ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση εφαρμογής για κινητό τηλέφωνο, σχετικά με τις διατροφικές συνήθειες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΙΚΗΤΑΚΗ ΒΑΣΙΛΕΙΟΥ

Επιβλέπουσα: Μπίμπη Σταματία

Επίκουρη Καθηγήτρια

ΚΟΖΑΝΗ/ΙΟΥΛΙΟΣ/2023

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL
& COMPUTER ENGINEERING
ece.uowm.gr

Implementation of a Mobile App about the eating habits

THESIS

by
NIKITAKIS VASILEIOS

SUPERVISOR: Bibi Stamatia
Assistant Professor

KOZANI/JULY/2023

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δήλωση μη Λογοκλοπής και Ανάληψης Προσωπικής Ευθύνης

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "**Υλοποίηση εφαρμογής για κινητό τηλέφωνο, σχετικά με τις διατροφικές συνήθειες**" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κα. **Μπίμπη Σταματία** αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορέςκαι μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Υπογραφή Φοιτητή: __ _____

Περίληψη

Η πτυχιακή εργασία με τίτλο "**Υλοποίηση εφαρμογής για κινητό τηλέφωνο, σχετικά με τις διατροφικές συνήθειες** " εστιάζει στη δημιουργία μιας εφαρμογής για smartphones με χρήση σύγχρονων τεχνολογιών και αρχιτεκτονικών (React Native, React, JavaScript και Firebase). Η εφαρμογή αυτή παρέχει ένα εκπαιδευτικό και ενημερωτικό περιβάλλον για την υγιεινή διατροφή και την ευεξία. Η εφαρμογή περιλαμβάνει ένα quiz με ερωτήσεις σχετικά με τις διατροφικές συνήθειες, δίνοντας στους χρήστες την ευκαιρία να εκπαιδευτούν και να επιβεβαιώσουν τις γνώσεις τους σε αυτό τον τομέα. Το quiz παρέχει αποτελέσματα και αναλύσεις για την απόδοση του χρήστη, ενθαρρύνοντας τη συνεχή βελτίωση και ευαισθητοποίηση για τη διατροφή. Επιπλέον, η εφαρμογή παρέχει λειτουργίες παρακολούθησης της κατανάλωσης θερμίδων και υδάτων, καθώς και δυνατότητα σάρωσης γραμμωτών κωδικών προϊόντων για εύκολη παρακολούθηση της διατροφής του χρήστη. Οι εγγεγραμμένοι χρήστες έχουν τη δυνατότητα να δημιουργήσουν ένα προσωπικό προφίλ με στόχους για την υγεία τους και να παρακολουθήσουν την πρόοδό τους. Η απόφαση για τη χρήση του Expo σαν πλατφόρμας ανάπτυξης επιτρέπει την απλούστευση της διαδικασίας ανάπτυξης και τη δυνατότητα παραγωγής εφαρμογής για διάφορες πλατφόρμες (iOS, Android) με κοινό κώδικα. Συνολικά, αυτή η πτυχιακή εργασία συνδυάζει την εκπόνηση ενός πρακτικού μέρους, που απαιτεί χρονοβόρα ανάπτυξη και αντιμετωπίζει τεχνικές προκλήσεις, με ένα θεωρητικό μέρος που αναλύει τις τεχνολογίες που χρησιμοποιήθηκαν και τονίζει τη σημασία της υγιεινής διατροφής. Μέσω αυτής της εργασίας, αναδεικνύεται η συνεισφορά των νέων τεχνολογιών στον τομέα της υγείας και της ευεξίας, προσφέροντας ένα εργαλείο που προωθεί την ενημέρωση και την υιοθέτηση υγιεινών διατροφικών συνηθειών.

Λέξεις Κλειδιά

React, React Native, Firebase, JavaScript, Expo

Abstract

The bachelor's thesis titled "**Implementation of a Mobile App about the eating habits** " focuses on the development of a mobile application using React Native, React, JavaScript, and Firebase technologies. This application provides an educational and informative environment for healthy eating and well-being. The application features a quiz section with questions related to dietary habits, allowing users to educate themselves and test their knowledge in this field. The quiz provides results and analysis of the user's performance, encouraging continuous improvement and awareness of nutrition. Additionally, the application offers functions for tracking calorie and water intake, as well as a barcode scanner for an easy tracking of consumed products into the calorie counter. Registered users have the ability to create a personal profile with health goals and monitor their progress. The decision to utilize Expo as the development platform simplifies the development process and enables the production of the application for multiple platforms (iOS, Android) using shared code. Overall, this bachelor's thesis combines a practical part, which involves time-consuming development and tackles technical challenges, with a theoretical part that analyzes the technologies used and highlights the importance of healthy eating. Through this work, the contribution of new technologies to the field of health and well-being is emphasized, providing a tool that promotes education and adoption of healthy dietary habits.

Keywords

React, React Native, Firebase, JavaScript, Expo

Ευχαριστίες

Ευχαριστώ την οικογένειά μου που με στήριξε καθ' όλη την διάρκεια της φοιτητικής μου πορείας, τους φίλους μου που στις δύσκολες στιγμές ήταν εκεί να με ανεβάζουν και έκαναν όλο αυτό το ταξίδι πιο ευχάριστο.

Τέλος, ευχαριστώ την επιβλέπουσα καθηγήτρια κα. Μπίμπη Σταματία που με εμπιστεύθηκε να φέρω εις πέρας αυτό το εγχείρημα.

ΚΟΖΑΝΗ/ΙΟΥΛΙΟΣ/2023

ΑΥΤΗ Η ΣΕΛΙΔΑ ΕΙΝΑΙ ΣΚΟΠΙΜΑ ΛΕΥΚΗ

Περιεχόμενα

Κατάλογος Εικόνων

Πρόλογος

Η παρούσα διπλωματική εργασία με τίτλο «**Υλοποίηση εφαρμογής για κινητό τηλέφωνο, σχετικά με τις διατροφικές συνήθειες»** δημιουργήθηκε από τις αρχές Φεβρουαρίου 2023 ως τον Ιούλιο 2023 στην πόλη της Κοζάνης υπό την επίβλεψη της επίκουρης καθηγήτριας Μπίμπης Σταματία. Η εργασία αυτή αποτελεί μια μελέτη που επικεντρώνεται στη δημιουργία μιας εφαρμογής quiz με θέμα τις διατροφικές συνήθειες. Στόχος της εργασίας είναι η ανάπτυξη μιας πλατφόρμας που θα παρέχει στους χρήστες πληροφορίες σχετικά με τη σωστή διατροφή και τις υγιεινές συνήθειες, ενώ ταυτόχρονα θα προσφέρει εργαλεία για τη μέτρηση των θερμίδων και του νερού που καταναλώνουν.

Η εφαρμογή υλοποιείται με χρήση των τεχνολογιών React Native, Expo, React και JavaScript, με την υποστήριξη της πλατφόρμας Firebase για τη διαχείριση των δεδομένων. Επιπλέον, η εφαρμογή περιλαμβάνει έναν μετρητή θερμίδων και νερού, καθώς και έναν σαρωτή barcode, που επιτρέπει στον χρήστη να καταγράφει την κατανάλωση των προϊόντων και να λαμβάνει αντίστοιχες πληροφορίες. Μέσω του quiz, οι χρήστες θα μπορούν να διευρύνουν τις γνώσεις τους σχετικά με τις διατροφικές συνήθειες και να ενημερωθούν για σημαντικές πτυχές της υγιεινής διατροφής.

Η εργασία αποτελείται από δύο κύρια στάδια. Αρχικά, περιλαμβάνει μια αναλυτική έρευνα σχετικά με τις διατροφικές συνήθειες και τις σχετικές πληροφορίες που πρέπει να παρέχονται στους χρήστες. Στη συνέχεια, παρουσιάζεται η ανάπτυξη της εφαρμογής, περιγράφοντας τις τεχνολογίες ανάπτυξης, την υλοποίηση, τη δομή και τη λειτουργικότητά της. Στο τέλος της εργασίας, γίνεται μια συνολική αξιολόγηση του έργου, αναλύονται τα αποτελέσματα που προέκυψαν και προτείνονται μελλοντικές επεκτάσεις και βελτιώσεις για την εφαρμογή. Με την παρούσα πτυχιακή εργασία, επιδιώκουμε να παρέχουμε ένα εργαλείο που θα ενθαρρύνει τους χρήστες να αναπτύξουν υγιείς διατροφικές συνήθειες και να ενημερωθούν για τη σημασία της διατροφής για την υγεία και την ευεξία τους.

Κεφάλαιο 1: Εισαγωγή

Η React Native και το Expo αποτελούν δύο αξιόλογες τεχνολογίες ανάπτυξης για τη δημιουργία διασκεδαστικών και πρωτοποριακών κινητών εφαρμογών. Με την React Native, μπορούμε να αναπτύξουμε εφαρμογές για διάφορες πλατφόρμες, όπως το iOS και το Android, χρησιμοποιώντας μια κοινή γλώσσα προγραμματισμού, τη JavaScript. Αυτό μας επιτρέπει να δημιουργήσουμε εφαρμογές με υψηλή απόδοση και ευελιξία, επαναχρησιμοποιώντας κώδικα και μειώνοντας τον χρόνο ανάπτυξης. Το Expo είναι μια πλατφόρμα ανάπτυξης που υποστηρίζει την ανάπτυξη React Native εφαρμογών με μεγάλη ευκολία. Παρέχει ένα πλήθος από έτοιμα εργαλεία, βιβλιοθήκες και υπηρεσίες που επιτρέπουν στους προγραμματιστές να αναπτύξουν εφαρμογές γρήγορα και αποτελεσματικά. Η Expo αναλαμβάνει την παροχή υποδομής για τη διανομή των εφαρμογών, καθιστώντας την διαδικασία ανάπτυξης και την αναβάθμιση των εφαρμογών πιο απλές και ευέλικτες. Η χρήση της React Native και του Expo προσφέρει πολλά πλεονεκτήματα στους προγραμματιστές. Καταρχάς, μειώνει το κόστος ανάπτυξης, καθώς ο κώδικας γράφεται μία φορά και μπορεί να χρησιμοποιηθεί για τις διάφορες πλατφόρμες. Επιπλέον, η ανάπτυξη γίνεται γρήγορα και αποτελεσματικά, καθώς το Expo παρέχει ένα πλούσιο σύνολο εργαλείων και βιβλιοθηκών που επιτρέπουν την εύκολη πρόσβαση σε δημοφιλείς λειτουργίες. Το Expo επιτρέπει επίσης την ενσωμάτωση του Firebase για την διαχείριση χρηστών και την αποθήκευση δεδομένων. Με τη χρήση του Firebase, μπορούμε να δημιουργήσουμε λογαριασμούς χρηστών, να διατηρούμε προσωπικά στοιχεία και να αποθηκεύουμε/επεξεργαζόμαστε δεδομένα. Συνοψίζοντας, η συνδυασμένη χρήση της React Native και του Expo προσφέρει μία ισχυρή και αποδοτική πλατφόρμα ανάπτυξης για κινητές εφαρμογές. Οι προγραμματιστές μπορούν να απολαύσουν τα πλεονεκτήματα της επαναχρησιμοποίησης κώδικα, της γρήγορης ανάπτυξης και της ευελιξίας, ενώ το Expo παρέχει ένα ολοκληρωμένο περιβάλλον ανάπτυξης για τη διανομή και την αναβάθμιση των εφαρμογών. Με αυτήν την ισχυρή συνεργασία, η ανάπτυξη και η παρουσίαση καινοτόμων κινητών εφαρμογών γίνεται πιο αποτελεσματική και προσβάσιμη για όλους.

1.1 Αντικείμενο της Διπλωματικής

Η πτυχιακή εργασία επικεντρώνεται στη δημιουργία μιας εφαρμογής παιχνιδιού quiz με πολλές ακόμα λειτουργίες. Στο τέλος κάθε quiz, παρέχονται τα αποτελέσματα του κάθε χρήστη με βαθμολογία έως και πέντε, οι σωστές απαντήσεις του παιχνιδιού που μόλις τελείωσε, η επιλογή να ξαναπαίξει ή και να εξερευνήσει μια από τις επτά κατηγορίες που παρέχονται. Μια από τις λειτουργίες του έργου είναι η δυνατότητα εγγραφής νέων χρηστών μέσω της Firebase ή σύνδεσης για τους εγγεγραμμένους χρήστες. Στην οθόνη του προφίλ, υπάρχουν διάφορες επιλογές, όπως το water/calorie counter, όπου ο χρήστης μπορεί να καταγράψει την κατανάλωση νερού/θερμίδων που καταναλώνει, την καταγραφή στόχων που θέτει ο χρήστης και το barcode scanner για να σαρώνει προϊόντα που καταναλώνει και να τα προσθέτει στον θερμιδομετρητή.

1.2 Σκοπός και στόχος της εργασίας

Ο σκοπός της πτυχιακής εργασία μου ήταν να αναπτύξω μια ευκολόχρηστη εφαρμογή που προσφέρει διασκέδαση εκπαιδεύοντας τον χρήστη με ερωτήσεις, αλλά ταυτόχρονα να παρέχει επιπλέον λειτουργίες που βοηθούν τον χρήστη στην παρακολούθηση της κατανάλωσης του νερού και των θερμίδων του, καθώς και στην επίτευξη των προσωπικών του στόχων. Επιπλέον, η δυνατότητα σάρωσης των γραμμωτών κωδικών των προϊόντων μέσω του barcode scanner παρέχει μια εύκολη μέθοδο για την προσθήκη των καταναλωθέντων προϊόντων στον προσωπικό θερμιδομετρητή. Αυτές οι λειτουργίες δημιουργήθηκαν προκειμένου να προσθέσουν αξία στην εμπειρία του χρήστη και του παρέχουν μια ολοκληρωμένη εφαρμογή που μπορεί να του φανεί χρήσιμη στην παρακολούθηση της διατροφής και της υγείας του.

1.3 Οργάνωση του Τόμου

Η παρακάτω μελέτη διαμορφώνεται ως εξής:

Η Ενότητα 2 στοχεύει την ανάλυση και επεξήγηση των περιπτώσεων χρήσης, των απαιτήσεων και του διαγράμματος βάσης δεδομένων.

Στην Ενότητα 3 παρουσιάζεται τις τεχνολογίες ανάπτυξης, τα περιβάλλοντα και λοιπές τεχνολογίες που χρησιμοποιήθηκαν.

Η Ενότητα 4 περιέχει τον τρόπο υλοποίησης και τα βήματα που ακολουθήθηκαν για την υλοποίηση αυτού του έργου.

Η Ενότητα 5 περιλαμβάνει τα αποτελέσματα της εφαρμογής.

Κεφάλαιο 2: Ανάλυση και σχεδίαση εφαρμογής

Σε αυτό το κεφάλαιο θα αναλύσουμε σε βάθος τις περιπτώσεις χρήσης, τις απαιτήσεις και το διάγραμμα βάσης δεδομένων καθώς επίσης και τους τρόπους με τους οποίους θα τα προσεγγίσουμε.

2.1 Ανάλυση Μελέτης Περίπτωσης

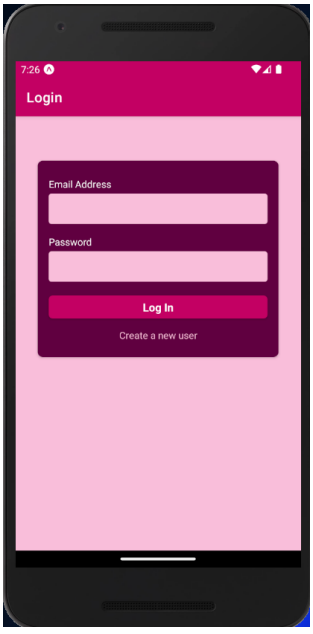
Στο πλαίσιο αυτής της διπλωματικής χρησιμοποιήθηκαν κάποια από τα πιο δημοφιλή και σύγχρονα framework και βιβλιοθήκες σε γλώσσες που κυριαρχούν περισσότερο στον κόσμο της τεχνολογίας. Δεν υπήρχαν κάποια συγκεκριμένα κριτήρια για την επιλογή αυτών των τεχνολογιών. Σκοπός είναι να μελετηθεί η πορεία κάθε έργου από την πρώτη φορά κυκλοφορίας του μέχρι σήμερα.

2.1.1 Περιπτώσεις Χρήσης

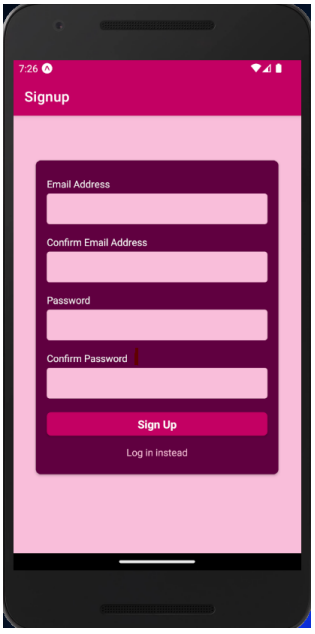
Η εφαρμογή που αναπτύσσεται στα πλαίσια αυτής της πτυχιακής εργασίας προσφέρει στους χρήστες τη δυνατότητα να απολαύσουν μια ποικιλία λειτουργιών που σχετίζονται με την υγεία τους. Αυτό σημαίνει ότι οι χρήστες μπορούν να εμπλουτίσουν τις γνώσεις τους σχετικά με τις διατροφικές συνήθειες, αντιμετωπίζοντας την εφαρμογή σαν ένα μέσω ψυχαγωγίας, σαν ένα πρόγραμμα καταγραφής στόχων καθώς και σαν ένα μικρό φορητό μετρητή στον οποίο κάθε χρήστης θα μπορεί να καταγράψει ότι χρειάζεται.

Οι περιπτώσεις χρήσης που μπορούν να αναλυθούν για αυτήν την πτυχιακή εργασία είναι οι εξής:

- Εγγραφή/Σύνδεση:
    - Ο χρήστης μπορεί να δημιουργήσει έναν λογαριασμό και να εγγραφεί στην εφαρμογή.
    - Αν ο χρήστης έχει δημιουργήσει λογαριασμό, μπορεί να συνδεθεί στον λογαριασμό του χρησιμοποιώντας τα διαπιστευτήριά του.



