλλά και κατάρτιση στην επιστήμη της πληροφορικής, η οποία αποτελεί ένα πολύ σημαντικό κομμάτι της ζωής μου καθημερινά.

1.5 Διάρθρωση Διπλωματικής Εργασίας

Στο παρόν κεφάλαιο έγινε μια εισαγωγή στις fitness εφαρμογές κινητών και στο ρόλο που αυτές καλούνται να παίξουν στη σημερινή εποχή.

Στο κεφάλαιο 2 θα γίνει εμβάθυνση σε εφαρμογές τέτοιου είδους και μέσω ανάλυσης του ανταγωνισμού θα καθοριστούν οι στόχοι και οι απαιτήσεις της εφαρμογής που θα υλοποιηθεί. Επίσης θα γίνει λόγος σχετικά με την ευχρηστία της εφαρμογής και συγκεκριμένα θα γίνει μελέτη πάνω στους επίσημους οδηγούς που παρέχονται από την Apple για την ανάπτυξη εύχρηστων εφαρμογών.

Το κεφάλαιο 3 αφορά τις σχεδιαστικές αποφάσεις οι οποίες πάρθηκαν και καθορίστηκαν από τους στόχους και τις απαιτήσεις της εφαρμογής. Συγκεκριμένα γίνεται εκτενής αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν από το σύνολο της εφαρμογής, καθώς και σε εργαλεία που βοήθησαν κατά την ανάπτυξη της.

Στο κεφάλαιο 4 αναλύεται η ανάπτυξη της εφαρμογής σε όλο της το εύρος, με παραδείγματα κώδικα για την καλύτερη κατανόηση. Ουσιαστικά το κεφάλαιο 4 αποτελεί τεχνική αναφορά του έργου που υλοποιήθηκε. Στο τέλος του παρατίθενται όλες οι οθόνες της εφαρμογής σε μορφή screenshot και γίνεται ανάλυση του τελικού προϊόντος και των περιπτώσεων χρήσης εφαρμογής.

Τέλος, το κεφάλαιο 5 αφορά μελλοντικές υλοποιήσεις και βελτιώσεις της εφαρμογής, με σκοπό την εξέλιξη της σε ένα ολοκληρωμένο ανταγωνιστικό προϊόν.

Κεφάλαιο 2ο: Εισαγωγή στην Εφαρμογή

Στο παρόν κεφάλαιο αρχικά γίνεται μελέτη της αγοράς των Fitness εφαρμογών και ανάλυση κάποιων εκ των βασικότερων με σκοπό την αναγνώριση των σημαντικότερων τους λειτουργειών. Αναγνωρίζοντας τις λειτουργίες αυτές αλλά και τις πιθανές ελλείψεις που μπορεί να έχουν, καθορίζονται οι στόχοι και οι απαιτήσεις της εφαρμογής που θα υλοποιηθεί. Τέλος γίνεται εκτενής αναφορά στα Apple Developer Guidelines που καθοδηγούν τον μηχανικό λογισμικού σχετικά με τον τρόπο σχεδιασμού και ανάπτυξης εύχρηστων και λειτουργικών εφαρμογών.

2.1 Παρόμοιες εφαρμογές – Ανάλυση Ανταγωνισμού

Η συντριπτική πλειοψηφία των Fitness εφαρμογών για κινητά τηλέφωνα μπορεί να χωριστεί σε δύο κατηγορίες:

- Trackers. Ως Trackers ορίζονται οι εφαρμογές στις οποίες ο χρήστης εισάγει τα αθλητικά / διατροφικά του δεδομένα και με βάση αυτά μπορεί να δέχεται εξατομικευμένο περιεχόμενο. Συνήθως στις εφαρμογές αυτές υπάρχει και ένα τμήμα στο οποίο γίνεται ανάλυση και παρουσιάζεται το ιστορικό του χρήστη για την εκάστοτε δραστηριότητα.
- Ευέλικτες Πλατφόρμες Γυμναστικής. Οι εφαρμογές αυτές επιτρέπουν στους χρήστες να παραμετροποιούν ένα σύνολο παραμέτρων και λειτουργούν ως βοηθός κατά τη διάρκεια μια δραστηριότητας. Ταυτόχρονα, συνήθως παρέχονται εξατομικευμένα προγράμματα γυμναστικής με βάση την αθλητική ικανότητα αλλά και τις προτιμήσεις του χρήστη. Τα προγράμματα αυτά ουσιαστικά λειτουργούν ως ένας προσωπικός ψηφιακός γυμναστής.

Δύο από τις πιο δημοφιλείς Fitness εφαρμογές παρουσιάζονται παρακάτω.

2.1.1 MyFitnessPal

Το MyFitnessPal [5] είναι ίσως η πιο επιτυχημένη εφαρμογή που αφορά την παρακολούθηση φυσικής κατάστασης και υγείας, παρέχοντας στους χρήστες δυνατότητα καταγραφής προσωπικών γευμάτων αλλά και τη δημιουργία εξατομικευμένων προπονητικών προγραμμάτων. Είναι μια εφαρμογή γυμναστικής που επικεντρώνεται κυρίως στην παρακολούθηση της κατανάλωσης και της δαπάνης θερμίδων, βοηθώντας τους χρήστες να διαχειριστούν το βάρος τους - είτε πρόκειται για απώλεια βάρους, αύξηση ή διατήρησή του.

Οι χρήστες μπορούν να καταγράφουν την καθημερινή κατανάλωση τροφίμων και ποτών τους, τα οποία επιλέγουν από εκτενή βάση δεδομένων με εκατομμύρια είδη τροφίμων, συμπεριλαμβανομένων επώνυμων και γενικών επιλογών. Επίσης οι χρήστες μπορούν να σαρώσουν barcodes σε συσκευασμένα τρόφιμα για να τα προσθέσουν γρήγορα στο ημερολόγιο τροφίμων τους, απλοποιώντας τη διαδικασία καταχώρισης. Με την καταχώριση του κάθε γεύματος, υπολογίζεται αυτόματα η διατροφική αξία και εντάσσεται στην εφαρμογή. Το MyFitnessPal έχει στη βάση δεδομένα του ακόμη και δημοφιλή γεύματα από γνωστές αλυσίδες φαγητού και ποτού, παράλληλα όμως επιτρέπει στους χρήστες να δημιουργούν και δικά τους προσαρμοσμένα γεύματα αν αυτοί το επιθυμούν. Η καταγραφή τον παραπάνω στοιχείων αποσκοπεί στην επίτευξη στόχων οι οποίοι καθορίζονται από το χρήστη. Συγκεκριμένα υπάρχουν στόχοι καύσης και κατανάλωσης θερμίδων, κατανάλωσης μακροθρεπτικών αλλά και μικροθρεπτικών συστατικών.

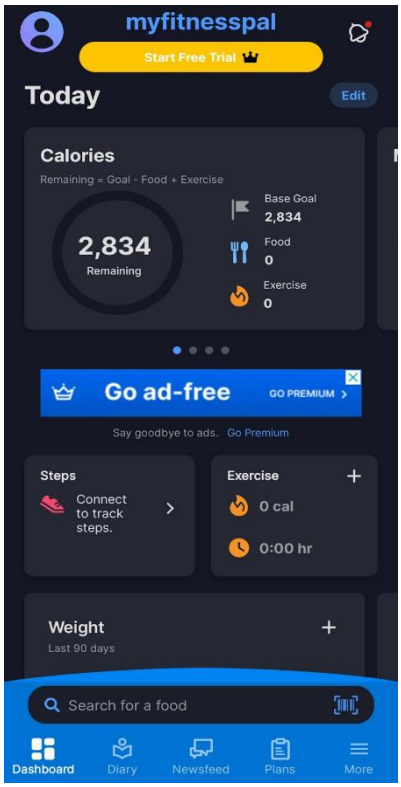
Πέρα από το κομμάτι της διατροφής, μέσω του MyFitnessPal οι χρήστες μπορούν να καταγράφουν την αθλητική και σωματική τους δραστηριότητα. Ο χρήστης επιλέγει και εκτελεί από ένα ευρύ σύνολο

ασκήσεων και μέσα από τα δεδομένα που παρέχονται για την κάθε άσκηση μπορεί να γνωρίζει και τις καύσεις που κάνει σε θερμίδες. Οι χρήστες μπορούν να θέσουν συγκεκριμένους στόχους υγείας και φυσικής κατάστασης που μπορεί να αφορούν την απώλεια βάρους, συντήρηση ή αύξηση μυών.

Η εφαρμογή μέσω διαγραμμάτων και γραφημάτων επιτρέπει στο χρήστη να ελέγχει την πρόοδο του σε όλους τους παραπάνω τομείς. Αυτό τον βοηθά να ερμηνεύει καλύτερα αλλαγές που αφορούν τη σωματοδομή και τον τρόπο άθλησης του. Μέσα από ομαδικούς στόχους και επιτεύγματα, η εφαρμογή προάγει τη χρήση της μαζί με άλλους χρήστες, δίνοντάς της έτσι ένα κοινωνικό προφίλ. Αυτό δημιουργεί μια κοινότητα χρηστών οι οποίοι μπορούν μέσω forum να συνεργάζονται και να ανταλλάσσουν συμβουλές και εμπειρίες.

Ένα άλλο πολύ σημαντικό χαρακτηριστικό αποτελεί η συνεργασία με έξυπνα ρολόγια και wearables, όπως τα Fitbit, Garmin και Apple Watch, για να εισάγει αυτόματα δεδομένα δραστηριότητας και βημάτων.

Ορισμένα από τα αρνητικά στοιχεία, κυρίως αφορούν την εκτεταμένη βάση δεδομένων τροφίμων και τον αυξημένο αριθμό των λειτουργιών που προαναφέρθηκαν. Πολλές φορές ο πλουραλισμός σε επιλογές μπορεί να μπερδέψει το χρήστη και να μην έχει το επιθυμητό αποτέλεσμα. Επίσης είναι σημαντικό όταν οι διαθέσιμες δυνατότητες είναι πολλές να υπάρχει κατάλληλη καθοδήγηση μέσω wizard και οπτικών βοηθειών έτσι ώστε ο χρήστης να μπορεί εύκολα να εξοικειωθεί με το σύνολο της εφαρμογής. Το MyFitnessPal είναι κατά ένα μέρος δωρεάν, παρέχοντας τις βασικές λειτουργίες από αυτές που περιγράφηκαν, ενώ προσφέρει και μια premium συνδρομή που ξεκλειδώνει τις προχωρημένες λειτουργίες, όπως προηγμένες διατροφικές πληροφορίες, προγραμματισμός γευμάτων και εμπειρία χωρίς διαφημίσεις.



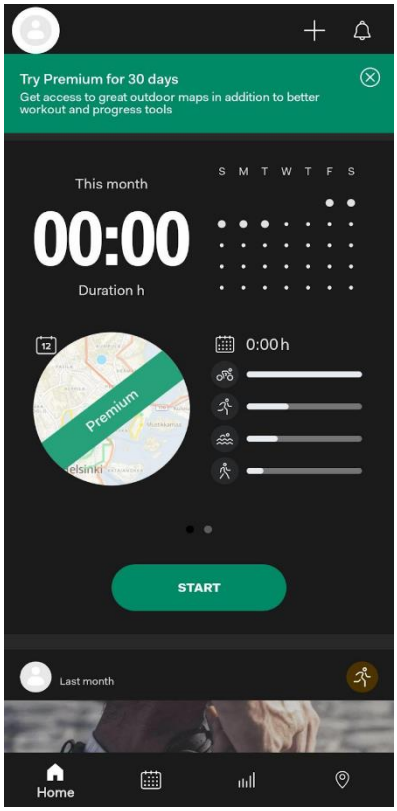
Σχήμα 2.1 MyFitnessPal

2.1.2 Sports Tracker Running Cycling

Το Sports Tracker Running Cycling [6] περιέχει ένα σύνολο αθλητικών δραστηριοτήτων όπως τρέξιμο, ποδηλασία, περπάτημα, πεζοπορία, ποδηλασία βουνού, σκι, κ.α. . Βασικός σκοπός της εφαρμογής είναι η ακριβής καταγραφή δεδομένων προπόνησης, χρησιμοποιώντας τεχνολογίες όπως το GPS του κινητού τηλεφώνου. Όπως και στην περίπτωση του MyFitnessPal τα δεδομένα που καταγράφονται αναλύονται και παρουσιάζονται στο χρήστη με τη μορφή διαγραμμάτων για την μελέτη και κατανόηση τους.

Πολύ ισχυρό είναι επίσης το κοινωνικό προφίλ της εφαρμογής. Οι χρήστες μπορούν να ακολουθούν ο ένας τον άλλο και να μοιράζονται τις επιδώσεις τους καθώς επίσης και φωτογραφίες, αγαπημένες διαδρομές, χάρτες και πολλά άλλα. Αυτές είναι τυπικές λειτουργίες που συναντώνται σε social media εφαρμογές.

Βασικό μειονέκτημα της εφαρμογής αποτελεί το premium mode. Για το μεγαλύτερο μέρος των λειτουργιών της εφαρμογής, πολλές φορές ακόμη και για βασικές λειτουργίες, ο χρήστης καλείται να πληρώσει για να τις χρησιμοποιήσει. Τέτοιες λειτουργίες είναι η πρόσβαση στο ιστορικό, η ανάλυση των αποτελεσμάτων της άσκησης, οι προτάσεις διαδρομών και προπονητικού πλάνου κ.α. Χωρίς να είναι ενεργοποιημένο το premium mode, η εφαρμογή καταλήγει πολύ φτωχή, σύμφωνα με σχόλια και παράπονα χρηστών.



Σχήμα 2.2 Sports Tracker

2.2 Σκοπός της Εφαρμογής

Βασισμένοι στη μελέτη των παραπάνω αλλά και άλλων εφαρμογών, καθώς και στην εμπειρία και τις προσωπικές μας ανάγκες, αποφασίστηκαν οι στόχοι της εφαρμογής Thesis Fitness App. Το Thesis Fitness App θα είναι κατά κόρων μια tracker εφαρμογή, με ένα σύνολο από λειτουργίες που θα βοηθούν το χρήστη και κατά την διάρκεια της άθλησής του. Βασικό του πλεονέκτημα σε σχέση με τις ήδη

υπάρχουσες υλοποιήσεις αποτελεί η δωρεάν χρήση του συνόλου των λειτουργιών που προσφέρονται. Είναι σύνηθες σε παρόμοιες εφαρμογές να υπάρχουν μέθοδοι χρηματοδότησης, όπως για παράδειγμα η αναβάθμιση ενός χρήστη σε προνομιούχο με αντάλλαγμα μηνιαίας συνδρομής ή αποκλεισμός χρηστών από συγκεκριμένο περιεχόμενο εκτός αν αυτοί καταβάλουν το απαιτούμενο ποσό (pay-gate). Το Thesis Fitness App αποσκοπεί στο να είναι ένα πλήρες δωρεάν σύστημα το οποίο θα παρέχει σε γενικές γραμμές αντίστοιχη προστιθέμενη αξία συγκριτικά με άλλες δημοφιλείς εφαρμογές.

Εφόσον στην εφαρμογή θα υπάρχουν χρήστες, το αρχικό στάδιο της θα περιλαμβάνει την αυθεντικοποίηση των χρηστών καθώς και την καταγραφή κάποιον δεδομένων με βάση τα οποία θα επαυξάνεται η λειτουργικότητα της υπόλοιπης εφαρμογής. Έπειτα, η εφαρμογής χωρίζεται στα παρακάτω σύνολα βασικών λειτουργιών:

- Αθλητική Δραστηριότητα
- Διατροφή
- Ιστορικό

Στη συνέχει περιγράφεται συνοπτικά η κάθε μια από τις παραπάνω λειτουργίες. Στο κεφάλαιο 4 φαίνεται το πως αυτές οι λειτουργίες μετατράπηκαν σε οθόνες και περιπτώσεις χρήσης της εφαρμογής.

2.2.1 Αθλητική Δραστηριότητα

Πέρα από την παρακολούθηση δεδομένων αθλητικής δραστηριότητας, θέλουμε η εφαρμογή να λειτουργεί ως βοήθημα αλλά και ως κίνητρο στο χρήστη για να αθληθεί και να είναι πιο ενεργός. Μέσα από ένα σύνολο οθονών ο χρηστής θα μπορεί να επιλέγει τις αγαπημένες του δραστηριότητες και να τις εκτελεί βάζοντας προαιρετικά ένα στόχο άθλησης και λαμβάνοντας σε πραγματικό χρόνο δεδομένα σχετικά με την άθληση του. Τα δεδομένα αυτά προσαρμόζονται και αλλάζουν με βάση το είδος της άσκησης. Επίσης ο χρήστης θα λαμβάνει προκλήσεις βασισμένες στις αγαπημένες του ασκήσεις, Η πρόοδος στις προκλήσεις αυτές θα ανεβαίνει όσο ο χρήστης αθλείται περισσότερο. Συνεπώς οι προκλήσεις αυτές θα λειτουργούν ως κίνητρο αλλά και ως ένας διασκεδαστικός τρόπος αλληλεπίδρασης με την εφαρμογή. Όλα τα παραπάνω δεδομένα θα συλλέγονται και θα αποθηκεύονται στη βάση δεδομένων. Η αξιοποίηση τους θα γίνεται στο κομμάτι του ιστορικού το οποίο περιγράφεται παρακάτω.

2.2.2 Διατροφή

Στο κομμάτι της διατροφής, ο χρήστης θα μπορεί να καταγράφει ένα σύνολο δεδομένων που αφορούν την ημερήσια κατανάλωσής νερού, θερμίδων και μακροθρεπτικών συστατικών (πρωτεΐνες, υδατάνθρακες, λίπη). Θέλουμε να κάνουμε την εισαγωγή των δεδομένων αυτών όσο πιο εύκολη γίνεται για το χρήστη, προσφέροντας του διαφορετικούς τρόπου εισαγωγής δεδομένων. Ο χρήστης θα μπορεί είτε να εισάγει κατευθείαν τις διατροφικές τιμές είτε να αναζητά τις τροφές που έφαγε από μια πληθώρα διαθέσιμων και να εισάγει την ποσότητα κατανάλωσης. Η εφαρμογή στη συνέχεια θα υπολογίζει αυτόματα τα ζητούμενα στοιχεία και θα τα προσθέτει στα ημερήσια διατροφικά δεδομένα. Όπως και στην περίπτωση των αθλητικών δραστηριοτήτων, όλα τα διατροφικά δεδομένα του χρήστη εντάσσονται στο ιστορικό του για ανάλυση και πορίσματα σε δεύτερο χρόνο.

2.2.3 Ιστορικό

Τόσο για της αθλητικές δραστηριότητες όσο και για τη διατροφή, ο χρήστης θα μπορεί να βρει ομαδοποιημένα τα δεδομένα του σε μορφή διαγραμμάτων στο τμήμα του ιστορικού. Αρχικά ο χρήστης θα επιλέγει το χρονικό πλαίσιο για το οποίο επιθυμεί να δει δεδομένα ιστορικού και έπειτα θα

κατασκευάζονται με βάση αυτό ένα σύνολο διαγραμμάτων που θα απεικονίζουν την πρόοδο του χρήστη κατά την πάροδο του χρόνου. Ουσιαστικά η εφαρμογή με τη χρήση του ιστορικού λειτουργεί και ως ψηφιακό ημερολόγιο.

2.3 Ευχρηστία και Apple Developer Guidelines

Εφόσον έχουν πλέον καθοριστεί οι απαιτήσεις της εφαρμογής, είναι σημαντικό να παρθούν ενημερωμένες σχεδιαστικές αποφάσεις για την επίτευξη του βέλτιστου δυνατού αποτελέσματος, μιας εφαρμογής που χαρακτηρίζεται από ευχρηστία και λειτουργικότητα.

2.3.1 Βασικά Χαρακτηριστικά Σχεδίασης

Κατά το σχεδιασμό μιας iOS εφαρμογής είναι απαραίτητο να γίνουν κατανοητά τα παρακάτω βασικά χαρακτηριστικά, τα οποία καθιστούν την εμπειρία χρήσης μιας iOS συσκευής μοναδική. Τα χαρακτηριστικά αυτά μπορεί να διαφοροποιούνται από συσκευή σε συσκευή, συνεπώς με την κατανόηση αυτών μπορούν να παρθούν ενημερωμένες αποφάσεις κατά το σχεδιασμό, έτσι ώστε η εφαρμογή να εκτιμηθεί από το χρήστη και να είναι απολαυστική κατά την χρήση. Τα εν λόγω χαρακτηριστικά είναι τα εξής:

- **Display / Οθόνη.** Τα iPhone έχουν κατά κανόνα οθόνες μετρίου μεγέθους και υψηλής ανάλυσης. Ενδεικτικά η μικρότερη κατηγορία οθονών iPhone έχει μήκος 3.5 ίντσες (8,9 εκ.) και ανάλυση 640 x 960 ενώ η μεγαλύτερη έχει μήκος 6.7 ίντσες (17 εκ.) και ανάλυση 1284 x 2778.
- **Ergonomics / Εργονομία.** Οι χρήστες κατά κανόνα κρατούν μια iPhone συσκευή με το ένα ή σπανιότερα και τα δύο χέρια και εναλλάσσουν τον προσανατολισμό από portrait σε landscape και αντίστροφα ανά περίπτωση. Κατά την αλληλεπίδραση με μια συσκευή, σε μια τυπική χρήση, ο χρήστης δεν απέχει παραπάνω από 0.6 μέτρα από αυτή.
- **Inputs / Είσοδοι. Χειρονομίες πολλαπλής αφής**, **πληκτρολόγια** και **φωνητικές εντολές** επιτρέπουν στους χρήστης να πραγματοποιούν ενέργειες και να πετυχαίνουν ουσιαστικούς για αυτούς στόχους όσο βρίσκονται εν κινήσει. Επίσης, πολλές φορές οι χρήστης μέσω των **προσωπικών δεδομένων**, δεδομένων από το **γυροσκόπιο** και το **επιταχυνσιόμετρο** της συσκευής, δεδομένων τοποθεσίας μέσω **δορυφόρων** κ.α. πραγματοποιεί λειτουργίες που αφορούν την τοποθεσία και την κίνηση του στο χώρο.
- **App Interactions / Αλληλεπιδράσεις εφαρμογών.** Ένα τυπικό σενάριο χρήσης, είτε αυτή διαρκεί λίγο χρόνο (1 - 2 λεπτά) είτε πολύ (1+ ώρες), είναι η εναλλαγή μεταξύ εφαρμογών. Τυπικά, ο χρήστης έχει πολλές εφαρμογές ανοιχτές και εναλλάσσεται μεταξύ τους. Είναι σημαντικό για τη σχεδίαση να υπάρχει πρόληψη για την εναλλαγή αυτή της εφαρμογής από το **προσκήνιο (foreground mode)** στο **παρασκήνιο (background mode)**.
- **System features / Λειτουργίες συστήματος.** Το λειτουργικό iOS παρέχει μια πληθώρα λειτουργιών οι οποίες βοηθούν το χρήστη να αλληλοεπιδρά με την εφαρμογή και το σύστημα με τρόπους οι οποίοι είναι ήδη γνώριμοι για αυτόν. Αναφορικά παρέχονται:
    - **Widgets**
    - **Home Screen Quick Actions**
    - **Spotlight**
    - **Shortcuts**
    - **Activity Views**

2.3.2 Βέλτιστες Πρακτικές Σχεδίασης

Για να παρέχονται μοναδικές εμπειρίες στο χρήστη, οι εξέχοντες εφαρμογές φροντίζουν να ενσωματώνουν τις δυνατότητες, τόσο του λειτουργικού όσο και της συσκευής, οι οποίες έχουν αποδειχθεί ως οι πιο χρήσιμες για το χρήστη. Είναι συνεπώς απαραίτητο να δοθεί προτεραιότητα στην υλοποίηση και τη σχεδίαση κατ' αυτόν τον τρόπο, κάτι που μπορεί να πραγματοποιηθεί ακολουθώντας τις παρακάτω ενδεικτικές πρακτικές.

- Ώθηση του χρήστη στο να επικεντρώνεται στις κύριες λειτουργίες και στόχους. Αυτό επιτυγχάνεται με την μείωση των διαθέσιμων κουμπιών / τρόπων αλληλεπίδρασης. Η διεπαφή χρήστη πρέπει πάντοτε να παραμένει ξεκάθαρη, γνώριμη και κατανοητή. Οποιαδήποτε δευτερεύουσα πληροφορία ή ενέργεια θα πρέπει να είναι ανιχνεύσιμη από το χρήστη με τον μικρότερο και απλούστερο συνδυασμό ενεργειών.
- Απρόσκοπτη προσαρμογή στις αλλαγές εμφάνισης περιβάλλοντος (π.χ. κατεύθυνση συσκευής, dark mode / light mode). Σε αυτό συμπεριλαμβάνεται και παροχή δυνατότητας στο χρήστη να παραμετροποιεί την εμφάνιση της διεπαφής έτσι ώστε να ταιριάζει στις προτιμήσεις του και στον τρόπο με τον οποίο έχει συνηθίσει.
- Σχεδιασμός διεπαφής και αλληλεπίδρασης έχοντας κατά νου τον τρόπο με τον οποίο ο χρήστης κρατά τη συσκευή. Π.χ. είναι αποδεδειγμένο ότι είναι ευκολότερο και αποτελεσματικότερο στη χρήση, κουμπιά ή άλλου είδους εργαλεία διεπαφής να βρίσκονται στις μεσαίες και χαμηλές περιοχές της συσκευής.
- Εμπλουτισμός της εμπειρίας διεπαφής με λειτουργίες που παρέχονται από το σύστημα (σύστημα πληρωμών, βιομετρική αυθεντικοποίηση, τοποθεσία χρήστη). Απαραίτητη προϋπόθεση η συναίνεση του χρήστη σχετικά με τη διαχείριση και την ασφάλεια προσωπικών δεδομένων.

Τα παραπάνω αποτελούν μια εισαγωγή, καθώς είναι τα απολύτως απαραίτητα και βασικά στοιχεία σχεδίασης μια διεπαφής. Στη συνέχεια θα γίνει ανάλυση και εμβάθυνση σε πιο συγκεκριμένα θέματα σχεδίασης. Αναφορικά θα μελετηθούν οι *θεμελιώδεις τεχνολογίες*, καθώς και οι *βέλτιστες πρακτικές και μοτίβα*.

2.3.3 Θεμελιώδεις Τεχνολογίες

Στο κεφάλαιο αυτό γίνεται μελέτη και παρατίθενται στοιχεία για την κατανόηση θεμελιωδών τεχνολογιών και στοιχείων τα οποία βοηθούν στη δημιουργία πλούσιων και ευχάριστων στη χρήση εφαρμογών. Για πρακτικούς λόγους θα αναλυθούν στοιχεία και τεχνολογίες που χρησιμοποιούνται στο fitness app, τα οποία αποτελούν υποσύνολο από αυτά που παρατίθενται στα guideline που παρέχονται από την Apple.

2.3.3.1 Λογότυπο Εφαρμογής, Εικονίδια Εφαρμογής και SF Symbols

Ένα μοναδικό και αξιομνημόνευτο εικονίδιο εφαρμογής είναι το καταλληλότερο μέσο για να επικοινωνήσει άμεσα στο χρήστη την προσωπικότητα, το σκοπό και τη λειτουργικότητα μιας εφαρμογής. Ένα εικονίδιο θα πρέπει να αποτελεί ένα γραφικό κομμάτι της εφαρμογής που εκφράζει κάτι ξεκάθαρο, κάτι που ο κάθε χρήστης μπορεί να κατανοήσει στιγμιαία. Είναι συνεπώς απαραίτητο να δοθεί δέουσα προσοχή τόσο στο λογότυπο της εφαρμογής, όσο και στα εικονίδια και σύμβολα που χρησιμοποιούνται στο εσωτερικό αυτής.

2.3.3.1.1 Λογότυπο Εφαρμογής

Ένα όμορφο λογότυπο αποτελεί απαραίτητο στοιχείο της κάθε εφαρμογής. Θα πρέπει όμως να δοθεί προσοχή κατά τη σχεδίαση του, έτσι ώστε αυτό να είναι συμβατό με τις διάφορες iOS πλατφόρμες, διατηρώντας τη μοναδικότητα του και επιτυγχάνοντας συνεκτικότητα. Για να επιτευχθούν αυτοί οι στόχοι η Apple συνιστά να τηρηθούν οι ακόλουθες πρακτικές.

- **Έμφαση στην απλότητα.** Συνήθως τα απλά λογότυπα είναι αυτά τα οποία είναι και πιο κατανοητά προς το χρήστη. Η αποφυγή των πολλών λεπτομερειών βοηθά επίσης στην καθαρότητα της εικόνας του λογότυπου, ιδιαίτερα στις μικρότερες συσκευές στις οποίες καθίσταται εξ ορισμού δύσκολο στο χρήστη να τις διακρίνει. Στην ίδια κατευθυντήρια, το παρασκήνιο (background) του λογότυπου θα πρέπει να είναι λιτό και να μην αποσπά την προσοχή του χρήστη. Παραδείγματα λογοτύπων που ακολουθούν τις παραπάνω πρακτικές και αποτελούν πλέον trademarks της βιομηχανίας φαίνονται στο Σχήμα 2.3.

Σχήμα 2.3 Λογότυπα δημοφιλών εφαρμογών

- **Αποφυγή χρήσης κειμένου.** Η χρήση κειμένου στο λογότυπο θα πρέπει να αποφεύγεται σε κάθε περίπτωση, εκτός και αν παίζει καθοριστικό ρόλο για την εφαρμογή. Το κείμενο στα λογότυπα συνήθως είναι πολύ μικρό, με αποτέλεσμα να μην γίνεται κατανοητό από το χρήστη. Επίσης, συνήθως το όνομα της εφαρμογής φαίνεται κάτω από αυτήν, συνεπώς το να συμπεριληφθεί στο λογότυπο καθίσταται περιττό.
- **Χρήση πρωτότυπου γραφικού και όχι φωτογραφίας.** Κατά κανόνα μια φωτογραφία περιέχει πληθώρα λεπτομερειών που όπως προαναφέρθηκε, είναι δύσκολα να αναγνωριστούν σε μικρότερα μεγέθη συσκευών. Στην ίδια λογική, το να γίνει χρήση ενός screenshot από το εσωτερικό της εφαρμογής θεωρείται κακή πρακτική και πρέπει να αποφευχθεί. Τέλος, αν και προφανές, το να χρησιμοποιηθεί ως λογότυπο γραφικό το οποίο προσομοιάζει λογότυπα άλλων εφαρμογών του συστήματος της Apple είναι παράνομο και πρέπει να αποφεύγεται σε κάθε περίπτωση.

Τα τεχνικά χαρακτηριστικά ενός λογότυπου εφαρμογής που αφορούν τόσο τη γραφική σχεδίαση όσο και το μέγεθος του φαίνονται στο Σχήμα 2.4 και στο Σχήμα 2.5 αντίστοιχα.

Platform	Layers	Transparency	Asset shape
iOS, iPadOS	Single	No	Square
macOS	Single	Yes, as appropriate	Square with rounded corners
tvOS	Multiple	No	Rectangle
watchOS	Single	No	Square

Σχήμα 2.4 Τεχνικά χαρακτηριστικά λογότυπου εφαρμογής

@2x (pixels)	@3x (pixels) iPhone only	Usage
120x120	180x180	Home Screen on iPhone
167x167	–	Home Screen on iPad Pro
152x152	–	Home Screen on iPad, iPad mini
80x80	120x120	Spotlight on iPhone, iPad Pro, iPad, iPad mini
58x58	87x87	Settings on iPhone, iPad Pro, iPad, iPad mini
76x76	114x114	Notifications on iPhone, iPad Pro, iPad, iPad mini

Σχήμα 2.5 Μεγέθη λογότυπου εφαρμογής για συσκευές iPhone

2.3.3.1.2 Εικονίδια Εφαρμογής

Σε κάθε εφαρμογή γίνεται χρήση μια πληθώρας εικονιδίων έτσι ώστε ο χρήστης να μπορεί να κατανοήσει τους τρόπους με τον οποίο μπορεί να αλληλοεπιδράσει με αυτή, καθώς και τις ενέργειες τις οποίες μπορεί να εκτελέσει.

Σε αντίθεση με τα λογότυπα εφαρμογών, τα οποία θα πρέπει να χαρακτηρίζονται από μοναδικότητα και πρωτοτυπία στο σχεδιασμό έτσι ώστε να συνάδουν με την προσωπικότητα της κάθε εφαρμογής, κατά το σχεδιασμό των εικονιδίων μιας εφαρμογής θα πρέπει να εφαρμοστεί μια τακτική με έμφαση στην απλότητα. Με τον τρόπο αυτό, αυξάνεται η πιθανότητα ο χρήστης να έχει ξαναχρησιμοποιήσει ίδια ή παρόμοια εικονίδια κατά τη χρήση άλλων εφαρμογών, καθιστώντας έτσι γνώριμη και φυσική την αλληλεπίδραση του και με την εν λόγω εφαρμογή. Ένας καλός και συνήθης τρόπος εφαρμογής της πρακτικής αυτής, είναι το κάθε εικονίδιο να εκφράζει μέσω μιας γνώριμης οπτικής μεταφοράς την ενέργεια η οποία θα εκτελεστεί όταν πραγματοποιηθεί αλληλεπίδραση με το χρήστη (π.χ. χρήση εικονιδίου φωτογραφικής μηχανής οδηγεί σε ενέργεια που αφορά την χρήση κάμερας του κινητού).

Η Apple εφιστά προσοχή στη συνοχή των εικονιδίων σε όλο το εύρος της εφαρμογής. Η συνοχή αφορά το μέγεθος, τη σχεδιαστική λεπτομέρεια, το βάρος και την προοπτική των εικονιδίων. Σαν γενική οδηγία προτείνεται η συνοχή αυτή να επεκτείνεται και σε κείμενο που μπορεί να υπάρχει δίπλα ή κοντά από τα εικονίδια αυτά. Με τον τρόπο αυτό και με τη χρήση κατάλληλης στοίχισης, μπορεί να δοθεί έμφαση σε συγκεκριμένο περιεχόμενο και η γενική εμφάνιση της εφαρμογής βελτιώνεται σημαντικά. Στις

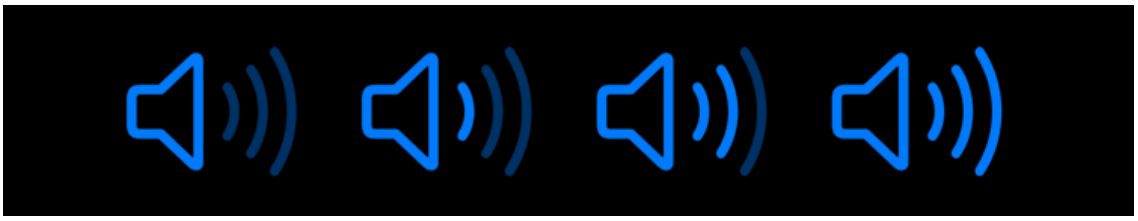
περιπτώσεις που αυτό είναι απαραίτητο, θα πρέπει να παρέχονται πολλαπλές μορφές ενός εικονιδίου, με βασικό παράδειγμα την επιλεγμένη και μη επιλεγμένη μορφή.

Σαν τεχνική οδηγία, η Apple συνιστά όλα τα εικονίδια της εφαρμογής να παρέχονται σε διανυσματική μορφή (PDF ή SVG). Ο λόγος είναι ότι το λειτουργικό σύστημα μπορεί να επιλύσει αυτόματα θέματα κλίμακας χωρίς να τίθεται κίνδυνος να επηρεαστεί η ανάλυση του εικονιδίου. Σε αντίθετη περίπτωση, θα πρέπει το εικονίδιο να παρέχεται σε πολλαπλές διαστάσεις, τόσο υψηλής όσο και χαμηλής ανάλυσης έτσι ώστε να μπορεί να χρησιμοποιηθεί σε όλο το εύρος των κινητών συσκευών.

2.3.3.1.3 SF Symbols

Τα SF Symbols αποτελούν ένα σύνολο από χιλιάδες εικονίδια (σύμβολα) τα οποία ακολουθούν όλες τις απαραίτητες προδιαγραφές που αναφέρθηκαν παραπάνω. Ένα από τα σημαντικότερα χαρακτηριστικά τους είναι η το πόσο παραμετροποιήσιμα είναι, καθιστώντας τα ικανά να ικανοποιήσουν ένα μεγάλο εύρος απαιτήσεων. Οι τρόποι παραμετροποίησης που είναι διαθέσιμοι είναι:

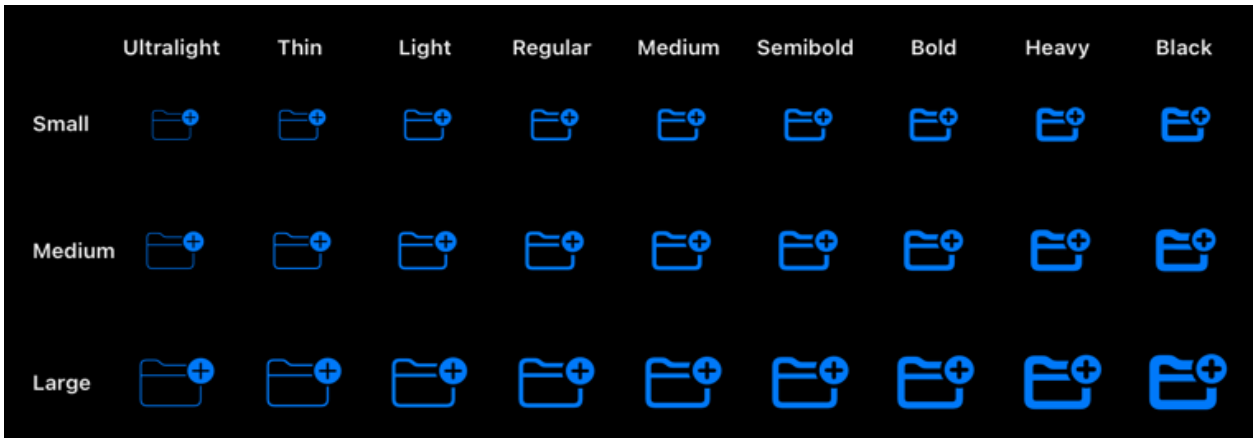
- **Rendering Modes**. Δίνεται η δυνατότητα χρήσης τεσσάρων rendering modes. Τα σύμβολα χωρίζονται σε ένα σύνολο από βαθμίδες (layers), στις οποίες εφαρμόζεται το επιθυμητό rendering mode, διαφοροποιώντας έτσι την εμφάνιση του εικονιδίου. Τα rendering modes που διατίθενται είναι το μονόχρωμο, ιεραρχικό, παλέτας και πολύχρωμο.
- **Variable Color**. Με τη χρήση του variable color, ένα σύμβολο μπορεί να αλλάζει ένα ή περισσότερα χαρακτηριστικά του, ανεξάρτητα από το rendering mode του. Η αλλαγή αυτή είναι μετρήσιμη σε ποσοστό επί τοις 100 (0% - 100%) και εφαρμόζεται σε μια ή περισσότερες βαθμίδες του εικονιδίου. Χαρακτηριστικό παράδειγμα αποτελεί το σύμβολο speaker.wave.3, το οποίο με τη χρήση του variable color παρουσιάζει την αυξομείωση στην ένταση, όπως φαίνεται στο Σχήμα 2.6.



Σχήμα 2.6 Speaker.wave.3 με τη χρήση variable color

- **Weights and Scales**. Τα SF Symbols παρέχουν ένα σύνολο από διαθέσιμα βάρη, αντίστοιχα με αυτά της γραμματοσειράς San Francisco, καθώς και τρία διαθέσιμα μεγέθη (small, medium, large). Στο Σχήμα 2.7 και στο Σχήμα 2.8 φαίνονται τα διαθέσιμα μεγέθη και βάρη των SF Symbols αντίστοιχα.

Σχήμα 2.7 Διαθέσιμα μεγέθη



Σχήμα 2.8 Διαθέσιμα βάρη

- **Animations**. Τα animations που παρέχονται από τα SF Symbols προσδίδουν ποικιλία και εκφραστικότητα στην εφαρμογή, αυξάνοντας σημαντικά την εμπειρία του χρήστη. Τα animations μπορούν να εφαρμοστούν σε οποιοδήποτε σύμβολο, ασχέτως από οποιαδήποτε άλλη παραμετροποίηση η οποία μπορεί να έχει εφαρμοστεί σε αυτό. Το λειτουργικό δίνει πλήρη έλεγχο στον προγραμματιστή, ο οποίος μπορεί να παραμετροποιήσει τη διάρκεια του animation, τον αριθμό των φορών που θα εκτελεστεί, την αρχή και το τέλος εκτέλεσης καθώς και την εκτέλεση υπό συγκεκριμένη συνθήκη. Για περισσότερες πληροφορίες και παραδείγματα από διαφορετικά ήδη animation, παραπέμπουμε στο documentation της Apple [7].

2.3.3.2 Light – Dark Mode

Για το μεγαλύτερο μέρος των χρηστών, είναι πλέον σύνηθες να έχουν ως προεπιλογή ενεργοποιημένο το dark mode ως στιλ διεπαφής. Συνεπώς ένας χρήστης που χρησιμοποιεί κατά κόρων το dark mode, περιμένει και οι εφαρμογές που χρησιμοποιεί να έχουν προνοήσει και να διαθέτουν αντίστοιχη υλοποίηση. Η διαφορά του dark mode σε σχέση με το light mode, είναι ότι το πρώτο χρησιμοποιεί μια σκουρόχρωμη χρωματική παλέτα, η οποία παρέχει ευκολία και άνεση στην ανάγνωση, διατηρώντας τα μάτια ξεκούραστα ακόμη και εκτεταμένη χρονικά χρήση. Ειδικότερα σε περιπτώσεις όπου το περιβάλλον έχει χαμηλό φωτισμό, το dark mode καθιστά τη θέαση αρκετά πιο ξεκούραστη.

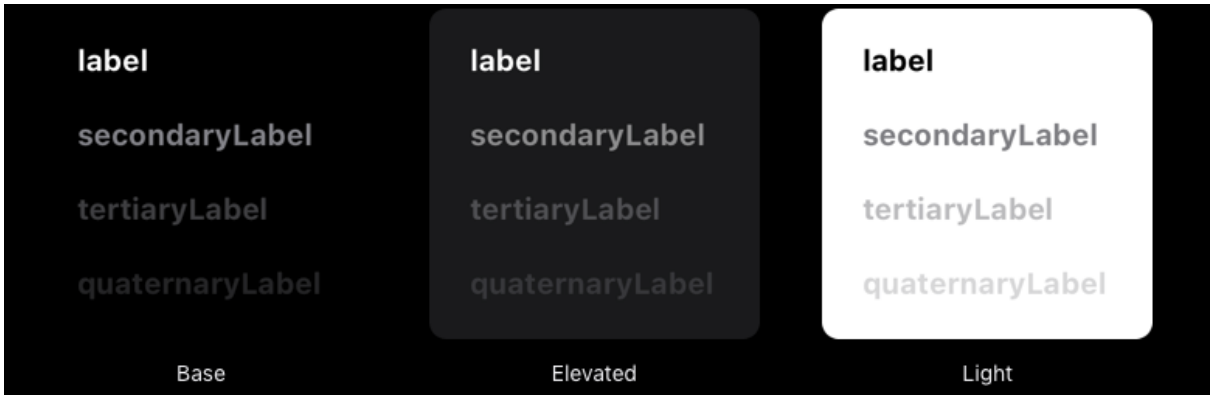
Η σημαντικότερη συμβουλή της Apple σχετικά με το dark mode είναι και η προφανής. Θα πρέπει να υπάρχει η απαραίτητη μέριμνα έτσι ώστε η εφαρμογή να λειτουργεί άπταιστα στις αλλαγές μεταξύ light και dark mode. Αυτό θα πρέπει να γίνεται ακόμη και δυναμικά, μιας και ο χρήστης έχει την επιλογή αφήσει το σύστημα να επιλέγει σε πραγματικό χρόνο το στιλ θέασης, ανάλογα με τις συνθήκες φωτισμού. Με βάση αυτό, η εφαρμογή θα πρέπει να είναι έτοιμη στο να αλλάζει στιλ θέασης, ακόμη και όταν είναι ανοιχτή. Πρέπει να δοθεί ιδιαίτερη έμφαση στο συνδυασμό των χρωμάτων παρασκηνίου και των χρωμάτων βασικών στοιχείων της εφαρμογής (labels, buttons), έτσι ώστε όταν το χρώμα παρασκηνίου σκουραίνει σε dark mode, τότε να υπάρχει αρκετή αντίθεση για να είναι ευανάγνωστες όλες οι πληροφορίες που φαίνονται στην οθόνη εκείνη τη στιγμή.

Η Apple συνιστά να μην υπάρχει εσωτερική ρύθμιση στην εφαρμογή για αλλαγή από light σε dark mode. Η εφαρμογή θα πρέπει να αναγνωρίζει τη γενική επιλογή του χρήστη και να συμμορφώνεται στο ανάλογο στυλ, ακόμη και σε πραγματικό χρόνο. Έτσι ο χρήστης δεν μπαίνει στη διαδικασία να κάνει ακόμη μια ρύθμιση και απολαμβάνει την εμπειρία που προσφέρει η εφαρμογή απρόσκοπτα. Σε σπάνιες

περιπτώσεις και ανάλογα με το είδος της εφαρμογής, συνίσταται η υποστήριξη μόνο του dark mode και όχι του light mode. Αυτό βέβαια αφαιρεί δυνατότητα παραμετροποίησης από το χρήστη, οπότε θα πρέπει να γίνεται μόνο όταν πραγματικά ταιριάζει με το ύφος και το περιεχόμενο της εφαρμογής.

Με τη χρήση χρωμάτων και εικονιδίων του συστήματος (System colors, SF Symbols) η μετάβαση από light σε dark mode και αντίστροφα γίνεται αυτόματα από την apple. Έτσι ο προγραμματιστής χρειάζεται να γράψει ελάχιστο επιπλέον κώδικα για να υποστηρίζει και τα δύο στιλ θέασης. Στην περίπτωση όμως που χρησιμοποιούνται custom χρώματα και εικόνες πρέπει να δοθεί έξτρα προσοχή, για να παρέχεται μια ολοκληρωμένη εμπειρία διεπαφής στο χρήστη.

Ειδικότερα σε iOS συσκευές, συνίσταται η χρήση των System Colors, ιδιαίτερα για τα χρώματα παρασκηνίου, λόγω της ιδιαιτερότητας του dark mode στις συγκεκριμένες συσκευές. Το σύστημα στις συσκευές αυτές χρησιμοποιεί δύο σετ χρωματικών παλετών το βασικό (base) και το υπερυψωμένο (elevated). Τα set αυτά εναλλάσσονται αυτόματα από το σύστημα και προσδίδουν καλύτερα την αίσθηση του βάθους. Στο base set τα χρώματα είναι πιο θαμπά, με αποτέλεσμα οι διεπαφές παρασκηνίου να φαίνεται ότι υποχωρούν, ενώ στο elevated set τα χρώματα είναι πιο φωτεινά, δίνοντας την αντίθετη αίσθηση. Τα παραπάνω φαίνονται και στο Σχήμα 2.9.



Σχήμα 2.9 Base – Elevated – Light set

Με τη χρήση των χρωμάτων του συστήματος, η εναλλαγές μεταξύ των παραπάνω set γίνονται αυτόματα, ακόμη και όταν η εφαρμογή είναι στο προσκήνιο. Στην περίπτωση των custom χρωμάτων, η διαχείριση θα πρέπει να γίνεται από τον προγραμματιστή με τέτοιο τρόπο, ώστε να προσομοιάζει την αντίστοιχη λειτουργικότητα του συστήματος. Για τη σωστή υλοποίηση και προσαρμογή της εφαρμογής στα σωστά mode, η apple παρέχει το εξής βίντεο [8].

2.3.4 Βέλτιστες Πρακτικές και Μοτίβα

Κατά την ανάπτυξη οποιασδήποτε εφαρμογής, προκύπτουν συγκεκριμένα σενάρια χρήσης τα οποία στις περισσότερες περιπτώσεις είναι κοινά μεταξύ των εφαρμογών. Παραδείγματος χάριν, ένα σύνηθες σενάριο χρήσης αποτελεί την εισαγωγή δεδομένων από το χρήστη. Για τέτοιου είδους συνήθη μοτίβα, η Apple παρέχει την κατάλληλη καθοδήγηση έτσι ώστε να τηρούνται οι βέλτιστες πρακτικές και να ενισχύεται η εμπειρία του χρήστη. Η δική μας εφαρμογή δεν αποτελεί εξαίρεση, συνεπώς παρακάτω αναλύονται κάποια βασικά μοτίβα τα οποία υπάρχουν και στο fitness app, και σχεδιάστηκαν με βάση τις οδηγίες που παρέχει η Apple. Τα μοτίβα παρατίθενται με τη σειρά που εμφανίζονται και στην εφαρμογή, και είναι τα παρακάτω:

- **Διαχείριση Λογαριασμών**

- **Εισαγωγή Δεδομένων - Onboarding**
- **Διαχείριση Ειδοποιήσεων**
- **Ρυθμίσεις**
- **Διαγραμματική Απεικόνιση Δεδομένων**

2.3.4.1 Διαχείριση Λογαριασμών

Αν ο χρήστης αποφασίσει να αξιοποιήσει πλήρως τις δυνατότητες που του προσφέρει το fitness app, είναι απαραίτητο να δημιουργήσει λογαριασμό. Ο τρόπος που θα παρουσιάσουμε αυτή τη δυνατότητα στο χρήστη, καθώς και οι επιλογές που του παρέχονται για την σύντομη διεκπεραίωση της διαδικασίας αυτής, θα παίξουν καθοριστικό ρόλο στην εμπειρία του.

Κατά κανόνα, η δημιουργία λογαριασμού θα πρέπει να προσφέρεται στο χρήστη σαν επιλογή, μόνο όταν είναι απαραίτητη για τις λειτουργίες της εκάστοτε εφαρμογής. Ιδανικά θα ήταν καλό ο χρήστης να μπορεί να χρησιμοποιήσει την κάθε εφαρμογή σε όλο της το εύρος χωρίς τη χρήση λογαριασμού. Ακόμα και στις περιπτώσεις, όπως η δική μας, όπου η ύπαρξη λογαριασμού κρίνεται σχεδόν απαραίτητη, θα πρέπει να εξηγήσουμε στο χρήστη με τον καλύτερο δυνατό τρόπο ποια είναι η εξατομικευμένη εμπειρία που θα του παρέχουμε με τη χρήση του λογαριασμού του. Μια πολύ καλή στρατηγική κίνηση επίσης, είναι να καθυστερήσουμε την εγγραφή / σύνδεση του χρήστη όσο το δυνατόν περισσότερο. Με τον τρόπο αυτό ο χρήστης μπορεί να δοκιμάσει κάποιες από τις βασικές λειτουργίες της εφαρμογής κατευθείαν και σε περίπτωση που μείνει ικανοποιημένος, να δημιουργήσει το λογαριασμό του έτσι ώστε να επωφεληθεί από το πλήρες εύρος λειτουργιών που του παρέχονται. Με τις παραπάνω σχεδιαστικές αποφάσεις, ενημερώνουμε το χρήστη γιατί να δημιουργήσει λογαριασμό και του παρέχουμε τη δυνατότητα δοκιμής της εφαρμογής χωρίς λογαριασμό, προσφέροντας έτσι ευελιξία και εμπνέοντας εμπιστοσύνη.

Επειδή ο λογαριασμός ενός χρήστη ουσιαστικά αποτελεί ένα σύνολο από προσωπικά του δεδομένα, πρέπει να δοθεί πολύ μεγάλη έμφαση στο κομμάτι της ασφάλειας. Ίσως ο καλύτερος αλλά και πιο σύντομος τρόπος εξασφάλισης μιας ασφαλούς διαχείρισης ενός λογαριασμού, είναι η χρήση εξωτερικής αυθεντικοποίησης (external authentication). Με τη χρήση του external authentication, ο μηχανισμός αυθεντικοποίησης πραγματοποιείται σε ξεχωριστό server, από πιστοποιημένους οργανισμούς οι οποίοι εξασφαλίζουν την ασφάλεια και την επικαιροποίηση του χρήστη. Η χρήση αυτής της μεθόδου αυθεντικοποίησης είναι πλέον η προτιμητέα τόσο από τους χρήστες όσο και από τους σχεδιαστές, διότι προσφέρει αμεσότητα και ασφάλεια. Στη δική μας περίπτωση, χρησιμοποιούμε το μηχανισμό αυθεντικοποίησης της google (Google Sign In) για την είσοδο στην εφαρμογή.

Εφόσον παρέχεται η δυνατότητα δημιουργίας λογαριασμού σε μια εφαρμογή, κρίνεται απαραίτητο να παρέχεται και η δυνατότητα διαγραφής αυτού. Για το συγκεκριμένο κομμάτι υπάρχει συγκεκριμένο νομικό πλαίσιο το οποίο διέπει τις ενέργειες οι οποίες πρέπει να παρθούν κατά τη διαγραφή ενός λογαριασμού και της αποδέσμευσης των προσωπικών δεδομένων του χρήστη. Σε κάθε περίπτωση, τα βήματα που θα ακολουθήσει ο χρήστης για να διαγράψει το λογαριασμό του θα πρέπει να είναι ξεκάθαρα σε αυτόν και όπως ο χρήστης ενημερώνεται για την επιτυχή δημιουργία του λογαριασμού του, έτσι θα πρέπει να ενημερώνεται και για την επιτυχή διαγραφή αυτού.

Σε γενικές γραμμές, η διαχείριση λογαριασμών θα πρέπει να είναι μια σύντομη και ξεκάθαρη διαδικασία η οποία χαρακτηρίζεται από διαφάνεια και συνέπεια, τόσο στη δημιουργία όσο και στη διαγραφή.

2.3.4.2 Εισαγωγή Δεδομένων – Onboarding

Στην περίπτωση του fitness app, όταν ο χρήστης πραγματοποιήσει με επιτυχία την εγγραφή του, του ζητούνται κάποια δεδομένα με βάση τα οποία δημιουργούμε εξατομικευμένη εμπειρία κατά τη χρήση. Συνήθως η εισαγωγή δεδομένων προκαλεί δυσαρέσκεια στο χρήστη και αν ο χειρισμός της είναι κακός μπορεί να οδηγήσει ακόμη και στη διαγραφή της εφαρμογής. Με τη χρήση συγκεκριμένων προτεινόμενων πρακτικών, προσπαθούμε να κάνουμε τη διαδικασία αυτή όσο πιο σύντομη και ευχάριστη για το χρήστη.

Μια από τις πρακτικές αυτές είναι να βεβαιωθούμε ότι δεν μπορούμε να αντλήσουμε την επιθυμητή πληροφορία από αλλού, πριν ζητήσουμε την εισαγωγή της. Για παράδειγμα, κατά την επιτυχή σύνδεση με Google, μπορούμε να αντλήσουμε στοιχεία του χρήστη όπως το email ή το ονοματεπώνυμο του κατευθείαν από τα metadata, αν ο χρήστης έχει δώσει τις αντίστοιχες άδειες (permissions) στο google λογαριασμό του. Έτσι μπορούμε ιδανικά να αποφύγουμε να ζητήσουμε τη πληροφορία αυτή κατά το onboarding, γλιτώνοντας έτσι ένα βήμα και κάνοντας τη διαδικασία συντομότερη.

Συμπληρωματικά, μπορούμε να προσυμπληρώσουμε κάποια πεδία που ζητούνται με κάποιο παράδειγμα αποδεκτής τιμής. Αυτό κάνει τη διαδικασία πιο κατανοητή και κατά συνέπεια την επιταχύνει. Στο ίδιο πλαίσιο, είναι σημαντικό να χρησιμοποιούνται και άλλες τεχνικές, όπως η δυναμική επικύρωση τιμών κατά την εισαγωγή τους, η χρήση ασφαλών στοιχείων εισαγωγής (secure text-entry fields) για τα ευαίσθητα δεδομένα και η δυνατότητα συμπλήρωσης πεδίων με εναλλακτικούς τρόπους πέρα από κείμενο (χρήση drop down menus, picker views). Όλα τα παραπάνω παρέχουν στο χρήστη βελτιστοποιημένη εμπειρία σε μια διαδικασία η οποία συνήθως δεν είναι ευχάριστη για αυτόν.

Για την εισαγωγή δεδομένων το λειτουργικό παρέχει ένα σύνολο στοιχείων τα οποία χρησιμοποιούνται εκτενώς στην εφαρμογή, τόσο στο onboarding όσο και σε εσωτερικές λειτουργίες όπως φαίνεται και στα παρακάτω screenshot. Αναφορικά γίνεται χρήστη TextField, Keyboard, PickerView.

Πέρα από τη διαδικασία της εισαγωγής δεδομένων, κατά το onboarding συνήθως παρέχονται στο χρήστη και οι απαραίτητες εξηγήσεις σχετικά με τον τρόπο χρήσης της εφαρμογής. Ιδανικά ο χρήστης θα πρέπει να καταλαβαίνει με φυσικό τρόπο τις λειτουργίες της εφαρμογής κατά τη χρήση, πολλές φορές όμως κρίνεται απαραίτητο να του εξηγηθούν περαιτέρω κάποιες λειτουργίες. Ακολουθώντας τα προηγούμενα μοτίβα σχεδίασης, σε περίπτωση που οι οδηγίες χρήσης παρέχονται με τη μορφή wizard, τότε αυτό θα πρέπει να είναι συνοπτικό, ξεκάθαρο και με δυνατότητα παράλειψης ανά πάσα στιγμή. Το wizard κατά κανόνα εμφανίζεται μόνο την πρώτη φορά κατά την εκκίνηση της εφαρμογής στο onboarding και έπειτα στο προφίλ ή στις ρυθμίσεις της εφαρμογής, έτσι ώστε ο χρήστης να μπορεί να το ξαναπαρακολουθήσει on demand σε περίπτωση που το επιθυμεί.

Ένας εναλλακτικός τρόπος καθοδήγησης, ο οποίος πολλές φορές είναι και προτιμητέος διότι είναι πιο διαδραστικός, είναι η εμφάνιση μικρών υποδείξεων όσο ο χρήστης χρησιμοποιεί φυσικά την εφαρμογή. Ειδικά αν η εμφάνιση των υποδείξεων αυτών γίνεται με περίτεχνο και ενδιαφέρον τρόπο (χρήση animations) μπορεί να κεντρίσει το ενδιαφέρον του χρήστη και να εμπνεύσει ενθουσιασμό. Φυσικά μια τέτοιο είδους προσέγγιση απαιτεί προσεκτική και καλόγουστη διαχείριση, έτσι ώστε να εκπληρώνει το σκοπό της ο οποίος είναι να συμπληρώσει / υποδείξει λειτουργίες και όχι να αποσπάσει την προσοχή του χρήστη καταλήγοντας να τον μπερδεύει.

Σαν συμπληρωματική συμβουλή για το onboarding, η Apple προτείνει οποιαδήποτε πληροφορία αφορά άδειες χρήσης, αποποίησης ευθυνών (disclaimers) κ.ο.κ να μην εμφανίζεται στο onboarding, αλλά να παρέχεται τόσο στο store έτσι ώστε ο χρήστης να μπορεί να τις διαβάσει πριν κατεβάσει την εφαρμογή,

αλλά και σε κατάλληλο τμήμα εντός αυτής (συνήθως στο more / περισσότερα ή στα settings / ρυθμίσεις).

2.3.4.3 Διαχείριση Ειδοποιήσεων

Οι ειδοποιήσεις είναι ένα πολύ χρήσιμο εργαλείο μιας εφαρμογής, διότι παρέχουν στο χρήστη χρήσιμες πληροφορίες ακόμη και όταν η συσκευή είναι κλειδωμένη. Ως αντεπιχείρημα, επειδή μια ειδοποίηση αποσπά και διακόπτει το χρήστη από τη ροή χρήσης που ακολουθεί τη δεδομένη στιγμή, απαιτείται ο απαραίτητος χειρισμός έτσι ώστε ο χρονισμός για την κάθε ειδοποίηση να είναι πάντοτε ο κατάλληλος.

Ο βασικότερος κανόνες ο οποίος πρέπει να τηρηθεί χωρίς εξαιρέσεις και αφορά τις ειδοποιήσεις, είναι ότι **η εφαρμογή χρειάζεται την άδεια (permission)** του χρήστη **για να στέλνει ειδοποιήσεις**. Η άδεια αυτή είναι παραμετροποιήσιμη από τις ρυθμίσεις της συγκεκριμένης εφαρμογής στις ρυθμίσεις του συστήματος. Συνεπώς είναι επιλογή του χρήστη το αν θα λαμβάνει ειδοποιήσεις από την εφαρμογή, όσο σημαντικές και αν θεωρούμε εμείς ότι είναι για την εμπειρία του.

Η Apple κατηγοριοποιεί τις ειδοποιήσεις τις σε δύο βασικές κατηγορίες: τις **communication** και τις **noncommunication.**

- **Communication Notifications**. Είναι όλες οι ειδοποιήσεις που σχετίζονται με επικοινωνία (όπως τηλεφωνήματα ή μηνύματα)
- **Noncommunication** Notifications. Είναι όλα τα υπόλοιπα είδη ειδοποιήσεων.