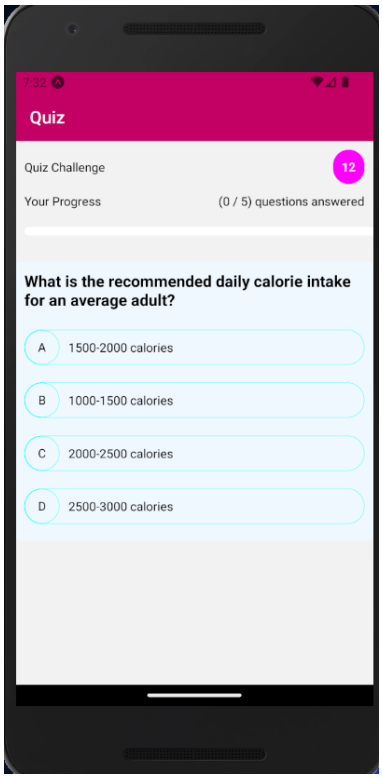
Εικόνα 0-Οθόνη LoginScreen



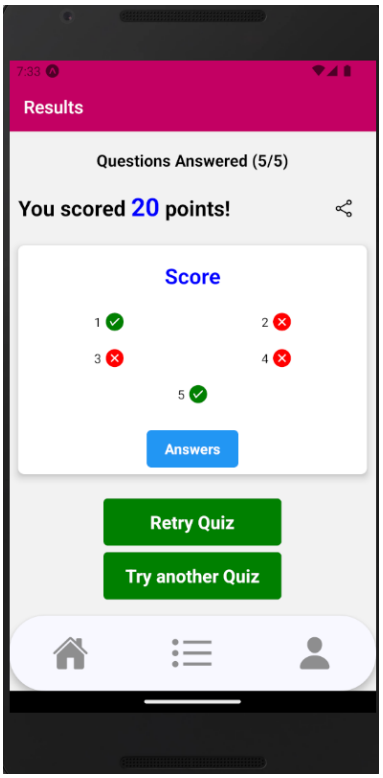
Εικόνα 1-Οθόνη SignupScreen

- Εκτέλεση ερωτηματολογίων και απαντήσεις στο τέλος:
    - Ο χρήστης μπορεί να εκτελέσει ερωτηματολόγια που αφορούν την υγεία και τις διατροφικές του συνήθειες.

- Οι ερωτήσεις περιλαμβάνουν θέματα όπως το λίπος, οι βιταμίνες, η χρησιμότητα του ύπνου κ.α.
- Ο χρήστης απαντά σε αυτές τις ερωτήσεις και καταχωρεί τις απαντήσεις του.
- Στο τέλος, ο χρήστης μπορεί να ενημερωθεί για τις σωστές απαντήσεις των συγκεκριμένων ερωτήσεων που έπαιξε, μπορεί να δει τους συνολικούς πόντους που πέτυχε, καθώς επίσης και να κοινοποιήσει τα αποτελέσματά του στους φίλους του ή σε οποιαδήποτε πλατφόρμα επιθυμεί.

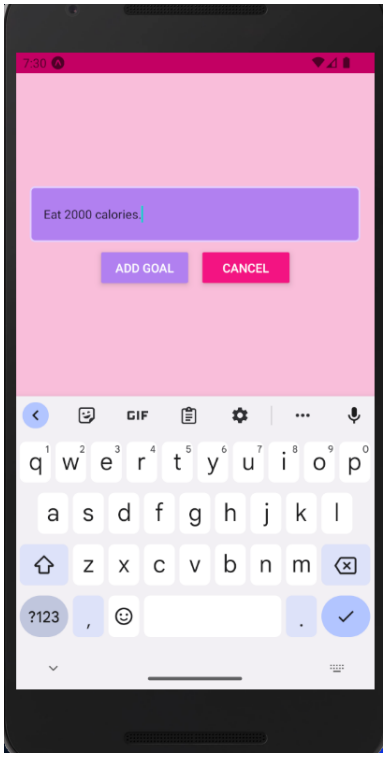


Εικόνα 2-Οθόνη παιχνιδιού quiz

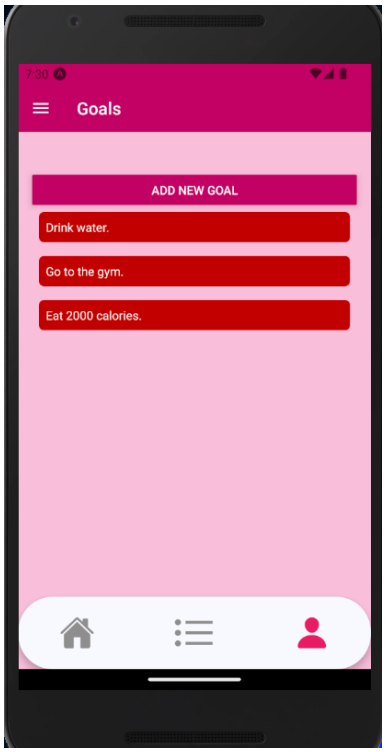


Εικόνα 3-Οθόνη Αποτελεσμάτων quiz

- Διαχείριση στόχων:
  - Ο χρήστης μπορεί να ορίσει προσωπικούς στόχους σχετικά με τη διατροφή, την άσκηση ή οτιδήποτε άλλο επιθυμεί. Οι στόχοι μπορούν να είναι π.χ. η κατανάλωση ενός συγκεκριμένου αριθμού θερμίδων ανά ημέρα ή η πραγματοποίηση μιας συγκεκριμένης δραστηριότητας.
  - Ο χρήστης μπορεί να παρακολουθεί την πρόοδό του προς την επίτευξη των στόχων του και να λαμβάνει ειδοποιήσεις για την επίτευξή τους.

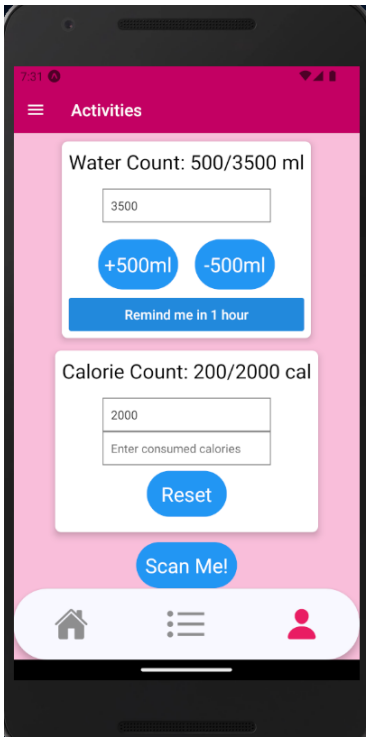


Εικόνα 4-Οθόνη καταγραφής στόχων



Εικόνα 5-Οθόνη προβολής στόχων

- Μέτρηση νερού και θερμίδων:
  - Ο χρήστης μπορεί να καταγράφει την κατανάλωση του νερού και των θερμίδων που λαμβάνει κατά τη διάρκεια της ημέρας.
  - Μπορεί να καταχωρεί τις ποσότητες νερού που πίνει και τα τρόφιμα που καταναλώνει.
  - Επιπλέον, υπάρχει ένα κουμπί ειδοποίησης που υπενθυμίζει στον χρήστη να καταγράφει την κατανάλωση νερού και θερμίδων.



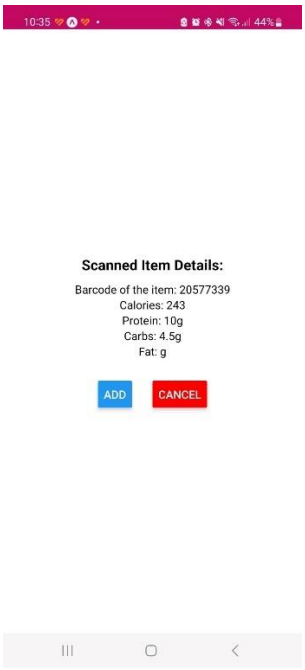
Εικόνα 6-Οθόνη προβολής μετρητών

- Σάρωση γραμμωτού κώδικα και ανάκτηση πληροφοριών διατροφής μέσω της Open Food Facts API:

  - Ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή για να σκανάρει τον γραμμωτό κώδικα ενός προϊόντος. Στη συνέχεια, η εφαρμογή αναζητά τις αντίστοιχες πληροφορίες για τις διατροφικές αξίες του προϊόντος στη βάση δεδομένων της Open Food Facts.

  - Ο χρήστης λαμβάνει πληροφορίες σχετικά με τις θερμίδες, την πρωτεΐνη, τους υδατάνθρακες και τα λιπαρά τα οποία περιέχει το τρόφιμο το οποίο επέλεξε να σκανάρει.

Εικόνα 7-Οθόνη ενεργοποίησης κάμερας



Εικόνα 8-Οθόνη προβολής διατροφικών στοιχείων

2.1.2 Απαιτήσεις

Προκειμένου οι χρήστες να μπορούν να έχουν πρόσβαση στην εφαρμογή και σε όλες τις λειτουργίες που προσφέρει, θα πρέπει να αναφερθούν οι παρακάτω απαιτήσεις:

- Συμβατότητα με smartphone: Η εφαρμογή πρέπει να είναι συμβατή με smartphones που λειτουργούν με τα λειτουργικά συστήματα Android και iOS. Αυτό θα επιτρέψει στους χρήστες με αυτές τις συσκευές να εγκαταστήσουν και να χρησιμοποιήσουν την εφαρμογή.
- Πρόσβαση στο internet για σκανάρισμα προϊόντων: Ο χρήστης πρέπει να έχει πρόσβαση στο internet για να μπορεί να σκανάρει τα προϊόντα και να εμφανίζονται τα στοιχεία της Open Food Facts. Αυτό είναι απαραίτητο για να ληφθούν οι απαραίτητες πληροφορίες για τα προϊόντα που σκανάρονται.
- Αποδοχή δικαιωμάτων: Ο χρήστης πρέπει να αποδεχθεί τις απαραίτητες άδειες ή δικαιώματα πρόσβασης που απαιτούνται από την εφαρμογή για να λειτουργήσει σωστά. Αυτά τα δικαιώματα μπορεί να περιλαμβάνουν την πρόσβαση στην κάμερα για το σκανάρισμα barcode.

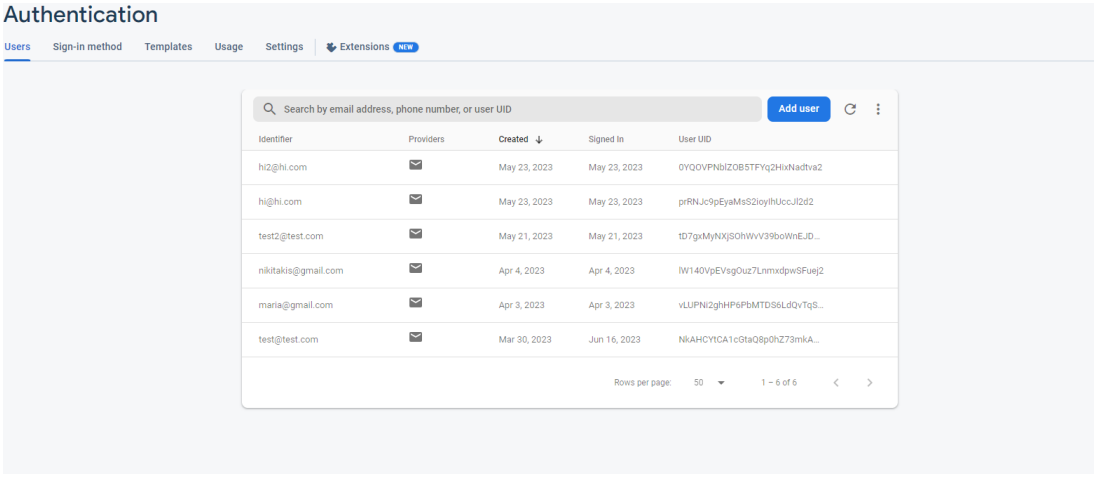
2.1.3 Διάγραμμα Βάσης Δεδομένων

Για την υλοποίηση της πτυχιακής εργασίας χρησιμοποιήθηκε η βάση δεδομένων Firebase για την αποθήκευση των δεδομένων. Η επιλογή της βάσης δεδομένων Firebase έγινε λόγω των ακόλουθων αιτιών. Αρχικά, η ευκολία χρήσης της, καθώς παρέχει μια ευέλικτη και εύχρηστη πλατφόρμα που διευκολύνει τη δημιουργία και τη διαχείριση της βάσης δεδομένων. Ακόμα, η Firebase παρέχει αυτόματο συγχρονισμό των δεδομένων ανάμεσα στην εφαρμογή και τη βάση δεδομένων, εξασφαλίζοντας ότι τα δεδομένα είναι πάντα ενημερωμένα και συνεπή. Τέλος, παρέχει αυξημένα μέτρα ασφαλείας για την προστασία των

δεδομένων των χρηστών, εξασφαλίζοντας ότι μόνο εξουσιοδοτημένα άτομα έχουν πρόσβαση σε αυτά. Με τη χρήση της Firebase ως βάση δεδομένων, η εφαρμογή μπορεί να αξιοποιήσει την αξιοπιστία, την ευελιξία και την απλότητα που παρέχει η πλατφόρμα για την αποθήκευση και τη διαχείριση των δεδομένων των χρηστών, των ερωτήσεων και των πόντων, καθώς και άλλα σχετικά δεδομένα.

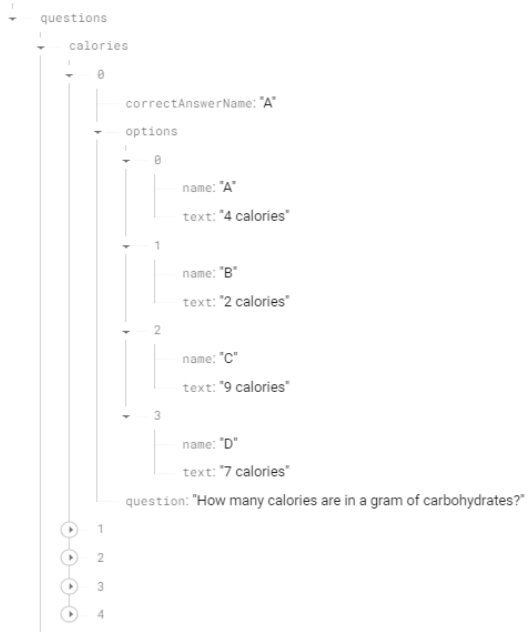
Συγκεκριμένα, η βάση δεδομένων Firebase περιλαμβάνει τα εξής στοιχεία:

- Λογαριασμοί χρηστών: Οι πληροφορίες των χρηστών, όπως τα ονόματα, τα email και οι κωδικοί πρόσβασης, αποθηκεύονται στη βάση δεδομένων Firebase. Μέσω αυτής της αποθήκευσης, οι χρήστες μπορούν να εγγραφούν και να συνδεθούν στην εφαρμογή.



Εικόνα 9-Περιεχόμενο αποθήκευσης λογαριασμών χρηστών 1

- Ερωτήσεις: Η βάση δεδομένων Firebase περιλαμβάνει τις ερωτήσεις που ήδη υπάρχουν στην εφαρμογή, καθώς και τις ερωτήσεις που δημιουργούν οι χρήστες. Αυτές οι ερωτήσεις αποθηκεύονται και ανακτώνται από τη βάση δεδομένων, επιτρέποντας την εμφάνισή τους στην εφαρμογή κατά την εκτέλεση των ερωτηματολογίων. Η αποθήκευση των ήδη υπαρχόντων γίνεται μέσω ενός αρχείου JSON που αποθηκεύεται στην Firebase.



Εικόνα 10-Περιεχόμενο αποθήκευσης ερωτήσεων 1

- Πόντοι χρηστών: Οι πόντοι κάθε χρήστη καταγράφονται και αποθηκεύονται στη βάση δεδομένων Firebase. Κάθε φορά που ένας χρήστης απαντά σε ένα ερωτηματολόγιο, οι πόντοι του ενημερώνονται ανάλογα.

Εικόνα 11-Περιεχόμενο αποθήκευσης πόντων χρηστών

Κεφάλαιο 3: Τεχνολογίες Ανάπτυξης

Σε αυτήν την ενότητα θα παρουσιάσω τις τεχνολογίες ανάπτυξης, τα περιβάλλοντα και λοιπές τεχνολογίες που είναι απαραίτητες για την υλοποίηση αυτής της εργασίας.