Στη δική μας εφαρμογή χρησιμοποιούμε καθαρά noncommunication ειδοποιήσεις, συνεπώς δεν θα ασχοληθούμε άλλο με τις communication μιας και δεν είναι της παρούσης. Οι noncommunication ειδοποιήσεις χαρακτηρίζονται από ένα επίπεδο διακοπής (interruption level), το οποίο το σύστημα χρησιμοποιεί για να καθορίσει το χρονισμό με τον οποίο θα παραδώσει την ειδοποίηση. Τα επίπεδα διακοπής που παρέχονται είναι τέσσερα:

- Παθητικό / Passive. Χαρακτηρίζει ειδοποιήσεις των οποίων οι πληροφορίες δεν είναι τόσο καίριες για το χρήστη, συνεπώς μπορεί να τις δει αργότερα στον ελεύθερό του χρόνο (π.χ. σύσταση για ένα εστιατόριο).
- Ενεργό / Active. Αποτελεί το default επίπεδο διακοπής. Χαρακτηρίζει ειδοποιήσεις των οποίων οι πληροφορίες είναι αρκετά σημαντικές για το χρήστη, έτσι ώστε αν εκείνος τις λάβει το συντομότερο δυνατό να το εκτιμήσει και να μην νιώσει ότι διακόπτεται (π.χ σε μια εφαρμογή θέασης αθλητικών αγώνων, μια αλλαγή στο σκορ ενός αγώνα).
- Χρονικής Ευαισθησίας / Time sensitive. Στο επίπεδο αυτό η πληροφορία της ειδοποίησης είναι πολύ σημαντική, σε βαθμό που πρέπει να γίνει γνωστή στο χρήστη άμεσα ακόμη και αν αυτό τον διακόψει από άλλες ενέργειες (π.χ. παραβίαση τραπεζικού λογαριασμού, παραλαβή ενός πακέτου κ.α.).
- Κρίσιμες / Critical. Κρίσιμες ειδοποιήσεις χαρακτηρίζονται αυτές των οποίων οι πληροφορίες είναι ζωτικής σημασίας και αφορούν την υγεία και την ασφάλεια του χρήστη. Τέτοιου είδους ειδοποιήσεις συνήθως στέλνονται από κυβερνητικούς παράγοντες ή από εφαρμογές που έχουν να κάνουν με ασφάλεια (π.χ εφαρμογή διαχείρισης έξυπνου σπιτιού) και γενικότερα είναι ιδιαίτερα σπάνιες.

Στο Σχήμα 2.10 φαίνεται η συμπεριφορά μιας ειδοποίησης ανάλογα με το επίπεδο διακοπής της.

Interruption level	Overrides scheduled delivery	Breaks through Focus	Overrides Ring/Silent switch on iPhone and iPad
Passive	No	No	No
Active	No	No	No
Time Sensitive	Yes	Yes	No
Critical	Yes	Yes	Yes

Σχήμα 2.10 Συμπεριφορά ειδοποίησης και επίπεδο διακοπής

Με βάση τα παραπάνω, είναι προφανές ότι ο χρονισμός μιας ειδοποίησης είναι ένα ζήτημα το οποίο χρήζει προσοχή και σοβαρότητα. Η κάθε εφαρμογή πρέπει να εκτιμά σωστά τη σημασία των ειδοποιήσεων της διότι έτσι δείχνει ενεργά ότι εκτιμά το χρόνο του χρήστη. Στη δική μας περίπτωση, όλες οι ειδοποιήσεις που στέλνουμε ανήκουν στο ενεργό επίπεδο διακοπής, μιας και είναι σημαντικές έτσι ώστε ο χρήστης να τις λαμβάνει άμεσα, όχι αρκετά σημαντικές όμως έτσι ώστε να παρακάμπτει τον τρόπο με τον οποίο ο ίδιος επιθυμεί να λαμβάνει ειδοποιήσεις.

2.3.4.4 Ρυθμίσεις

Ένα πολύ θετικό στοιχείο σε μια εφαρμογή, είναι να προσφέρεται παραμετροποίηση σε διάφορα σημεία της έτσι ώστε ο χρήστης να προσαρμόζει το περιεχόμενο και τις λειτουργίες της με βάση τις δικές του ανάγκες. Από την άλλη, ο κάθε χρήστης έχει την απαίτηση από μια εφαρμογή να λειτουργεί σωστά, χωρίς να χρειάζεται να κάνει ο ίδιος επιπλέον ενέργειες. Συνεπώς, όταν κρίνεται απαραίτητο ή όταν έχει να προσφέρει κάτι, είναι καλό να παρέχονται ρυθμίσεις σε μια εφαρμογή. Κατά τη σχεδίαση όμως πρέπει να λάβουμε υπόψη ότι ο χρήστης σπανίως επισκέπτεται την οθόνη των ρυθμίσεων (settings), έτσι οι ρυθμίσεις πρέπει να ενσωματώνονται κατάλληλα και να χρησιμοποιούνται με φειδώ.

Μια βασική οδηγία που παρέχεται από την Apple σχετικά με τις ρυθμίσεις, είναι να υπάρχουν διαθέσιμες στο χρήστη στο σημείο όπου αυτές επηρεάζουν. Για να γίνει πιο κατανοητό το παραπάνω, υποθέτουμε ότι υπάρχει μια λίστα από περιεχόμενο, το οποίο είναι παραμετροποιήσιμο έτσι ώστε ο χρήστης να μπορεί να το φιλτράρει με βάση τις δικές του προτιμήσεις. Η ρύθμιση για το φιλτράρισμα αυτό θα πρέπει να βρίσκεται στην ίδια οθόνη, κοντά στη λίστα την οποία αφορά έτσι ώστε να είναι απευθείας κατανοητό από το χρήστη.

Γενικότερα, είναι κακή πρακτική ο χρήστης να κατευθύνεται σε οθόνη ρυθμίσεων για τέτοιου είδους ενέργειες. Η οθόνη την ρυθμίσεων εξυπηρετεί για ρυθμίσεις που αφορούν το σύνολο της εφαρμογής, όπως για παράδειγμα την αλλαγή θέματος (theme). Σε περίπτωση που υπάρχει μια τέτοια οθόνη στην εφαρμογή (τονίζεται ότι δεν είναι απαραίτητο μια εφαρμογή να έχει οθόνη ρυθμίσεων) πρέπει να δοθεί προσοχή και στον αριθμό των διαθέσιμων επιλογών προς ρύθμιση. Πολλαπλές επιλογές παραμετροποίησης συνήθως επιφέρουν περισσότερα αρνητικά απ’ ότι θετικά αποτελέσματα. Ο χρήστης μπερδεύεται από το βομβαρδισμό πληροφορίας και ξοδεύει πολύ χρόνο ρυθμίζοντας την εφαρμογή και λιγότερο αξιοποιώντας την. Επίσης, όσο περισσότερες είναι οι ρυθμίσεις τόσο δυσκολότερο καθίσταται για το χρήστη η αναζήτηση και η εύρεση μιας συγκεκριμένης.

Αν και προφανές, είναι πολύ σημαντικό επίσης να μην μπερδεύουμε τις ρυθμίσεις που παρέχει το λειτουργικό για μια εφαρμογή με τις εσωτερικές ρυθμίσεις εφαρμογής. Ρυθμίσεις που παρέχονται σε

επίπεδο συστήματος δεν θα πρέπει να υπάρχουν και στο εσωτερικό της εφαρμογής. Παράδειγμα τέτοιας ρύθμισης είναι η διαχείριση τοποθεσίας χρήστη και γενικότερα η ρύθμιση για τα location services.

2.3.4.5 Διαγραμματική Απεικόνιση Δεδομένων

Η διαγραμματική απεικόνιση δεδομένων αποτελεί έναν αποτελεσματικό τρόπο μετάδοσης περίπλοκης πληροφορίας χωρίς να κουράζει το χρήστη. Λόγω της γραφικότητας των διαγραμμάτων, μπορούμε να επικοινωνήσουμε ευκολότερα και να εκφράσουμε με ενδιαφέρον τρόπο την πληροφορία που θέλουμε να μεταδώσουμε, κάνοντας την εμπειρία πιο ενδιαφέρουσα και ενισχύοντας τη διεπαφή χρήστη. Συνήθως χρησιμοποιούμε διαγράμματα με σκοπό:

- Να αναλύσουμε δεδομένα βάση ιστορικού και να εξάγουμε προτιμήσεις (trends)
- Να οπτικοποιήσουμε την πρόοδο σε ένα στόχο, την αλλαγή ποσότητας μιας τιμής ή γενικότερα τη μεταβολή κάποιας μεταβλητής.

Επειδή τα διαγράμματα καλύπτουν συνήθως μεγάλο μέρος της οθόνης, αλλά και λόγω της ίδιας τους της φύσης είναι έντονα και χτυπητά στο χρήστη, θα πρέπει να χρησιμοποιούνται μόνο σε πληροφορίες μεγάλης σημασίας, άξιες προσοχής. Για άλλη μια φορά, θα πρέπει να δοθεί έμφαση στην αμεσότητα και την απλότητα της πληροφορίας ενός διαγράμματος. Σε περίπτωση που ένα διάγραμμα θα είναι υπεύθυνο να μεταδώσει μεγάλο όγκο πληροφορίας, είναι προτιμότερο να δίνεται η δυνατότητα στον χρήστη να πλοηγείται σε κομμάτια της πληροφορίας σταδιακά. Στην άλλη περίπτωση, μεγάλος όγκος πληροφορίας μπορεί να μπερδέψει τον χρήστη και να τον οδηγήσει σε σύγχυση.

Κατά κανόνα, είναι καλό να χρησιμοποιούνται τα κοινότυπα διαγράμματα που παρέχονται από το σύστημα. Ο λόγος έχει να κάνει με την αναγνωρισιμότητα τους από το χρήστη, ο οποίος νιώθει άνετα βλέποντας και χρησιμοποιώντας κάτι γνώριμο. Σε περίπτωση που για δημιουργικούς ή άλλους λόγους κρίνεται απαραίτητο να παρεκκλίνουμε από τα κοινότυπα διαγράμματα, είναι σημαντικό να παρέχουμε και τις απαραίτητες οδηγίες, έτσι ώστε ο χρήστης να μπορεί να ερμηνεύσει το διάγραμμα το οποίο βλέπει. Εφόσον αποφασιστεί το είδος του διαγράμματος που θα χρησιμοποιηθεί και αντιστοιχιστεί με την εκάστοτε πληροφορία, τότε θα πρέπει να υπάρχει συνοχή και σε επόμενες χρήσεις του. Η χρήση διαφορετικών διαγραμμάτων για την περιγραφή της ίδιας πληροφορίας θα οδηγήσει σε το χρήστη σε σύγχυση και θα τον μπερδέψει. Η Apple, πέρα από τις παραπάνω οδηγίες παρέχει και τις εξής οδηγίες για τη χρήση διαγραμμάτων, σε μορφή βίντεο [9] [10] [11].

Κεφάλαιο 3ο: Μεθοδολογία και Σχεδιάστηκες Αποφάσεις

Έχοντας πλέον καθορίσει το σκοπό και τις απαιτήσεις του Thesis Fitness App, πρέπει να πραγματοποιηθεί ο κατάλληλος σχεδιασμός για τα εργαλεία που θα χρησιμοποιηθούν κατά την ανάπτυξη του έργου στο πλήρες εύρος του. Τα κριτήρια επιλογής έχουν να κάνουν με μια πληθώρα παραγόντων. Ένας από αυτούς είναι η άμεση ελεύθερη διαθεσιμότητα (open source) των εργαλείων αυτών. Επίσης η αποδεδειγμένη χρήση σε άλλου είδους αντίστοιχων state-of-the-art εφαρμογών και γενικότερα η εδραίωση τους στο χώρο προσφέρουν σιγουριά και σταθερότητα στο βάθος της υλοποίησης. Τέλος η προσωπική προϋπάρχουσα γνώση και επαγγελματική εμπειρία χρήσης των εργαλείων αυτών είναι κάτι το οποίο θα επιταχύνει τη διαδικασία ανάπτυξης και θα αυξήσει την ποιότητα του τελικού αποτελέσματος.

Στο κεφάλαιο αυτό θα αναλυθούν οι διάφορες τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του πρακτικού μέρους (εφαρμογής) της διπλωματικής εργασίας. Οι τεχνολογίες αυτές μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες:

- Front-end Development: Τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της mobile εφαρμογής
- Back-end Development: Τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν για την διαχείριση των δεδομένων της εφαρμογής και τη δημιουργία της βάσης δεδομένων

Παραδείγματα και περιπτώσεις χρήσης των παραπάνω θα παρατεθούν σε επόμενο κεφάλαιο το οποίο θα αφορά την ανάπτυξη της εφαρμογής. Το παρόν κεφάλαιο αποτελεί ένα είδος οδηγού έτσι ώστε ο αναγνώστης να έχει μια καλή εικόνα σχετικά με όλα τα εργαλεία που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής. Αρχικά θα γίνει αναφορά στις γλώσσες προγραμματισμού που επιλέχθηκαν για την υλοποίηση και έπειτα σε ένα σύνολο από τεχνολογίες τόσο του Back-End όσο και του Front-End που συνολικά συνέβαλαν στην επίτευξη του τελικού στόχου.

3.1 Επιλογή Γλωσσών Προγραμματισμού

Για την ανάπτυξη στο κομμάτι του Back-End επιλέχθηκε η χρήση του Firestore Cloud Database, μιας document-based βάσης δεδομένων η οποία δεν απαιτεί τη χρήση κώδικα για την υλοποίηση της. Το γεγονός αυτό την καθιστά εύχρηστη και απλή στην κατανόηση ενώ συνάμα πληροί όλες τις απαραίτητες προδιαγραφές και καλύπτει τις ανάγκες της εφαρμογής. Η χρήση του Firestore Cloud Database είναι ελεύθερη με περιορισμούς στον αριθμό των αιτημάτων που μπορούν να εκτελεστούν στο διάστημα μιας μέρας, περιορισμοί οι οποίοι δεν μας εμποδίζουν στη συγκεκριμένη υλοποίηση η οποία για την ώρα δεν έχει πολλούς χρήστες και συνεπώς η δικτυακή κίνηση είναι σχετικά περιορισμένη.

Σχετικά με το περιβάλλον του Firestore καθώς και με τη δομή των οντοτήτων του γίνεται λόγος στο κεφάλαιο 4. Εφόσον λοιπόν για το Back-End δεν έγινε χρήση κάποιας γλώσσας προγραμματισμού, απομένει το κομμάτι της ανάπτυξης του Front-End, δηλαδή της εφαρμογής iOS.

3.1.1 Γλώσσα Προγραμματισμού Swift

Για την ανάπτυξη της mobile εφαρμογής επιλέχθηκε η γλώσσα προγραμματισμού Swift. Η Swift είναι ευρέως διαδεδομένη και χρησιμοποιείται από τη συντριπτική πλειοψηφία των προγραμματιστών εφαρμογών iOS. Είναι μια αντικειμενοστραφής γλώσσα, ειδικά σχεδιασμένη από την Apple με βάση τις απαιτήσεις του συνόλου των λειτουργικών συστημάτων της. Μερικά από τα πλεονεκτήματα της είναι ο μοντέρνος σχεδιασμός και η ευκολία στη χρήση (αυτόματη διαχείριση δεδομένων, απλότητα

στη διαχείριση του concurrency, loosely typed), η ελεύθερη πρόσβαση (open source), η δυνατότητα συνδυασμού με κώδικα Objective-C και C++ κ.α. Ένα άλλο σημαντικό πλεονέκτημα αποτελεί η πληθώρα του διαθέσιμου υλικού που υπάρχει, τόσο στη μορφή documentation της Apple όσο και ως οδηγούς και πληροφορίες από ανεξάρτητες πηγές. Το υλικό αυτό βοηθά την κατανόηση και τη σωστή χρήση της Swift, καθιστώντας έτσι ακόμη και αρκετά περίπλοκα tasks ευκολότερα στην επίλυση. Η ανάπτυξη του Thesis Fitness App έγινε με την έκδοση Swift 5.0 [12]. Περισσότερες πληροφορίες σχετικά με τη γλώσσα προγραμματισμού Swift υπάρχουν και στο επίσημο documentation της Apple [13].

Σχήμα 3.1 Εικονίδιο Γλώσσας προγραμματισμού Swift και Cloud Firestore Database

3.2 Επιλογή Εργαλείων - Τεχνολογιών

Η Swift και η βάση δεδομένων Cloud Firestore αποτελούν τα δύο βασικά εργαλεία ανάπτυξης της εφαρμογής. Πέρα από αυτά όμως χρησιμοποιήθηκαν και μια πληθώρα από άλλες τεχνολογίες τόσο στο Front-End όσο και στο Back-End οι οποίες είναι άξιες μνείας και αναλύονται στη συνέχεια.

3.2.1 Τεχνολογίες Back-End

Οι τεχνολογίες που χρησιμοποιήθηκαν και αφορούν το Back-End, πέρα από τη βάση δεδομένων Cloud Firestore, είναι οι παρακάτω.

3.2.1.1 HTTP Request

Η ανταλλαγή δεδομένων από τη βάση προς την mobile εφαρμογή και αντίστροφα επιτυγχάνεται με τη χρήση του πρωτοκόλλου HTTP και συγκεκριμένα μέσω HTTP Request. Ένα HTTP Request πραγματοποιείται από ένα client, στην περίπτωση μας τη mobile εφαρμογή, σε ένα server με σκοπό την πρόσβαση σε κάποια δεδομένα. Για την επίτευξη του request απαιτείται ένα URL ή αλλιώς μια αναφορά στο path στο οποίο βρίσκεται το δεδομένο στο οποίο ζητείται πρόσβαση [14]. Για τα request στη συγκεκριμένη υλοποίηση χρησιμοποιούνται βιβλιοθήκες οι οποίες απλουστεύουν τη διαδικασία ανάπτυξης. Για τη διαχείριση όλων των request προς τη βάση είναι υπεύθυνος η κλάση NetworkManager.

```
final class NetworkManager {

    // MARK: - Methods

    static func getDocument<ResponseType: Mappable>(_ responseType: ResponseType.Type, _
collection: CollectionTypeEnum, _ documentId: String, completion: @escaping
(Result<ResponseType, Error>) -> Void) {
```

```
        let db = Firestore.firestore()

        db.collection(collection.rawValue).document(documentId).getDocument { (document,
error) in
            if let error = error {
                completion(.failure(error))
            } else {
                if let document = document, document.exists, let data = document.data() {
                    guard let response = ResponseType.init(document: data) else {
                        return
                    }
                    completion(.success(response))
                }
            }
        }
    }

    static func getDocuments<ResponseType: Mappable>(_ responseType: ResponseType.Type, _
collection: CollectionTypeEnum, completion: @escaping (Result<([ResponseType], [String]),
Error>) -> Void) {
        let db = Firestore.firestore()

        db.collection(collection.rawValue).getDocuments { (querySnapshot, error) in
            if let error = error {
                completion(.failure(error))
            } else {
                var data: [ResponseType] = []
                var ids: [String] = []
                guard let documents = querySnapshot?.documents else {
                    return
                }
                documents.forEach { document in
                    if let object = ResponseType.init(document: document.data()) {
                        data.append(object)
                        ids.append(document.documentID)
                    }
                }
                completion(.success((data, ids)))
            }
        }
    }

    static func addDocument<DocumentType: Mappable>(_ document: DocumentType, _
collection: CollectionTypeEnum, completion: @escaping (Result<DocumentReference?, Error>)
-> Void) {
        let db = Firestore.firestore()
        var ref: DocumentReference?

        ref = db.collection(collection.rawValue).addDocument(data: document.toJSON()) {
err in
            if let error = err {
                completion(.failure(error))
            } else {
                completion(.success(ref))
            }
        }
    }

    static func updateDocument(with data: [String: Any], _ collection:
CollectionTypeEnum, _ documentId: String, completion: @escaping (Result<Bool, Error>) ->
Void) {
        let db = Firestore.firestore()

        db.collection(collection.rawValue).document(documentId).getDocument { (document,
error) in
```

```
            if let error = error {
                completion(.failure(error))
            } else {
                document?.reference.updateData(data)
                completion(.success(true))
            }
        }
    }
}
```

3.2.1.2 JSON

Μετά από ένα επιτυχημένο request στη βάση δεδομένων, το αποτέλεσμα που λαμβάνει η εφαρμογή είναι σε μορφή JSON (JavaScript Object Notation). Με τον όρο JSON αναφερόμαστε σε μια εδραιωμένη μορφή ανταλλαγής δεδομένων, η οποία είναι εύκολη στην κατανόηση από τους ανθρώπους αλλά και στη μετάφραση και τη δημιουργία από υπολογιστικά συστήματα [15]. Τα JSON δεδομένα μετατρέπονται σε μοντέλα της εφαρμογής μετά τη λήψη τους. Παρακάτω φαίνεται παράδειγμα από μια απάντηση αιτήματος στη βάση, σε μορφή JSON.


```
{
  "kind": "identitytoolkit#GetAccountInfoResponse",
  "users": [
    {
      "localId": "4OH8vBlJJHXpAIhOpxRQ1rQquAJ2",
      "email": "nerodantetm@gmail.com",
      "displayName": "Panos Tob",
      "photoUrl":
"https://lh3.googleusercontent.com/a/AAcHTtcj9nrSQbrkCKJkQ2V6W-
RXej8EF_xDG9SCdI2DJcf57wk=s96-c",
      "emailVerified": true,
      "providerUserInfo": [
        {
          "providerId": "google.com",
          "displayName": "Panos Tob",
          "photoUrl":
"https://lh3.googleusercontent.com/a/AAcHTtcj9nrSQbrkCKJkQ2V6W-
RXej8EF_xDG9SCdI2DJcf57wk=s96-c",
          "federatedId": "104297717395306981854",
          "email": "nerodantetm@gmail.com",
          "rawId": "104297717395306981854"
        }
      ],
      "validSince": "1692528374",
      "lastLoginAt": "1694094582265",
      "createdAt": "1692528374784",
      "lastRefreshAt": "2023-09-07T13:49:42.265Z"
    }
  ]
}
```

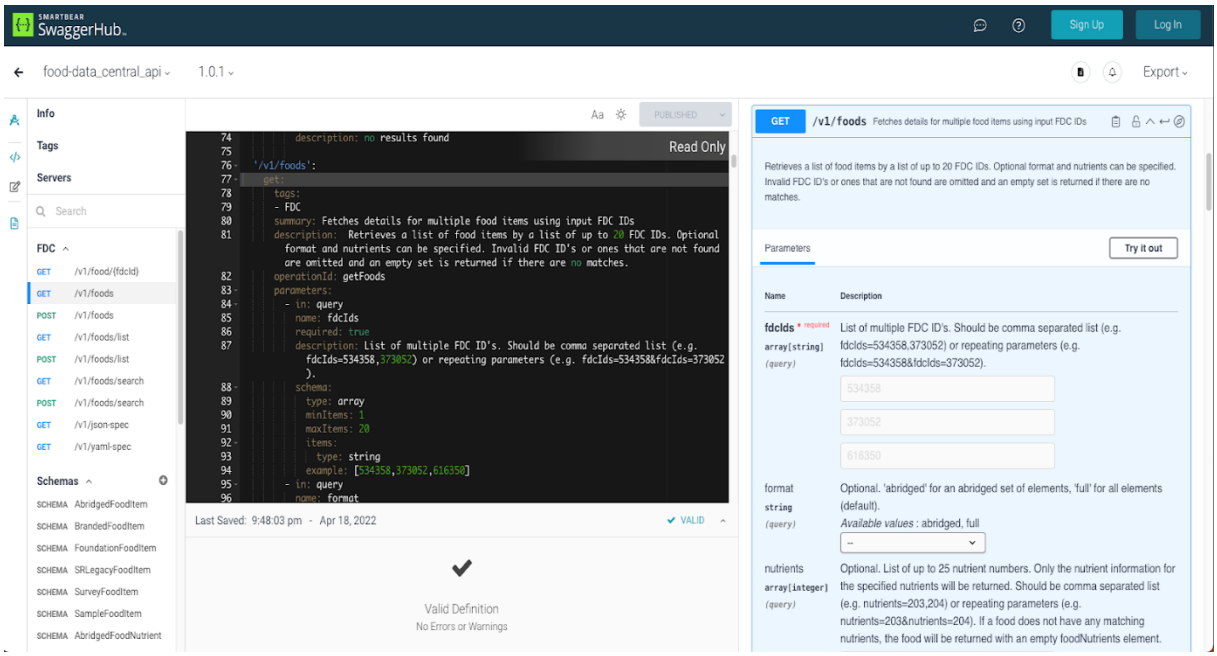
3.2.1.3 External Authentication – Google Sign In

Μια από τις κύριες απαιτήσεις της εφαρμογής είναι η ύπαρξη και η διαχείριση χρηστών, έτσι ώστε να μπορεί να παρέχεται προσωποποιημένη πληροφορία σχετικά με το διατροφικό ιστορικό και το ιστορικό δραστηριοτήτων και αθλημάτων. Για να είμαστε σίγουροι ότι η διαχείριση των χρηστών και ιδιαίτερα των προσωπικών τους δεδομένων είναι απόλυτη ασφαλής, αλλά και για λόγους ευχρηστίας της εφαρμογής αποφασίσαμε να ακολουθήσουμε τη μέθοδο της εξωτερικής αυθεντικοποίησης (external authentication). Ουσιαστικά αντί να είναι η ίδια η εφαρμογή υπεύθυνη να αυθεντικοποιεί τους χρήστες της, αναθέτει αυτή την λειτουργία σε κάποιον έμπιστο third-party οργανισμό που παρέχει τέτοιου είδους υπηρεσίες [16]. Η χρήση τέτοιων μεθόδων είναι πλέον ιδιαίτερα γνώριμη στους χρήστες γεγονός που αυξάνει την εμπειρία και προσδίδει αξιοπιστία.

Σαν μέθοδο εξωτερικής αυθεντικοποίησης επιλέξαμε το Google Sign In, διότι επιτρέπει ελεύθερη χρήση και είναι εύκολο στην υλοποίηση. Με το Google Sign In, ο χρήστης αρχικά αυθεντικοποιείται από τη Google και κατά την επιτυχία επιστρέφεται ένα AccessToken. Με το Token αυτό μπορεί να γίνει αυθεντικοποίηση του χρήστη στο Firebase και εν συνεχεία να εγγραφεί στη βάση δεδομένων Cloud Firestore. Για την υλοποίηση του Google Sign In στην πλατφόρμα iOS ακολουθήθηκε το σχετικό documentation που παρέχεται από τη Google [17].

3.2.1.4 FoodData Central A.P.I.

Κατά τη χρήση της εφαρμογής, ο χρήστης έχει τη δυνατότητα να επιλέξει μια τροφή από ένα σύνολο τροφών και εισάγοντας την ποσότητα που έχει καταναλώσει να υπολογίζεται η διατροφική αξία της τροφής αυτής και να προστίθεται στην ημερήσια διατροφή του. Για τη λίστα των τροφών αυτών χρησιμοποιήθηκε το FoodData Central, ένα open source A.P.I. το οποίο παρέχει μια πληθώρα τροφών και δεδομένα για όλα τα απαραίτητα μακροθρεπτικά συστατικά της κάθε τροφής [18]. Επιπλέον δίνει τη δυνατότητα αναζήτησης συγκεκριμένης τροφής από τις παρεχόμενες. Η ελεύθερη χρήση δίχως περιορισμούς καθώς και η αξιοπιστία τόσο στην παρεχόμενη υπηρεσία όσο και στην εγκυρότητα των δεδομένων καθιστούν το FoodData Central μια πολύ καλή επιλογή για το Thesis Fitness App. Για την ευκολότερη χρήση και κατανόηση του A.P.I παρέχεται και swagger όπου αναλύονται τα διαθέσιμα endpoints και η μορφή των αποτελεσμάτων [19]. Στην Εικονα 3.2 *FoodData Central A.P.I Swagger,* φαίνεται το περιβάλλον του swagger και έπειτα ένα παράδειγμα απάντησης JSON του FoodData Central A.P.I.



Σχήμα 3.2 FoodData Central A.P.I Swagger


```
{
  "dataType": "Branded",
  "description": "NUT 'N BERRY MIX",
  "fdcId": 534358,
  "foodNutrients": [
    {
      "number": 303,
      "name": "Iron, Fe",
      "amount": 0.53,
      "unitName": "mg",
      "derivationCode": "LCCD",
      "derivationDescription": "Calculated from a daily value percentage per serving size measure"
    }
  ],
  "publicationDate": "4/1/2019",
  "brandOwner": "Kar Nut Products Company",
  "gtinUpc": "077034085228",
  "ndbNumber": 7954,
  "foodCode": "27415110"
}
```

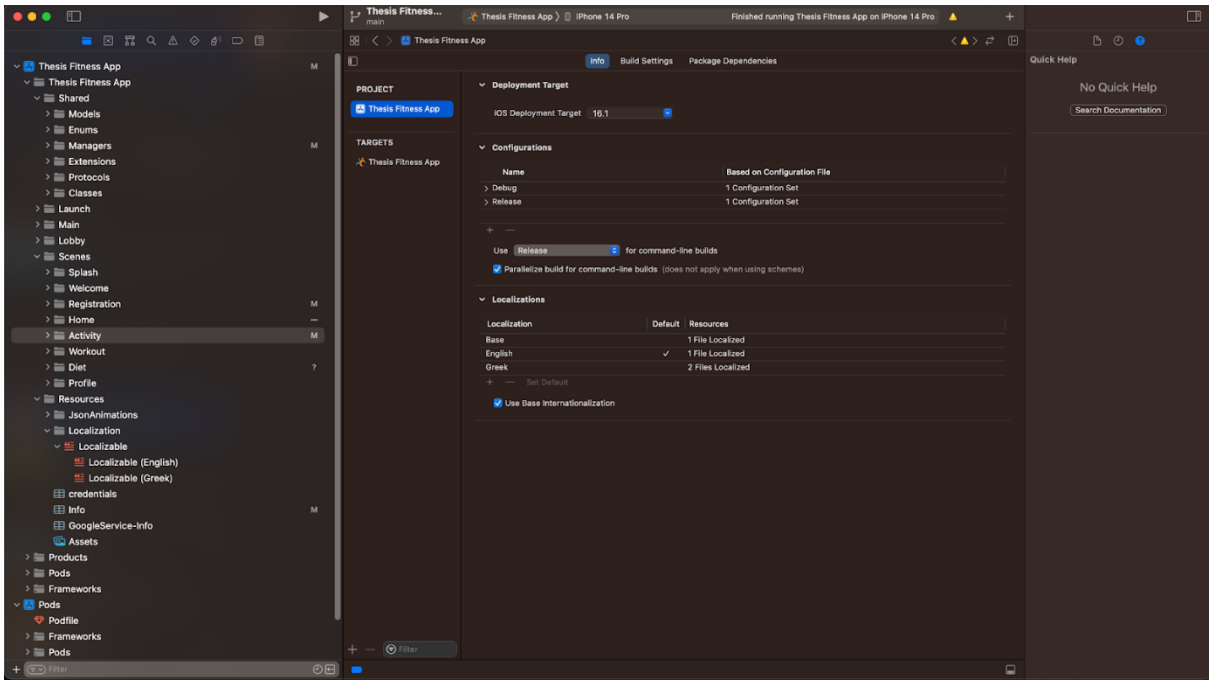

3.2.2 Τεχνολογίες Front-End

Οι τεχνολογίες που χρησιμοποιήθηκαν και αφορούν το Front-End, πέρα από τη γλώσσα προγραμματισμού Swift, είναι οι παρακάτω.