3.1 Γλώσσες προγραμματισμού

3.1.1 JavaScript

Η JavaScript (JS) είναι γλώσσα προγραμματισμού για Η/Υ. Η JavaScript αποτελεί μέρος της υλοποίησης των browsers, ώστε τα client-side scripts να μην μπορούν να επικοινωνήσουν με τον χρήστη, τα δεδομένα να αλλάζουν ασύγχρονα και το περιεχόμενο του εγγράφου που εμφανίζεται να αλλάζει δυναμικά. Η JavaScript είναι μια δυναμική prototype-based γλώσσα σεναρίων, με ασθενείς τύπους και συναρτήσεις ως αντικείμενα πρώτης τάξης. Η C έχει επηρεάσει τη σύνταξη της JavaScript. Η Java δανείζει πολλά ονόματα και συμβάσεις στην JavaScript, αλλά έχουν πολύ διαφορετική σημασιολογία και δεν σχετίζονται μεταξύ τους. Από τις γλώσσες προγραμματισμού Self και Scheme προέρχονται οι βασικές αρχές σχεδιασμού της JavaScript. Είναι multi-paradigm γλώσσα, υποστηρίζοντας αντικειμενοστραφές, συναρτησιακό και προστακτικό στυλ προγραμματισμού. Το JavaScript χρησιμοποιείται και σε web applications υπάρχουν παραδείγματα όπως τα PDF, οι εξειδικευμένοι browsers και τα desktop widgets. Οι καινούργιες εικονικές μηχανές και τα πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).[1]

3.1.2 React

Το React είναι μια ανοικτού κώδικα βιβλιοθήκη JavaScript για τη δημιουργία χρήσιμων και αποδοτικών διεπαφών χρήστη (UI). Αναπτύχθηκε από την Facebook και έχει γίνει ευρέως αποδεκτό εργαλείο στην κοινότητα των προγραμματιστών για την ανάπτυξη δυναμικών και αποκρίνοντων εφαρμογών web. Το React ακολουθεί την έννοια του component-based development, όπου η διεπαφή χρήστη αποτελείται από μικρές ανεξάρτητες μονάδες που ονομάζονται components. Αυτό το μοντέλο ανάπτυξης επιτρέπει την επαναχρησιμοποίηση, τη συντήρηση και την αποδοτική διαχείριση του κώδικα. Ο κεντρικός στόχος του React είναι η αποδοτικότητα. Χρησιμοποιεί έναν αλγόριθμο Virtual DOM που επιτρέπει στο React να εντοπίζει και να εφαρμόζει μόνο τις απαραίτητες αλλαγές στο UI, αντί να ανακατασκευάζει ολόκληρο το DOM σε κάθε ενημέρωση. Αυτή η προσέγγιση καθιστά το React γρήγορο και αποδοτικό σε σχέση με άλλες μεθόδους ανάπτυξης UI. Το React είναι επίσης ευέλικτο, καθώς μπορεί να συνδυαστεί με άλλες τεχνολογίες και βιβλιοθήκες. Μια από τις δημοφιλείς συνδυασμένες τεχνολογίες είναι το Redux, ένα state management tool που βοηθά στη διαχείριση της κατάστασης των εφαρμογών React. Συνολικά, το React έχει επαναπροσδιορίσει τον τρόπο με τον οποίο οι προγραμματιστές προσεγγίζουν τη δημιουργία UI στο web. Με την ισχυρή του κοινότητα και τη συνεχή εξέλιξή του, το React συνεχίζει να αποτελεί ένα από τα πιο δημοφιλή εργαλεία ανάπτυξης για τη δημιουργία σύγχρονων και αποδοτικών εφαρμογών web.[2]

3.1.3 JSX

Το JSX είναι μια σύνταξη που χρησιμοποιείται στη React και σε άλλα παρόμοια πλαίσια ανάπτυξης JavaScript. Αντιπροσωπεύει τη συνδυασμένη χρήση της σύνταξης JavaScript και HTML/XML για τη δημιουργία δομής και εμφάνισης των στοιχείων της διεπαφής χρήστη. Το JSX επιτρέπει στους προγραμματιστές να περιγράφουν τα στοιχεία της διεπαφής χρήστη ως ιεραρχική δομή, παρόμοια με τον τρόπο που γράφουμε HTML. Μπορούμε να δημιουργήσουμε στοιχεία, να τα συνθέσουμε μεταξύ τους και να τους προσθέσουμε ιδιότητες, ακριβώς όπως κάνουμε με τις ετικέτες HTML. Η κύρια διαφορά μεταξύ

JSX και HTML είναι ότι στο JSX χρησιμοποιούμε JavaScript αντί για στατικό κείμενο. Αυτό σημαίνει ότι μπορούμε να ενσωματώσουμε μεταβλητές, να αξιοποιήσουμε συναρτήσεις και να γράψουμε δυναμικό κώδικα μέσα στο JSX. Αυτή η δυνατότητα επιτρέπει την ευέλικτη δημιουργία διεπαφών χρήστη με δυνατότητα αλληλεπίδρασης. Η χρήση του JSX καθιστά την ανάπτυξη εφαρμογών React πιο ευανάγνωστη και ευκολότερη για τους προγραμματιστές, καθώς μπορούν να συνδυάσουν την ισχύ της JavaScript με την κατανοητή δομή του HTML για τη δημιουργία πολύπλοκων διεπαφών χρήστη.[3]

3.1.4 JSON

Το JSON (JavaScript Object Notation) είναι μια μορφή αποθήκευσης και ανταλλαγής δεδομένων. Αποτελεί ένα απλό, αναγνώσιμο ανθρώπινα κατανοητό κείμενο που χρησιμοποιείται για την αναπαράσταση δομημένων πληροφοριών. Αρχικά αναπτύχθηκε για τη γλώσσα προγραμματισμού JavaScript, αλλά σήμερα χρησιμοποιείται σε πολλές γλώσσες προγραμματισμού. Ο κώδικας JSON αποτελείται από αντικείμενα και πίνακες. Τα αντικείμενα είναι συλλογές με ζευγάρια "κλειδί-τιμή", όπου το κλειδί είναι μια συμβολοσειρά και η τιμή μπορεί να είναι αριθμός, συμβολοσειρά, λογική τιμή, αντικείμενο ή πίνακας. Οι πίνακες είναι συλλογές τιμών που μπορούν να είναι αριθμοί, συμβολοσειρές, λογικές τιμές, αντικείμενα ή άλλοι πίνακες. Ο JSON χρησιμοποιείται ευρέως για τη μεταφορά δεδομένων μεταξύ εφαρμογών και την αποθήκευση δομημένων πληροφοριών. Είναι ανεξάρτητο από γλώσσες προγραμματισμού και υποστηρίζεται από πολλές πλατφόρμες και περιβάλλοντα προγραμματισμού. Η απλότητα του JSON και η ευκολία ανάγνωσής του από ανθρώπους και μηχανές το έχουν καταστήσει δημοφιλές στην ανταλλαγή δεδομένων σε διαδικτυακές εφαρμογές.[4]

3.2 Περιβάλλοντα Ανάπτυξης

3.2.1 Visual Studio Code

Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE - Integrated Development Environment) που παρέχεται από τη Microsoft. Αποτελεί ένα ισχυρό εργαλείο για τη δημιουργία εφαρμογών σε διάφορες πλατφόρμες, όπως το λειτουργικό σύστημα Windows, τον περιηγητή ιστού Microsoft Edge, το Azure cloud και άλλα. Το Visual Studio προσφέρει πλούσιες δυνατότητες για την ανάπτυξη, τη δοκιμή, την αποσφαλμάτωση και τη συντήρηση εφαρμογών. Παρέχει ένα φιλικό περιβάλλον με ευέλικτες επιλογές προγραμματισμού, που καλύπτουν μια ευρεία γκάμα γλωσσών, όπως η C#, η C++, η JavaScript και άλλες. Επίσης, υποστηρίζει την ανάπτυξη εφαρμογών για διάφορες πλατφόρμες, όπως το Desktop, το Web, το Mobile, το Cloud και το IoT. Το Visual Studio παρέχει επίσης πλούσια σύστημα εργαλείων για την ομαδική συνεργασία, την έκδοση ελέγχου (version control), την αυτοματοποίηση διαδικασιών (automation), τη διαχείριση των εξαρτήσεων και πολλά άλλα. Επιπλέον, προσφέρει ευέλικτες δυνατότητες για τη διαμόρφωση του περιβάλλοντος ανάπτυξης, με πρόσθετα (extensions) που επεκτείνουν τις λειτουργίες του IDE. Συνολικά, το Visual Studio είναι ένα ισχυρό και πλήρες εργαλείο ανάπτυξης που επιτρέπει στους προγραμματιστές να δημιουργήσουν εφαρμογές υψηλής ποιότητας και αποδοτικότητας για διάφορες πλατφόρμες και σενάρια ανάπτυξης.[5]

3.2.2 Android Studio

Το Android Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που αναπτύχθηκε από την Google για τη δημιουργία εφαρμογών για το λειτουργικό σύστημα Android. Αποτελεί το κύριο εργαλείο για τους προγραμματιστές που επιθυμούν να δημιουργήσουν εφαρμογές για την πλατφόρμα Android. Το Android Studio προσφέρει ένα πλούσιο σύνολο εργαλείων και λειτουργιών που βοηθούν στην ανάπτυξη, επεξεργασία και αποσφαλμάτωση των εφαρμογών Android. Περιλαμβάνει έναν γρήγορο Android emulator για τη δοκιμή των εφαρμογών σε διάφορες συσκευές, έναν πλούσιο σε δυνατότητες επεξεργαστή κώδικα, συστήματα οπτικής ανάπτυξης διεπαφών χρήστη (UI) και δυνατότητες παρακολούθησης της απόδοσης.[6]

3.3 Frameworks

3.3.1 React Native

Η React Native είναι μια ανοικτού κώδικα πλατφόρμα ανάπτυξης εφαρμογών για τη δημιουργία κινητών εφαρμογών για Android και iOS χρησιμοποιώντας την JavaScript και το πλαίσιο React. Με τη χρήση της React Native, μπορούν να αναπτυχθούν εφαρμογές για κινητές συσκευές με επαναχρησιμοποιήσιμο κώδικα, μειώνοντας έτσι τον χρόνο και την προσπάθεια που απαιτείται για την ανάπτυξη εφαρμογών για κάθε πλατφόρμα ξεχωριστά. Η React Native προσφέρει κάποιες συναρπαστικές δυνατότητες, όπως γρήγορη ανανέωση (hot-reloading), οπτικοποίηση πραγματικού χρόνου (real-time rendering), και δυνατότητες πρόσβασης σε υπηρεσίες συστήματος όπως η κάμερα, οι ειδοποιήσεις και οι αισθητήρες της συσκευής. Η ανάπτυξη μιας εφαρμογής React Native δίνει την δυνατότητα σε κάθε προγραμματιστή, να έχει μια ομοιόμορφη εμπειρία χρήσης ανεξάρτητα από το λειτουργικό σύστημα της συσκευής, καθώς μοιράζεται τον ίδιο κώδικα για τις διάφορες πλατφόρμες.[7]

3.3.2 Expo

Το Expo είναι μια ανοικτού κώδικα πλατφόρμα για την ανάπτυξη εφαρμογών React Native. Παρέχει ένα σύνολο εργαλείων και υπηρεσιών που διευκολύνουν την ανάπτυξη, τον αποστολέα και τον κατανεμημένο έλεγχο εφαρμογών React Native. Η Expo επιτρέπει στους προγραμματιστές να αναπτύξουν εφαρμογές χωρίς την ανάγκη για εγκατάσταση εξωτερικών εργαλείων ανάπτυξης, όπως το Android Studio ή το Xcode.[7]

3.4 Περιβάλλον Εκτέλεσης

3.4.1 Node.js

Το Node.js είναι ένα περιβάλλον εκτέλεσης JavaScript που βασίζεται στον JavaScript κινητήρα V8 της Google και επιτρέπει την εκτέλεση JavaScript κώδικα έξω από τον περιηγητή ιστού (browser). Αναπτύχθηκε αρχικά από τον Ryan Dahl το 2009 και έχει γίνει δημοφιλές για τη δημιουργία επιδραστικών και ευέλικτων διακομιστών (servers) και εφαρμογών. Ο στόχος του Node.js είναι να παρέχει ένα αποδοτικό, ασύγχρονο για τη διαχείριση των πόρων για αιτήματα εισόδου/εξόδου I/O μοντέλο, το οποίο το καθιστά κατάλληλο για αναπτυξιακά σενάρια που απαιτούν υψηλή κλίμακα και απόδοση σε πραγματικό χρόνο. Επίσης, χρησιμοποιείται για τη δημιουργία web εφαρμογών, ανταλλαγή δεδομένων μέσω API, ανάπτυξη εργαλείων για τον περιηγητή ιστού, εκτέλεση εργασιών γραμμής εντολών και πολλά άλλα. Ένα από τα βασικά πλεονεκτήματα του Node.js είναι η χρήση του μοντέλου μη-μπλοκαρισμένης (non-blocking) I/O, που επιτρέπει την ασύγχρονη επεξεργασία πολλαπλών αιτημάτων ταυτόχρονα, χωρίς να απαιτείται η δημιουργία νέων νημάτων για κάθε αίτημα. Αυτό οδηγεί σε αυξημένη απόδοση και αποτελεσματικότητα του συστήματος. Το Node.js διαθέτει επίσης ένα εκτεταμένο οικοσύστημα πακέτων (package ecosystem) μέσω του npm (Node Package Manager), το οποίο παρέχει πρόσβαση σε χιλιάδες πακέτα κώδικα που μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών. Συνολικά, το Node.js προσφέρει ένα δυναμικό και ευέλικτο περιβάλλον για την ανάπτυξη διακομιστών και εφαρμογών, επιτρέποντας στους προγραμματιστές να δημιουργήσουν γρήγορα και αποδοτικά λογισμικά.[8]

3.5 Βάση Δεδομένων

3.5.1 Firebase

Στην πληροφορική, μια βάση δεδομένων είναι μια οργανωμένη συλλογή δεδομένων που αποθηκεύονται και αποκτώνται ηλεκτρονικά από ένα σύστημα υπολογιστή. Όπου οι βάσεις

δεδομένων είναι πιο περίπλοκες, συχνά αναπτύσσονται χρησιμοποιώντας επίσημες τεχνικές σχεδιασμού και μοντελοποίησης. Η Firebase είναι μια NoSQL βάση δεδομένων που φτιάχτηκε από την Google. Είναι μια cloud based βάση υπηρεσία δηλαδή τρέχει στο διαδίκτυο σε κάποιον απομακρυσμένο server της Google και όχι σε τοπικό server όπως οι συνηθισμένες βάσεις. Παρέχει στους προγραμματιστές μια ποικιλία εργαλείων και υπηρεσιών για να τους βοηθήσει να αναπτύξουν ποιοτικές εφαρμογές. Η Firebase χωρίζει τις υπηρεσίες της σε κατηγορίες, για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν σχεδόν όλες οι υπηρεσίες της βάσης.[9]

3.5.1.1 Real Time Database

Η Realtime database είναι μια υπηρεσία της Firebase η οποία είναι ο πιο γρήγορος τρόπος αποθήκευσης, ανάκτησης και ενημέρωσης δεδομένων σε πραγματικό χρόνο. Τα δεδομένα αποθηκεύονται σε μορφή JSON και είναι διαθέσιμα ακόμα και εκτός σύνδεσης. Το ελάττωμα αυτής της υπηρεσίας είναι ότι δεν είναι εύκολο να γίνει ανάκτηση των δεδομένων με φίλτρα. Αυτή η υπηρεσία χρησιμοποιήθηκε για τα προσωπικά δεδομένα των χρηστών καθώς επίσης και για την δημιουργία νέων ερωτήσεων στα παιχνίδια quiz.[9]

3.5.1.2 Authentication

Η υπηρεσία Authentication του Firebase παρέχει μηχανισμούς για την ασφαλή αυθεντικοποίηση χρηστών στις εφαρμογές. Προσφέρει δυνατότητες όπως εγγραφή, σύνδεση, ανάκτηση κωδικού πρόσβασης και διαχείριση προφίλ χρηστών. Αυτό επιτρέπει στις εφαρμογές να ελέγχουν την πρόσβαση και την ασφάλεια των χρηστών τους.[9]

3.6 API

3.6.1 Open Food Facts API

Το Open Food Facts API είναι μια δωρεάν, ανοικτή πηγή δεδομένων που περιέχει πληροφορίες για τα τρόφιμα. Το API παρέχει πρόσβαση σε μια μεγάλη βάση δεδομένων με πληροφορίες για τα συστατικά, τις θρεπτικές αξίες, τις ετικέτες, τις αλλεργίες και πολλά άλλα για διάφορα τρόφιμα. Αυτό το API είναι χρήσιμο για ανάπτυξη εφαρμογών που ασχολούνται με τη διατροφή, την υγεία ή την αναζήτηση προϊόντων. Χρησιμοποιώντας το Open Food Facts API, μπορείτε να ανακτήσετε και να χρησιμοποιήσετε αυτές τις πληροφορίες για τη δημιουργία εφαρμογών που παρέχουν πληροφορίες για τα τρόφιμα, επιτρέποντας στους χρήστες να ενημερώνονται για την ποιότητα και την υγιεινή των τροφίμων που καταναλώνουν.[10]

3.6.2 Firebase API

Η Firebase API είναι μια πλατφόρμα ανάπτυξης που παρέχει ένα ολοκληρωμένο σετ εργαλείων για τη δημιουργία και τη διαχείριση εφαρμογών. Με τη χρήση της Firebase API, οι προγραμματιστές μπορούν να δημιουργήσουν γρήγορα και εύκολα εφαρμογές για διάφορες πλατφόρμες, όπως ιστοσελίδες, κινητές συσκευές και έξυπνες συσκευές. Η Firebase API παρέχει πολλές λειτουργίες, όπως αυθεντικοποίηση χρηστών, αποθήκευση δεδομένων σε πραγματικό χρόνο, αποστολή ειδοποιήσεων και πολλές ακόμη. Επιπλέον, προσφέρει έναν αποτελεσματικό τρόπο για τη διαχείριση χρηστών, την ανάπτυξη κλιμάκωσης εφαρμογών και την παροχή αναλυτικών πληροφοριών και στατιστικών. Με την Firebase API, οι προγραμματιστές μπορούν να επωφεληθούν από ένα αξιόπιστο και ευέλικτο σύστημα που διευκολύνει την ανάπτυξη και τη συντήρηση εφαρμογών. Επιπλέον, η Firebase API είναι ένας ισχυρός σύμμαχος για την επιτάχυνση της ανάπτυξης λογισμικού και τη βελτίωση της εμπειρίας των χρηστών.[9]

Κεφάλαιο 4: Υλοποίηση

Σε αυτήν την ενότητα θα αναλυθεί ο τρόπος υλοποίησης και τα βήματα που ακολουθήθηκαν για την υλοποίηση αυτής της εφαρμογής.

4.1 Σχεδίαση Εφαρμογής

Για την υλοποίηση του project, πρέπει να αναφέρω ότι εργάστηκα σε λογισμικό Windows, οπότε οι εντολές που βασίστηκε αυτή η εργασία, λειτουργούν για λογισμικό Windows.

4.1.1 Πρώτο Στάδιο

Αρχικά, γίνεται η εγκατάσταση του Node.js από την επίσημη ιστοσελίδα του Node.js (https://nodejs.org). Το Expo απαιτεί το Node.js για την εκτέλεσή του. Έπειτα, πρέπει να γίνει εγκατάσταση του Expo CLI, η οποία γίνεται μέσω του τερματικού ή τη γραμμή εντολών και εκτελώντας την εντολή **npm install -g expo-cli** για να εγκατασταθεί το Expo Command Line Interface (CLI).