3.2.2.1 Xcode I.D.E

Το Xcode IDE είναι το περιβάλλον που χρησιμοποιείται για ανάπτυξη Apple εφαρμογών σε όλο το εύρος των λειτουργικών συστημάτων (iOS, MacOS, WatchOS) [20]. Είναι ένα εργαλείο το οποίο παρέχεται δωρεάν από την Apple και παρέχει ένα ολοκληρωμένο περιβάλλον για την υλοποίηση του UI, του Business Logic καθώς και όλων των λειτουργιών που έχουν να κάνουν με τη διαμόρφωση του τελικού εκτελέσιμου αρχείου .ipa, του πακεταρίσματος του κώδικα κ.α. Όλες οι απαραίτητες ρυθμίσεις του project και οποιαδήποτε άλλη ενέργεια έχει να κάνει με αυτό πραγματοποιούνται στο περιβάλλον

του Xcode. Πέρα από αυτά παρέχεται επίσης και η δυνατότητα αποσφαλμάτωσης κώδικα (debugging), η οποία επιταχύνει και εξομαλύνει κατά πολύ τη διαδικασία ανάπτυξης. Μια γενική εικόνα του Xcode IDE φαίνεται στο Σχήμα 3.3.



Σχήμα 3.3 Περιβάλλον ανάπτυξης Xcode

3.2.2.2 UIKit

Για την ανάπτυξη της γραφικής διεπαφής, προσφέρονται από την Apple δύο τρόποι υλοποίησης. Ένας εκ των δύο είναι το SwiftUI [21]. Στο SwiftUI η γραφική διεπαφή αναπτύσσεται καθαρά με κώδικα Swift και δεν γίνεται χρήση XML αρχείων. Είναι μια τεχνική declarative UI, η οποία είναι σχετικά καινούρια μιας και δημοσιεύθηκε προς χρήση στις 3 Ιουνίου του 2019. Για το λόγο αυτό δεν είναι ακόμη αρκετά διαδεδομένη και δεν υποστηρίζει όσες δυνατότητες υποστηρίζει η πιο παραδοσιακή τεχνική UIKit. Για αυτό το λόγο σε συνδυασμό με την εύλογη προϋπάρχουσα εμπειρία με τη χρήση του UIKit, το SwiftUI δεν επιλέχθηκε για την ανάπτυξη του Thesis Fitness App.

Στο UIKit [22], το οποίο είναι και η τεχνική που επιλέχθηκε για την ανάπτυξη, χρησιμοποιούνται .xib και storyboard αρχεία και η γραφική διεπαφή αναπτύσσεται σε γραφικό περιβάλλον drag and drop. Το UIKit αποτελεί μια component based τεχνική, όπου η κάθε οθόνη αποτελείται από ένα σύνολο στοιχείων όπως UIView, UILabel, UIButton κ.α.

Για την παραμετροποίηση των στοιχείων αυτών μπορεί να χρησιμοποιηθεί και κώδικας Swift, πράγμα το οποίο είναι αρκετά συχνό συνεπώς η τελική μορφή του UI καθορίζεται από συνδυασμό κώδικα και drag n drop ανάπτυξης. Σε περίπτωση που αυτό είναι επιθυμητό, μπορεί να υπάρξει συνδυασμός των μεθόδων UIKit και SwiftUI, για τη συγκεκριμένη εφαρμογή όμως η ανάπτυξη έγινε στο 100% με τη χρήση του UIKit.

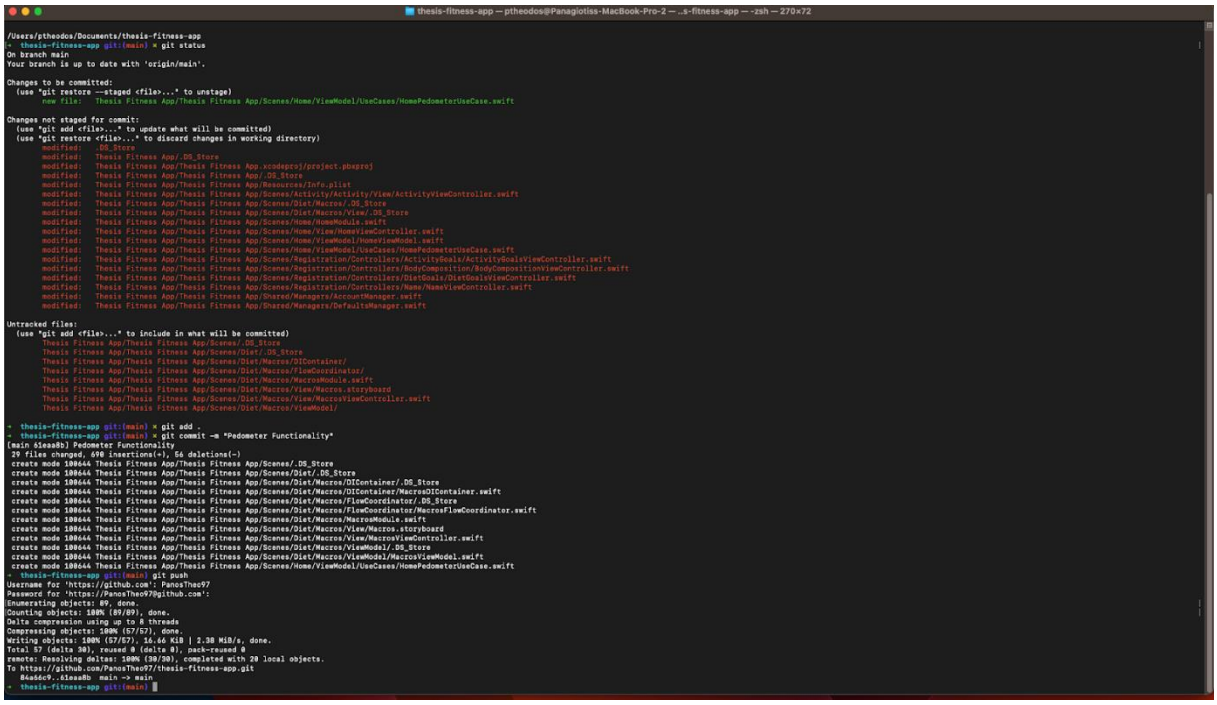
3.2.2.3 Git

Το Git είναι ένα κατανεμημένο σύστημα ελέγχου εκδόσεων, ικανό να διαχειριστεί οποιοδήποτε project με ασφάλεια και αξιοπιστία [23]. Είναι ένα λογισμικό ανοιχτού κώδικα το οποίο είναι διατεθειμένο

δωρεάν προς χρήση και είναι πολύ χρήσιμο για την οργάνωση των εκδόσεων ενός project, για συνεργατικό προγραμματισμό και για πολλούς άλλους λόγους.

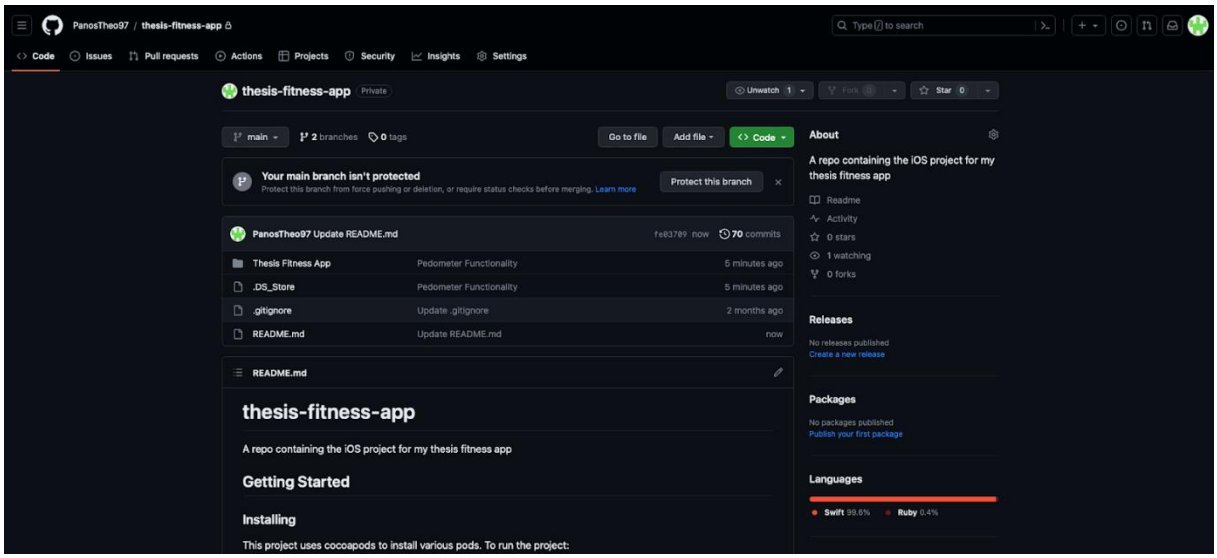
Το git χωρίζει τα στάδια υλοποίησης σε branches με βασικό το main branch. Για κάθε ξεχωριστή φάση υλοποίησης ή για κάθε αυτοτελή λειτουργία συστήνεται να δημιουργείται ένα καινούριο branch, διασφαλίζοντας έτσι ότι ο καινούργιος κώδικας δεν μπορεί να επηρεάσει αρνητικά ή να δημιουργήσει προβλήματα στον παλαιότερο. Όταν η καινούργια υλοποίηση είναι έτοιμη και έχει ελεγχθεί επαρκώς, τότε μπορεί να ενωθεί (merge) με το branch πατέρα της. Η διαδικασία αυτή επαναλαμβάνεται και όσο συνεχίζεται η υλοποίηση επεκτείνεται το “δέντρο” του Git. Για να ενταχθεί ένα κομμάτι κώδικα στο υπάρχων branch πρώτα πρέπει να εκτελεστεί η εντολή add όπου προσθέτει τα επιθυμητά προς ένταξη αρχεία. Έπειτα με τη χρήση της εντολής commit, τα αρχεία προετοιμάζονται προς ένταξη. Συνήθως η εντολή αυτή συνοδεύεται και από ένα commit message το οποίο είναι εμφανές στο ιστορικό και περιγράφει σύντομα τις λειτουργίες που προστέθηκαν σε αυτό το commit. Τέλος με την εντολή push πραγματοποιείται η συγχώνευση στο branch. Το git προσφέρει αρκετές παραπάνω δυνατότητες και εντολές οι οποίες φαίνονται στο επίσημο git documentation [24], οι βασικές όμως είναι αυτές που περιγράφονται παραπάνω.

Για την ανάπτυξη του Thesis Fitness App χρησιμοποιήθηκε το GitHub [25], το οποίο διατηρεί το σύνολο του project σε ένα online repository. Για τις διάφορες εντολές που πρέπει να εκτελεστούν για την ενημέρωση του repository μπορεί να χρησιμοποιηθεί το μενού source control που παρέχει το Xcode, η εφαρμογή του GitHub που επιτρέπει την ενημέρωση και την εκτέλεση εντολών μέσω γραφικού περιβάλλοντος ή πιο παραδοσιακά το command line, όπως φαίνεται στο Σχήμα 3.4.



Σχήμα 3.4 Git με χρήση Command Line

Στο Σχήμα 3.5 φαίνεται η αρχική σελίδα του GitHub repository για τη συγκεκριμένη εφαρμογή. Στο σημείο αυτό στο κάτω μέρος υπάρχει το αρχείο Readme, το οποίο ενημερώνει για τα βασικά στοιχεία του repository, τη σωστή χρήση του και οποιεσδήποτε άλλες πληροφορίες κρίνονται απαραίτητες από τον προγραμματιστή.



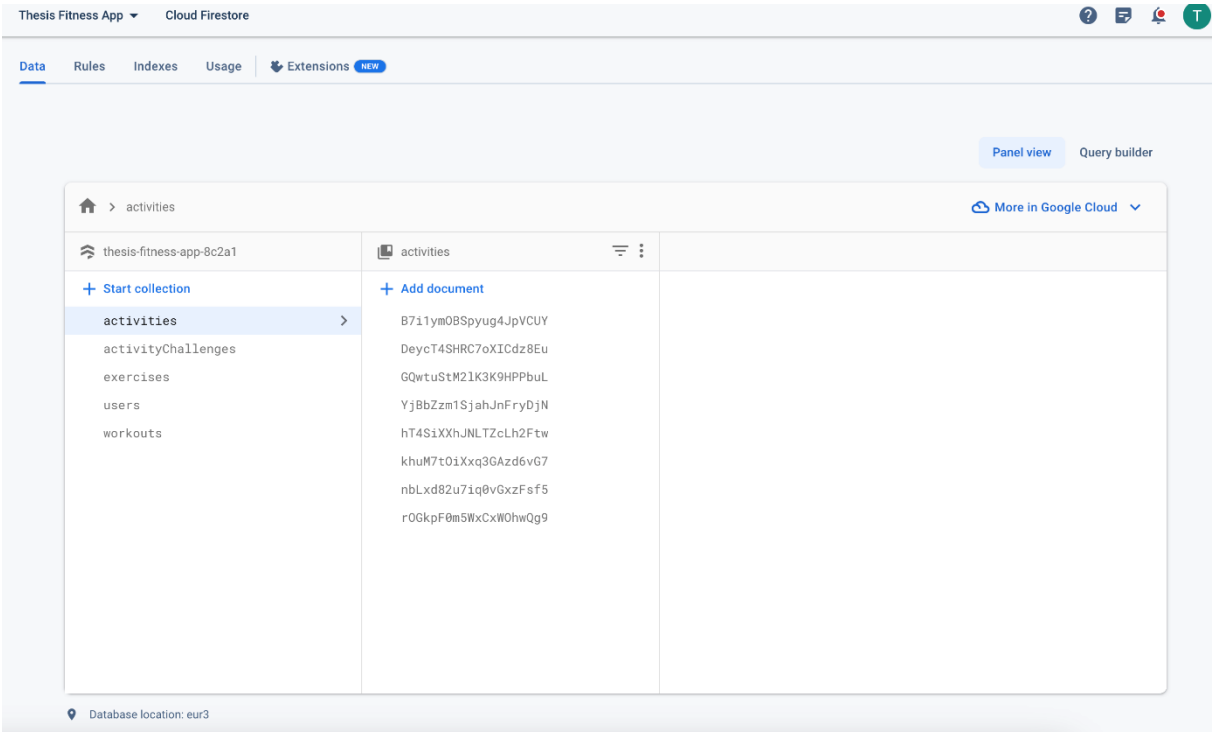
Σχήμα 3.5 Περιβάλλον GitHub

Κεφάλαιο 4ο: Ανάπτυξη Εφαρμογής και Περιγραφή λειτουργίας

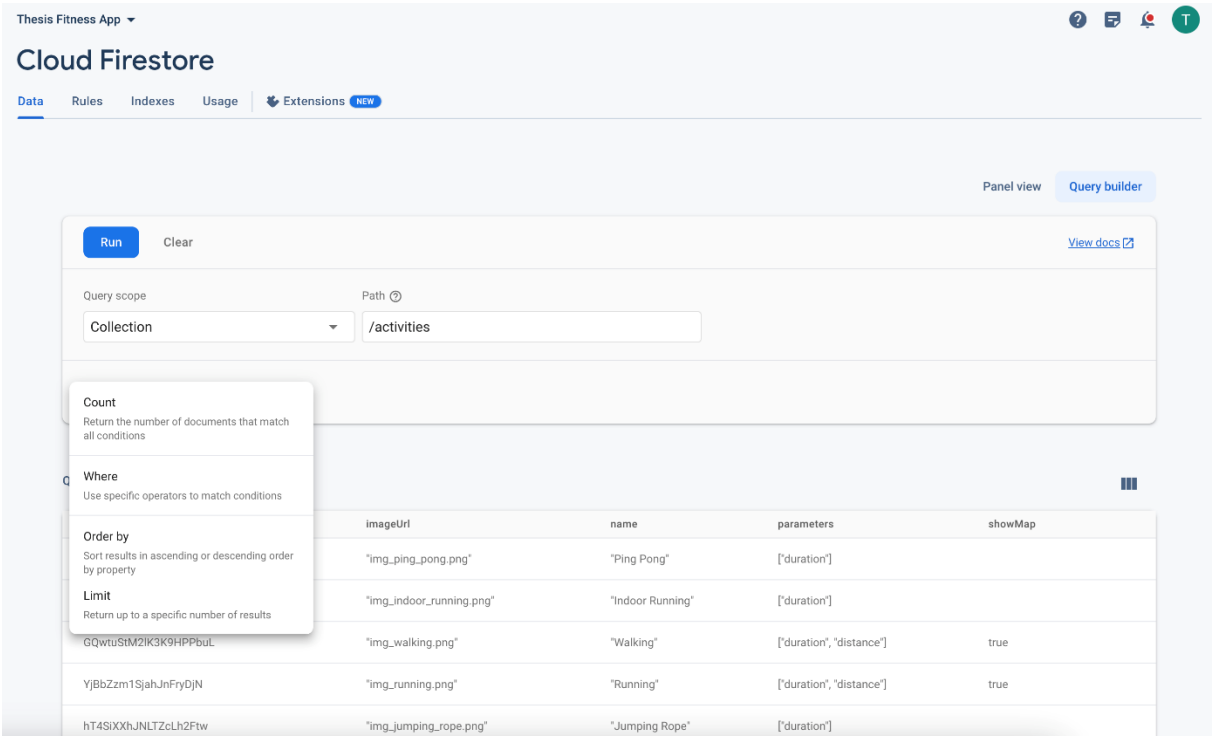
Με βάση τις σχεδιάστηκες αποφάσεις που πάρθηκαν και αναλύθηκαν παραπάνω, στο συγκεκριμένο κεφάλαιο παρουσιάζεται η πρακτική εφαρμογή τους στην υλοποίηση της εφαρμογής στο σύνολο της. Αρχικά γίνεται λόγος για την ανάπτυξη της βάσης δεδομένων Firestore και παρουσιάζονται οι οντότητες αυτής. Έπειτα αναλύεται η ανάπτυξη του mobile app με τη χρήση του Xcode, δίνοντας έμφαση στην αρχιτεκτονική που επιλέχθηκε για την υλοποίηση και στο πως αυτή μεταφράζεται σε ένα σύνολο διαφορετικών οντοτήτων του project. Τέλος παρατίθενται σενάρια και περιπτώσεις χρήσης που αναδεικνύουν τη λειτουργικότητα της εφαρμογής.

4.1 Ανάπτυξη Back-End

Όπως αναλύθηκε και στο προηγούμενο κεφάλαιο, για τη βάση δεδομένων της εφαρμογής χρησιμοποιήθηκε το Firebase Cloud Firestore, μια NoSQL βάση δεδομένων. Για την ανάπτυξη της δεν κρίθηκε απαραίτητη η χρήση κώδικα, μιας και τα δεδομένα προς διαχείριση δημιουργούνται εξ ολοκλήρου από την εφαρμογή και οι οντότητες της δημιουργούνται σε γραφικό περιβάλλον που παρέχεται από τη βάση. Σε περίπτωση που υπάρχει ανάγκη, παρέχεται και η δυνατότητα του Query builder για γρήγορη αναζήτηση. Στο Σχήμα 4.1 φαίνεται το περιβάλλον της κονσόλας για το Firestore Database και στο Σχήμα 4.2 ο Query builder με τις διαθέσιμες παραμέτρους του.



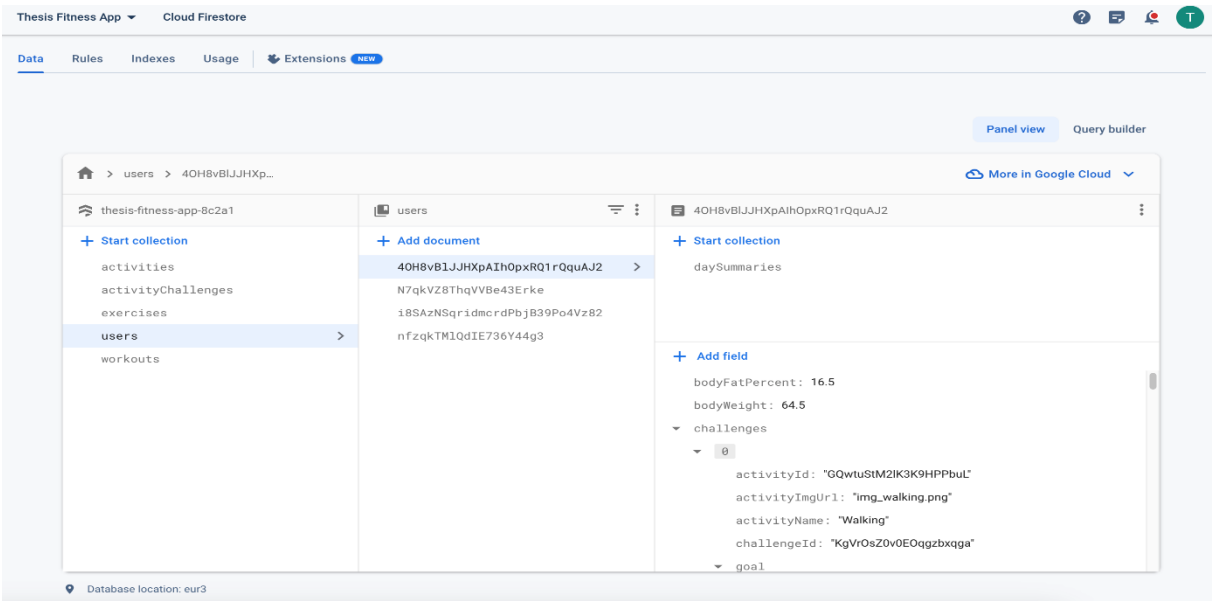
Σχήμα 4.1 Firestore Console Panel View



Σχήμα 4.2 Firestore Console Query Builder

4.1.1 Σχεδίαση Βάσης Δεδομένων

Όπως φαίνεται και στο Σχήμα 4.1 τα βασικά συστατικά μιας βάσης δεδομένων Firestore είναι τα collections και τα documents. Ένα document αποτελεί μια εγγραφή ενός collection η οποία αποτελείται από ένα σύνολο τιμών σε μορφή key-value pair. Εκτός από τις τιμές αυτές, ένα document μπορεί να περιέχει και ένα ή περισσότερα collections. Στο Σχήμα 4.3 φαίνεται ένα από τα documents που εμπεριέχονται στο collection users. Στα δεξιά του panel παρατηρούμε ότι το συγκεκριμένο document αποτελείται από ένα σύνολο key-value pairs και από ένα collection με όνομα daySummaries.

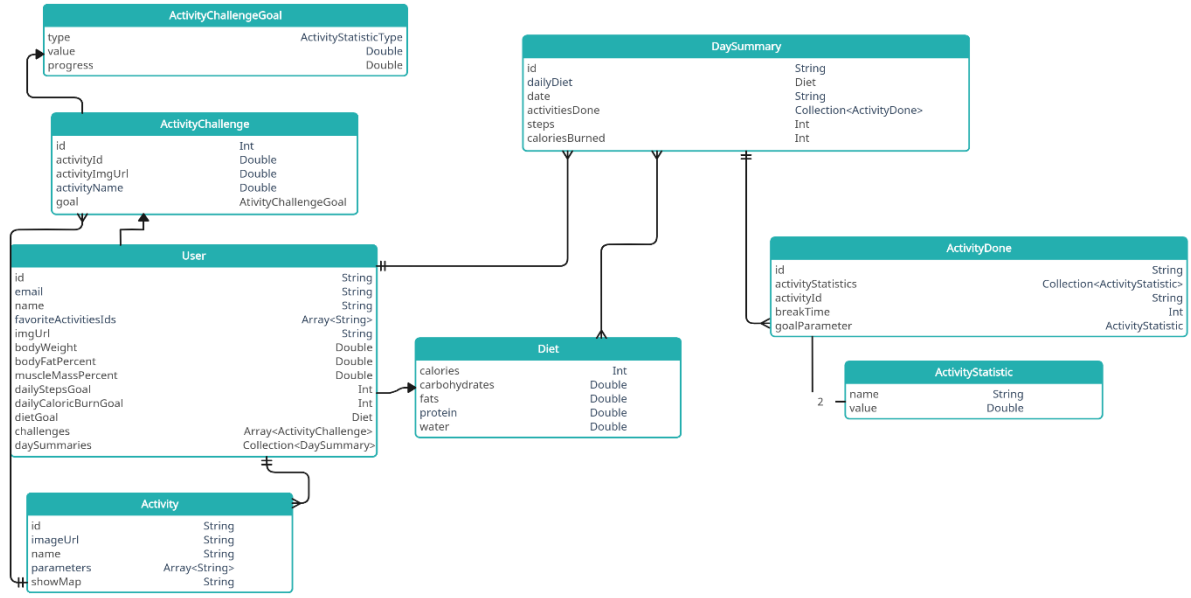


Σχήμα 4.3 User Document

Ένα collection είναι ένας χώρος αποθήκευσης που αποτελείται από documents. Σε αντίθεση με τα documents, ένα collection μπορεί να αποτελείται μόνο από documents και δεν μπορεί να περιέχει άλλα collections αλλά ούτε και raw values. Στο Σχήμα 4.3 στο αριστερό κομμάτι του panel φαίνονται κάποια από τα collection της βάσης δεδομένων.

4.1.2 Προσδιορισμός Οντοτήτων

Λαμβάνοντας υπόψη τη μορφολογία του Cloud Firestore αλλά και τις ανάγκες τις εφαρμογής σχεδιάστηκαν οι οντότητες της βάσης δεδομένων, όπως φαίνονται στην Σχήμα 4.4.



Σχήμα 4.4 Διάγραμμα Οντοτήτων Βάσης Δεδομένων

4.1.2.1 User

Η οντότητα User είναι ίσως η βασικότερη οντότητα της βάσης δεδομένων. Στα documents τύπου user αποθηκεύονται όλα τα απαραίτητα δεδομένα για ένα χρήστη της εφαρμογής. Μέσα σε αυτά συμπεριλαμβάνονται και το ημερήσιο ιστορικό του, το οποίο αφορά τις δραστηριότητες και τη διατροφή την οποία ακολούθησε για μια συγκεκριμένη μέρα. Τα δεδομένα του ιστορικού αποθηκεύονται ως ξεχωριστά documents ανά μέρα στο collection daySummaries. Κατά την επιτυχή σύνδεση ενός καινούριου χρήστη, δημιουργείται ένα καινούριο document και αποθηκεύεται στο collection users, το οποίο αποθηκεύει όλους τους users της εφαρμογής.

4.1.2.2 Activity

Η οντότητα activity αφορά το σύνολο των δραστηριοτήτων ο οποίες προσφέρονται στο χρήστη από την εφαρμογή. Το collection activities αποθηκεύει όλα τα διαθέσιμα activities της εφαρμογής τα οποία αφορούν όλους τους χρήστες. Ένα document τύπου activity αποτελείται από πληροφορίες που αφορούν το activity όπως το όνομα, την εικόνα, το σύνολο των παραμέτρων του κ.α.

4.1.2.3 Diet

Η οντότητα diet αποθηκεύει το σύνολο των μακροθρεπτικών συστατικών, το νερό και τις θερμίδες τα οποία καταναλώνει ένας χρήστης ημερησίως. Τα documents τύπου diet αποθηκεύονται στο daySummary του κάθε χρήστη σε αντιστοιχία ένα προς ένα.

4.1.2.4 ActivityChallenge

Τα ActivityChallenges είναι προκλήσεις τις οποίες ο χρήστης λαμβάνει ανάλογα με τις αγαπημένες δραστηριότητες του. Οι προκλήσεις αυτές είναι προκαθορισμένες από την εφαρμογή και αποθηκεύονται στο collection activityChallenges. Μια εγγραφή ActivityChallenge αποθηκεύει δεδομένα σχετικά με τον τύπο της δραστηριότητας καθώς και με το στόχο τον οποίο ο χρήστης πρέπει να επιτεύξει.

4.1.2.5 ActivityChallengeGoal

Η οντότητα ActivityChallengeGoal σχετίζεται με τα activityChallenges και αποθηκεύει πληροφορίες σχετικά με το στόχο μια πρόκλησης. Συγκεκριμένα διατηρεί δεδομένα για τον τύπο και την πρόοδο του χρήστη στην εκπλήρωση του στόχου.

4.1.2.6 DaySummary

Όπως αναφέρθηκε και προηγουμένως, για κάθε μέρα χρήσης της εφαρμογής δημιουργείται ένα document daySummary και αποθηκεύεται στο collection daySummaries του χρήστη. Τα documents τύπου daySummary αποθηκεύουν όλες τις πληροφορίες που σχετίζονται με το ιστορικό του χρήστη για τη συγκεκριμένη ημέρα καθώς και την ημερομηνία. Συγκεκριμένα οι πληροφορίες αυτές αφορούν την ημερήσια πρόσληψη μακροθρεπτικών, νερού, θερμίδων, την ημερήσια δραστηριότητα του χρήστη, τα βήματα που έκανε και τις θερμίδες που κατανάλωσε.

4.1.2.7 ActivityDone

Η οντότητα ActivityDone είναι μέρος του daySummary ενός χρήστη και αποθηκεύει δεδομένα για μια δραστηριότητα την οποία ο χρήστης έχει εκτελέσει. Συγκεκριμένα, πέρα από το είδος της δραστηριότητας αποθηκεύονται και άλλου είδους στατιστικά, όπως ο χρόνος διαλείμματος, ο στόχος άθλησης αν αυτός τέθηκε από το χρήστη κ.α.

4.1.2.8 ActivityStatistics

Η οντότητα activityStatistics αποθηκεύει δεδομένα για μια δραστηριότητα. Τα δεδομένα αυτά μπορεί να αφορούν το χρόνο άθλησης και την απόσταση που διανύθηκε, σε περίπτωση που η άσκηση έχει να κάνει με απόσταση.

4.2 Ανάπτυξη Front-End

Η ανάπτυξη της mobile εφαρμογής έγινε με τη χρήση του Xcode IDE με τη γλώσσα προγραμματισμού Swift. Στο υποκεφάλαιο αυτό θα γίνει ανάλυση της αρχιτεκτονικής που χρησιμοποιήθηκε για την οργάνωση του κώδικα, καθώς και των βιβλιοθηκών που επιλέχθηκαν για την εκπλήρωση των απαιτήσεων της εφαρμογής.

4.2.1 Αρχιτεκτονική – Οντότητες Front End

Σε αυτό το υποκεφάλαιο γίνεται ανάλυση των στρατηγικών που ακολουθήθηκαν, για την οργάνωση του κώδικα και την επίλυση των ποικίλων προβλημάτων και απαιτήσεων της εφαρμογής. Μαζί με την

ανάλυση του κάθε μοτίβου σχεδίασης (design pattern) που επιλέχθηκε, παρατίθενται και περιπτώσεις χρήσης από το project για την καλύτερη κατανόηση.

4.2.1.1 MVVM Design Pattern

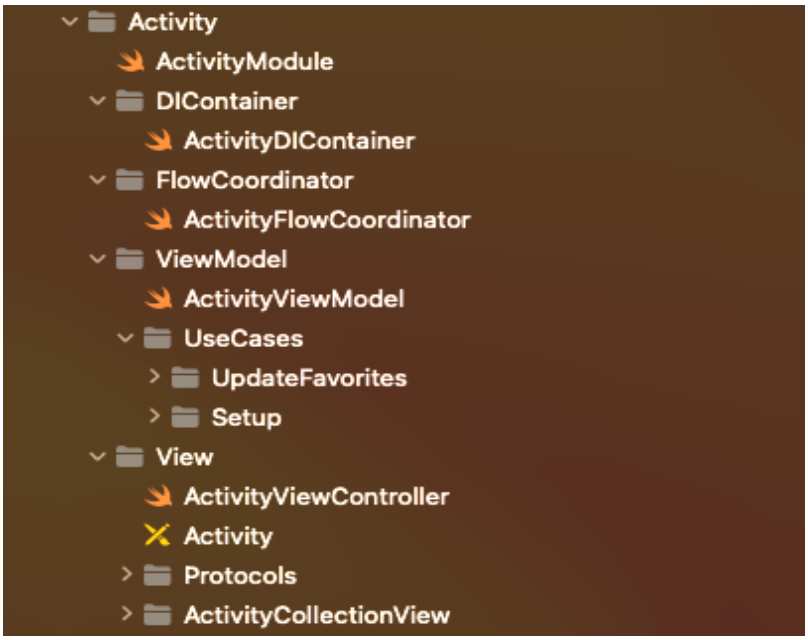
Για να επιτευχθεί η ανάπτυξη της εφαρμογής ακολουθώντας στο καλύτερο δυνατό βαθμό τις αρχές της αρχιτεκτονικής clean code, επιλέχθηκε το Model - View - ViewModel design pattern (MVVM). Ο σκοπός που εκπληρώνει το MVVM είναι ο διαχωρισμός της λογικής της γραφικής διεπαφής (GUI logic), από το Business logic και το Back-End logic.

Το παραδοσιακό MVVM, μπορεί να αναλυθεί στις εξής επιμέρους βαθμίδες (layers):

- Model: Η ορολογία Model αναφέρεται στο σύνολο των δεδομένων της εφαρμογής. Το "ταξίδι" των δεδομένων ξεκινά από τη λήψη τους από τη βάση δεδομένων και καταλήγει στην παρουσίαση τους στο χρήστη μέσω της γραφικής διεπαφής. Σε όλη αυτή τη διαδικασία, τα δεδομένα περνούν από διάφορες βαθμίδες στις οποίες και επιδέχονται την απαραίτητη επεξεργασία για να αποκτήσουν την τελική τους μορφοποίηση.
- View: Με τον όρο View αναφερόμαστε στο σύνολο των δομών που αποτελούν το περιβάλλον το οποίο βλέπει και αλληλεπιδρά ο χρήστης. Το View είναι υπεύθυνο για την αναπαράσταση των δεδομένων, καθώς και για την λήψη των ενεργειών του χρήστη κατά την αλληλεπίδραση του με την εφαρμογή και την προώθηση τους στο ViewModel.
- View Model: Το ViewModel δρα ως δίαυλος επικοινωνίας μεταξύ του View και του Model. Το ViewModel είναι υπεύθυνο για την επεξεργασία των δεδομένων έτσι ώστε αυτά να έρθουν στην κατάλληλη μορφή για να τα χρησιμοποιήσει το View. Είναι επίσης υπεύθυνο για τη διαχείριση των ενεργειών του χρήστη, της οποίες λαμβάνει μέσω του View και την εκτέλεση τους συνήθως με τη χρήση ενός Use Case. Ουσιαστικά το ViewModel αποτελεί τον ενδιάμεσο μεταξύ του Model και του View.

4.2.1.2 MVVM Design Pattern στο Thesis Fitness App

Στο υποκεφάλαιο αυτό αναλύεται η συγκεκριμένη custom υλοποίηση του MVVM pattern στο Thesis Fitness App. Η υλοποίηση αυτή είναι βασισμένη στο γενικό MVVM μοντέλο, λαμβάνοντας υπόψη τις απαιτήσεις της εφαρμογής και τις ιδιαιτερότητες της ανάπτυξης στο περιβάλλον του XCode με γλώσσα Swift. Η κάθε επιμέρους οθόνη της εφαρμογής ακολουθεί αυτό το μοτίβο και αποτελείται από συγκεκριμένα layers τα οποία αλληλοεπιδρούν μεταξύ τους. Τα layers που χρησιμοποιήθηκαν στην εν λόγω υλοποίηση φαίνονται στο Σχήμα 4.5.



Σχήμα 4.5 Activity MVVM

Συγκεκριμένα, στο παραπάνω σχήμα βλέπουμε τα εξής layers:

4.2.1.2.1 View

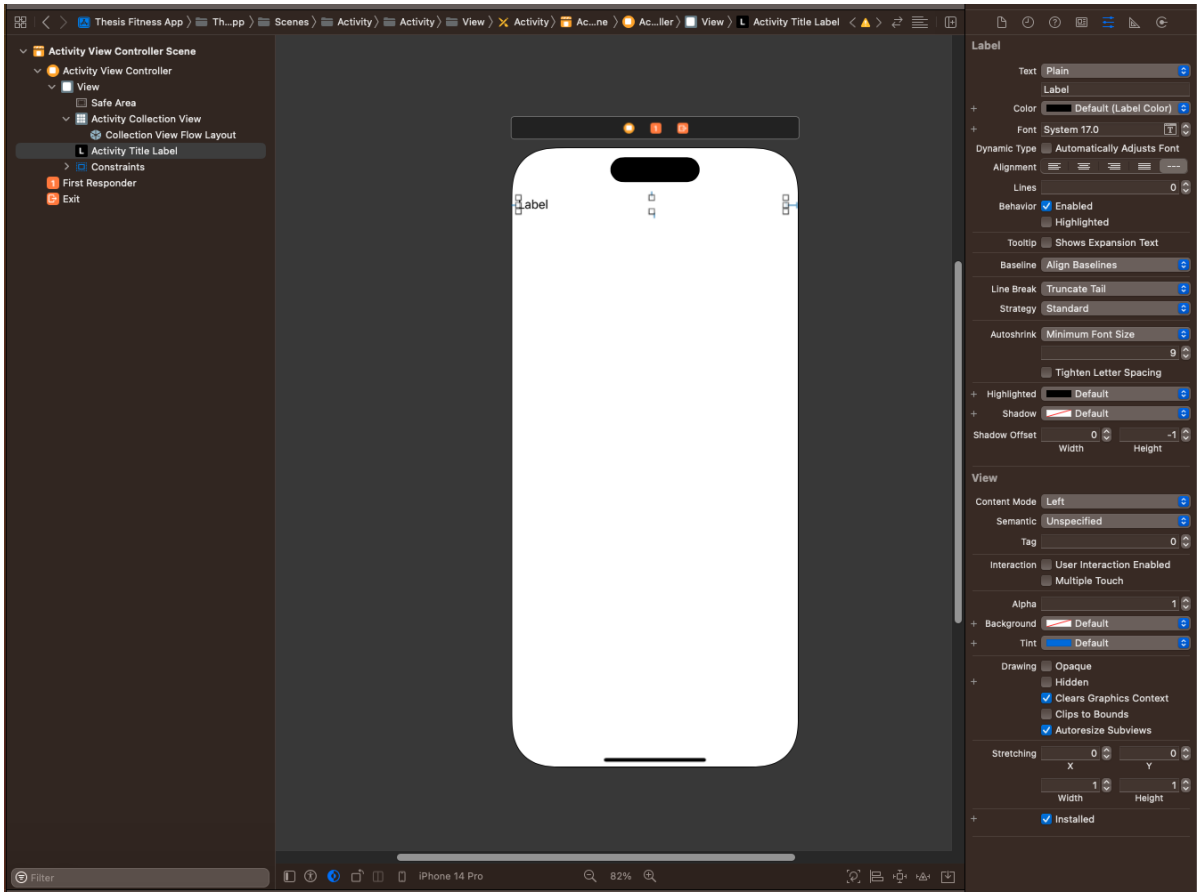
Στο View layer, όπως αναφέρθηκε προηγουμένως, ανήκουν όλα τα αρχεία και ο κώδικας που έχει να κάνει με το layout της εφαρμογής, την παρουσίαση και την αλληλεπίδραση με τον χρήστη. Συγκεκριμένα, οι δύο βασικοί τύποι αρχείων που συναντάμε στο View layer ειναι οι ViewControllers και τα Storyboards, καθώς και όλες οι κλάσεις που αφορούν στοιχεία διεπαφής της εφαρμογής, όπως π.χ. TableViews, CollectionViews κ.α. Αναλυτικότερα για τους 2 αυτούς τύπους αρχείων ισχύουν τα παρακάτω:

- **ViewController**: Οι ViewControllers της εφαρμογής είναι τα αρχεία τα οποία παραμετροποιούν τα στοιχεία διεπαφής και λαμβάνουν τις ενέργειες που προκύπτουν από την αλληλεπίδραση με αυτά. Η σύνδεση των στοιχείων από το Storyboard γίνεται με τη μορφή IBOutlet. Επιπροσθέτως, κάθε ViewController της εφαρμογής υλοποιεί το BaseProtocol, από το οποίο έχει πρόσβαση στο ViewModel και τον FlowCoordinator. Του δίνεται επίσης η δυνατότητα να παρακολουθεί (observe) αλλαγές που γίνονται σε συγκεκριμένες μεταβλητές του ViewModel και να ενημερώνει το UI όταν αυτό κρίνεται απαραίτητο.

```
import Foundation
protocol BaseProtocol: AnyObject {

    associatedtype ViewModel
    associatedtype FlowCoordinator

    // MARK: - Properties

    var viewModel: ViewModel? { get set }
    var flowCoordinator: FlowCoordinator { get set }

    // MARK: - Methods

    func registerBaseObservers()
```

```
}

extension BaseProtocol {

    func registerBaseObservers() {
        self.registerLoadingObservers()
    }

    func registerLoadingObservers() {
        (viewModel as? BaseViewModelProtocol)?.isLoading.addObserver({ [weak self]
isLoading in
            guard let flowCoordinator = self?.flowCoordinator as?
BaseFlowCoordinatorProtocol else { return }
            flowCoordinator.handleLoader(isLoading)
        })
    }
}
```

- **Storyboard**: Τα Storyboards είναι XML αρχεία στα οποία υλοποιείται το UI της εφαρμογής. Για την υλοποίηση αυτή χρησιμοποιούνται συνήθως γραφικές μέθοδοι, όπου τα UI Elements γίνονται drag and drop μέσα στην οθόνη και οριοθετούνται στο χώρο με περιορισμούς (constraints). Υπάρχει επίσης η δυνατότητα υλοποίησης με XML κώδικα, όμως αυτή χρησιμοποιείται σπανίως έως και καθόλου διότι δεν είναι καθόλου εύχρηστη για τον προγραμματιστή. Παράδειγμα από το περιβάλλον ανάπτυξης που περιγράφεται φαίνεται στην Σχήμα 4.6.



Σχήμα 4.6 Activity View Controller Storyboard

Στο συγκεκριμένο παράδειγμα φαίνεται να είναι επιλεγμένο ένα UILabel, ένα element παρουσίασης λεκτικών. Δεξιά υπάρχει ένα μενού με ένα σύνολο επιλογών το οποίο δείχνει στοιχεία που αφορούν τη θέση, τη μορφοποίηση, το είδος του element κ.α. Στο συγκεκριμένο παράδειγμα είναι επιλεγμένο το μενού μορφοποίησης ιδιοτήτων (attributes inspector) από το οποίο μπορούμε να παραμετροποιήσουμε την εμφάνιση του UIElement (στην περίπτωση του Label το χρώμα του κειμένου, το μέγεθος και το είδος της γραμματοσειράς κ.α.). Όλες οι δυνατότητες που προσφέρονται από τα μενού δεξιά, μπορούν να επιτευχθούν και με κώδικα Swift στον αντίστοιχο ViewController. Μάλιστα αυτή είναι μια προτιμητέα πρακτική διότι καθιστά τον κώδικα και τη σχεδίαση πιο κατανοητά. Το συγκεκριμένο UILabel λαμβάνει με αυτό τον τρόπο μορφοποίηση μέσα από το IBOutlet του στον ActivityViewController.

```
class ActivityViewController: UIViewController, BaseProtocol,
TabBarViewControllerProtocol {
    // MARK: - IBProperties

    @IBOutlet weak var activityTitleLabel: UILabel! {
        didSet {
            activityTitleLabel.textColor = .App.mainText
            activityTitleLabel.font = .systemFont(ofSize: 20, weight: .bold)
            activityTitleLabel.text = "Activity_activityTitleLabel".getLocalized()
        }
    }
```

Με βάση τα παραπάνω παραδείγματα έχουμε μια εικόνα για την υλοποίηση ολόκληρης της εφαρμογής σε ότι αφορά το View Layer. Η κάθε οθόνη της εφαρμογής είναι ένας συνδυασμός από ένα Storyboard με UIElements και ένα ViewController με IBOutlets από τα Elements αυτά και κώδικα μορφοποίησης.

4.2.1.2.2 ViewModel

Τα ViewModel της εφαρμογής είναι το σημείο στο οποίο εκτελείται όλη η λογική επεξεργασίας δεδομένων. Αυτό συμπεριλαμβάνει τη συλλογή δεδομένων, τη μετατροπή τους σε Domain Models τα οποία χρησιμοποιούνται από το View layer για την αναπαράσταση τους στο UI, το φιλτράρισμα τους ή την παραμετροποίηση τους όπου χρειάζεται και ανάλογα με τις ανάγκες της εφαρμογής και τέλος την ειδοποίηση του View όταν ολοκληρωθούν οι παραπάνω διαδικασίες, συνήθως με τη χρήση Observable μεταβλητών. Επεξήγηση σχετικά με τους όρους Domain Model και Observable υπάρχει σε παρακάτω υποκεφάλαια. Κάθε ViewModel της εφαρμογής υλοποιεί το BaseViewModelProtocol, το οποίο του παρέχει τη μεταβλητή isLoading που χρησιμοποιείται για την εμφάνιση του loader όταν εκτελείται μια λειτουργία η οποία απαιτεί την αναμονή του χρήστη (π.χ. η λήψη δεδομένων από τη βάση δεδομένων).

```
import Foundation
protocol BaseViewModelProtocol {
    var isLoading: Observable<Bool?> { get set }
}
```

Με σκοπό την καλύτερη οργάνωση του κώδικα, οι παραπάνω λειτουργίες χωρίζονται στο ViewModel με τη μορφή των UseCases. Ένα UseCase είναι υπεύθυνο για να εκτελέσει όλες τις απαραίτητες ενέργειες για την επίτευξη ενός συγκεκριμένου σκοπού. Έτσι ένα ViewModel αποτελείται από ένα ή περισσότερα UseCase που στο σύνολο τους διεκπεραιώνουν όλες τις λειτουργίες για τις οποίες το ViewModel είναι υπεύθυνο.

Σε μια τυπική ροή μεταξύ των UseCases, του ViewModel και του View, το View καλεί μια μέθοδο execute του ViewModel, η οποία είναι υπεύθυνη για την εκτέλεση ενός UseCase. Τα αποτελέσματα του

UseCase αυτού μετά την εκτέλεση του επιστρέφονται στο ViewModel και από εκεί με τη χρήση μιας Observable μεταβλητής στο View, το οποίο εκτελεί τις απαραίτητες ενέργειες ενημέρωσης του UI.

Τα παρακάτω κομμάτια κώδικα αποτελούν παράδειγμα της ροής αυτής στη Home οθόνη. Αρχικά ο HomeViewController, κάθε φορά που εμφανίζεται εκτελεί το HomeSetupUseCase.

```
override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        self.viewModel?.executeHomeSetupUseCase()
    }
```

Η μέθοδος executeHomeSetupUseCase του HomeViewModel ενεργοποιεί τον Loader και εκτελεί το εν λόγω UseCase, τα αποτελέσματα του οποίου επιστρέφονται κατά την ολοκλήρωση μέσω ενός completionHandler.

```
func executeHomeSetupUseCase() {
        self.isLoading.value = true
        self.homeSetupUseCase.execute { user in
            self.isLoading.value = false
            self.user.value = user
        }
    }
```

Το HomeSetupUseCase πραγματοποιεί την λήψη ενός χρήστη από τη βάση δεδομένων και όταν αυτή στεφθεί με επιτυχία, τη μετατροπή του χρήστη από το DataModel στο DomainModel και την επιστροφή με τον completionHandler.

```
import Foundation
protocol HomeSetupUseCaseProtocol: AnyObject {
    func execute(completion: @escaping (UserModel?) -> Void)
}
class HomeSetupUseCase: HomeSetupUseCaseProtocol {

    func execute(completion: @escaping (UserModel?) -> Void) {
        guard let documentId = AccountManager.shared.userDocimentId else {
            completion(nil)
            return
        }
        NetworkManager.getDocument(FUserModel.self, .users, documentId) { result in
            switch result {
            case .success(let user):
                completion(UserModel(from: user))
            case .failure:
                completion(nil)
            }
        }
    }
}
```

Η ροή που μόλις περιγράφηκε παρατηρείται σε μεγάλο βαθμό σε όλο το εύρος της εφαρμογής και αποτελεί βασικό συστατικό της MVVM αρχιτεκτονικής, διαχωρίζοντας το Infrastructure Layer που αφορά από το Application Layer και το Domain Layer όπως αυτά περιγράφονται στην Clean Code Αρχιτεκτονική [26].

4.2.1.2.3 FlowCoordinator

Οι FlowCoordinators, όπως το αναφέρει και η ονομασία τους, είναι υπεύθυνοι για τη διαχείριση της ροής της εφαρμογής. Με τον όρο αυτό εννοούμε την εναλλαγή από οθόνη σε οθόνη, την εμφάνιση προειδοποιήσεων και μηνυμάτων και γενικότερα οποιαδήποτε ενέργεια οδηγεί το χρήστη σε

διαφορετικό σημείο στην εφαρμογή από αυτό που βρισκόταν πριν. Κάθε FlowCoordinator της εφαρμογής υλοποιεί το BaseFlowCoordinatorProtocol και έτσι μπορεί αυτόματα να εμφανίζει τον loader της εφαρμογής όποτε αυτό κρίνεται απαραίτητο.

```
import Foundation
import SVProgressHUD
protocol BaseFlowCoordinatorProtocol {

    // MARK: - Methods

    func handleLoader(_ isLoading: Bool?)
}
extension BaseFlowCoordinatorProtocol {

    func handleLoader(_ isLoading: Bool?) {
        if isLoading ?? false {
            SVProgressHUD.show()
        } else {
            DispatchQueue.main.async {
                SVProgressHUD.dismiss(withDelay: 0.5)
            }
        }
    }
}
```

Για να κληθεί κάποια μέθοδος του FlowCoordinator, συνήθως πραγματοποιείται μια αλλαγή στην τιμή του RoutingEnum στο ViewModel. Το ViewModel αλλάζει την τιμή του routing enum με τη μέθοδο updateRouting, το οποίο είναι τύπου Observable property και συνεπώς ενημερώνει τον ViewController για την αλλαγή αυτή. Έπειτα ο ViewController, έχοντας πρόσβαση στον FlowCoordinator καλεί την κατάλληλη μέθοδο και τότε εκείνος πραγματοποιεί τη μετάβαση που απαιτείται. Στο παράδειγμα του Activity MVVM, τα παραπάνω υλοποιούνται ως εξής.

Τμήμα του Activity View Model:

```
class ActivityViewModel: BaseViewModelProtocol {
    // MARK: - Observables

    var isLoading = Observable<Bool?>(value: nil)
    var routingEnum = Observable<Activity.RoutingEnum?>(value: nil)

// MARK: - Methods

    func update(routing: Activity.RoutingEnum) {
        self.routingEnum.value = routing
    }
```

Τμήμα του Activity View Controller:

```
private func registerObservers() {

        registerBaseObservers()

        viewModel?.routingEnum.addObserver({ [weak self] routingEnum in
            switch routingEnum {
            case .activityParameter(let activityId):
                self?.flowCoordinator?.moveToActivityParameter(activityId: activityId)
            default: ()
            }
        })
```

ActivityFlowCoordinator:

```
import Foundation
class ActivityFlowCoordinator: BaseFlowCoordinatorProtocol {

    // MARK: - Properties
    var activityDIContainer: ActivityDIContainer

    weak var activityViewController: ActivityViewController?

    // MARK: - Life cycle

    init(activityDIContainer: ActivityDIContainer, activityViewController:
ActivityViewController?) {
        self.activityDIContainer = activityDIContainer
        self.activityViewController = activityViewController
    }

    // MARK: - Methods
    func moveToActivityParameter(activityId: String) {
        guard let activityParameterViewController =
self.activityDIContainer.activityParameterModule.makeActivityParameterViewController(acti
vityId: activityId) else {
            return
        }

        self.activityViewController?.navigationController?.pushViewController(activityPar
ameterViewController, animated: true)
    }

}
```

4.2.1.2.4 DIContainer

Οι Dependency Injection Containers ή εν συντομία DIContainers, είναι υπεύθυνοι να κρατούν ως αναφορά και κατά συνέπεια να δίνουν πρόσβαση στα Modules άλλων MVVM οθονών της εφαρμογής. Τα Modules αυτά χρησιμοποιούνται από τους FlowCoordinators για την δημιουργία μια καινούριας οθόνης και όλου του MVVM "πακέτου" που της αναλογεί, με απώτερο σκοπό τη μετάβαση σε αυτή την οθόνη. Περισσότερα σχετικά με τα Modules θα συζητηθούν στο αμέσως επόμενο υποκεφάλαιο. Παρακάτω βλέπουμε τον ActivityDIContainer καθώς και το ActivityParameterModule ως μεταβλητή του.

```
import Foundation
final class ActivityDIContainer {

    let activityParameterModule = ActivityParameterModule()

}
```

4.2.1.2.5 Module

Για την δημιουργία όλων των παραπάνω κλάσεων, αλλά και τη σύνδεση μεταξύ τους έτσι ώστε να επιτυγχάνεται η αρχιτεκτονική MVVM χρησιμοποιούνται τα Modules. Ένα Module αποτελείται εξ ολοκλήρου από μεθόδους που εκτελούν τις παραπάνω λειτουργίες και δεν εκτελεί καμία άλλη λειτουργικότητα. Ουσιαστικά τα Modules δεν επιδρούν στο πώς συμπεριφέρεται η εφαρμογή αλλά λειτουργούν ως συνδετικός κρίκος μεταξύ των άλλων κλάσεων της αρχιτεκτονικής MVVM. Όλες οι ενέργειες ενός Module θα μπορούσαν να διασπαστούν και να εκτελεστούν στις επιμέρους MVVM κλάσεις, όμως με αυτό τον τρόπο θα υπάρχει σύγχυση, δυσκολία στην κατανόηση και ο κώδικας θα

γίνονταν αρκετά περιπλοκότερος. Το HomeModule με όλες τις απαραίτητες μεθόδους διαμορφώνεται ως εξής:

```
import Foundation
import UIKit
class HomeModule {

    func makeHomeNavigationController(tabBarController: LobbyTabBarController) ->
UINavigationController? {
        if let viewController = self.makeHomeViewController(tabBarController:
tabBarController) {
            let homeNavigationController = UINavigationController(rootViewController:
viewController)
            homeNavigationController.navigationBar.tintColor = .App.mainText
            homeNavigationController.navigationBar.barTintColor = .clear
            return homeNavigationController
        }
        return nil
    }

    func makeHomeViewController(tabBarController: LobbyTabBarController) ->
HomeViewController? {
        if let viewController = HomeViewController.create(storyboardName: "Home") {
            viewController.lobbyTabBarController = tabBarController
            viewController.viewModel = makeHomeViewModel()
            viewController.flowCoordinator = makeHomeFlowCoordinator(viewController)
            return viewController
        }
        return nil
    }

    private func makeHomeViewModel() -> HomeViewModel {
        HomeViewModel(homeSetupUseCase: makeHomeSetupUseCase(),
                      homePedometerUseCase: makeHomePedometerUseCase())
    }

    private func makeHomeFlowCoordinator(_ viewController: HomeViewController) ->
HomeFlowCoordinator {
        HomeFlowCoordinator(homeDIContainer: makeHomeDIContainer(),
                            homeViewController: viewController)
    }

    private func makeHomeDIContainer() -> HomeDIContainer {
        HomeDIContainer()
    }

    private func makeHomeSetupUseCase() -> HomeSetupUseCaseProtocol {
        HomeSetupUseCase()
    }

    private func makeHomePedometerUseCase() -> HomePedometerUseCaseProtocol {
        HomePedometerUseCase()
    }
}
```

4.2.1.3 Singleton Pattern

Το Singleton pattern επιτρέπει σε ένα instance μιας κλάσης να μοιράζεται τον εαυτό του σε ολόκληρη την εφαρμογή (global scope). Το instance αυτό αρχικοποιείται μέσα στην ίδια την κλάση σε μορφή static μεταβλητής. Με τον τρόπο αυτό, δεν χρειάζεται να κάνουμε αρχικοποίηση ενός καινούριου instance της singleton κλάσης κάθε φορά που χρειαζόμαστε λειτουργικότητα της. Το singleton pattern

χρησιμοποιείται αρκετά σε Manager κλάσεις, οι οποίες περιγράφονται παρακάτω. Ένα παράδειγμα χρήσης μεθόδου μιας singleton κλάσης αποτελεί το εξής:

```
class WelcomeSignInUseCase: WelcomeSignInUseCaseProtocol {

    func execute(completion: @escaping (Bool) -> Void) {
        AccountManager.shared.signIn { successfulSignIn in
            completion(successfulSignIn)
        }
    }
}
```

4.2.1.4 Delegation Pattern

Το Delegation pattern δίνει τη δυνατότητα σε μια κλάση να αναθέσει κάποιες από τις λειτουργίες που εκτελεί σε μια οντότητα διαφορετικού τύπου. Στη Swift, για να υλοποιηθεί το Delegation Pattern απαιτείται η δημιουργία ενός delegate protocol το οποίο εμπεριέχει όλη την επιθυμητή λειτουργικότητα, μια delegator κλάση η οποία αναθέτει τη λειτουργικότητα και μια κλάση η οποία υλοποιεί το delegate protocol και με αυτό τον τρόπο παρέχει την επιθυμητή λειτουργικότητα. Το Delegation Pattern χρησιμοποιείται εκτενώς και natively από τη Swift, κυρίως για τη διαχείριση των δεδομένων ενός TableView ή ενός CollectionView. Στο παρακάτω παράδειγμα φαίνεται η χρήση ενός custom delegate pattern, στο οποίο ο MacrosViewController αναθέτει στον DietViewController τον υπολογισμό των μακροθρεπτικών συστατικών που ο χρήστης επέλεξε να προσθέσει και την ανανέωση της διεπαφής με βάση τα νέα σύνολα μακροθρεπτικών συστατικών.

```
weak var macrosViewControllerDelegate: MacrosViewControllerProtocol?
```

```
protocol MacrosViewControllerProtocol: AnyObject {
    func addMacros(data: [DietModel])
}
```

```
extension DietViewController: MacrosViewControllerProtocol {
    func addMacros(data: [DietModel]) {
        self.presentedViewController?.dismiss(animated: true)
        self.viewModel?.dietData.value = data
    }
}
```

4.2.1.5 Observable Pattern

Το Observable pattern στη Swift χρησιμοποιείται με σκοπό την ενημέρωση του UI όταν προκύπτει κάποια αλλαγή στα δεδομένα της εφαρμογής. Ουσιαστικά το observable pattern αποτελεί ένα δίαυλο επικοινωνίας μεταξύ του ViewModel layer και του View layer. Όταν τα δεδομένα στο ViewModel layer αλλάζουν, πυροδοτείται ένα notification στο οποίο αν το View layer είναι subscribed, μπορεί να "ακούει" αυτές τις αλλαγές στα δεδομένα και να πραγματοποιεί κάποια ενέργεια όταν αυτό κρίνεται απαραίτητο. Στη συγκεκριμένη εφαρμογή χρησιμοποιείται η εξής custom υλοποίηση του Observable pattern.

```
class Observable<T> {

    // MARK: - Properties

    var value: T {
        didSet {
```

```
            self.valueChanged?(self.value)
        }
    }

    private var valueChanged: ((T) -> Void)?

    // MARK: - Life cycle

    init(value: T) {
        self.value = value
    }

    // MARK: - Methods
    func addObserver(fireNow: Bool = false, _ onChange: ((T) -> Void)?) {
        valueChanged = onChange
        if fireNow {
            onChange?(value)
        }
    }

    func removeObserver() {
        valueChanged = nil
    }

}
```

Το παρακάτω αποτελεί παράδειγμα χρήσης αυτής της υλοποίησης του Observable Pattern. Η μεταβλητή dietData που είναι τύπου Observable και βρίσκεται στο DietViewModel κοινοποιεί την τιμή της όταν αυτή αλλάζει και ο DietViewController λαμβάνει ειδοποίηση για την αλλαγή αυτή και ενημερώνει το UI της συγκεκριμένης οθόνης.

```
class DietViewModel: BaseViewModelProtocol {

    // MARK: - Observables

    var isLoading = Observable<Bool?>(value: nil)
    var routingEnum = Observable<Diet.RoutingEnum?>(value: nil)

    var dietData = Observable<[DietModel]>(value: [])
```

```
private func registerObservers() {

        registerBaseObservers()

        viewModel?.routingEnum.addObserver({ [weak self] routingEnum in
            switch routingEnum {
            case .macros:
                self?.flowCoordinator?.moveToMacros()
            default: ()
            }
        })

        viewModel?.dietData.addObserver({ [weak self] data in
            self?.dietTableView.setup(data: data)
        })

    }
```

4.2.1.6 Models

Τα models είναι οι κλάσεις δεδομένων της εφαρμογής. Τα δεδομένα τα οποία λαμβάνονται από τη βάση δεδομένων είναι σε μορφή JSON, καθιστώντας έτσι την διαχείριση τους δύσκολη. Για την επίλυση του

προβλήματος αυτού σε μια Object Oriented γλώσσα όπως η Swift μετατρέπουμε τα JSON δεδομένα σε Model αντικείμενα, τα οποία είναι σαφώς πιο διαχειρίσιμα και εύχρηστα. Τα Models της εφαρμογής μπορούν να χωριστούν σε δύο υποκατηγορίες, τα DataModels και τα DomainModels.