Για δημιουργία ενός νέου Expo project, μεταβαίνουμε στον κατάλογο όπου θέλουμε να δημιουργήσουμε το νέο μας project με την εντολή **cd Projects** και εκτελείται η εντολή **expo init Quiz** , όπου Quiz είναι το όνομα του project που θέλουμε να δημιουργήσουμε. Μετά πρέπει να ακολουθήσουμε τις οδηγίες για την επιλογή του template και την αρχικοποίηση του project. Στην δική μας περίπτωση επιλέξαμε την επιλογή **blank** η οποία δηλώνει ότι τα αρχεία που θα δημιουργηθούνε είναι κενά. Για την εκτέλεση του Expo project πρέπει να μεταβούμε στον φάκελο του νέου μας project (cd Quiz) και εκτελέσουμε την εντολή **expo start** ή **npm start** για να ξεκινήσει το Expo development server. Μόλις εκτελεστούν αυτά τα βήματα, μπορούμε να κλείσουμε τον τερματικό ή τη γραμμή εντολών.

Για την επεξεργασία των αρχείων που δημιουργήθηκαν θα πρέπει να γίνει η εγκατάσταση του Visual Studio Code (VS Code) από την επίσημη ιστοσελίδα του Visual Studio Code (https://code.visualstudio.com/). Ακόμα, θα εγκαταστήσουμε το Android Studio από την επίσημη ιστοσελίδα του Android Studio (https://developer.android.com/studio). Αφού γίνει η εγκατάσταση αυτών, ανοίγουμε το Android Studio και κατεβάζουμε ένα smartphone emulator το οποίο έχει πρόσβαση σε Google Play εφαρμογές μέσω της επιλογής του Device Manager. Ανοίγουμε τον emulator, κατεβάζουμε την εφαρμογή Expo ή Expo Go από το Google Play Store. Στη συνέχεια, θα ανοίξουμε στο VS Code τον φάκελο Quiz που δημιουργήσαμε, ανοίγουμε τον τερματικό του VS Code και εκτελούμε τις εντολές **cd Projects** και **cd Quiz** για να μεταβούμε στον φάκελο που θα εργαστούμε. Έπειτα εκτελούμε την εντολή **npm start** και εμφανίζονται ορισμένες εντολές οι οποίες μας δίνουν την δυνατότητα να εκτελέσουμε το project είτε σε έναν emulator Android ή iOS (το iOS emulator λειτουργεί μόνο σε υπολογιστές με λογισμικό MacOS), είτε σε μια σελίδα για Web App, καθώς και ένα QR code το οποίο μπορούμε να το σκανάρουμε από τις πραγματικές συσκευές μας, έχοντας κατεβασμένη την εφαρμογή Expo ή Expo Go.

Τέλος, εκτελείται η εντολή **npm install** για την εγκατάσταση πακέτων και βιβλιοθηκών τόσο για το front-end όσο και το back-end.

4.1.2 Βιβλιοθήκες

Οι βιβλιοθήκες είναι συλλογές προγραμματιστικού κώδικα που έχουν σχεδιαστεί και υλοποιηθεί για να παρέχουν συγκεκριμένες λειτουργίες και δυνατότητες. Είναι χρήσιμα εργαλεία που μας επιτρέπουν να επωφεληθούμε από το έργο και τις γνώσεις άλλων προγραμματιστών, καθιστώντας τη διαδικασία ανάπτυξης ευκολότερη, ταχύτερη και πιο αξιόπιστη.[7] Μπορούμε επίσης να ορίσουμε σαν βιβλιοθήκη συναρτήσεις, κλάσεις ή/και οτιδήποτε άλλο έχουμε δημιουργήσει και θέλουμε να χρησιμοποιηθεί για την υλοποίηση ενός καινούριου αρχείου.

Στην παρακάτω εικόνα βλέπουμε τον τρόπο με τον οποίο μπορούμε να εισάγουμε ορισμένα στοιχεία/εργαλεία από πακέτα που έχουμε κατεβάσει, καθώς και ορισμένες συναρτήσεις που έχουμε

αποθηκευμένες σε άλλα αρχεία και θέλουμε να χρησιμοποιηθούν στο συγκεκριμένο αρχείο.

```
import { useContext, useEffect, useState } from "react";
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { StatusBar } from "expo-status-bar";
import AsyncStorage from "@react-native-async-storage/async-storage";
import LoadingOverlay from "./components/ui/LoadingOverlay";
import { Dimensions } from "react-native";

import { createDrawerNavigator } from "@react-navigation/drawer";

//sreens
import LoginScreen from "./screens/LoginScreen";
import SignupScreen from "./screens/SignupScreen";
import WelcomeScreen from "./screens/WelcomeScreen";
import CategoriesScreen from "./screens/CategoriesScreen";
import MyAccountScreen from "./screens/MyAccountScreen";

import { Ionicons } from "@expo/vector-icons";

import { Colors } from "./constants/styles";
import AuthContextProvider, { AuthContext } from "./store/auth-context";
import IconButton from "./components/ui/IconButton";
```

Εικόνα 12-Παράδειγμα αρχικοποίησης βιβλιοθηκών

4.1.3 Navigation

Οι πλοηγήσεις είναι ένας σημαντικός και απαραίτητος μηχανισμός που αναφέρονται στον τρόπο με τον οποίο οι χρήστες μεταβαίνουν από ένα στοιχείο σε ένα άλλο μέσα στην εφαρμογή. Οι πλοηγήσεις συχνά ευθύνονται στη δημιουργία διαφόρων οθονών και τη δυνατότητα μετάβασης μεταξύ αυτών.[7] Μια κοινή πρακτική είναι η χρήση των "stack navigators", οι οποίοι επιτρέπουν στους χρήστες να πραγματοποιούν μεταβάσεις με τη μορφή στοίβας (stack) οθονών. Αυτό σημαίνει ότι οι νέες οθόνες τοποθετούνται πάνω στη στοίβα και οι προηγούμενες οθόνες παραμένουν κάτω από αυτές. Με την χρήση αυτών μπορούμε να δημιουργήσουμε μηχανισμούς πλοήγησης όπως τα "tabs" (καρτέλες) ή και μηχανισμούς πλοήγησης όπως τα "drawers" (συρτάρια).

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν όλα όσα προαναφέρθηκαν.

```
function AuthStack() {
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: "white",
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen name="Login" component={LoginScreen} />
      <Stack.Screen name="Signup" component={SignupScreen} />
    </Stack.Navigator>
  );
}
```

Εικόνα 13-Παράδειγμα αρχικοποίησης οθονών σε Stack Navigator

Η παραπάνω εικόνα δείχνει την αρχικοποίηση των οθονών “Log in” και “Sign up” σε μορφή στοίβας. Οι δύο αυτές οθόνες καλούνται πρώτες κατά την εκκίνηση της εφαρμογής καθώς ο

χρήστης πρέπει πρώτα να δημιουργήσει λογαριασμό ή να συνδεθεί αν έχει δημιουργήσει λογαριασμό στο παρελθόν, προκειμένου να μπορέσει να χρησιμοποιήσει την εφαρμογή. Η αρχικοποίηση αυτών των οθονών γίνεται με την χρήση του “Stack.Navigator” το οποίο είναι ένα εξάρτημα που παρέχεται από τη βιβλιοθήκη πλοήγησης React Navigation και εντός αυτού μπορούμε να ορίσουμε τις οθόνες που θέλουμε να προβληθούν, με την σειρά την οποία επιθυμούμε. Στην “Stack.Navigator” καθώς και στις υπόλοιπες πλοηγήσεις που θα αναφέρουμε παρακάτω, μπορούμε να ορίσουμε ορισμένες επιλογές οι οποίες μας βοηθάνε να ρυθμίσουμε την εμφάνιση που θα έχουν οι οθόνες που θα προβάλλουμε (με την χρήση του screenOptions).

Μια άλλη μορφή πλοήγησης που χρησιμοποιούμε στην εφαρμογή είναι η “Tab” η οποία μας επιτρέπει να μεταβαίνουμε στις οθόνες επιλέγοντας τις ‘’καρτέλες’’ που απεικονίζονται στο κάτω μέρος της εφαρμογής. Η αρχικοποίηση των οθονών και η ρύθμιση εμφάνισης αυτών, γίνεται με παρόμοιο τρόπο όπως γίνεται και με την “Stack.Navigator”. Οι παρακάτω εικόνες μας δείχνει ένα παράδειγμα της αρχικοποίησης των οθονών εντός του “Tab.Navigator”.

```
function AuthenticatedStack() {
  const authCtx = useContext(AuthContext);
  return (
    <Tab.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: "white",
        contentStyle: { backgroundColor: Colors.primary100 },
        tabBarShowLabel: false, // remove text label for all screens
        showLabel: false,
        tabBarActiveTintColor: "#e91e63",
        tabBarStyle: {
          position: "absolute",
          bottom: 0,
          left: 0,
          right: 0,
          borderRadius: 40,
          height: 85,
          elevation: 5, // This property adds the shadow to the tab bar
          backgroundColor: "#f8f8ff",
          shadowColor: "#000000",
          shadowOffset: {
            width: 0,
            height: 3,
          },
          shadowOpacity: 0.3,
          shadowRadius: 5,
        },
      }}
    >
      <Tab.Screen
        name="Home"
        component={WelcomeScreen}
        options={{
          tabBarIcon: ({ color, size }) => (
            <Ionicons name="home" color={color} size={iconSize} />
          ),

          headerRight: ({ tintColor }) => (
            <IconButton
              icon="exit"
```

Εικόνα 14-Παράδειγμα αρχικοποίησης Tab Navigator

```
        <Tab.Screen
          name="GoalsScreen"
          component={DrawerNavigator}
          options={{
            tabBarButton: () => null,
            tabBarVisible: false,
            title: "GoalsScreen",
          }}
        />

        <Tab.Screen
          name="ActivitiesScreen"
          component={DrawerNavigator}
          options={{
            tabBarButton: () => null,
            tabBarVisible: false,
            title: "ActivitiesScreen",
          }}
        />

        <Tab.Screen
          name="ScanMe"
          component={DrawerNavigator}
          options={{
            tabBarStyle: { display: "none" },
            tabBarButton: () => null,
            tabBarVisible: false,
            title: "Scan",
          }}
        />
      </Tab.Navigator>
    );
}
```

Εικόνα 15-Παράδειγμα αρχικοποίησης Tab οθονών

Τέλος, η τελευταία μορφή πλοήγησης που χρησιμοποιούμε στην εφαρμογή είναι η “Drawer” η οποία μας επιτρέπει να μεταβαίνουμε στις οθόνες επιλέγοντας μια από τις επιλογές που εμφανίζονται εντός του “συρταριού” που ανοίγει πατώντας το κουμπί που απεικονίζεται στην αριστερή πλευρά της οθόνης. Η αρχικοποίηση των οθονών και η ρύθμιση εμφάνισης αυτών, γίνεται με παρόμοιο τρόπο όπως γίνεται και με την “Stack.Navigator” και την “Tab.Navigator”. Οι παρακάτω εικόνες μας δείχνει ένα παράδειγμα της αρχικοποίησης των οθονών εντός του “Drawer.Navigator”.

```
function DrawerNavigator() {
  return (
    <Drawer.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: "white",
        contentStyle: { backgroundColor: Colors.primary100 },
        drawerActiveTintColor: Colors.primary500,
        zIndex: 1,
      }}
    >
      <Drawer.Screen
        name="Profile"
        component={MyAccountScreen}
        options={{
          drawerIcon: ({ color, size }) => (
            <Ionicons name="person" color={color} size={size} />
          ),
          swipeEnabled: true,
        }}
      />
      <Drawer.Screen
        name="Goals"
        component={GoalsScreen}
        options={{
          drawerIcon: ({ color, size }) => (
            <Ionicons
              name="checkmark-circle-outline"
              color="black"
              size={size}
            />
          ),
          swipeEnabled: true,
        }}
```

Εικόνα 16-Παράδειγμα αρχικοποίησης Drawer Navigator

```
      <Drawer.Screen
        name="Activities"
        component={ActivitiesScreen}
        options={{
          drawerIcon: ({ color, size }) => (
            <Ionicons name="body-outline" color="black" size={size} />
          ),
          swipeEnabled: true,
        }}
      />
      <Drawer.Screen
        name="Scan"
        component={ScanMeScreen}
        options={{
          drawerIcon: ({ color, size }) => (
            <Ionicons name="scan-outline" color="black" size={size} />
          ),
          swipeEnabled: true,
        }}
      />
    </Drawer.Navigator>
  );
}
```

Εικόνα 17-Παράδειγμα αρχικοποίησης Drawer οθονών

4.1.4 Οθόνες

4.1.4.1 Οθόνη Εγγραφής/Σύνδεσης

Η οθόνη εγγραφής είναι υπεύθυνη για την δημιουργία λογαριασμών των χρηστών, ενώ η οθόνη σύνδεσης ευθύνεται για την είσοδο των χρηστών στην εφαρμογή. Η υλοποίηση αυτών απαρτίζεται από πολλά στάδια.

Αρχικά, και οι δύο συναρτήσεις αυτών των οθονών περιέχουν έλεγχο για τα στοιχεία που θα εισάγουν οι χρήστες κατά την δημιουργία λογαριασμών τους ή κατά την σύνδεσή τους και καλούν την συνάρτηση “AuthContent”.

```
  async function signupHandler({ email, password }) {
    setIsAuthenticating(true);
    try {
      const token = await createUser(email, password);
      authCtx.authenticate(token);
    } catch (error) {
      Alert.alert(
        'Authentication failed',
        'Could not create user, please check your input and try again later.'
      );
      setIsAuthenticating(false);
    }
  }

  if (isAuthenticating) {
    return <LoadingOverlay message="Creating user..." />;
  }

  return <AuthContent onAuthenticate={signupHandler} />;
}
```

Εικόνα 18-Στιγμιότυπο της SignUpScreen

```
async function loginHandler({ email, password }) {
  setIsAuthenticating(true);
  try {
    const token = await login(email, password);
    authCtx.authenticate(token);
  } catch (error) {
    Alert.alert(
      'Authentication failed!',
      'Could not log you in. Please check your credentials or try again later!'
    );
    setIsAuthenticating(false);
  }
}

if (isAuthenticating) {
  return <LoadingOverlay message="Logging you in..." />;
}

return <AuthContent isLogin onAuthenticate={loginHandler} />;
}
```

Εικόνα 19-Στιγμιότυπο της LogInScreen

Η συνάρτηση “AuthContent” λαμβάνει δύο παραμέτρους: την “isLogin”, που υποδηλώνει αν η φόρμα πιστοποίησης είναι για σύνδεση ή εγγραφή, και την “onAuthenticate”, που είναι η συνάρτηση που καλείται όταν υποβάλλεται η φόρμα πιστοποίησης. Ακόμα, ορίζεται μια μεταβλητή κατάσταση “credentialsInvalid”, η οποία διαχειρίζεται το αν οι διαπιστευτήρια που εισάγονται στη φόρμα είναι έγκυρα ή όχι. Η συνάρτηση “switchAuthModeHandler” καλείται όταν ο χρήστης πατάει το κουμπί για αλλαγή λειτουργίας πιστοποίησης (είσοδος/εγγραφή). Ανάλογα με την τρέχουσα κατάσταση, η μεταβλητή “navigation” αντικαθίσταται με την αντίθετη οθόνη (αν είναι στην οθόνη εισόδου, γίνεται οθόνη εγγραφής και αντίστροφα).

```
function AuthContent({ isLogin, onAuthenticate }) {
  const navigation = useNavigation();

  const [credentialsInvalid, setCredentialsInvalid] = useState({
    email: false,
    password: false,
    confirmEmail: false,
    confirmPassword: false,
  });

  function switchAuthModeHandler() {
    if (isLogin) {
      navigation.replace('Signup');
    } else {
      navigation.replace('Login');
    }
  }
```

Εικόνα 20-Στιγμιότυπο της AuthContent συνάρτησης

Η “submitHandler” καλείται όταν ο χρήστης υποβάλλει τη φόρμα πιστοποίησης. Παίρνει τα διαπιστευτήρια που εισάγονται και εκτελεί διάφορους ελέγχους έγκυρης εισόδου (π.χ. έγκυρη διεύθυνση email, μήκος κωδικού πάνω από 6 χαρακτήρες, ίδια email και επιβεβαίωση κωδικού αν δεν είναι είσοδος).

Σε περίπτωση λάθους εμφανίζεται και το αντίστοιχο μήνυμα, αλλιώς καλείται η “onAuthenticate” με τα διαπιστευτήρια του χρήστη.

```
function submitHandler(credentials) {
  let { email, confirmEmail, password, confirmPassword } = credentials;

  email = email.trim();
  password = password.trim();

  const emailIsValid = email.includes('@');
  const passwordIsValid = password.length > 6;
  const emailsAreEqual = email === confirmEmail;
  const passwordsAreEqual = password === confirmPassword;

  if (
    !emailIsValid ||
    !passwordIsValid ||
    (!isLogin && (!emailsAreEqual || !passwordsAreEqual))
  ) {
    Alert.alert('Invalid input', 'Please check your entered credentials.');
    setCredentialsInvalid({
      email: !emailIsValid,
      confirmEmail: !emailIsValid || !emailsAreEqual,
      password: !passwordIsValid,
      confirmPassword: !passwordIsValid || !passwordsAreEqual,
    });
    return;
  }
  onAuthenticate({ email, password });
}
```

Εικόνα 21-Στιγμιότυπο της submitHandler συνάρτησης

Η “AuthContent” με τη σειρά της καλεί την “AuthForm” η οποία είναι ένας συνδυασμός κειμένων εισαγωγής (Input) και κουμπιών (Button) που χρησιμοποιούνται για την απεικόνιση και υποβολή της φόρμας πιστοποίησης αναλόγως με την οθόνη στην οποία βρίσκεται ο χρήστης. Ακόμα, είναι υπεύθυνο για τη συλλογή των στοιχείων που εισήγαγε ο χρήστης και την προώθησή τους στην υποβολή.

```
function updateInputValueHandler(inputType, enteredValue) {
  switch (inputType) {
    case 'email':
      setEnteredEmail(enteredValue);
      break;
    case 'confirmEmail':
      setEnteredConfirmEmail(enteredValue);
      break;
    case 'password':
      setEnteredPassword(enteredValue);
      break;
    case 'confirmPassword':
      setEnteredConfirmPassword(enteredValue);
      break;
  }
}
```

Εικόνα 22-Στιγμιότυπο 1 της AuthForm συνάρτησης

```
      )}
      <Input
        label="Password"
        onUpdateValue={updateInputValueHandler.bind(this, 'password')}
        secure
        value={enteredPassword}
        isInvalid={passwordIsInvalid}
      />
      {!isLogin && (
        <Input
          label="Confirm Password"
          onUpdateValue={updateInputValueHandler.bind(
            this,
            'confirmPassword'
          )}
          secure
          value={enteredConfirmPassword}
          isInvalid={passwordsDontMatch}
        />
      )}
      <View style={styles.buttons}>
        <Button onPress={submitHandler}>
          {isLogin ? 'Log In' : 'Sign Up'}
        </Button>
      </View>
    </View>
```

Εικόνα 23-Στιγμιότυπο 2 της AuthForm συνάρτησης

Επιπλέον, και οι δύο αυτές οθόνες σε περίπτωση επιτυχής εγγραφής και σύνδεσης, αποθηκεύουν τα tokens και τα ids από τα στοιχεία που εισήγαγαν οι χρήστες, στον χώρο των προσωπικών τους τηλεφώνων προκειμένου να μην χρειάζεται να συνδέονται κάθε φορά που είναι να χρησιμοποιήσουν την εφαρμογή. Αυτό επιτυγχάνεται με την κλήση της συνάρτησης"AuthContext" και πιο συγκεκριμένα την συνάρτηση "authenticate" η οποία στέλνει ένα αίτημα HTTP στην βάση δεδομένων και συλλέγει τα δεδομένα.

```
function AuthContextProvider({ children }) {
  const [authToken, setAuthToken] = useState();
  const [uid, setUid] = useState();

  async function authenticate(token) {
    setAuthToken(token);

    //apothikeysh stoixeiwn toy xrhsth
    AsyncStorage.setItem("token", token);

    // retrieve uid of the user from db (it will be needed in making the requests inside the pages)
    const res = await axios.post(
      `https://identitytoolkit.googleapis.com/v1/accounts:lookup?key=${API_KEY}`,
      {
        idToken: token,
      }
    );

    const uid = res?.data?.users?.[0]?.localId;

    if (uid) setUid(uid);
```

Εικόνα 24-Στιγμιότυπο της AuthContextProvider συνάρτησης

Τα tokens διαγράφονται από την μνήμη του κινητού σε περίπτωση που οι χρήστες επιλέξουν να

αποσυνδεθούν από την εφαρμογή.

```
function logout() {
  setAuthToken(null);
  setUid(null);
  AsyncStorage.removeItem("token");
}
```

Εικόνα 25-Στιγμιότυπο διαγραφής των token

Τέλος, η συνάρτηση “LoginScreen” καλεί την συνάρτηση “login” και η συνάρτηση “SignupScreen” καλεί την συνάρτηση “createUser”. Και οι δύο συναρτήσεις μας παραπέμπουν στο αρχείο “auth”, το οποίο αναλαμβάνει να πραγματοποιήσει την επικοινωνία με το API του Firebase Authentication για εκτέλεση λειτουργιών σύνδεσης και εγγραφής χρήστη. Παίρνει ως παραμέτρους το mode που μπορεί να είναι "signUp" για εγγραφή νέου χρήστη ή "signInWithPassword" για σύνδεση υπάρχοντος χρήστη, καθώς επίσης και το email και password για τα στοιχεία του χρήστη που υποβάλλονται για σύνδεση ή εγγραφή.

```
export const API_KEY = "AIzaSyBvC3Vv-yKqu6P_jGrEWx1bWS6oqPnTk34";

async function authenticate(mode, email, password) {
  const url = `https://identitytoolkit.googleapis.com/v1/accounts:${mode}?key=${API_KEY}`;

  const response = await axios.post(url, {
    email: email,
    password: password,
    returnSecureToken: true,
  });

  const token = response.data.idToken;

  return token;
}

export function createUser(email, password) {
  return authenticate("signUp", email, password);
}

export function login(email, password) {
  return authenticate("signInWithPassword", email, password);
}
```

Εικόνα 26-Στιγμιότυπο της authenticate συνάρτησης

4.1.4.2 Αρχική Οθόνη

Η αρχική οθόνη αποτελείται από:

- Ένα κουμπί αποσύνδεσης το οποίο αρχειοθετείται στην “App.js” και η λειτουργικότητα αυτού αναλύθηκε προηγουμένως.

```
<Tab.Screen
  name="Home"
  component={WelcomeScreen}
  options={{
    tabBarIcon: ({ color, size }) => (
      <Ionicons name="home" color={color} size={iconSize} />
    ),

    headerRight: ({ tintColor }) => (
      <IconButton
        icon="exit"
        color={tintColor}
        size={24}
        onPress={authCtx.logout}
      />
    ),
  }}
/>
```

Εικόνα 27-Στιγμιότυπο αρχικοποίησης αρχικής οθόνης και κουμπιού αποσύνδεσης

- Ένα κουμπί που μας παραπέμπει στην οθόνη κατηγοριών.

```
<Pressable
  onPress={() => navigation.navigate("Categories")}
  style={styles.pressableContainer}
>
  <Text style={styles.pressableTextContainer}>Categories</Text>
</Pressable>
```

Εικόνα 28-Στιγμιότυπο κουμπιού πλοήγησης στην οθόνη κατηγοριών

- Ένα κουμπί που ενεργοποιεί ένα BottomSheet το οποίο μας δίνει τις απαραίτητες πληροφορίες για τα παιχνίδια quiz.

```
<Pressable
  onPress={toggleBottomNavigationView}
  style={styles.pressableContainerInfo}
>
  <Text style={styles.pressableTextContainer}>Information</Text>
</Pressable>
<BottomSheet
  visible={visible}
  onBackButtonPress={toggleBottomNavigationView}
  onBackdropPress={toggleBottomNavigationView}
>
  <View style={styles.bottomNavigationView}>
    <View
      style={{
        flex: 1,
        flexDirection: "column",
        justifyContent: "space-between",
        marginBottom: 20,
      }}
    >
      <Text style={styles.bottomSheetTextContainer}>
        What you need to know before you start:
      </Text>
      <View
        style={{
          padding: 45,
          paddingTop: 20,
          backgroundColor: Colors.primary100,
          borderRadius: 10,
        }}
      >
```

Εικόνα 29-Στιγμιότυπο κουμπιού πληροφοριών παιχνιδιού

- Ένα κείμενο καλωσορίσματος με το logo της εφαρμογής.

```
<View style={styles.rootContainer}>
  <Image
    style={{
      height: 350,
      width: "140%",
      resizeMode: "contain",
      marginLeft: -10,
    }}
    source={require("../assets/ZoomedLOGO.png")}
  />
  <Text style={styles.title}>Welcome!</Text>

  <Text style={styles.texts}>
    Choose your category and learn more about your eating habits!
  </Text>
```

Εικόνα 30-Στιγμιότυπο δεδομένων στην οθόνη καλωσορίσματος

4.1.4.3 Οθόνη Κατηγοριών

Η οθόνη κατηγοριών αποτελείται από μια απλή σχεδίαση από επιλογές οι οποίες μας παραπέμπουν στις αντίστοιχες οθόνες Quiz της κατηγορίας που επιλέγουν οι χρήστες.

```
<ScrollView style={{ backgroundColor: Colors.primary100 }}>
  <View style={styles.innerContainer}>
    <Pressable
      onPress={() => navigation.navigate("QuizOverview")}
      style={styles.buttonCal}
    >
      <Text style={styles.text}>Calories</Text>
    </Pressable>
    <Pressable
      onPress={() => navigation.navigate("FatQuiz")}
      style={styles.buttonFat}
    >
      <Text style={styles.text}>Fats</Text>
    </Pressable>
    <Pressable
      onPress={() => navigation.navigate("SugarQuiz")}
      style={styles.buttonSugar}
    >
      <Text style={styles.text}>Sugar</Text>
    </Pressable>
    <Pressable
      onPress={() => navigation.navigate("ProteinQuiz")}
      style={styles.buttonProtein}
    >
      <Text style={styles.text}>Protein</Text>
    </Pressable>
```

Εικόνα 31-Στιγμιότυπο κώδικα οθόνης κατηγοριών

4.1.4.4 Οθόνες Quiz

Οι ερωτήσεις του παιχνιδιού που έχουν σχεδιαστεί είναι οργανωμένες σε διάφορες κατηγορίες. Όταν ο χρήστης επιλέξει το κουμπί “Play”, του παρέχεται η δυνατότητα να επιλέξει μια από αυτές τις κατηγορίες. Αφού επιλέξει μια κατηγορία, ο χρήστης έχει τρεις επιλογές:

1. Να παίξει το quiz με τις ερωτήσεις που ήδη υπάρχουν: Ο χρήστης μπορεί να ξεκινήσει το quiz και να απαντήσει σε μια σειρά ερωτήσεων που ανήκουν στην επιλεγμένη κατηγορία. Καθώς προχωράει, λαμβάνει αμέσως την ανατροφοδότηση για τις απαντήσεις του απαντώντας τες εντός χρόνου (15 δευτερόλεπτα).

2. Να δημιουργήσει μια δική του ερώτηση: Ο χρήστης έχει τη δυνατότητα να προσθέσει μια νέα ερώτηση στο quiz της επιλεγμένης κατηγορίας. Μπορεί να εισάγει το κείμενο της ερώτησης, τις πιθανές απαντήσεις και τη σωστή απάντηση. Η νέα ερώτηση θα προστεθεί στο quiz και θα είναι διαθέσιμη για τους χρήστες που παίζουν το quiz.

3. Να δει τις ερωτήσεις του quiz: Ο χρήστης μπορεί να προβάλλει τις υπάρχουσες ερωτήσεις που ανήκουν στην επιλεγμένη κατηγορία.

Με αυτόν τον τρόπο, ο χρήστης έχει την ελευθερία να επιλέξει πώς θέλει να αλληλεπιδράσει με το quiz, είτε παίζοντας το, είτε συνεισφέροντας με νέες ερωτήσεις, είτε εξερευνώντας το περιεχόμενο του.

Οι κατηγορίες των ερωτήσεων έχουν δημιουργηθεί σε μορφή κουμπιών, οι οποίες σε παραπέμπουν στις ανάλογες οθόνες. Σε κάθε μια από αυτές τις οθόνες γίνεται έλεγχος για το αν έχει ξεκινήσει το quiz ή όχι (αν ο χρήστης έχει πατήσει την επιλογή να παίξει ή όχι).

Σε περίπτωση που δεν έχει ξεκινήσει το παιχνίδι καλείται η οθόνη “QuizSettingsScreen” το οποίο είναι υπεύθυνο για να προβάλλει όλες τις επιλογές που αναφέρθηκαν προηγουμένως.

```
import { View, Text, StyleSheet } from "react-native";
import React from "react";
import { useNavigation } from "@react-navigation/native";
import Button from "../../components/ui/Button";

function QuizSettingsScreen({ setStartQuiz, type }) {
  const navigation = useNavigation();

  // navigate the user to the appropriate screen, depending on which button presses
  const viewAllQuestions = () =>
    navigation.navigate("AllQuestions", { type: type });
  const startQuiz = () => setStartQuiz(true);
  const createQuestion = () =>
    navigation.navigate("CreateQuestion", { type: type });

  return (
    <View style={styles.wrapper}>
      <Text style={styles.title}>
        Please select one of the following options.
      </Text>

      <View style={styles.buttons}>
        <Button
          onPress={startQuiz}
          extraStyle={{
            button: {
              minWidth: 300,
              marginBottom: 25,
            },
          }}
        >
          Start quiz
        </Button>
        <Button
          onPress={createQuestion}
          extraStyle={{
            button: {
              minWidth: 300,
              marginBottom: 25,
            },
          }}
        >
          Add a new question
        </Button>
        <Button
          onPress={viewAllQuestions}
          extraStyle={{
            button: {
              minWidth: 300,
            },
          }}
        >
          View all questions
        </Button>
      </View>
    </View>
  );
}

export default QuizSettingsScreen;
```

Εικόνα 32-Στιγμιότυπο κώδικα για τις επιλογές του χρήστη

Μόλις επιλεχτεί μια από τις επιλογές, προκύπτει και το αντίστοιχο αποτέλεσμα.

```
const viewAllQuestions = () =>
  navigation.navigate("AllQuestions", { type: type });
const startQuiz = () => setStartQuiz(true);
const createQuestion = () =>
  navigation.navigate("CreateQuestion", { type: type });
```

Εικόνα 33-Στιγμιότυπο κώδικα πλοήγησης επιλογών

Με την επιλογή της προβολής των ερωτήσεων, καλείται η συνάρτηση “viewAllQuestions” η οποία μας παραπέμπει στην οθόνη “AllQuestionsScreen”. Η “AllQuestionsScreen” στέλνει ένα αίτημα HTTP στην βάση δεδομένων και ανακτά τον τύπο των ερωτήσεων που πρέπει να κληθούν για προβολή.

```
const getQuestions = async (type) => {
  const data = (
    await axios.get(
      `https://quiz-dipl-default-rtdb.europe-west1.firebasedatabase.app/questions/${type}.json`
    )
  )?.data;

  if (data) setQuestions(data);
};

useEffect(() => {
  getQuestions(route.params.type);
}, [route.params.type]);

useFocusEffect(
  useCallback(() => {
    getQuestions(route.params.type);
  }, [route.params.type])
);
```

Εικόνα 34-Στιγμιότυπο κώδικα για αίτημα HTTP

Στην συνέχεια γίνεται η προβολή.

```
return (
  <View style={styles.container}>
    <ScrollView>
      {questions?.length ? (
        <View style={styles.wrapper}>
          {questions.map(({ question }, index) => {
            return (
              <View key={index} style={styles.questions}>
                <Text
                  style={styles.questionText}
                >{`\u2022 ${question}`}</Text>
              </View>
            );
          })}
        </View>
      ) : null}
    </ScrollView>
    <Button title="Back" color="#c30b64" onPress={handlePrevious} />
  </View>
);
```

Εικόνα 35-Στιγμιότυπο κώδικα προβολής ερωτήσεων

Με την επιλογή της δημιουργίας ερώτησης, καλείται η συνάρτηση “createQuestion” η οποία μας παραπέμπει στην οθόνη “CreateQuestionScreen”. Η συνάρτηση αυτή στέλνει επίσης ένα αίτημα HTTP στην βάση δεδομένων και ανακτά τα δεδομένα.

```
const data = (
  await axios.get(
    `https://quiz-dipl-default-rtdb.europe-west1.firebasedatabase.app/questions.json`
  )
)?.data;
```

Εικόνα 36-Στιγμιότυπο κώδικα ανάκτησης δεδομένων

Εμφανίζει πλαίσια κειμένων στα οποία ο χρήστης μπορεί να εισάγει ερωτήσεις, και απαντήσεις.

```
<View>
  <Text style={styles.title}>Please fill in the fields below</Text>
  <Input
    label="Question"
    onUpdateValue={(value) => setQuestionText(value)}
    value={questionText}
    extraStyle={{
      text: { color: Colors.primary800 },
      inputContainer: { minWidth: "90%", maxWidth: "90%" },
    }}
  />

  <Input
    label="Correct answer"
    onUpdateValue={(value) => setCorrectAnswer(value)}
    value={correctAnswer}
    extraStyle={{
      text: { color: Colors.primary800 },
      inputContainer: { minWidth: "90%", maxWidth: "90%" },
    }}
  />

  <Input
    label="First false answer"
    onUpdateValue={(value) => setFalseAnswerA(value)}
    value={falseAnswerA}
    extraStyle={{
      text: { color: Colors.primary800 },
      inputContainer: { minWidth: "90%", maxWidth: "90%" },
    }}
  />

  <Input
    label="Second false answer"
    onUpdateValue={(value) => setFalseAnswerB(value)}
    value={falseAnswerB}
    extraStyle={{
      text: { color: Colors.primary800 },
```

Εικόνα 37-Στιγμιότυπο κώδικα δημιουργίας ερωτήσεων

Κατανέμει τις απαντήσεις των ερωτήσεων σε τυχαία σειρά.

```
const updatedData = data?.[route.params.type]
  ? [...data?.[route.params.type]]
  : [];
let options = [];

const randomlyOrderedNames = ["A", "B", "C", "D"]
  .map((value) => ({ value, sort: Math.random() }))
  .sort((a, b) => a.sort - b.sort)
  .map(({ value }) => value);

options.push({
  name: randomlyOrderedNames[0],
  text: correctAnswer,
});

options.push({
  name: randomlyOrderedNames[1],
  text: falseAnswerA,
});

if (falseAnswerB)
  options.push({
    name: randomlyOrderedNames[2],
    text: falseAnswerB,
  });

if (falseAnswerC)
  options.push({
    name: randomlyOrderedNames[3],
    text: falseAnswerC,
  });

options.sort(compareNames);
```

Εικόνα 38-Στιγμιότυπο κώδικα τυχαίας κατανομής ερωτήσεων

Τέλος, ενημερώνει τις αλλαγές που έχουν δημιουργηθεί στην βάση δεδομένων.

```
updatedData.push({
  question: questionText,
  options: options,
  correctAnswerName: randomlyOrderedNames[0],
});

// update the db with the new array of questions
const res = await axios.put(
  `https://quiz-dipl-default-rtdb.europe-west1.firebasedatabase.app/questions.json`,
  { ...data, [route.params.type]: updatedData }
);
```

Εικόνα 39-Στιγμιότυπο κώδικα ενημέρωσης βάσης δεδομένων 1

Σε περίπτωση που ο χρήστης επιλέξει να ξεκινήσει το quiz γίνονται τα ακόλουθα:

- Το σύστημα ανακτά τις ερωτήσεις της συγκεκριμένης κατηγορίας στέλνοντας ένα αίτημα HTTP στην βάση δεδομένων.

```
const getQuestions = async (type) => {
  const data = (
    await axios.get(
      `https://quiz-dipl-default-rtdb.europe-west1.firebasedatabase.app/questions/${type}.json`
    )
  )?.data;
```