Τα DataModels είναι τα μοντέλα τα οποία παράγονται από τη μετατροπή των JSON δεδομένων. Τα DataModels είναι τύπου Mappable. Ως mapping ορίζουμε τη διαδικασία μετατροπής JSON δεδομένων σε δεδομένων τύπου DataModels. Επειδή ως βάση δεδομένων της εφαρμογής χρησιμοποιείται το Firestore Database, όλα τα DataModels της εφαρμογής εμφανίζουν το γράμμα F στην αρχή του ονόματος τους. Το FDietModel είναι ένα από τα DataModels της εφαρμογής που αποθηκεύει πληροφορίες σχετικά με τη διατροφή του χρήστη.

```
import Foundation
import ObjectMapper
class FDietModel: Mappable {

    // MARK: - Properties

    var calories: Double?
    var carbohydrates: Double?
    var fats: Double?
    var protein: Double?
    var water: Double?

    // MARK: - Life cycle

    required init?(map: Map) { }

    init(calories: Double?, carbohydrates: Double?, fats: Double?, protein: Double?,
water: Double?) {
        self.calories = calories
        self.carbohydrates = carbohydrates
        self.fats = fats
        self.protein = protein
        self.calories = calories
    }

    // MARK: - Methods

    func mapping(map: Map) {
        calories <- map["calories"]
        carbohydrates <- map["carbohydrates"]
        fats <- map["fats"]
        protein <- map["protein"]
        water <- map["water"]
    }
}
```

Για κάθε DataModel της εφαρμογής υπάρχει μια αντιστοιχία σε ένα DomainModel. Τα DomainModels είναι μοντέλα τα οποία συνήθως αρχικοποιούνται από DataModels και χρησιμοποιούνται για να εκτελούν λειτουργίες επεξεργασία και μορφοποίησης των δεδομένων τους. Για παράδειγμα, αν υπάρχει η ανάγκη μετατροπής τιμής μιας μεταβλητής σε τιμή διαφορετικού τύπου ή αν χρειαζόμαστε το άθροισμα δύο μεταβλητών ενός μοντέλου, θα έπρεπε να υπάρχουν οι κατάλληλες μέθοδοι επεξεργασίας. Οι μέθοδοι αυτοί δεν μπορούν να ανήκουν σε ένα DataModel διότι το DataModel είναι υπεύθυνο μόνο για την μετατροπή δεδομένων JSON. Συνεπώς χρησιμοποιούμε ένα DomainModel και εκεί υλοποιούμε επιπλέον λειτουργικότητα αν αυτό απαιτείται. Το UserDietModel είναι ένα DomainModel το οποίο έχει έναν initializer που δέχεται ως παράμετρο το αντίστοιχο DataModel, το FDietModel.

```
import Foundation
class UserDietModel {

    // MARK: - Properties

    var calories: Double?
    var carbohydrates: Double?
    var fats: Double?
    var protein: Double?
    var water: Double?

    // MARK: - Life cycle

    init(calories: Double?, carbohydrates: Double?, fats: Double?, protein: Double?,
water: Double?) {
        self.calories = calories
        self.carbohydrates = carbohydrates
        self.fats = fats
        self.protein = protein
        self.calories = calories
    }

    init(from fDietModel: FDietModel?) {
        self.calories = fDietModel?.calories
        self.carbohydrates = fDietModel?.carbohydrates
        self.fats = fDietModel?.fats
        self.protein = fDietModel?.protein
        self.calories = fDietModel?.calories
    }

}
```

4.2.1.7 Enumerations

Τα enumerations στη Swift χρησιμοποιούνται για να δηλώσουν ένα κοινό τύπο (type) για ένα σύνολο διαφορετικών τιμών οι οποίες σχετίζονται μεταξύ τους [27]. Στην κάθε περίπτωση (case) ενός Enumeration μπορούμε προαιρετικά να θέσουμε μια τιμή είτε ως σταθερά, αναγκάζοντας έτσι και όλες τις υπόλοιπες περιπτώσεις να ακολουθούν το συγκεκριμένο τύπο τιμής, είτε ως συσχετισμένη τιμή (associated value) με τη μορφή παραμέτρου. Το NavigationButtonTypeEnum είναι ένα enumeration του οποίου οι τύποι δεν έχουν τιμή, πέραν από την περίπτωση title ή οποία δέχεται παραμετρικά κατά τη δημιουργία μια τιμή τύπου string.

```
import Foundation
enum NavigationButtonTypeEnum {
    case edit
    case done
    case next
    case history
    case settings
    case back
    case close
    case scan
    case title(title: String)
    case checkmark
}
enum NavigationButtonPositionEnum {
    case right
    case left
}
```

4.2.1.8 Managers

Οι managers είναι κλάσεις της εφαρμογής οι οποίες είναι υπεύθυνες για τη διεκπεραίωση συγκεκριμένων λειτουργιών. Είναι ένας τρόπος ομαδοποίησης κώδικα ο οποίος αφορά κάποια συγκεκριμένη ανάγκη της εφαρμογής. Ο AccountManager, τμήμα του οποίου παρατίθεται στη συνέχεια, είναι μια κλάση η οποία ακολουθεί το Singleton pattern, το οποίο περιγράφηκε προηγουμένως, και μέσα της συγκεντρώνει κώδικα που αφορά τη διαχείριση των χρηστών της εφαρμογής και συγκεκριμένα τη σύνδεση, την αποσύνδεση, τον έλεγχο για συνδεδεμένο χρήστη και τη διαγραφή χρήστη.

```
class AccountManager {

    // MARK: - Shared

    static let shared = AccountManager()

    // MARK: - Properties

    var user: User?
    var userDocimentId: String?

    // MARK: - Methods

    func signIn(completion: @escaping (Bool) -> Void) {
        guard let clientID = FirebaseApp.app()?.options.clientID else {
            print("No client ID")
            return
        }
        let config = GIDConfiguration(clientID: clientID)
        GIDSignIn.sharedInstance.signIn(with: config, presenting:
UIWindow.topMostViewController) { result, error in
            guard error == nil else {
                completion(false)
                return
            }

            guard let accessToken = result?.authentication.accessToken, let idToken =
result?.authentication.idToken else {
                print("Error getting the idToken or client token")
                completion(false)
                return
            }

            let credential = GoogleAuthProvider.credential(withIDToken: idToken,
                                                           accessToken: accessToken)

            Auth.auth().signIn(with: credential) { result, error in
                guard error == nil else {
                    completion(false)
                    return
                }
                self.user = result?.user
                completion(true)
            }
        }
    }
```

4.2.1.9 Extensions

Οι επεκτάσεις (extensions) προσδίδουν επιπλέον λειτουργικότητα σε μια κλάση ή ένα πρωτόκολλο με τη μορφή μεταβλητών ή μεθόδων. Πολλές φορές κατά την ανάπτυξη οι native δυνατότητες που παρέχονται δεν είναι επαρκείς και χρειάζεται custom παραμετροποίηση. Η επέκταση StringExtension,

που φαίνεται στη συνέχεια, προσθέτει κάποιες βοηθητικές μεθόδους στη κλάση String, οι οποίες χρησιμοποιούνται εκτενώς στην εφαρμογή και καθιστούν τον κώδικα ευανάγνωστο και κατανοητό.

```
import Foundation
import UIKit

extension String {

    func getLocalized() -> String {
        guard let bundle = appViewModel.localizationManagerProtocol.bundle else {
            return NSLocalizedString(self, comment: "")
        }
        return NSLocalizedString(self, tableName: nil, bundle: bundle, value: "",
comment: "")
    }

    func height(withConstrainedWidth width: CGFloat, font: UIFont) -> CGFloat {
        let constraintRect = CGSize(width: width, height: .greatestFiniteMagnitude)
        let boundingBox = self.boundingRect(with: constraintRect, options:
.usesLineFragmentOrigin, attributes: [NSAttributedString.Key.font: font], context: nil)

        return ceil(boundingBox.height)
    }

    func width(withConstrainedHeight height: CGFloat, font: UIFont) -> CGFloat {
        let constraintRect = CGSize(width: .greatestFiniteMagnitude, height: height)
        let boundingBox = self.boundingRect(with: constraintRect, options:
.usesLineFragmentOrigin, attributes: [NSAttributedString.Key.font: font], context: nil)
        return ceil(boundingBox.width)
    }
}
```

4.2.1.10 Protocols

Τα πρωτόκολλα (protocols) στη Swift χρησιμοποιούνται ως ένα blueprint, καθορίζοντας ιδιότητες, μεθόδους και άλλες λειτουργίες τις οποίες οντότητες που υιοθετούν το πρωτόκολλο πρέπει να υπακούν [28]. Πολλές φορές τα πρωτόκολλα χρησιμοποιούνται για να ομαδοποιήσουν ή να επεκτείνουν μια ιδιότητα μιας ομάδας οντοτήτων. Παράδειγμα της συγκεκριμένης περίπτωσης χρήσης αποτελεί το ReusableViewProtocol, το οποίο προσδίδει επιπλέον παραμετροποίηση σε όλα τα UIViews, με τον παρακάτω τρόπο.

```
import Foundation
import UIKit
protocol ReusableViewProtocol {
    static var reuseIdentifier: String { get }
}
extension ReusableViewProtocol {
    static var reuseIdentifier: String {
        return String(describing: self)+"Identifier"
    }
}
extension UIView: ReusableViewProtocol { }
```

4.2.1.11 Classes

Για την ανάπτυξη της εφαρμογής χρειάστηκε να δημιουργηθούν προσαρμοσμένες κλάσεις (custom classes), οι οποίες καλύπτουν διάφορες ανάγκες που αφορούν τη διεπαφή χρήστη, αλλά και εσωτερική λειτουργικότητα. Οι κλάσεις αυτές σε κάποιες περιπτώσεις κληρονομούν από κλάσεις του συστήματος, κυρίως όταν μιλάμε για custom UI classes, ενώ σε άλλες περιπτώσεις είναι απλά οντότητες που

διευκολύνουν στη λειτουργικότητα. Δύο παραδείγματα custom κλάσεων αποτελούν οι κλάσεις CustomAnnotation και Stopwatch, οι οποίες φαίνονται παρακάτω.

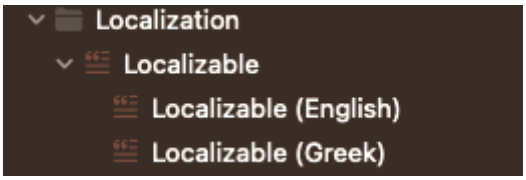
```
import Foundation
import MapKit
class CustomAnnotation: NSObject, MKAnnotation {
    let title: String?
    let subtitle: String?
    let coordinate: CLLocationCoordinate2D
    let coordinateType: CoordinateType

    init(coordinateType: CoordinateType, coordinate: CLLocationCoordinate2D) {
        self.coordinateType = coordinateType
        self.title = coordinateType == .start ?
"ActivitySession_annotation_start".getLocalized() :
"ActivitySession_annotation_finish".getLocalized()
        self.subtitle = coordinateType == .start ?
"ActivitySession_annotation_start_description".getLocalized() :
"ActivitySession_annotation_finish_description".getLocalized()
        self.coordinate = coordinate
    }
}
enum CoordinateType {
    case start
    case end
}
```

```
import Foundation
class Stopwatch {

    // MARK: - Properties

    private var startTime: Date?
    private var accumulatedTime: TimeInterval = 0

    // MARK: - Methods

    func start() {
        self.startTime = Date()
    }

    func stop() {
        self.accumulatedTime = self.elapsedTime()
        self.startTime = nil
    }

    func reset() {
        self.accumulatedTime = 0
        self.startTime = nil
    }

    func elapsedTime() -> TimeInterval {
        return -(self.startTime?.timeIntervalSinceNow ?? 0) + self.accumulatedTime
    }

    func getTimeString() -> String {
        let hours = Int(self.elapsedTime() / 3600)
        let minutes = Int(self.elapsedTime() / 60) % 60
        let seconds = Int(self.elapsedTime()) % 60
        let fractions = Int((self.elapsedTime().truncatingRemainder(dividingBy: 1)) *
100)
        let timeString = String(format: "%02d:%02d:%02d:%02d", hours, minutes, seconds,
fractions)
```

```
        return timeString
    }
}
```

4.2.2 Localization

Με τον όρο Localization αναφερόμαστε στη διαδικασία μετάφρασης και προσαρμογής της εφαρμογής σε πολλαπλές γλώσσες και περιοχές. Συγκεκριμένα για την επίτευξη του στόχου αυτού δημιουργείται ένα αρχείο Localizable, το οποίο χωρίζεται σε υπο-αρχεία ανάλογα με το πλήθος των γλωσσών τις οποίες θέλουμε η εφαρμογή να υποστηρίζει, όπως φαίνεται στο Σχήμα 4.7 . Στην περίπτωση του fitness app το αρχείο παρέχει λεκτικά για την Αγγλική και την Ελληνική γλώσσα. Η γλώσσα εφαρμογής αρχικά επιλέγεται με βάση τη γλώσσα συσκευής του χρήστη και έπειτα ο ίδιος μπορεί να την αλλάξει από την ενότητα του προφίλ του. Στο Σχήμα 4.8 φαίνεται τμήμα του Localizable αρχείου της εφαρμογής, τόσο στα Ελληνικά όσο και στα Αγγλικά.



Σχήμα 4.7 Διαχωρισμός αρχείου Localizable

```
// MARK: - Welcome

"Welcome_welcomeLabel" = "Welcome";
"Welcome_googleSignInLabel" = "Sign in with Google";
"Welcome_guestButton" = "Or continue as Guest";

// MARK: - Registration

"Registration_Name_titleLabel" = "What is your name?";
"Registration_Name_placeholder" = "Name";

"Registration_BodyComposition_titleLabel" = "Give us the following body composition measurements";
"Registration_BodyComposition_weightPlaceholder" = "Weight (in kg.)";
"Registration_BodyComposition_musclePlaceholder" = "Lean Muscle Mass Percentage (%) - Optional";
"Registration_BodyComposition_fatPlaceholder" = "Fat Mass Percentage (%) - Optional";

"Registration_ActivityGoals_titleLabel" = "If you want, set daily goals and we will help you track them!";
"Registration_ActivityGoals_stepsPlaceholder" = "Steps";
"Registration_ActivityGoals_caloriesPlaceholder" = "Calories Burned";

"Registration_DietyGoals_titleLabel" = "Almost Done!\nIf you want, set daily diet goals and track them in the diet tab!";
"Registration_DietGoals_caloriesPlaceholder" = "Calorie Intake (in kcal)";
"Registration_DietGoals_proteinPlaceholder" = "Protein Intake (in gram.)";
"Registration_DietGoals_carbsPlaceholder" = "Carbohydrates Intake (in gram.)";
"Registration_DietGoals_fatsPlaceholder" = "Fats Intake (in gram.)";
"Registration_DietGoals_waterPlaceholder" = "Water Intake (in mL.)";



// MARK: - Lobby

"Lobby_homeTitle" = "Home";
"Lobby_activityTitle" = "Activity";
"Lobby_workoutTitle" = "Workout";
"Lobby_dietTitle" = "Diet";
"Lobby_profileTitle" = "Profile";
```

```
// MARK: - Welcome

"Welcome_welcomeLabel" = "Καλωσήρθες";
"Welcome_googleSignInLabel" = "Σύνδεση με Google";
"Welcome_guestButton" = "Η συνέχισε ως καλεσμένος";

// MARK: - Registration

"Registration_Name_titleLabel" = "Ποιό είναι το όνομα σου;";
"Registration_Name_placeholder" = "Όνομα";

"Registration_BodyComposition_titleLabel" = "Δώσε τις παρακάτω μετρήσεις σχετικά με το σώμα σου";
"Registration_BodyComposition_weightPlaceholder" = "Βάρος (σε kg.)";
"Registration_BodyComposition_musclePlaceholder" = "Ποσοστό Μυικής Μάζας (%) - Προεραιτικά";
"Registration_BodyComposition_fatPlaceholder" = "Ποσοστό Λιπώδους Ιστού (%) - Προεραιτικά";

"Registration_ActivityGoals_titleLabel" = "Εαν θέλεις, θέσε ημερίσιους στόχους κι εμείς θα σε βοηθήσουμε να τους ελέγχεις!";
"Registration_ActivityGoals_stepsPlaceholder" = "Βήματα";
"Registration_ActivityGoals_caloriesPlaceholder" = "Καύση Θερμίδων";

"Registration_DietyGoals_titleLabel" = "Σχεδον τελείωσες!/nΕαν θέλεις, θέσε ημερίσιους διατροφικούς στόχος και έλεγξε τους στο αντίστοιχο παράθυρο!";
"Registration_DietGoals_caloriesPlaceholder" = "Πρόσληψη Θερμίδων (in kcal)";
"Registration_DietGoals_proteinPlaceholder" = "Πρόσληψη Πρωτεΐνης (σε γραμ.)";
"Registration_DietGoals_carbsPlaceholder" = "Πρόσληψη Υδατανθράκων (σε γραμ.)";
"Registration_DietGoals_fatsPlaceholder" = "Πρόσληψη Λίπους (σε γραμ.)";
"Registration_DietGoals_waterPlaceholder" = "Πρόσληψη Νερού (σε μλ.)";

// MARK: - Lobby

"Lobby_homeTitle" = "Αρχική";
"Lobby_activityTitle" = "Άθληση";
"Lobby_workoutTitle" = "Γυμναστήριο";
"Lobby_dietTitle" = "Διατροφή";
"Lobby_profileTitle" = "Προφίλ";
```

Σχήμα 4.8 Αρχείο Localizable στα Ελληνικά / Αγγλικά

4.2.3 Βιβλιοθήκες

Παρακάτω παρατίθενται όλες οι εξωτερικές βιβλιοθήκες που χρησιμοποιήθηκαν για την ανάπτυξη της mobile εφαρμογής. Οι βιβλιοθήκες αυτές είναι στο σύνολο τους open source και διατίθενται προς χρήση κατά κανόνα με το standard MIT License.

4.2.3.1 CocoaPods

Το CocoaPods είναι το εργαλείο που χρησιμοποιήθηκε για την εγκατάσταση και τη διαχείριση όλων των βιβλιοθηκών που θα αναφερθούν παρακάτω. Το CocoaPods είναι ένας διαχειριστής εξαρτημάτων (dependency manager) φτιαγμένος για Swift και Objective-C προτζεκτ [29]. Γραμμένος σε Ruby, αποτελεί είναι πολύ σημαντικό εργαλείο οργάνωσης και διαχείρισης έτσι ώστε όλες οι βιβλιοθήκες να λειτουργούν ομαλά κατά τον προγραμματισμό και τη λειτουργία της εφαρμογής. Αυτό επιτυγχάνεται με τη χρήση του podfile, το οποίο δημιουργείται από το CocoaPods και μέσα του καταγράφονται οι προς χρήση βιβλιοθήκες καθώς και ο απαραίτητος κώδικας αρχικοποίησης και παραμετροποίησης, καθώς και με τη χρήση command line εντολών για την εγκατάσταση, απεγκατάσταση, ενημέρωση αυτών κ.α.

4.2.3.2 Alamofire

Το Alamofire είναι μια βιβλιοθήκη δικτύωσης HTTP γραμμένη σε Swift [30]. Η βιβλιοθήκη παρέχει μια πληθώρα δυνατοτήτων στον προγραμματιστή, βασική της χρήση στην εν λόγω περίπτωση όμως αποτελεί η διευκόλυνση και η καθαρότητα στον προγραμματισμό των HTTP request της εφαρμογής. Σε αυτό το κομμάτι, η βιβλιοθήκη ουσιαστικά αποτελεί επέκταση των native frameworks που παρέχονται από τη Swift, και συγκεκριμένα του URLSession και URLSessionTask. Σημαντικό στοιχείο της βιβλιοθήκης, το οποίο αποτελεί το standard σε διαδικασίες δικτύωσης, είναι ότι αυτές πραγματοποιούνται ασύγχρονα.

4.2.3.3 ObjectMapper

Το ObjectMapper είναι μια βιβλιοθήκη που διευκολύνει τη μετατροπή δεδομένων από JSON σε μοντέλα (κλάσεις εφαρμογής - model objects) και το αντίστροφο, γραμμένη σε swift [31]. Η διαδικασία αυτή αναφέρεται ως mapping (εξου και ObjectMapper) και η βιβλιοθήκη αυτή αποτελεί χρήσιμο εργαλείο, ειδικά σε περιπτώσεις mapping με εμφωλευμένα αντικείμενα ή σε περιπτώσεις όπου τα δεδομένα χρειάζονται ειδική προσαρμογή κατά το mapping (custom transformations).

4.2.3.4 SwiftyJson

Το SwiftyJson, γραμμένο σε Swift, βοηθά στη διαχείριση JSON δεδομένων [32]. Συγκεκριμένα η χρησιμότητα του φαίνεται στη διαχείριση τύπων, καθιστώντας τον κώδικα πιο καθαρό και ευανάγνωστο.

4.2.3.5 Kingfisher

Το Kingfisher είναι μια Swift βιβλιοθήκη που χρησιμοποιείται στη λήψη και στη προσωρινή αποθήκευση (caching) εικόνων από το διαδίκτυο [33]. Ειδικά στο κομμάτι του caching προσφέρονται πολλές δυνατότητες όπως αυτόματο caching με διάφορες παραμέτρους (ημερομηνία λήξης, χώρος στη μνήμη), αυτόματη ακύρωση λήψης και χρήση προηγούμενων λήψεων για βελτιωμένη απόδοση κ.α.

4.2.3.6 SwiftyUserDefaults

Τα UserDefaults ή αλλιώς η user's default database αποτελεί μια βάση δεδομένων του συστήματος, στην οποία αποθηκεύονται δεδομένα με τη μορφή key-value ζευγαριών [34]. Τα δεδομένα αυτά χρησιμοποιούνται στην εφαρμογή για να καθορίσουν συμπεριφορές αυτής ανάλογα με τις προτιμήσεις του χρήστη καθώς και για την αποθήκευση και διατήρηση δεδομένων μετά από πολλαπλές εκκινήσεις. Η βιβλιοθήκη SwiftyUserDefaults, γραμμένη σε Swift, επεκτείνει την native κλάση NSUserDefaults και κάνει τη διαχείριση των UserDefaults ευκολότερη.

4.2.3.7 Lottie-ios

Το lottie-ios είναι μια βιβλιοθήκη που επιτρέπει την απόδοση κινουμένων σχεδίων σε μορφή διανυσμάτων (vector-based animation rendering) σε πραγματικό χρόνο με τη λιγότερη δυνατή χρήση κώδικα [35].

4.2.3.8 Floaty

Η βιβλιοθήκη floaty προσφέρει ένα νέο component το οποίο δεν παρέχεται natively από τη Swift, το floating action button [36].

4.2.3.9 IQKeyboardManager

Με τη χρήση του IQKeyboardManager γίνεται διαχείριση των περιπτώσεων όπου απαιτείται εισαγωγή δεδομένων από το χρήστη και το πληκτρολόγιο καλύπτει μεγάλο μέρος της οθόνης και κρύβει το πεδίο το οποίο ο χρήστης συμπληρώνει [37].

4.2.3.10 SVProgressHud

Πολλές φορές στη χρήση της εφαρμογής, κυρίως στις περιπτώσεις χρήσης όπου υπάρχουν κλήσεις στη βάση δεδομένων για λήψη ή αποθήκευση δεδομένων, απαιτείται από το χρήστη να περιμένει μέχρι να τελειώσει η εν λόγω διαδικασία ή οποία κατά το πέρας της πυροδοτεί αλλαγές στη διεπαφή. Στις περιπτώσεις αυτές είναι συνήθης πρακτική να εμφανίζεται ένας loader, ο οποίος δηλώνει ότι η εφαρμογή πραγματοποιεί κάποιο task. Η βιβλιοθήκη SVProgressHud, γραμμένη σε Objective-C, είναι υπεύθυνη για να δείξει και να παραμετροποιήσει τον loader [38].

4.2.3.11 GoogleSignIn-iOS

Το GoogleSignIn-iOS είναι η βιβλιοθήκη που παρέχεται από την Google έτσι ώστε η εφαρμογή να μπορεί να κάνει χρήση εξωτερικής αυθεντικοποίησης χρήστη με google [39]. Αντίστοιχη βιβλιοθήκη παρέχεται και στο Android, η οποία πραγματοποιεί ακριβώς την ίδια λειτουργία. Το GoogleSignIn-iOS χρησιμοποιείται συνδυαστικά μαζί με την αμέσως επόμενη βιβλιοθήκη για να γίνει η αυθεντικοποίηση και η εγγραφή του χρήστη στη βάση.

4.2.3.12 FirebaseAuth

Το FirebaseAuth είναι ένα από τα πακέτα της βιβλιοθήκης Firebase[40] (firebase-ios-sdk), το οποίο χρησιμοποιείται σε συνδυασμό με το GoogleSignIn-iOS για την αυθεντικοποίηση ενός χρήστη. Συγκεκριμένα ο χρήστης πρώτα αυθεντικοποιείται μέσω google και κατά την επιτυχή σύνδεση η εφαρμογή λαμβάνει ένα token, το οποίο χρησιμοποιείται από το FirebaseAuth για να γίνει πιστοποίηση του χρήστη στο Firebase.

4.2.3.13 FirebaseFirestore

Άλλο ένα πακέτο της βιβλιοθήκης Firebase, το Firebase Firestore χρησιμοποιείται εκτενώς από την εφαρμογή για οποιαδήποτε δικτυακή διαδικασία μεταξύ αυτής και της βάσης δεδομένων Firestore. Μέσα από τη βιβλιοθήκη αυτή παρέχονται μέθοδοι άντλησης, εισαγωγής, ενημέρωσης και διαγραφής δεδομένων από το Firestore.

4.2.3.14 Charts

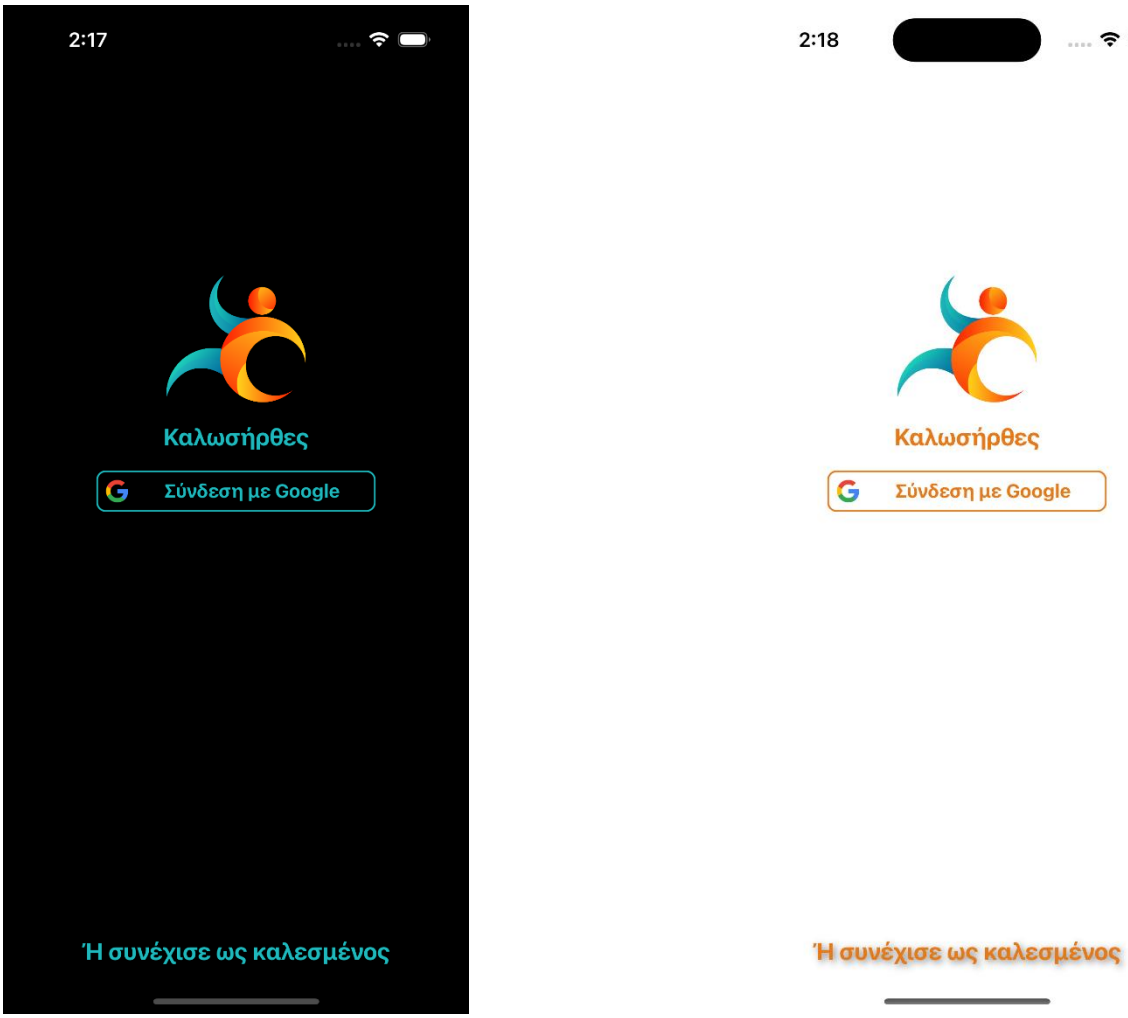
Η βιβλιοθήκη αυτή, γραμμένη σε Swift, είναι υπεύθυνη για όλα τα διαγράμματα τα οποία βρίσκονται στην εφαρμογή, συγκεκριμένα στο ιστορικό χρήστη [41]. Εμπνευσμένη από την αντίστοιχη Android βιβλιοθήκη MPAndroidChart, το Charts παρέχει ένα σύνολο διαγραμμάτων, εύκολα στη χρήση και την παραμετροποίηση τους.