Εικόνα 40-Στιγμιότυπο κώδικα ανάκτησης ερωτήσεων

- Επιλέγει τυχαία τις πέντε ερωτήσεις που θα προβληθούν στο χρήστη.

```
if (data) {
  const randomlyOrderedQuestions = data
    .map((value) => ({ value, sort: Math.random() }))
    .sort((a, b) => a.sort - b.sort)
    .map(({ value }) => value);
  const first5RandomQuestions = randomlyOrderedQuestions.slice(0, 5);

  setQuestions(first5RandomQuestions);
  if (noQuestionsExist) setNoQuestionsExist(false);
} else if (!noQuestionsExist) {
  // if there are not any questions in the db, then the user should see an appropriate message
  setNoQuestionsExist(true);
  setStartQuiz(false);
}
```

Εικόνα 41-Στιγμιότυπο κώδικα τυχαίας επιλογής ερωτήσεων

- Ξεκινάει να λειτουργεί ο χρόνος που έχει δημιουργηθεί με την χρήση της “useEffect()” η οποία είναι μια συνάρτηση της React που εκτελείται αυτόματα μετά από κάθε αλλαγή μιας από τις μεταβλητής που δηλώνεται. Στην προκειμένη περίπτωση μόλις ο χρήστης επιλέγει να παίξει, η “useEffect()” ενεργοποιείται και εκτελεί τις λειτουργίες που της έχουν ανατεθεί. Δηλαδή, αρχικοποιεί τον μετρητή σε δεκαπέντε(15) δευτερόλεπτα και μειώνει τον χρόνο αυτό μετά από ένα διάστημα 1000ms (1 δευτερόλεπτο) με την χρήση της “setTimeout”.

```
useEffect(() => {
  if (!startQuiz) return;

  const myInterval = () => {
    if (counter > 0) {
      setCounter((state) => state - 1);
    } else {
      setIndex(index + 1);
      setCounter(15);
    }
  };

  interval = setTimeout(myInterval, 1000);

  // Clean up
  return () => clearTimeout(interval);
}, [counter, index, startQuiz]);
```

Εικόνα 42-Στιγμιότυπο κώδικα χρόνου

- Γίνεται αύξηση των πόντων του χρήστη κάθε φορά που απαντάει σωστά μια από τις ερωτήσεις.

```
if (selectedAnswerName === questions[index]?.correctAnswerName) {
  setPoints((prevPoints) => {
    const updatedPoints = prevPoints + 10;
    return updatedPoints;
  });
```

Εικόνα 43-Στιγμιότυπο κώδικα ενημέρωσης πόντων

- Εμφανίζει μια μπάρα με το ποσοστό των ερωτήσεων που έχει απαντήσει μέχρι τώρα ο χρήστης.

```
<View
  style={{
    backgroundColor: "white",
    width: "100%",
    flexDirection: "row",
    alignItems: "center",
    height: 10,
    borderRadius: 20,
    justifyContent: "center",
    marginTop: 20,
    marginLeft: 10,
  }}
>
  <Text
    style={{
      backgroundColor: "#FFC0CB",
      borderRadius: 12,
      position: "absolute",
      left: 0,
      height: 10,
      right: 0,
      width: `${Math.floor((index / questions?.length) * 100)}%`,
      marginTop: 20,
    }}
  />
</View>
```

Εικόνα 44-Στιγμιότυπο κώδικα μπάρα προόδου

- Ελέγχει τις απαντήσεις το χρήστη.

```
<Pressable
  key={name}
  onPress={() =>
    selectedAnswerName === null && setSelectedAnswerName(name)
  }
  style={
    selectedAnswerName === name &&
    name === questions[index].correctAnswerName
      ? {
          flexDirection: "row",
          alignItems: "center",
          borderWidth: 0.5,
          borderColor: "#00FFFF",
          marginVertical: 10,
          backgroundColor: "green",
          borderRadius: 20,
        }
      : selectedAnswerName === name
      ? {
          flexDirection: "row",
          alignItems: "center",
          borderWidth: 0.5,
          borderColor: "#00FFFF",
          marginVertical: 10,
          backgroundColor: "red",
          borderRadius: 20,
        }
      : {
          flexDirection: "row",
          alignItems: "center",
          borderWidth: 0.5,
          borderColor: "#00FFFF",
          marginVertical: 10,
          borderRadius: 20,
        }
  }
>
```

Εικόνα 45-Στιγμιότυπο κώδικα ελέγχου απαντήσεων

4.1.4.5 Οθόνες Αποτελεσμάτων Quiz

Στο τέλος κάθε παιχνιδιού εμφανίζονται οι οθόνες αποτελεσμάτων. Κάθε κατηγορία παιχνιδιού έχει και την δικιά της οθόνη αποτελεσμάτων.

Κάθε τέτοια οθόνη προβάλλει:

- Τους συνολικούς πόντους που συγκέντρωσε ο χρήστης από τις σωστές απαντήσεις, καθώς και ενημερώνει την βάση δεδομένων για τους συνολικούς πόντους που έχει συγκεντρώσει ο χρήστης από κάθε κατηγορία.

```
<Text style={styles.pointsText}>
  You scored{" "}
  <Text style={styles.pointsContainer}>{route.params.points}</Text>{" "}
  points!
</Text>
```

Εικόνα 46-Στιγμιότυπο κώδικα προβολής πόντων

```
const updateTotalPoints = async () => {
  const data = (
    await axios.get(
      `https://quiz-dipl-default-rtdb.europe-west1.firebasedatabase.app/questions.json/`
    )
  )?.data;

  let updatedPoints = route.params.points;
  if (data?.points) updatedPoints = updatedPoints + data?.points;

  axios.put(
    `https://quiz-dipl-default-rtdb.europe-west1.firebasedatabase.app/questions.json`,
    { ...data, points: updatedPoints }
  );
};
```

Εικόνα 47-Στιγμιότυπο κώδικα ενημέρωσης βάσης δεδομένων 2

- Τις σωστές απαντήσεις.

```
const getFormattedQuestionsWithAnswers = (questions) => {
  return questions.map(({ question, options, correctAnswerName }) => {
    return {
      question,
      answer: options.find((option) => option.name === correctAnswerName)
        ?.text,
    };
  });
};
```

Εικόνα 48-Στιγμιότυπο κώδικα προβολής απαντήσεων

- Το ταμπλό σωστών/λάθος απαντήσεων.

```
<FlatList
  numColumns={2}
  data={route.params.answers}
  renderItem={({ item }) => (
    <View style={styles.flatlistContainer}>
      <Text>{item.question}</Text>
      {item.answer === true ? (
        <AntDesign
          name="checkcircle"
          size={20}
          color="green"
          style={styles.answerIcon}
        />
      ) : (
        <AntDesign
          name="closecircle"
          size={20}
          color="red"
          style={styles.answerIcon}
        />
      )}
    </View>
  )}
  keyExtractor={(item, index) => index.toString()}
/>
```

Εικόνα 49-Στιγμιότυπο κώδικα προβολής ταμπλό απαντήσεων

- Την επιλογή να κοινοποιήσει ο χρήστης τους πόντους του σε οποιαδήποτε εφαρμογή επιθυμεί.

```
const handleShare = () => {
  const message = `I scored ${route.params.points} points in the quiz!`;
  Share.share({
    message,
  });
};
```

Εικόνα 50-Στιγμιότυπο κώδικα συνάρτησης κοινοποίησης

- Την επιλογή να ξαναπαίξει την ίδια κατηγορία παιχνιδιού.

```
<Pressable
  onPress={() => navigation.navigate("QuizOverview")}
  style={styles.retryButton}
>
  <Text style={styles.retryButtonText}>Retry Quiz</Text>
</Pressable>
```

Εικόνα 51-Στιγμιότυπο κώδικα κουμπιού επανάληψης παιχνιδιού

- Την επιλογή να επιλέξει μια άλλη κατηγορία.

```
<Pressable
  onPress={() => navigation.navigate("Categories")}
  style={styles.retryButton}
>
  <Text style={styles.retryButtonText}>Try another Quiz</Text>
</Pressable>
```

Εικόνα 52-Στιγμιότυπο κώδικα πλοήγησης κουμπιού στην οθόνη κατηγοριών

4.1.4.6 Οθόνη Profile

Στην οθόνη Λογαριασμού προβάλλονται τα εξής χαρακτηριστικά:

- Συνολικοί πόντοι από όλες τις κατηγορίες των παιχνιδιών που έχει παίξει.

```
const getUserPoints = async () => {
  const data = (
    await axios.get(
      `https://quiz-dipl-default-rtdb.europe-west1.firebasedatabase.app/questions/points.json`
    )
  )?.data;

  // set the retrieved points so that user can see them in the page
  setPoints(data);
};
```

Εικόνα 53-Στιγμιότυπο κώδικα προβολής συνολικών πόντων

- Την επιλογή να καθοδηγηθεί στην οθόνη που μπορεί να εισάγει γραπτώς τους στόχους που θέλει να πετύχει ή/και να προβληθούν οι στόχοι που έχει ήδη εισάγει.

```
  <Text style={styles.bio}>
    After everything you learned, will you add some new goals?
  </Text>
</View>

<Pressable
  onPress={() => navigation.navigate("Goals")}
  style={{
    backgroundColor: Colors.primary500,
    padding: 14,
    marginLeft: "auto",
    marginRight: "auto",
    marginBottom: 20,
    borderRadius: 5,
  }}
>
  <Text style={{ color: "white", textAlign: "center" }}>
    ADD NEW GOALS
  </Text>
</Pressable>
```

Εικόνα 54-Στιγμιότυπο κώδικα καθοδήγησης σε οθόνη στόχων

- Το κουμπί ανοίγματος του "Drawer" προκειμένου να δει τις υπόλοιπες επιλογές που μπορεί να επιλέξει.

4.1.4.7 Οθόνη Στόχων

Στην οθόνη στόχων προβάλλονται οι συνολικοί στόχοι που έχει προσθέσει ο χρήστης και έχει την επιλογή να προσθέσει έναν στόχο.

```
<View style={styles.container}>
  <StatusBar style="auto" />
  <View style={styles.appContainer}>
    <Button
      title="Add New Goal"
      color="#c30b64"
      onPress={startAddGoalHandler}
    />
    <GoalInput
      visible={modalIsVisible}
      onAddGoal={addGoalHandler}
      onCancel={endAddGoalHandler}
    />
    <View style={styles.goalsContainer}>
      <FlatList
        data={Goals}
        renderItem={(itemData) => {
          return (
            <GoalItem
              text={itemData.item.text}
              id={itemData.item.id}
              onDeleteItem={deleteGoalHandler}
            />
          );
        }}
        keyExtractor={(item, index) => {
          return item.id;
        }}
```

Εικόνα 55-Στιγμιότυπο κώδικα οθόνης στόχων

Η συνάρτηση "GoalItem" είναι υπεύθυνη να εμφανίσει τους στόχους σε συγκεκριμένη μορφοποίηση έτσι ώστε όταν ένας από τους στόχους να πατηθεί, να ενημερωθεί η κατάλληλη συνάρτηση.

```
<View style={styles.goalItem}>
  <Pressable
    android_ripple={{ color: "#210644" }}
    onPress={props.onDeleteItem.bind(this, props.id)}
    style={({ pressed }) => pressed && styles.pressedItem}
  >
    <Text style={styles.goalText}>{props.text}</Text>
  </Pressable>
</View>
```

Εικόνα 56-Στιγμιότυπο κώδικα εμφάνισης στόχων

Κάθε στόχος που έχει εισαχθεί διαγράφεται σε περίπτωση που πατηθεί από τον χρήστη μέσω της συνάρτησης " deleteGoalHandler".

```
function deleteGoalHandler(id) {
  setGoals((currentGoals) => {
    return currentGoals.filter((goal) => goal.id !== id);
  });
}
```

Εικόνα 57-Στιγμιότυπο κώδικα διαγραφής στόχων

Μόλις ο χρήστης πατήσει το κουμπί προσθήκης στόχων εμφανίζεται ένα νέο παράθυρο στο οποίο μπορεί να εισάγει τους στόχους που επιθυμεί καλώντας την συνάρτηση "GoalInput". Σε περίπτωση που ο χρήστης δεν εισάγει τίποτα τότε εμφανίζεται το κατάλληλο μήνυμα.

```
<Modal visible={props.visible} animationType="slide">
  <View style={styles.inputContainer}>
    <TextInput
      style={styles.textInput}
      placeholder="Add your goals here!"
      onChangeText={goalInputHandler}
      value={enteredGoalText}
    />
    {errorMessage !== "" && (
      <Text style={styles.errorMessage}>{errorMessage}</Text>
    )}
    <View style={styles.buttonContainer}>
      <View style={styles.button}>
        <Button title="Add Goal" onPress={addGoalHandler} color="#b180f0" />
      </View>
      <View style={styles.button}>
        <Button title="Cancel" onPress={props.onCancel} color="#f31282" />
      </View>
    </View>
  </View>
</Modal>
```

Εικόνα 58-Στιγμιότυπο κώδικα πλαισίου εισαγωγής στόχων

4.1.4.8 Οθόνη Δραστηριοτήτων

Σε αυτή την οθόνη εμφανίζονται δύο τμήματα. Το πρώτο τμήμα αναφέρεται στην καταμέτρηση των λίτρων νερού που καταναλώνονται και το δεύτερο τμήμα αναφέρεται στην καταμέτρηση των θερμίδων που καταναλώνονται.

Το πρώτο περιλαμβάνει έναν μετρητή που αντιπροσωπεύει τον αριθμό των λίτρων νερού που έχουν

καταναλωθεί. Ο χρήστης μπορεί να αυξήσει τον μετρητή πατώντας το κουμπί "+500ml" ή να μειώσει τον μετρητή πατώντας το κουμπί "-500ml". Υπάρχει επίσης ένα πεδίο κειμένου όπου ο χρήστης μπορεί να εισάγει τον ημερήσιο στόχο κατανάλωσης σε λίτρα. Ακόμα, υπάρχει ένα κουμπί που θέτει μια ειδοποίηση για να υπενθυμίσει στον χρήστη να πιεί νερό μετά από μια ώρα.

```
<View style={styles.postContainer}>
  <Text style={styles.countText}>
    Water Count: {count}/{goal} ml
  </Text>

  <View style={styles.inputContainer}>
    <TextInput
      style={styles.input}
      placeholder="Enter your daily water goal"
      keyboardType="numeric"
      onChangeText={(text) => setGoal(parseInt(text))}
    />
  </View>

  <View style={styles.buttonContainer}>
    <TouchableOpacity style={styles.button} onPress={incrementCount}>
      <Text style={styles.buttonText}>+500ml</Text>
    </TouchableOpacity>
    <TouchableOpacity style={styles.button} onPress={decrementCount}>
      <Text style={styles.buttonText}>-500ml</Text>
    </TouchableOpacity>
  </View>

  <Button
    title={"Remind me in 1 hour"}
    onPress={scheduleNotificationHandler}
  />
</View>
```

Εικόνα 59-Στιγμιότυπο κώδικα προβολής μετρητή νερού

Το δεύτερο τμήμα περιλαμβάνει έναν μετρητή θερμίδων που αντιπροσωπεύει τον αριθμό των θερμίδων που έχουν καταναλωθεί. Ο χρήστης μπορεί να εισάγει τον στόχο του για την ημέρα σε θερμίδες και να εισάγει τον αριθμό των θερμίδων καταναλώνει. Ο χρήστης μπορεί επίσης να επαναφέρει τον στόχο θερμίδων στην αρχική τιμή. Υπάρχει επίσης ένα κουμπί που ονομάζεται "Scan Me!" και μεταφέρει τον χρήστη σε μια άλλη οθόνη που θα αναλύσουμε παρακάτω.

```
<View style={styles.postContainer}>
  <Text style={styles.countText}>
    Calorie Count: {calorieCount}/{calorieGoal} cal
  </Text>

  <View style={styles.inputCalorieContainer}>
    <TextInput
      style={styles.input}
      placeholder="Enter your daily calorie goal"
      keyboardType="numeric"
      onChangeText={updateCalorieGoal}
    />
    <TextInput
      style={styles.input}
      placeholder="Enter consumed calories"
      keyboardType="numeric"
      onChangeText={updateCalorieCount}
      onSubmitEditing={addCalories}
      value={consumedCalories}
      clearButtonMode="while-editing"
    />
    <TouchableOpacity
      style={styles.buttonReset}
      onPress={resetCalorieGoal}
    >
      <Text style={styles.buttonResetText}>Reset</Text>
    </TouchableOpacity>
  </View>
</View>
```

Εικόνα 60-Στιγμιότυπο κώδικα προβολής μετρητή θερμίδων

Κάθε φορά που ο χρήστης εισάγει θερμίδες ή κάθε φορά που γίνεται σάρωση κάποιου αντικειμένου μέσω του κουμπιού "Scan Me!", οι θερμίδες προστίθενται.

```
const updateCalorieCount = (text) => {
  setConsumedCalories(text);
};

const addCalories = () => {
  const enteredCalories = parseInt(consumedCalories);
  setConsumedCalories("");
  setCalorieCount((prevCount) => prevCount + enteredCalories);
  if (enteredCalories >= calorieGoal) {
    setBottomSheetVisible(true);
  }
};
```

Εικόνα 61-Στιγμιότυπο κώδικα προσθήκης θερμίδων

Τέλος, μόλις ο χρήστης φτάσει τον επιθυμητό στόχο κατανάλωσης νερού ή τον επιθυμητό στόχο κατανάλωσης θερμίδων, εμφανίζεται ένα “BottomSheet” το οποίο τον ενημερώνει.

```
<BottomSheet
  isVisible={bottomSheetVisible}
  containerStyle={styles.bottomSheetContainer}
>
  <View style={styles.bottomSheetContent}>
    {calorieCount >= calorieGoal ? (
      <Text style={styles.modalText}>
        Congratulations, you reached your goal!
      </Text>
    ) : (
      <Text style={styles.modalText}>
        Congratulations, you reached your goal!
      </Text>
    )}

    <Button
      title="Close"
      onPress={closeBottomSheet}
      buttonStyle={styles.closeButton}
      titleStyle={styles.closeButtonText}
    />
  </View>
</BottomSheet>
```