4.2.3.15 SwiftLint

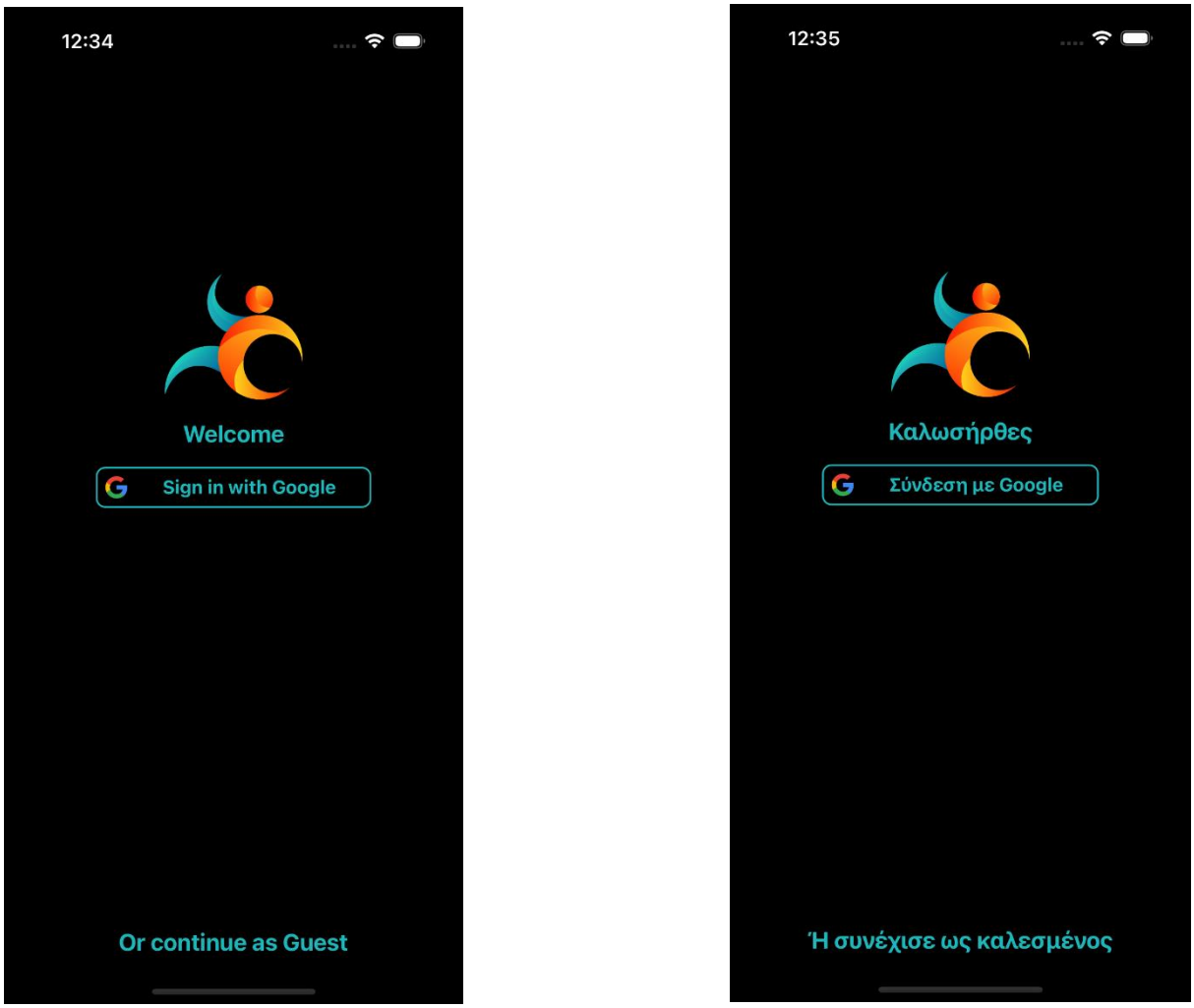
Η SwiftLint δεν προσφέρει κάποια λειτουργικότητα στην εφαρμογή, είναι όμως πολύ χρήσιμη κατά την ανάπτυξη και τον προγραμματισμό [42]. Με χρήση προειδοποιήσεων και άλλων οπτικών βοηθημάτων, υπενθυμίζει στον προγραμματιστή να γράφει καθαρό, ευανάγνωστο κώδικα και να ακολουθεί τα πρότυπα γραφής κώδικα swift. Πέρα από αυτά, υποδεικνύει περιπτώσεις στις οποίες μπορεί να προκύψουν crashes λόγω κακής υλοποίησης και αποτρέπει την εφαρμογή από τη λειτουργία μέχρι τα εν λόγω λάθη να διορθωθούν. Η αυστηρότητα στις υποδείξεις και τα οπτικά βοηθήματα είναι παραμετροποιήσιμη έτσι ώστε να ταιριάζει στις απαιτήσεις του εκάστοτε project.

4.2.4 Οθόνες Διεπαφής Χρηστών – Περιπτώσεις Χρήσης

Σε αυτό το σημείο γίνεται παρουσίαση της τελικής μορφής της εφαρμογής, όπως αυτή αναπτύχθηκε με τη χρήση όλων των παραπάνω τεχνολογιών και εργαλείων. Μέσα από περιπτώσεις χρήσης και screenshot των οθονών της εφαρμογής, θα πραγματοποιηθεί μια περιήγηση σε αυτή και στις λειτουργίες τις οποίες προσφέρει. Πριν από αυτό και για λόγους συντομίας, στο Σχήμα 4.9 και Σχήμα 4.10 παρουσιάζονται με παράδειγμα οι υποστηριζόμενες γλώσσες τις εφαρμογής (Ελληνικά – Αγγλικά) και η συμπεριφορά της διεπαφής σε light και dark mode. Όλες οι οθόνες τις εφαρμογής υποστηρίζουν τις λειτουργίες αυτές και συμπεριφέρονται με παρόμοιο τρόπο με τα σχήματα αυτά.



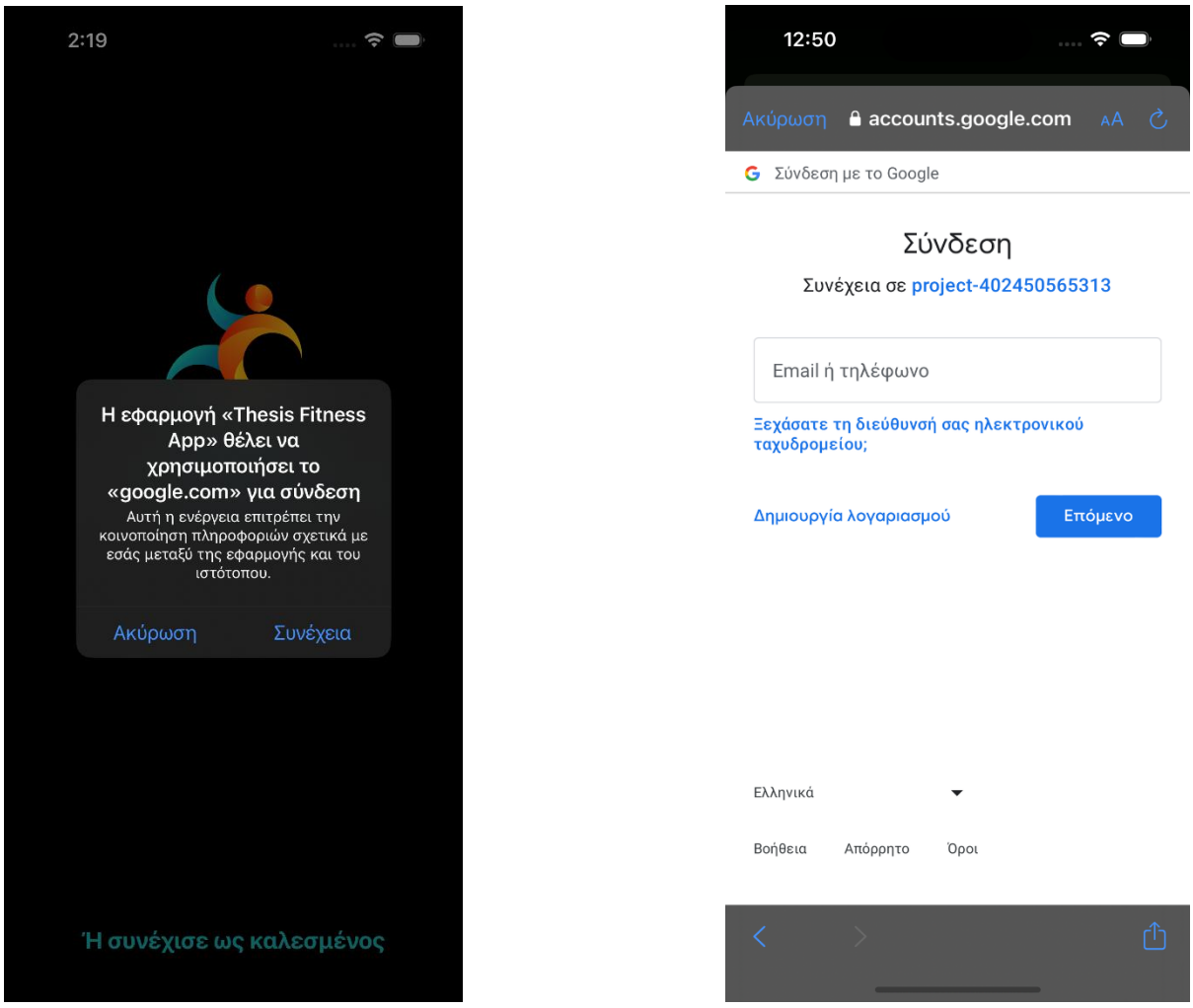
Σχήμα 4.9 Light – Dark Mode



Σχήμα 4.10 Παράδειγμα Localization

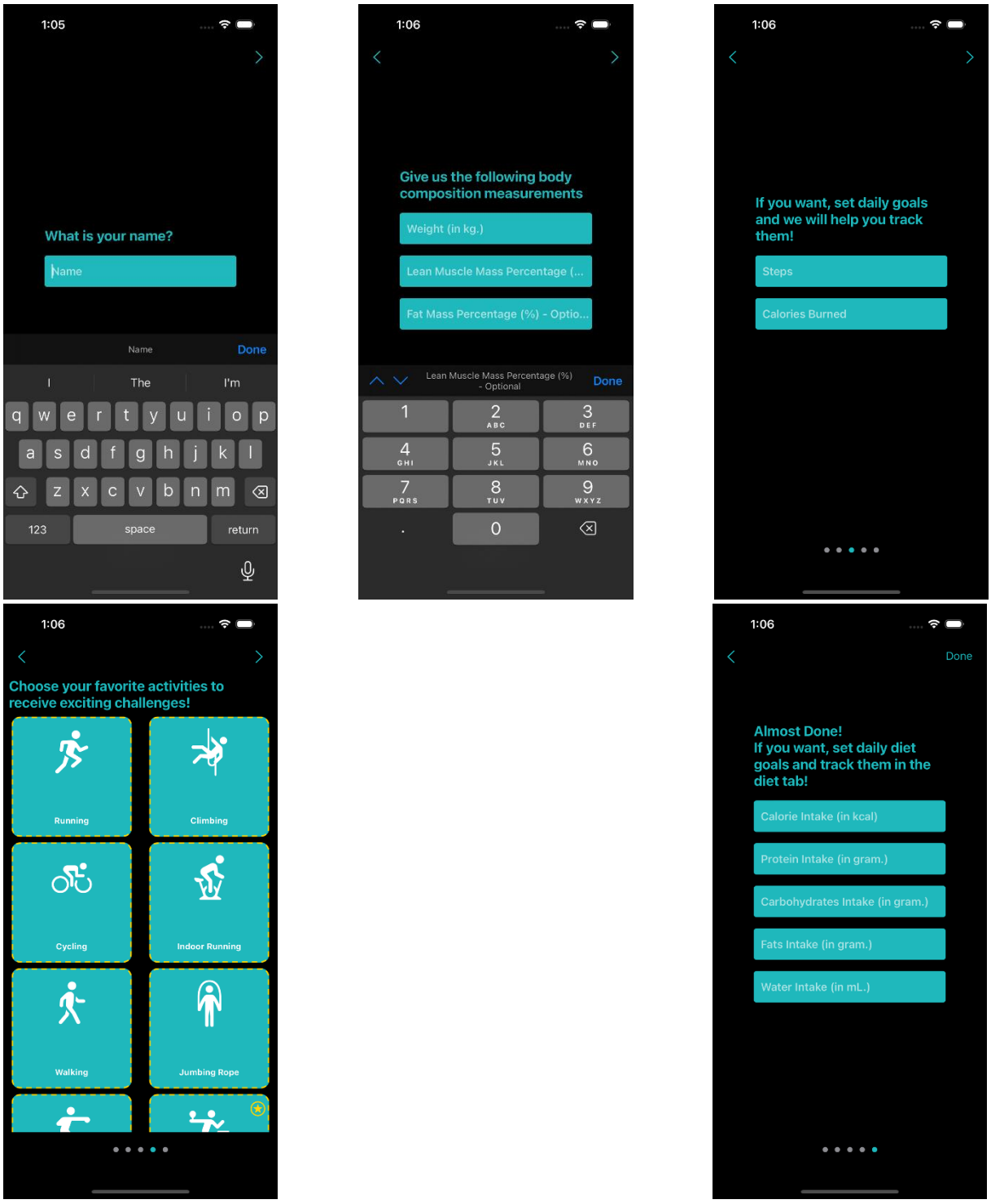
4.2.4.1 Welcome – Registration

Ο χρήστης, την πρώτη φορά που ανοίγει την εφαρμογή καταλήγει στην οθόνη Welcome όπως αυτή φαίνεται στο Σχήμα 4.9 και 4.10. Έπειτα, πατώντας το κουμπί Sign in with Google και δίνοντας την απαραίτητη έγκριση, ο χρήστης μπορεί να συνδεθεί με το google λογαριασμό του. Σε διαφορετική περίπτωση μπορεί απλώς να συνεχίσει ως καλεσμένος. Στην περίπτωση του καλεσμένου τα δεδομένα του από τη χρήση της εφαρμογής δεν αποθηκεύονται και συνεπώς χάνονται κατά το κλείσιμο της εφαρμογής.



Σχήμα 4.11 Σύνδεση με λογαριασμό Google

Κατά την πρώτη επιτυχή σύνδεση με έναν google λογαριασμό, ο χρήστης πρέπει να εκτελέσει τη διαδικασία του Registration. Το Registration της εφαρμογής αποτελείται στο σύνολο του από πέντε οθόνες και σκοπός του είναι να συλλέξει σημαντικές πληροφορίες από το χρήστη.

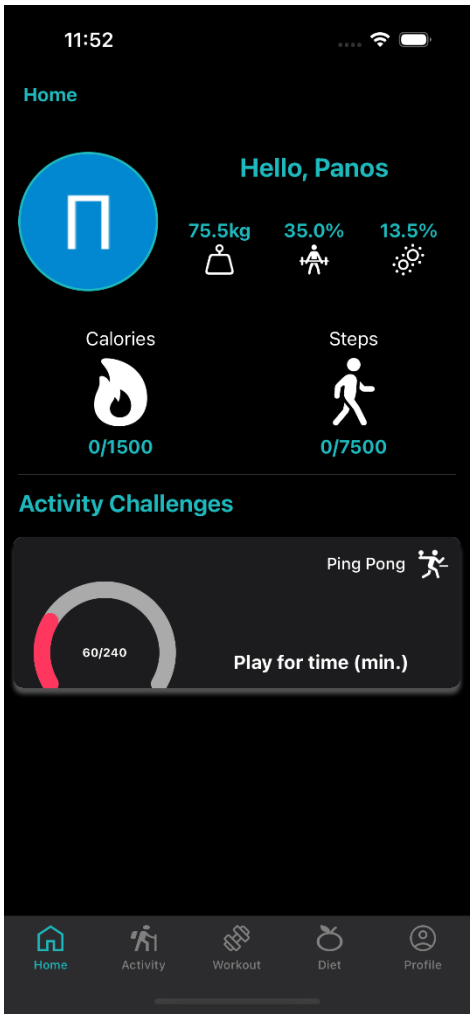


Σχήμα 4.12 Registration

Αν ο χρήστης έχει ξανασυνδεθεί με τον ίδιο λογαριασμό στο παρελθόν θα υπάρχει εγγραφή του στη βάση δεδομένων. Συνεπώς δεν ακολουθεί τη διαδικασία του registration και οδηγείται κατευθείαν στο Lobby της εφαρμογής και συγκεκριμένα στην οθόνη Home.

4.2.4.2 Home

Φτάνοντας στο Home Screen ο χρήστης μπορεί να δει μια περίληψη των δεδομένων του καθώς και το σύνολο των προκλήσεων που είναι διαθέσιμες για αυτόν, ανάλογα με τις αγαπημένες του δραστηριότητες.

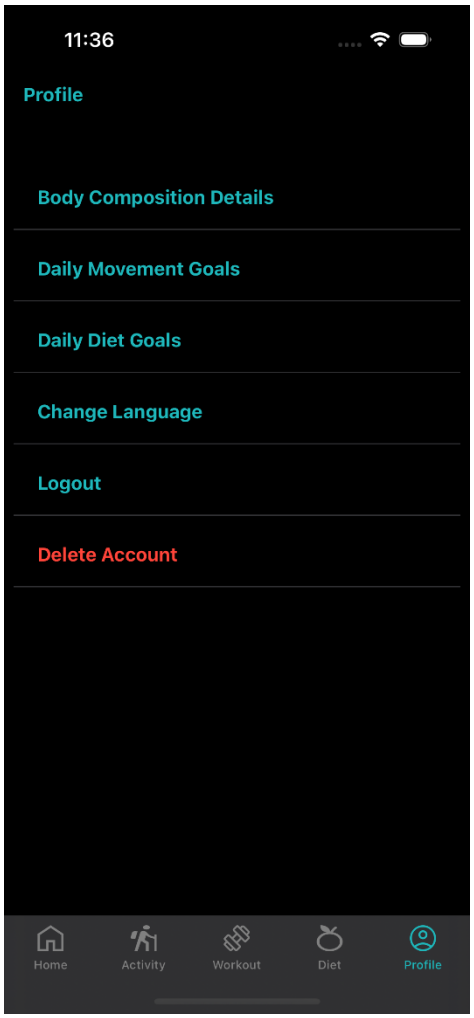


Σχήμα 4.13 Home Screen

Ξεκινώντας από το πάνω μέρος τις οθόνης συναντάμε το όνομα και την εικόνα του χρήστη (η οποία συμπληρώνεται αυτόματα από τον google λογαριασμό του). Έπειτα βλέπουμε το βάρος του χρήστη καθώς και τη σύνθεση του σώματος του εάν αυτός επέλεξε να τη συμπληρώσει κατά το registration. Ακριβώς από κάτω φαίνονται τα βήματα και οι θερμίδες που κατανάλωσε τη συγκεκριμένη μέρα (ο υπολογισμός των θερμίδων γίνεται κατά προσέγγιση βάση κιλών και κατάλληλου αλγορίθμου). Τέλος, στο κάτω μέρος της οθόνης υπάρχει το μενού πλοήγησης το οποίο οδηγεί το χρήστη στις υπόλοιπες οθόνες της εφαρμογής. Στα επόμενα υποκεφάλαια περιγράφονται οι οθόνες αυτές, ξεκινώντας από τα δεξιά προς τα αριστερά.

4.2.4.3 Profile

Η οθόνη του Profile αφορά βασικές ρυθμίσεις και ενημερώσεις των στοιχείων του χρήστη καθώς επίσης και τη διαχείριση του λογαριασμού του.

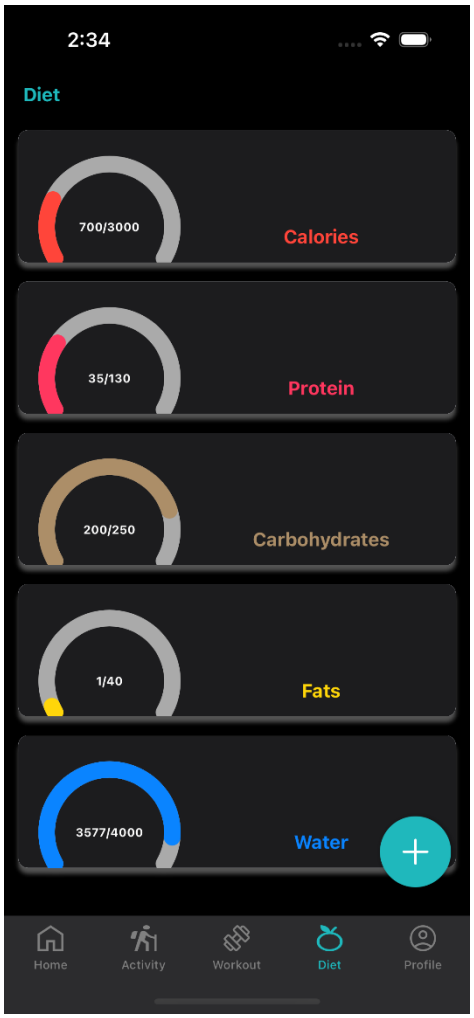


Σχήμα 4.14 Profile Screen

Συγκεκριμένα ο χρήστης μπορεί να αλλάξει τη σύνθεση του σώματος του και το βάρος του, τον καθημερινό στόχο του σχετικά με τα βήματα και τους ημερήσιους διατροφικούς του στόχους. Σε αυτό το σημείο επίσης γίνεται και η αλλαγή γλώσσας της εφαρμογής από Αγγλικά σε Ελληνικά και το αντίστροφο. Για την αλλαγή θέματος από light σε dark mode δεν απαιτείται δυνατότητα ρύθμισης μέσα από την εφαρμογή. Η εφαρμογή επιλέγει θέμα με βάση το προεπιλεγμένο θέμα για το σύνολο των εφαρμογών, το οποίο λαμβάνει με βάση τη δηλωμένη προτίμηση του χρήστη από το λειτουργικό σύστημα του κινητού. Από την οθόνη του προφίλ ο χρήστης μπορεί επίσης να αποσυνδεθεί ή να διαγράψει το λογαριασμό του. Με τη διαγραφή του λογαριασμού αφαιρείται από τη βάση το σύνολο των δεδομένων του χρήστη, όπως αυτό προβλέπεται από τα apple developer guidelines. Τόσο στη διαγραφή όσο και στην αποσύνδεση, ο χρήσης οδηγείται στην οθόνη Welcome όπως αυτή περιεγράφηκε παραπάνω.

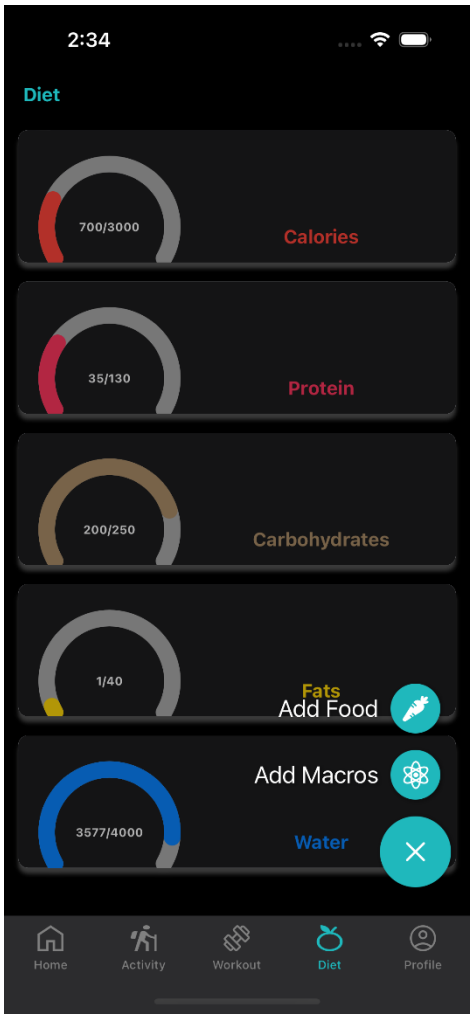
4.2.4.4 Diet

Στην υποκατηγορία Diet ο χρήστης διαχειρίζεται τα ημερήσια διατροφικά του δεδομένα με βάση τα γεύματα του και τις τροφές που καταναλώνει.



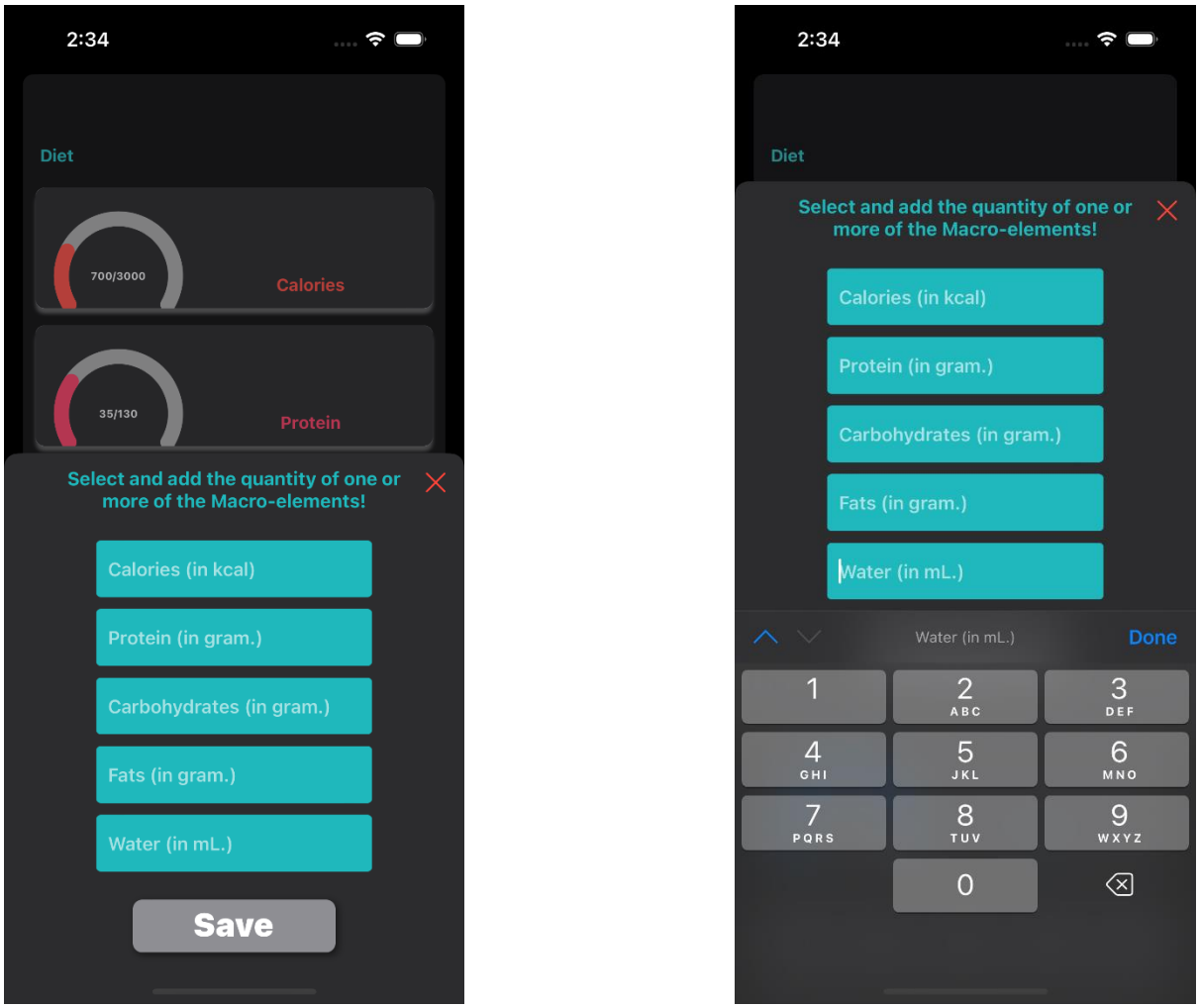
Σχήμα 4.15 Diet Screen

Το μεγαλύτερο μέρος της οθόνης Diet αποτελείται από τις πέντε μετρήσεις τις οποίες ο χρήστης μπορεί να διαχειριστεί. Οι μετρήσεις αυτές αφορούν την ημερήσια πρόσληψη και είναι από πάνω προς τα κάτω: θερμίδες, πρωτεΐνη, υδατάνθρακες, λίπη, νερό. Ο χρήστης μπορεί να θέσει ημερήσιους στόχους πρόσληψης για κάθε μια από αυτές τις τιμές. Πατώντας το κουμπί « + » που βρίσκεται στο κάτω δεξιά μέρος της οθόνης, ο χρήστης μπορεί να εισάγει δεδομένα από μια τροφή που κατανάλωσε με δύο τρόπους, όπως φαίνεται στο Σχήμα 4.16.



Σχήμα 4.16 Diet Floating Action Button

Πατώντας την επιλογή “add Macros” εμφανίζεται ένα μενού εισαγωγής μακροθρεπτικών όπως παρουσιάζεται στο Σχήμα 4.17. Εκεί ο χρήστης μπορεί να προσθέσει ένα ή περισσότερα μακροθρεπτικά στοιχεία. Πατώντας το κουμπί “save” τα στοιχεία αυτά αποθηκεύονται στη βάση δεδομένων, προκαλώντας έτσι και την ενημέρωση της γραφικής διεπαφής της οθόνης Diet.



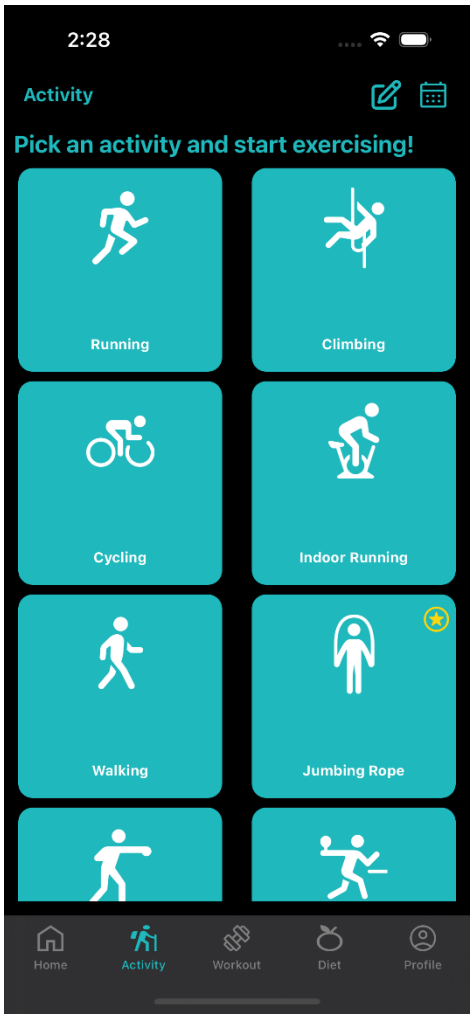
Σχήμα 4.17 Add Macros

Πατώντας την επιλογή “add Food” ο χρήστης οδηγείται στην οθόνη αναζήτησης φαγητών, τα δεδομένα της οποίας παρέχονται δωρεάν από το FoodData Centra A.P.I. . Στην οθόνη αυτή ο χρήστης αναζήτα κατευθείαν την τροφή την οποία κατανάλωσε και εισάγοντας την ποσότητα κατανάλωσης σε γραμμάρια οδηγείται πίσω στην οθόνη Diet. Στο παρασκήνιο, υπολογίζονται τα μακροθρεπτικά συστατικά από την τροφή και την ποσότητα κατανάλωσης, τα συστατικά αυτά αποθηκεύονται στη βάση και γίνεται ενημέρωση της γραφικής διεπαφής, όπως και στην περίπτωση “add Macros”. Στο Σχήμα 4.18 φαίνεται η εν λόγω οθόνη.

Σχήμα 4.18 Add Food

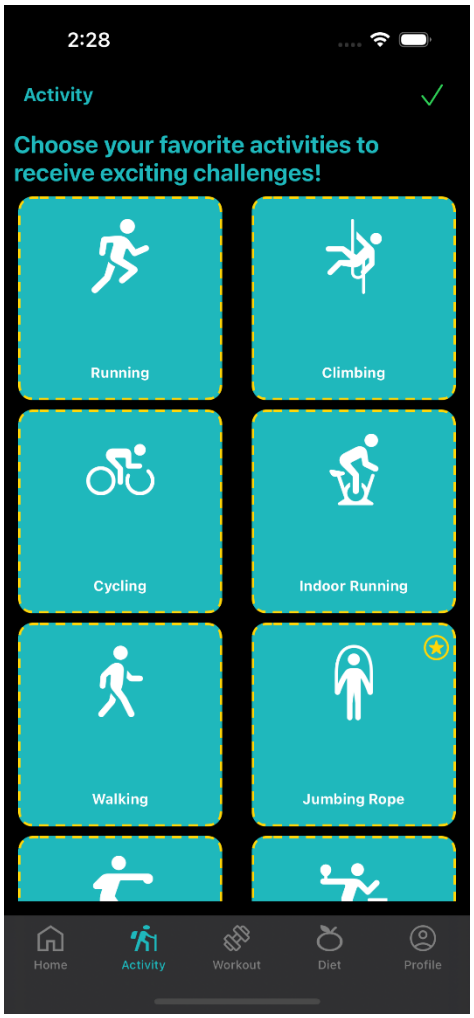
4.2.4.5 Activity

Η υποκατηγορία Activity αποτελείται από ένα σύνολο οθονών το οποίο λειτουργεί ως βοήθεια του χρήστη κατά τη διάρκεια μια αθλητικής δραστηριότητας. Ξεκινώντας με την οθόνη Activity, η οποία φαίνεται στο Σχήμα 4.19, παρουσιάζονται οι διαθέσιμες δραστηριότητες τις οποίες ο χρήστης μπορεί να εκτελέσει.



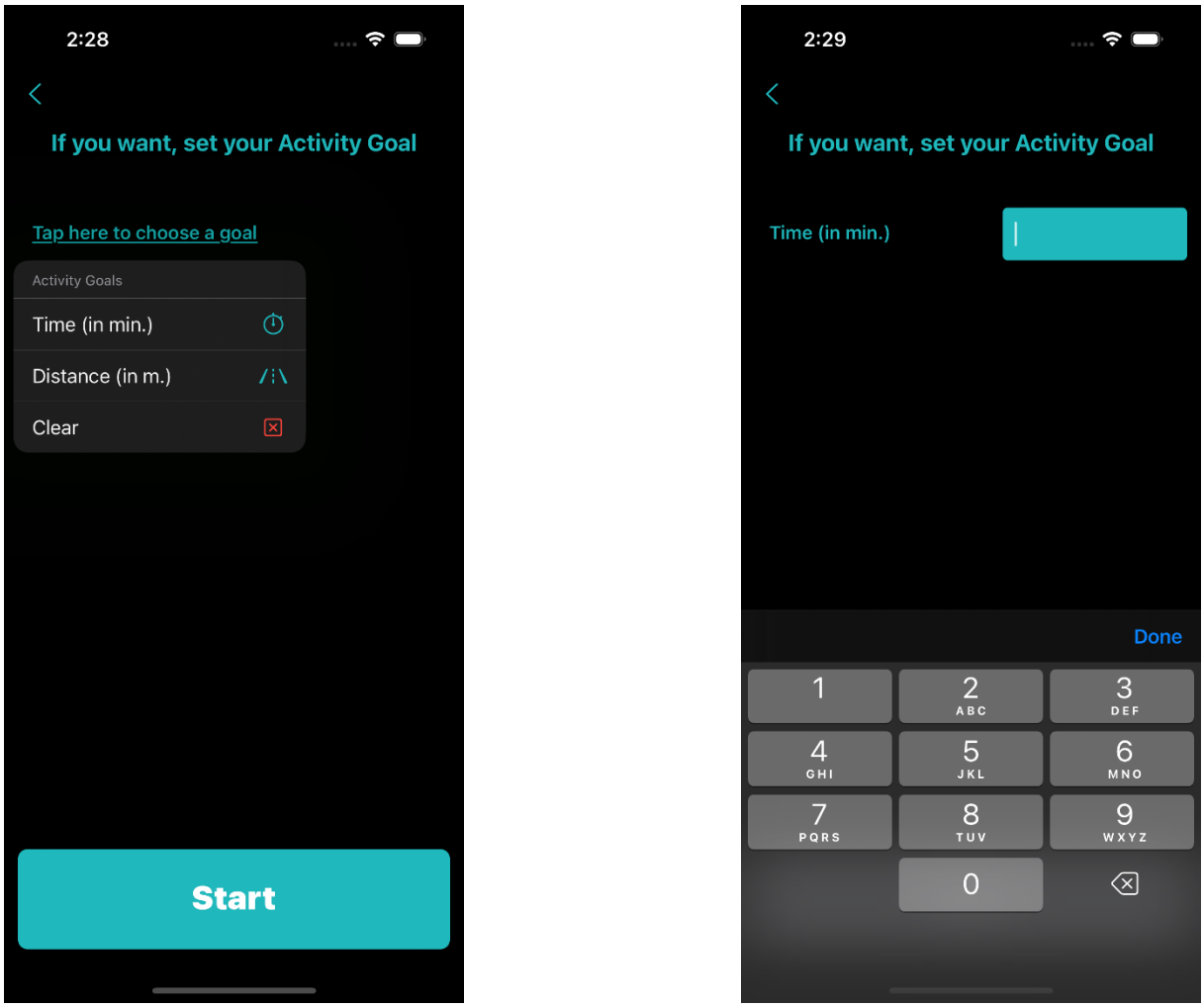
Σχήμα 4.19 Activity Screen

Πατώντας πάνω σε μια από τις δραστηριότητες ο χρήστης οδηγείται στην οθόνη Activity Parameter, η οποία περιγράφεται παρακάτω. Πατώντας το πρώτο από τα αριστερά κουμπί στο top navigation, η οθόνη activity αλλάζει και εισέρχεται σε edit mode. Εκεί ο χρήστης μπορεί να διαλέξει τις αγαπημένες του δραστηριότητες από το διαθέσιμο σύνολο. Οι αγαπημένες δραστηριότητες είναι αυτές που έχουν το εικονίδιο του άστρου στο πάνω δεξιά μέρος. Πατώντας το “ tick ” στο top navigation, οι αλλαγές στις αγαπημένες δραστηριότητες του χρήστη αποθηκεύονται στη βάση και προστίθενται οι αντίστοιχες προκλήσεις στην οθόνη Home, με βάση τις καινούριες αγαπημένες δραστηριότητες, Η οθόνη Activity σε edit mode φαίνεται στο Σχήμα 4.20.



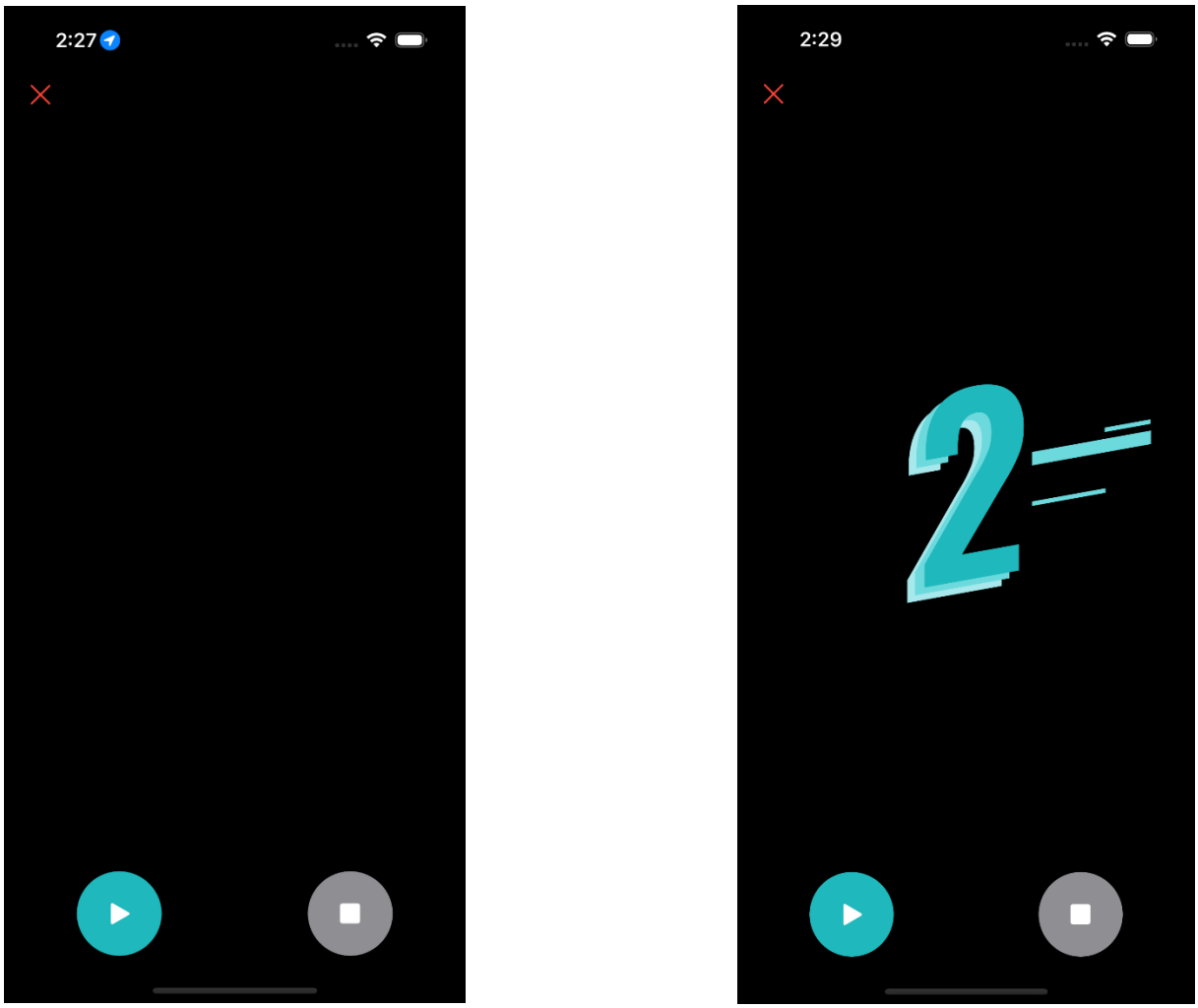
Σχήμα 4.20 Activity Screen – Edit Mode

Στην οθόνη Activity Parameter ο χρήστης προετοιμάζεται για την έναρξη της δραστηριότητας του, θέτοντας στόχους άθλησης αν αυτός το επιθυμεί.



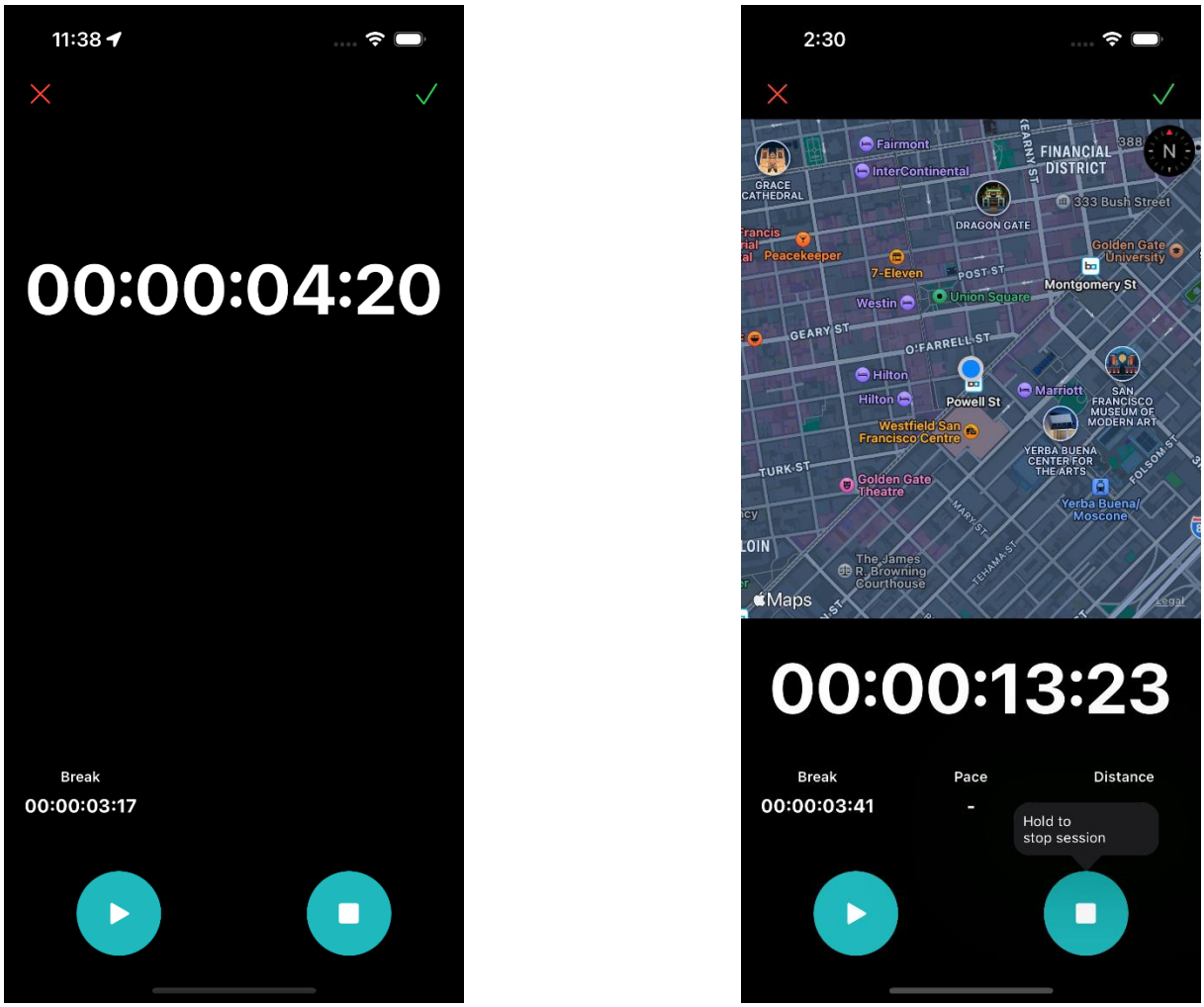
Σχήμα 4.21 Activity Parameter Screen

Όπως φαίνεται και στο παραπάνω σχήμα, ο χρήστης μπορεί είτε να πατήσει “start” και να μην θέσει κανένα στόχο άθλησης είτε να επιλέξει από ένα σύνολο διαθέσιμων στόχων ανάλογα με τη δραστηριότητα που θα εκτελέσει. Στη συνέχει οδηγείται στην οθόνη Activity Session, όπου και εκτελεί τη δραστηριότητα που επέλεξε. Για να ξεκινήσει την άθληση του ο χρήστης πρέπει να πατήσει το κουμπί “play” και να περιμένει την αντίστροφη μέτρηση. Σε περίπτωση που ο χρήστης μετανιώσει, μπορεί να πατήσει το κουμπί “ X ” του top navigation και να οδηγηθεί στην οθόνη activity, χωρίς να αποθηκευτεί κανένα δεδομένα για αυτό το session στη βάση, μιας και ουσιαστικά δεν εκκινήθηκε ποτέ.



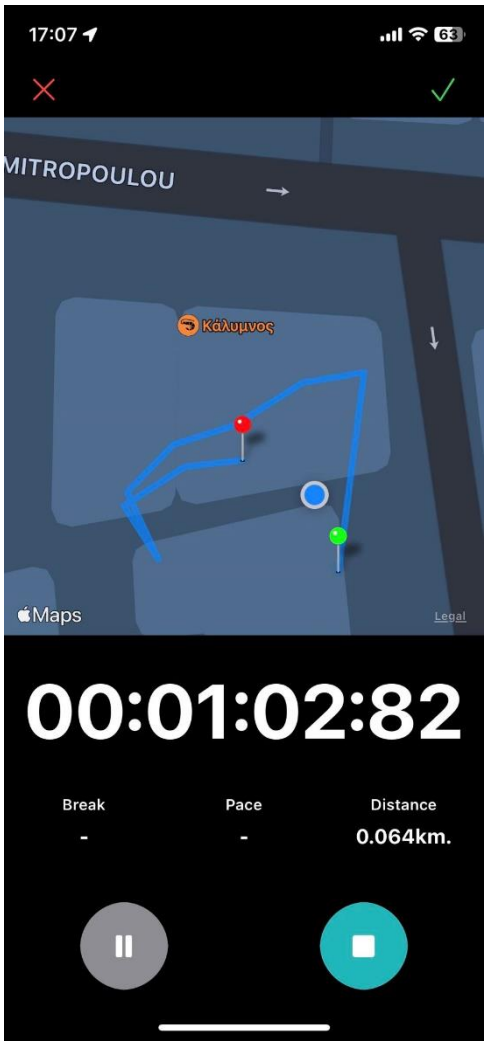
Σχήμα 4.22 Activity Session Screen – Εκκίνηση Δραστηριότητας

Η οθόνη Activity Session παραμετροποιείται ανάλογα με το είδος της δραστηριότητας. Αν η δραστηριότητα εμπεριέχει απόσταση (περπάτημα, τρέξιμο, ποδηλασία) τότε εμφανίζεται χάρτης καθώς και επιπλέον στοιχεία που καταγράφουν τη διαδρομή, το ρυθμό και την απόσταση που διανήθηκε. Σε οποιαδήποτε άλλη περίπτωση εμφανίζεται απλά ο χρόνος άσκησης και ο χρόνος διαλείμματος.



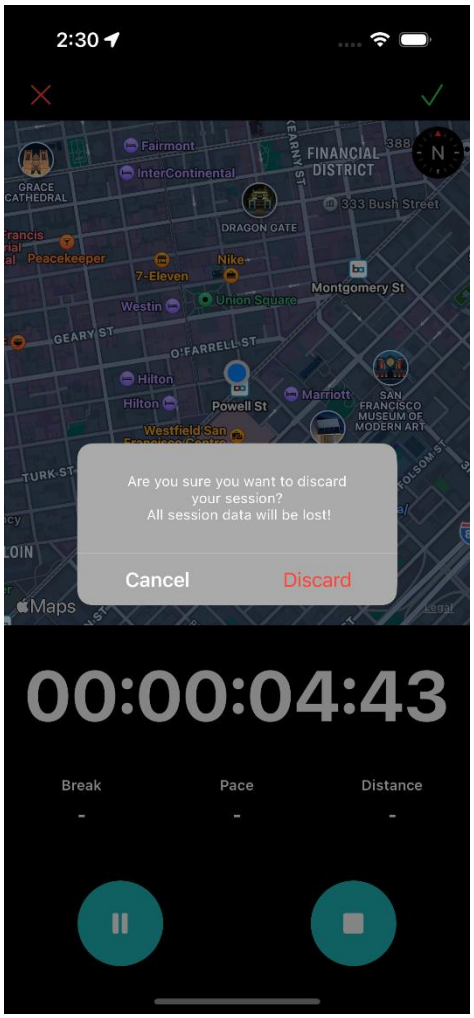
Σχήμα 4.23 Activity Session Screen – Τύποι Δραστηριότητας

Πατώντας το κουμπί “start – pause” πραγματοποιείται εναλλαγή από το χρονόμετρο διαλείμματος στο χρονόμετρο άσκησης και αντίστροφα. Στο παραπάνω σχήμα αν και δεν είναι εμφανές, βρισκόμαστε σε χρόνο διαλείμματος. Ο χρήστης στην περίπτωση του χάρτη μπορεί να βλέπει τον εαυτό του να μετακινείται εάν δώσει την άδεια στην εφαρμογή να διαχειρίζεται δεδομένα τοποθεσίας. Πατώντας παρατεταμένα το κουμπί “stop” ο χρήστης ολοκληρώνει την άσκηση του και παραμένει στην ίδια οθόνη όπου μπορεί να δει αναλυτικά τη διαδρομή που ακολούθησε.



Σχήμα 4.24 Activity Session Screen – Ολοκλήρωση Session

Έπειτα πατώντας το “tick” στο top navigation αποθηκεύει τα δεδομένα άθλησης στη βάση και ολοκληρώνει το session. Οποιαδήποτε στιγμή κατά τη διάρκεια της άθλησης, αν ο χρήστης επιθυμεί να μην ολοκληρώσει το session και να μην αποθηκευτούν τα δεδομένα του μπορεί να πατήσει το κουμπί “ X ” στο top navigation. Για να αποφύγουμε τα λάθος πατήματα, ο χρήστης ερωτάται αν είναι σίγουρος ότι δεν θέλει να ολοκληρώσει την άθληση του. Αν απαντήσει θετικά τότε το session ακυρώνεται, διαφορετικά το session συνεχίζεται κανονικά μέχρι τη μελλοντική ολκλήρωση και αποθήκευση ή ακύρωση του.



Σχήμα 4.25 Activity Session Screen – Ακύρωση Δραστηριότητας

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντική Εργασία

5.1 Συμπεράσματα

Η εφαρμογή Thesis Fitness App στην τωρινή μορφή της αποτελεί ένα ολοκληρωμένο σύστημα διαχείρισης των αθλητικών δραστηριοτήτων αλλά και των διατροφικής αξίας που προσλαμβάνει ένας χρήστης καθημερινά από τις τροφές που καταναλώνει. Με τη χρήση της εφαρμογής παρέχεται στο χρήστη ένα βοήθημα στην καθημερινή του δραστηριότητα και συλλέγονται μια πληθώρα δεδομένων τα οποία ο ίδιος στη συνέχει μπορεί να χρησιμοποιήσει για να ελέγξει την πρόοδο του στους τομείς που προαναφέρθηκαν. Επίσης, λόγω της επιλογής της αρχιτεκτονικής MVVM και του κομφορμισμού σε βέλτιστες πρακτικές σχεδίασης και υλοποίησης, ο κώδικας της εφαρμογής βρίσκεται σε ένα στάδιο από το οποίο μπορεί με ευκολία να επεκταθεί και να επαναχρησιμοποιηθεί, βελτιώνοντας με αυτό τον τρόπο τις είδη υπάρχουσες λειτουργίες και προσθέτοντας καινούριες.

Συμπερασματικά, το έργο αυτό αποτέλεσε μια προσωπική πρόκληση η οποία στέφθηκε με επιτυχία. Μέσα από την ανάπτυξη του Thesis Fitness App κατάφερα να κατανοήσω σε μεγαλύτερο βαθμό τις προκλήσεις που έχει η δημιουργία ενός τέτοιου έργου, κυρίως σε τομείς οι οποίοι δεν συμπεριλαμβάνονται στην καθημερινή μου επαγγελματική δραστηριότητα και στους οποίους συνεπώς δεν διαθέτω μεγάλη εμπειρία. Συγκεκριμένα αναφέρομαι στο κομμάτι του σχεδιασμού, του καθορισμού των προδιαγραφών και των απαιτήσεων, του διαμοιρασμού του χρόνου και του καθορισμού διοριών. Όλα τα παραπάνω αποτελούν ένα σημαντικό μάθημα το οποίο θα παίξει καθοριστικό ρόλο, τόσο στο μέλλον της συγκεκριμένης εφαρμογής όσο και στην ανάπτυξη λογισμικού που αφορά άλλα project.

5.2 Μελλοντική Εργασία και Βελτιστοποίηση

Παρόλο που η εφαρμογή στην παρούσα της μορφή αποτελεί ένα ολοκληρωμένο, πλήρως λειτουργικό σύστημα, το εύρος των πιθανών λειτουργιών που θα μπορούσαν να υλοποιηθούν και αφορούν αθλητική δραστηριότητα είναι πολύ μεγάλο. Το γεγονός αυτό δίνει αρκετό περιθώριο στην υπάρχουσα υλοποίηση να εντάξει επιπρόσθετες λειτουργίες που θα επαυξήσουν την προστιθέμενη αξία και θα ανεβάσουν την ποιότητα της εφαρμογής.

Συγκεκριμένα, στον αρχικό σχεδιασμό της εφαρμογής υπήρχε σχέδιο για υλοποίηση διαχειριστικού κομματιού που αφορά την άσκηση σε γυμναστήριο. Συγκεκριμένα ο χρήστης θα μπορούσε να επιλέξει ασκήσεις από ένα διευρυμένο ασκησιολόγιο και να δημιουργήσει τα δικά του workout με ένα σύνολο παραμέτρων (πλήθος ασκήσεων, πλήθος σετ και επαναλήψεων, χρόνος διαλείμματος, χρόνος άσκησης κ.α.). Στη συνέχεια, πέρα από την αποθήκευση και την κατ' απαίτηση εκτέλεση αυτών των workout, ο χρήστης θα μπορούσε να κοινοποιήσει τα workout αυτά σε άλλους χρήστες, δημιουργώντας έτσι μια κοινότητα γυμναστήριού. Συμπληρωματικά, με το πέρας κάθε workout τα δεδομένα του θα αποθηκεύονται στη βάση δεδομένων, έτσι ώστε ο χρήστης να μπορεί να δει το ιστορικό του και λάβει πληροφορίες σχετικά με την πρόοδο του, όπως ακριβώς μπορεί και στην υπάρχουσα υλοποίηση για τις κατηγορίες της διατροφής και της αθλητικής δραστηριότητας. Με την προσθήκη της κατηγορίας του γυμναστηρίου η εφαρμογή θα αυξήσει κατά πολύ το κοινό στο οποίο απευθύνεται και η δυνατότητα διαμοιρασμού των workout (share) μπορεί να λειτουργήσει και ως μέσο διαφήμισης αυξάνοντας περαιτέρω τους χρήστες της.

Πέρα από την κατηγορία του γυμναστηρίου, υπάρχουν ιδέες και για επιπρόσθετες υλοποιήσεις που σκοπό θα έχουν να βελτιώσουν την ήδη υπάρχουσα λειτουργικότητα χρησιμοποιώντας καινούριες και μοντέρνες τεχνολογίες. Συγκεκριμένα, η προσθήκη βασικών φωνητικών εντολών (παύση, εκκίνηση,

τερματισμός) στο κομμάτι των αθλητικών δραστηριοτήτων και του γυμναστήριού θα κάνουν την εφαρμογή πιο εύχρηστη, μιας και ελαχιστοποιούν τη διαδικασία αλληλεπίδρασης με χρήση gestures διευκολύνοντας έτσι τη διαδικασία άθλησης του χρήστη. Άλλη μια quality of life υλοποίηση που αφορά το κομμάτι της διατροφής είναι το σκανάρισμα των διατροφικών πινάκων από προϊόντα που ο χρήστης καταναλώνει. Με τη χρήση της κάμερας και της βιβλιοθήκης Vision της Apple, ο χρήστης θα μπορεί να σκανάρει τα τρόφιμα τα οποία καταναλώνει και προσθέτοντας την ποσότητα κατανάλωσης η εφαρμογή θα υπολογίζει τα μακροθρεπτικά συστατικά και θα τα προσθέτει στο διατροφικό ιστορικό του.

Τέλος, μετά τις παραπάνω λειτουργίες προγραμματίζεται η υλοποίηση αντίστοιχης εφαρμογής σε wearable (apple watch, WatchOS) οι οποία θα συνδέεται με το Thesis Fitness App στο iOS, αυξάνοντας την προστιθέμενη αξία και προσφέροντας επιπλέον ευκολία και αμεσότητα στη χρήση και των δύο πλέον εφαρμογών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Yongming Liu, “The Impact of Mobile Apps on Our Lives”, *Journal Computers in Human Behavior* , 2017.

[2] Daniel C. Denison, “The Rise of Mobile Apps and Their Impact on Society”, *Technological Forecasting and Social Change,* 2018.

[3] Sarah J. Wildman, “The Impact of Fitness Apps on Physical Activity and Health”, *Health Psychology,* 2019.

[4] Michelle M. Otto, “The Role of Mobile Apps in Promoting Physical Activity and Healthy Eating”, *Preventive Medicine,* 2020.

[5] MyFitnessPal Application [Online]. Available: https://apps.apple.com/us/app/myfitnesspal-calorie-counter/id341232718

[6] Sports Tracker Application [Online]. Available: https://apps.apple.com/us/app/sports-tracker-for-all-sports/id426684873

[7] Apple Human Interface Guidelines – SF Symbol Animations [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/sf-symbols#Animations

[8] Implementing Dark Mode on iOS [Online]. Available: https://developer.apple.com/videos/play/wwdc2019/214

[9] Design app experiences with charts [Online]. Available: https://developer.apple.com/videos/play/wwdc2022/110342

[10] Design an effective chart [Online]. Available: https://developer.apple.com/videos/play/wwdc2022/110340

[11] Swift Charts [Online]. Available: https://developer.apple.com/videos/play/wwdc2022/10136

[12] Swift Documentation [Online]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/

[13] Appel Developer Documentation – Swift [Online]. Available: https://developer.apple.com/swift/

[14] IBM Documentation – HTTP Requests [Online]. Available: https://www.ibm.com/docs/en/cics-ts/5.3?topic=protocol-http-requests

[15] JSON Documentation [Online]. Available: https://www.json.org/json-en.html

[16] IBM Documentation – What is external authentication [Online]. Available: https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=authorization-external-authentication

[17] Google Identity – Get Started on Google Sign-in for iOS and macOS [Online]. Available: https://developers.google.com/identity/sign-in/ios/start-integrating

[18] U.S Department of Agriculture – FoodData Central A.P.I [Online]. Available: https://fdc.nal.usda.gov/api-guide.html

[19] SwaggerHub – FoodData Central A.P.I [Online]. Available: https://app.swaggerhub.com/apis/fdcnal/food-data_central_api/1.0.1

[20] Apple Developer Documentation – Xcode [Online]. Available: https://developer.apple.com/xcode/

[21] Apple Developer Documentation – SwiftUI Overview [Online]. Available: https://developer.apple.com/xcode/swiftui/

[22] Apple Developer Documentation – UIKit Framework [Online]. Available: https://developer.apple.com/documentation/uikit

[23] Git [Online]. Available: https://git-scm.com

[24] Git Documentation – Reference Manual [Online]. Available: https://git-scm.com/docs

[25] GitHub [Online]. Available: https://github.com

[26] Robert Martin, *Clean Architecture - A Craftsman's Guide to Software Structure and Design*, PEARSON, 2017

[27] Swift Documentation – Enumerations [Online]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/enumerations/

[28] Swift Documentation – Protocols [Online]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/protocols/

[29] CocoaPods – Getting Started [Online]. Available: https://guides.cocoapods.org/using/getting-started.html

[30] GitHub – Alamofire [Online]. Available: https://github.com/Alamofire/Alamofire

[31] GitHub – ObjectMapper [Online]. Available: https://github.com/tristanhimmelman/ObjectMapper

[32] GitHub – SwiftyJson [Online]. Available: https://github.com/SwiftyJSON/SwiftyJSON

[33] GitHub – Kingfisher [Online]. Available: https://github.com/onevcat/Kingfisher

[34] GitHub – SwiftyUserDefaults [Online]. Available: https://github.com/sunshinejr/SwiftyUserDefaults

[35] GitHub – Lottie-ios [Online]. Available: https://github.com/airbnb/lottie-ios

[36] GitHub – Floaty [Online]. Available: https://github.com/kciter/Floaty

[37] GitHub – IQKeyboardManager [Online]. Available: https://github.com/Kilograpp/IQkeyboardManager

[38] GitHub – SVProgressHUD [Online]. Available: https://github.com/SVProgressHUD/SVProgressHUD

[39] GitHub – GoogleSigIn-iOS [Online]. Available: https://github.com/google/GoogleSignIn-iOS

[40] GitHub – Firebase iOS sdk [Online]. Available: https://github.com/firebase/firebase-ios-sdk

[41] GitHub – Charts [Online]. Available: https://github.com/danielgindi/Charts

[42] GitHub – SwiftLint [Online]. Available: https://github.com/realm/SwiftLint