Εικόνα 62-Στιγμιότυπο κώδικα BottomSheet

4.1.4.9 Οθόνη Σάρωσης

Αυτή η οθόνη χρησιμοποιείται προκειμένου ο χρήστης να μπορεί να κάνει σάρωση των προϊόντων που καταναλώνει. Αρχικά, γίνεται έλεγχος για την άδεια πρόσβασης στην κάμερα του χρήστη προκειμένου να μπορέσει να εκτελέσει αυτή την λειτουργία.

```
const askForCameraPermission = async () => {
  const { status } = await Camera.requestCameraPermissionsAsync();
  setHasPermission(status === "granted");
};
```

Εικόνα 63-Στιγμιότυπο κώδικα άδειας χρήσης κάμερας 1

```
if (hasPermission === null) {
  return (
    <View style={styles.container}>
      <Text>Requesting camera permission...</Text>
    </View>
  );
}
if (hasPermission === false) {
  return (
    <View style={styles.container}>
      <Text style={{ margin: 10 }}>No access to camera</Text>
      <Button title={"Allow Camera"} onPress={askForCameraPermission} />
    </View>
  );
}
```

Εικόνα 64-Στιγμιότυπο κώδικα άδειας χρήσης κάμερας 2

Στη συνέχεια, η οθόνη εμφανίζεται ένα κουμπί "Start Barcode Scanning". Όταν ο χρήστης πατάει το κουμπί, ενεργοποιείται η λειτουργία σάρωσης του barcode μέσω ενός παραθύρου.

```
<View style={styles.startButton}>
  <Button
    title={"Start Barcode Scanning"}
    onPress={startBarcodeScanning}
  />
</View>
<Modal
  visible={modalVisible}
  animationType="slide"
  onRequestClose={closeModal}
>
```

Εικόνα 65-Στιγμιότυπο κώδικα κουμπιού ενεργοποίησης κάμερας

Όταν εντοπίζεται ένας γραμμωτός κωδικός, καλείται η συνάρτηση “handleBarCodeScanned” που αναλαμβάνει να λάβει τα απαραίτητα δεδομένα θρεπτικής αξίας για το σκαναρισμένο προϊόν.

```
const handleBarCodeScanned = async ({ data }) => {
  const fetchedData = await fetchNutritionData(data);

  setNutritionData(fetchedData);
  setModalVisible(true);
  setText(data);
  setStartScanning(false);
};
```

Εικόνα 66-Στιγμιότυπο κώδικα της handleBarCodeScanned συνάρτησης

Με την χρήση της συνάρτησης “ fetchNutritionData” γίνεται ανάκτηση δεδομένων της θρεπτικής αξίας για ένα προϊόν μέσω της Open Food Facts API.

```
const fetchNutritionData = async (barcode) => {
  const apiUrl = `https://world.openfoodfacts.org/api/v0/product/${barcode}.json`;

  try {
    const response = await fetch(apiUrl);
    const data = await response.json();

    if (response.ok && data.status === 1 && data.product) {
      const product = data.product;
      const nutritionData = {
        calories: product.nutriments.energy_serving,
        protein: product.nutriments.proteins_serving,
        carbs: product.nutriments.carbohydrates_serving,
        fat: product.nutriments.fat_serving,
      };
      return nutritionData;
    } else {
      console.log("Error: Product not found");
      return null;
    }
  } catch (error) {
    console.log("Error:", error);
    return null;
  }
};
```

Εικόνα 67-Στιγμιότυπο κώδικα ανάκτησης δεδομένων από API

Έπειτα, εμφανίζεται ένα παράθυρο που περιέχει τις λεπτομέρειες του σκαναρισμένου προϊόντος, όπως τον γραμμωτό κωδικό, τις θερμίδες, την πρωτεΐνη, τους υδατάνθρακες και το λίπος του προϊόντος, καθώς και δύο κουμπιά, ένα για προσθήκη του σκαναρισμένου προϊόντος στην οθόνη "Activities" με τη μεταφορά των θερμίδων του σκαναρισμένου προϊόντος και ένα για ακύρωση της διαδικασίας.

```
<Modal
  visible={modalVisible}
  animationType="slide"
  onRequestClose={closeModal}
>
  <View style={styles.modalContainer}>
    <Text style={styles.modalText}>Scanned Item Details:</Text>
    <Text>Barcode of the item: {text}</Text>
    {nutritionData ? (
      <>
        <Text>Calories: {nutritionData.calories}</Text>
        <Text>Protein: {nutritionData.protein}g</Text>
        <Text>Carbs: {nutritionData.carbs}g</Text>
        <Text>Fat: {nutritionData.fat}g</Text>
      </>
    ) : (
      <Text>No nutrition data available</Text>
    )}
    <View style={styles.buttons}>
      <Button
        title="Add"
        onPress={() => addScannedItem(nutritionData)}
        style={{ borderRadius: 20 }}
      />
      <Button title="Cancel" color="red" onPress={closeModal} />
    </View>
  </View>
</Modal>
```

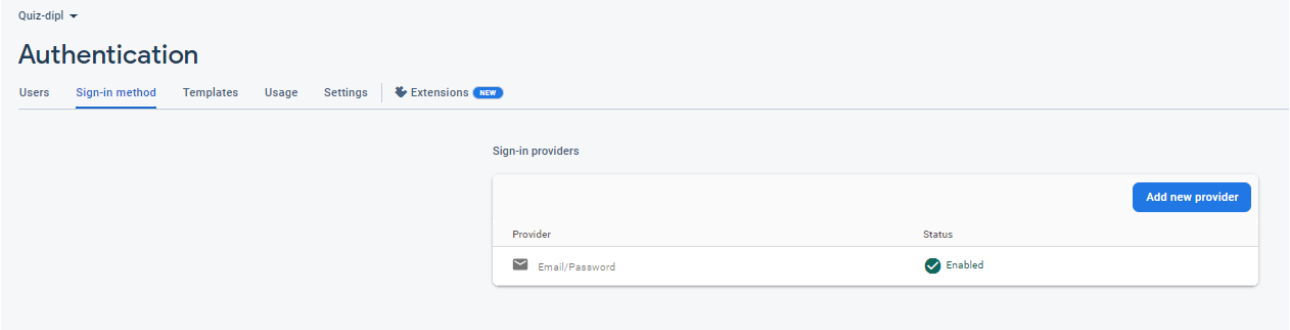
Εικόνα 68-Στιγμιότυπο κώδικα προβολής θρεπτικών αξιών

4.1.5 App.js

Η “App.js” είναι ο σημαντικότερος χώρος για την διαμόρφωση της εφαρμογής και την εισαγωγή των βασικών συστατικών που την αποτελούν. Η “App.js” αντιπροσωπεύει το κεντρικό σημείο εισόδου της εφαρμογής, καθώς από αυτήν ξεκινά η εκτέλεση της εφαρμογής. Σε αυτήν έγιναν χρήση πολλών βιβλιοθηκών, αρχικοποιήθηκαν όλες οι οθόνες που αναφέραμε προηγουμένως και κατηγοριοποιήθηκαν στις αντίστοιχες περιηγήσεις με τις κατάλληλες παραμέτρους. Σε θέματα όπως η σειρά που θα διεξαχθούν οι οθόνες, ο τρόπος σχεδίασης/εμφάνισης και τα ονόματα της κάθε συνάρτησης, ευθύνεται η App.js.

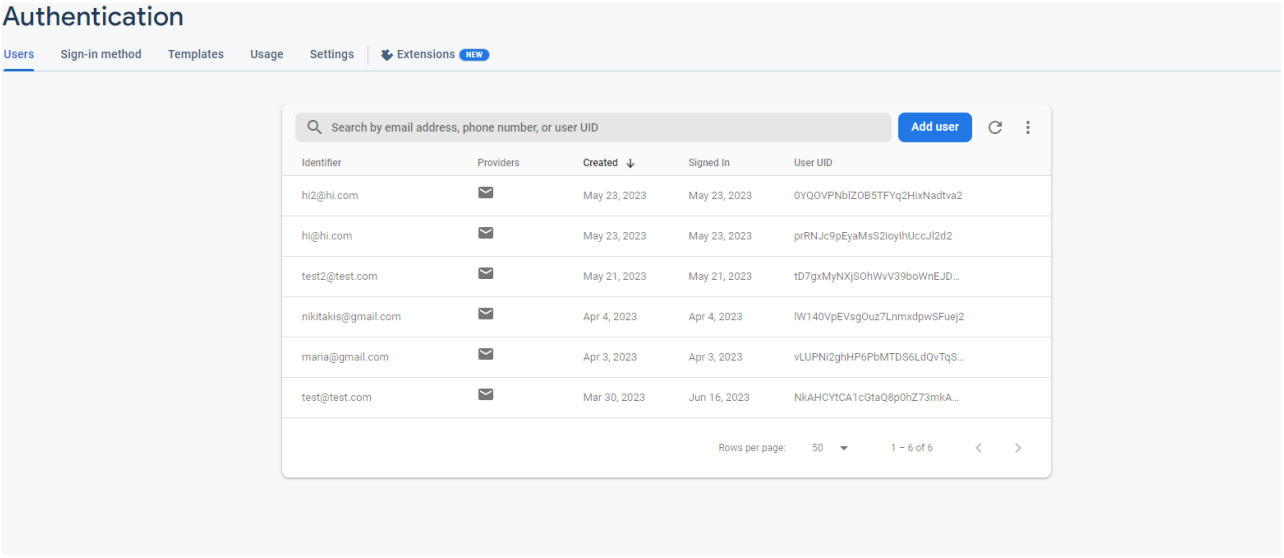
4.2 Σχεδίαση Βάσης Δεδομένων

Για την υλοποίηση του project χρησιμοποιήθηκε η βάση δεδομένων Firebase για την αποθήκευση των δεδομένων. Αρχικά, δημιούργησα έναν λογαριασμό στη Firebase από την επίσημη ιστοσελίδα της Firebase (https://firebase.google.com/), έφτιαξα ένα νέο έργο (project) με όνομα Quiz-dipl και ακολούθησα τις οδηγίες για την ολοκλήρωση του έργου. Στη συνέχεια, στον τομέα Authentication, επέλεξα την επιλογή Email/Password και ενεργοποίησα την επιλογή Email/Password προκειμένου να ενεργοποιηθούν οι λειτουργίες που περιγράφονται στην ιστοσελίδα https://firebase.google.com/docs/reference/rest/auth#section-create-email-password.



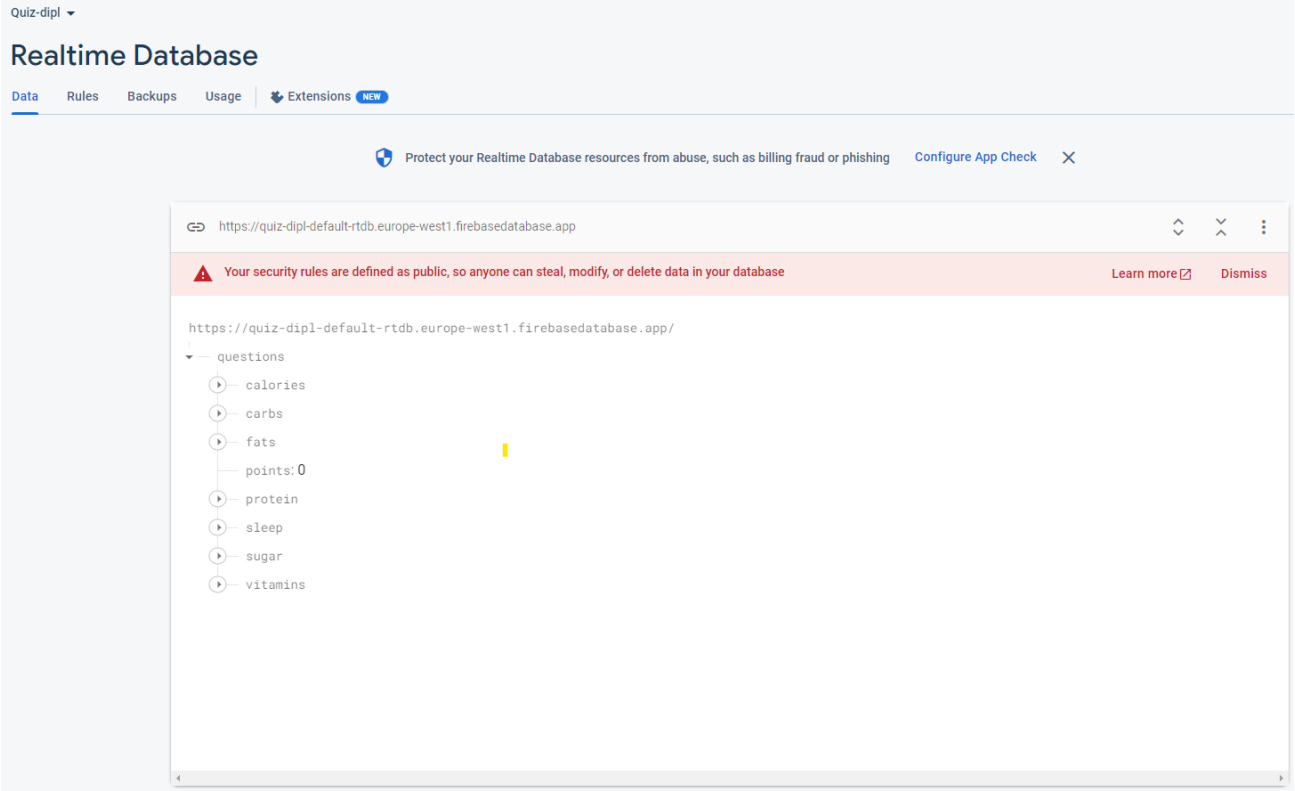
Εικόνα 69-Στιγμιότυπο ενεργοποίησης χρήσης email/password

Για την επικοινωνία μεταξύ εφαρμογής-βάσης δεδομένων χρησιμοποιήθηκε η https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY] με API_KEY να είναι το κλειδί που δίνεται σε κάθε χρήστη της Firebase στην Project Settings σελίδα. Με αυτόν τον τρόπο οι χρήστες που εγγράφονται στην εφαρμογή, να αποθηκεύονται στην βάση δεδομένων.



Εικόνα 70-Περιεχόμενο αποθήκευσης λογαριασμών χρηστών 2

Στον τομέα Real Time Database δίνεται η δυνατότητα εισαγωγής αρχείου JSON την οποία την αξιοποίησα για την αποθήκευση των πόντων του κάθε χρήστη, καθώς και των ερωτήσεων του quiz χωρισμένες σε κατηγορίες.



Εικόνα 71-Περιεχόμενο αποθήκευσης ερωτήσεων 2

Το αρχείο περιέχει κώδικα γραμμένο σε JSON όπως φαίνεται παρακάτω:

Εικόνα 72-Περιεχόμενο αρχείου JSON

4.3 Στοιχεία Επαναχρησιμοποίησης

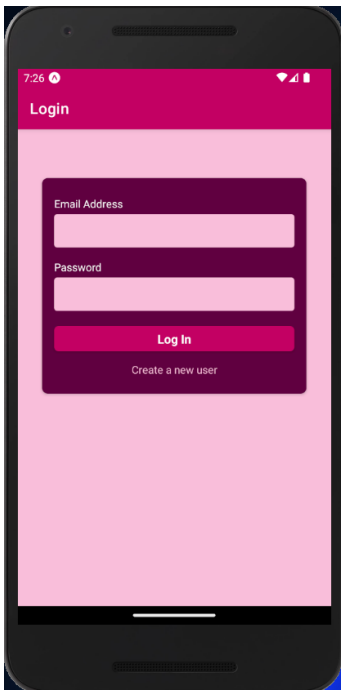
Όπως αναφέρθηκε και στο κεφάλαιο 3 η χρήση της γλώσσα React και React Native ενθαρρύνει τους προγραμματιστές να επαναχρησιμοποιήσουν εργαλεία που έχουν δημιουργηθεί από άλλους προγραμματιστές προκειμένου να επιτευχθεί η ταχύτερη υλοποίηση των εφαρμογών. Αυτά τα εργαλεία είναι έτοιμοι κώδικες που μπορούν να χρησιμοποιηθούν κατεβάζοντας third-party βιβλιοθήκες. Εκτός από την χρήση αυτών των εργαλείων, πρέπει να αναφέρω πως για την συγγραφή και υλοποίηση της εργασίας σημαντικό ρόλο συνέβαλλαν έτοιμα στοιχεία από κώδικες στο internet οι οποίοι αποτέλεσαν έμπνευση στο δικό μου έργο. Μερική επαναχρησιμοποίηση και τροποποίηση αυτών έγινε από σελίδες που έχω αναφέρει στη βιβλιογραφία. [11][12][13]

Κεφάλαιο 5: Εξαγωγή Αποτελεσμάτων

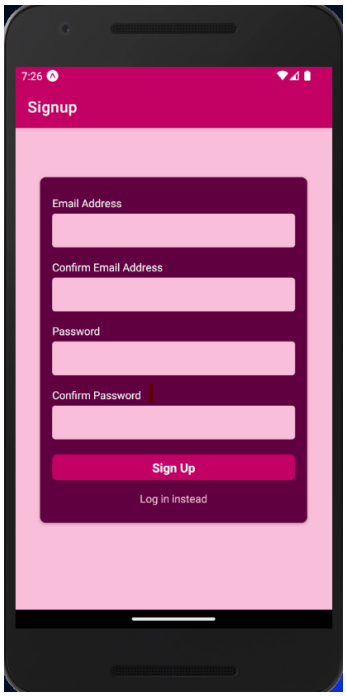
Σε αυτό το κεφάλαιο θα παρουσιαστούν και θα αναλυθούν τα αποτελέσματα όλης της εφαρμογής. Για την καλύτερη κατανόηση θα χρησιμοποιηθούν στιγμιότυπα από την εφαρμογή.

5.1 Αρχική οθόνη

Η πρώτη σελίδα που συναντά ο χρήστης είναι αυτή που περιλαμβάνει τα πεδία συμπλήρωσης στοιχείων σύνδεσης και εγγραφής.



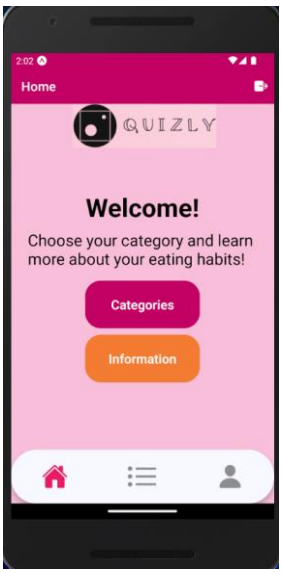
Εικόνα 1-Οθόνη LoginScreen



Εικόνα 2-Οθόνη SignupScreen

5.2 Οθόνη καλωσορίσματος

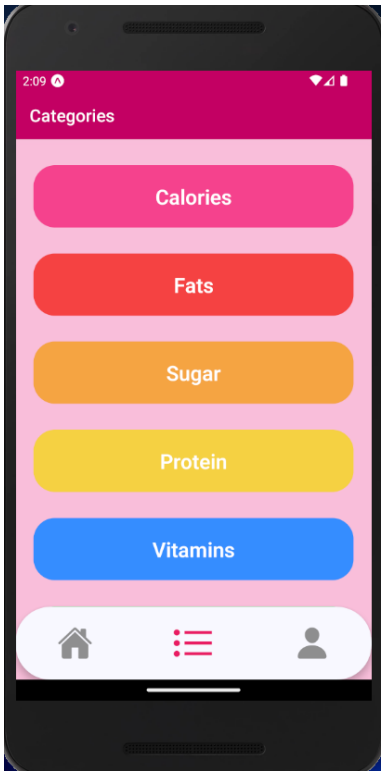
Αφού συνδεθεί, εμφανίζεται η οθόνη καλωσορίσματος η οποία λειτουργεί σαν το κεντρικό μενού της εφαρμογής.



Εικόνα 73-Οθόνη καλωσορίσματος

5.3 Οθόνη κατηγοριών

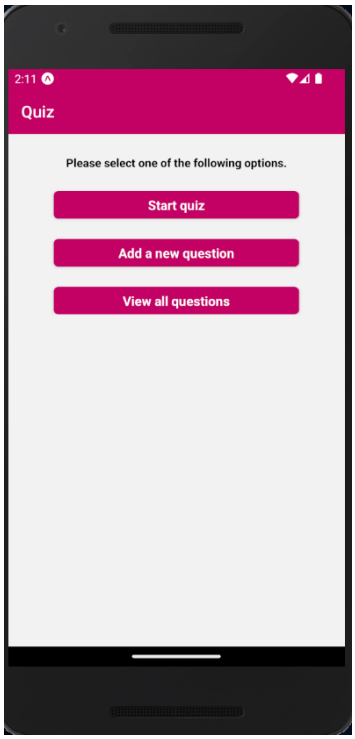
Η οθόνη αυτή προβάλλει τις κατηγορίες των παιχνιδιών quiz που μπορεί να επιλέξει ο χρήστης.



Εικόνα 74-Οθόνη κατηγοριών

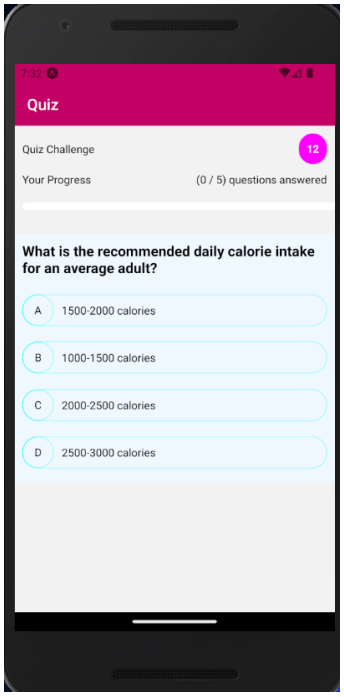
5.4 Οθόνες Quiz

Οι οθόνες αυτές προβάλλουν τρεις επιλογές.



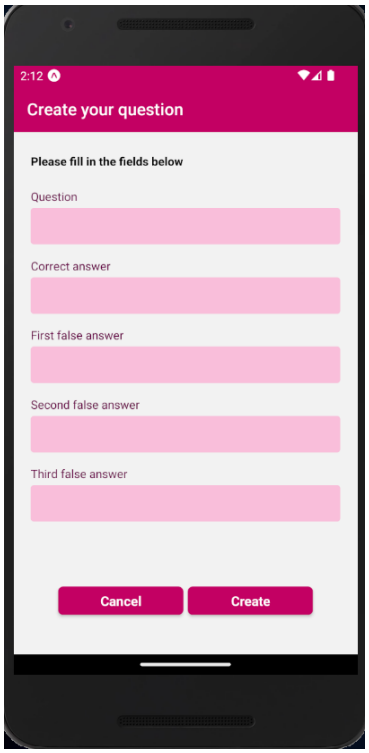
Εικόνα 75-Οθόνη επιλογών

- Η πρώτη επιλογή αφορά την έναρξη παιχνιδιού.



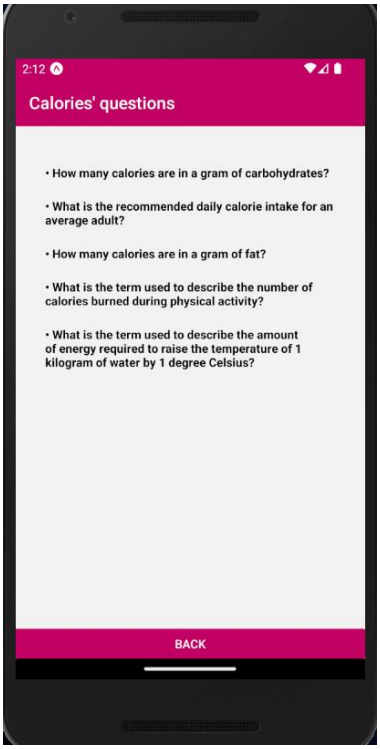
Εικόνα 3-Οθόνη παιχνιδιού quiz

- Η δεύτερη επιλογή αφορά την δημιουργία μίας ερώτησης και τις απαντήσεις της, προκειμένου να ενταχθεί στην συγκεκριμένη κατηγορία ερωτήσεων.



Εικόνα 76-Οθόνη δημιουργίας ερωτήσεων

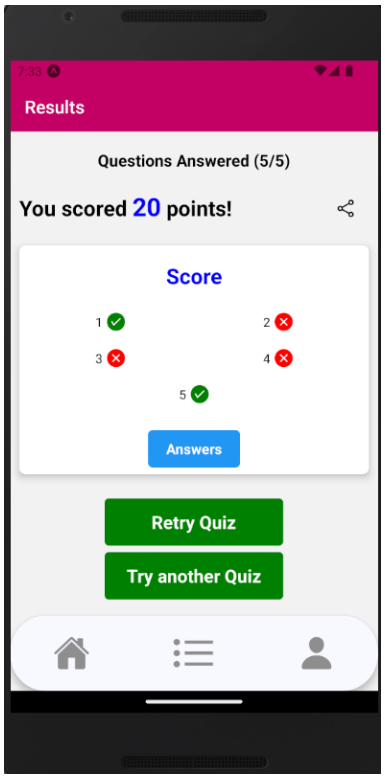
- Η τρίτη επιλογή εμφανίζει όλες τις ερωτήσεις που βρίσκονται στην επιλεγμένη κατηγορία.



Εικόνα 77-Οθόνη προβολής ερωτήσεων

5.5 Οθόνες Αποτελεσμάτων

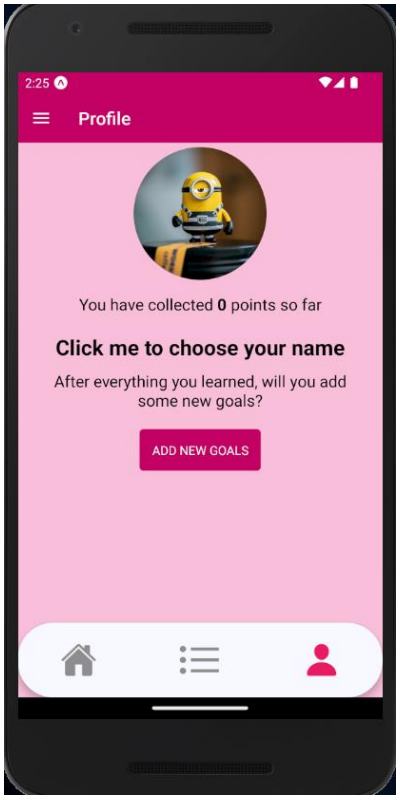
Στο τέλος κάθε παιχνιδιού quiz εμφανίζονται οι οθόνες αποτελεσμάτων οι οποίες προβάλλουν τα αποτελέσματα των απαντήσεων, τις σωστές απαντήσεις, τους συνολικούς πόντους που συγκεντρώθηκαν καθώς και την επιλογή κοινοποίησης.



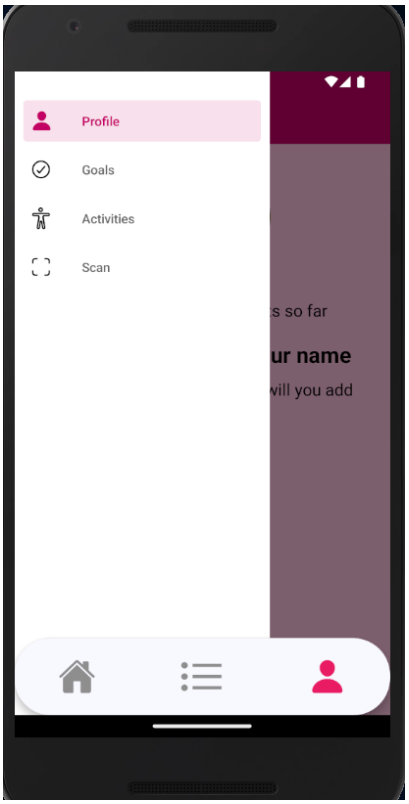
Εικόνα 4-Οθόνη Αποτελεσμάτων quiz

5.6 Οθόνη Profile

Στην οθόνη Profile, ο χρήστης συναντά τους συνολικούς πόντους που έχει συγκεντρώσει, το κουμπί ‘‘συρταριού’’ καθώς και την επιλογή μετάβασης στην οθόνη στόχων.



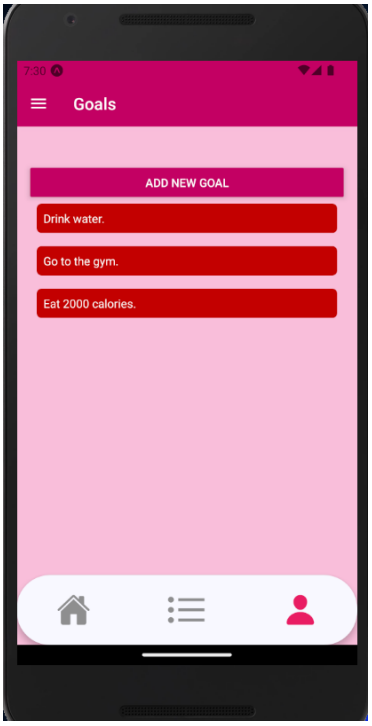
Εικόνα 78-Οθόνη Profile 1



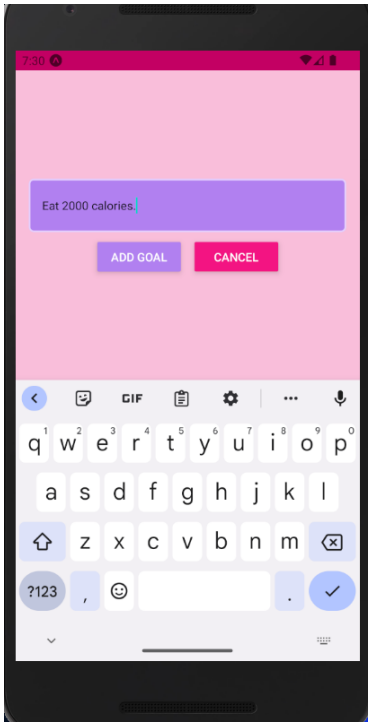
Εικόνα 79-Οθόνη Profile 2

5.7 Οθόνη Στόχων

Η οθόνη αυτή προβάλλει τους στόχους που έχει θέσει ο χρήστης καθώς και την επιλογή προσθήκης νέων.



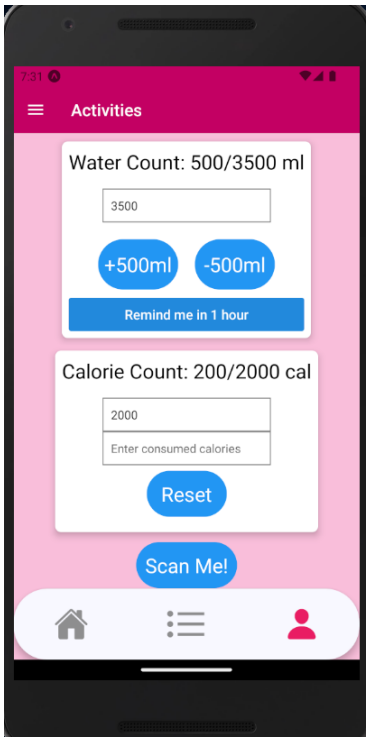
Εικόνα 5-Οθόνη καταγραφής στόχων



Εικόνα 6-Οθόνη προβολής στόχων

5.8 Οθόνη Δραστηριοτήτων

Η οθόνη δραστηριοτήτων προβάλλει τον μετρητή νερού και θερμίδων καθώς και την επιλογή μετάβασης στην οθόνη σάρωσης.

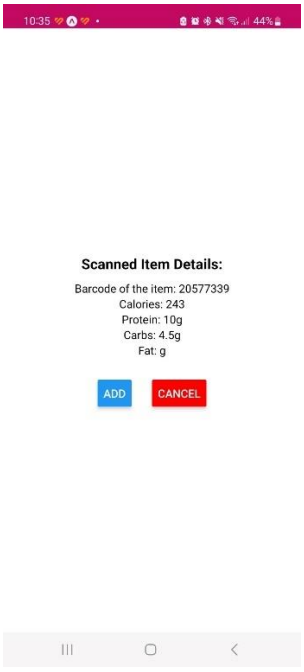


Εικόνα 7-Οθόνη προβολής μετρητών

5.9 Οθόνη Σάρωσης

Η οθόνη σάρωσης περιέχει ένα κουμπί με το οποίο ενεργοποιείται η κάμερα του smartphone και εμφανίζει τα αποτελέσματα των σκαναρισμένων προϊόντων.

Εικόνα 8-Οθόνη ενεργοποίησης κάμερας



Εικόνα 9-Οθόνη προβολής διατροφικών στοιχείων

Κεφάλαιο 6: Επίλογος

Η πτυχιακή εργασία αποτελεί μια πολύτιμη προσθήκη στον τομέα της ψηφιακής υγείας και παρέχει ένα εργαλείο που μπορεί να επιτρέψει στους χρήστες να λαμβάνουν διατροφικές πληροφορίες και να παρακολουθούν την υγιεινή τους κατανάλωση. Μέσω αυτής της εργασίας, αναδεικνύεται η σημασία της τεχνολογίας στον τομέα της υγείας που μπορεί να συμβάλλει στην ενημέρωση και την προώθηση ενός υγιεινού τρόπου ζωής. Μελλοντικά, μπορεί να επεκταθεί η λειτουργικότητα της εφαρμογής, προσθέτοντας νέες λειτουργίες βάσει των αναγκών και των στόχων των χρηστών.

Μελλοντική Επέκταση

Ολοκληρώνοντας και αναλύοντας την συγκεκριμένη διπλωματική, δημιουργείται η ευκαιρία για περαιτέρω έρευνα και επέκταση επί του θέματος. Μια ενδιαφέρουσα επέκταση θα ήταν η προσθήκη περισσότερων πληροφοριών για τα σκαναρισμένα προϊόντα χρησιμοποιώντας μια API που να αναλύει τα συστατικά του προϊόντος, αλλεργιογόνα, βιταμίνες και μέταλλα που περιέχει. Αυτό θα βοηθούσε τους χρήστες να πάρουν περισσότερες πληροφορίες για τη διατροφή τους και να κάνουν πιο ενημερωμένες επιλογές. Μια επιπλέον επέκταση θα ήταν η δυνατότητα αποθήκευσης του ιστορικού των σκαναρισμένων προϊόντων, έτσι ώστε οι χρήστες να μπορούν να παρακολουθούν την πρόοδο της διατροφής τους, να ελέγχουν τις επιλογές τους και να αντιλαμβάνονται καλύτερα τις διατροφικές τους συνήθειες. Τέλος, το ιδανικότερο σενάριο θα περιλάμβανε επίσης ένα σύστημα κατάταξης παικτών στο οποίο οι χρήστες θα μπορούσαν να συγκεντρώνουν πόντους από τα παιχνίδια που παίζουν και να παίζουν μαζί ή ενάντιας των φίλων τους. Με όλα τα προαναφερθέντα, η εφαρμογή θα παγιώσει το ενδιαφέρον και την εμπειρία χρήσης της εφαρμογής του χρήστη.

Βιβλιογραφία

[1] Zakas, Nicholas C. "Professional JavaScript for Web Developers." John Wiley & Sons, 2019.

[2] Porcello, Eve. "Learning React: Functional Web Development with React and Redux." O'Reilly Media, 2017.

[3] https://legacy.reactjs.org/docs/introducing-jsx.html

[4] https://en.wikipedia.org/wiki/JSON

[5] https://en.wikipedia.org/wiki/Visual_Studio

[6] https://en.wikipedia.org/wiki/Android_Studio

[7] Krick, E., & Schmitt, A. Fullstack React Native: The Complete Guide to React Native & Expo, 2020.

[8] Cantelon, M., Harter, M., & Rajlich, T. Node.js in Action. Manning Publications, 2018.

[9] Koller, F. Learning Firebase: Develop the Next Killer App Using Firebase, 2020.

[10] https://world.openfoodfacts.org/discover

[11] https://github.com/academind/react-native-practical-guide-code/tree/11-auth/code/01-starting-code

[12] https://www.youtube.com/watch?v=esWUVic6IcU&ab_channel=SujanAnand

[13] https://github.com/Ericnsamba/Quizflix/tree/master/App/assets

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

ml	milliliter
κλπ.	Και λοιπά
κ.α.	και άλλα
API	Application Programming Interface
JSON	JavaScript Object Notation
PDF	Portable Document Format
UI	User Interface
IDE	Integrated Development Environment
IoT	Internet of Things
CLI	Command Line Interface
npm	node package manager

Απόδοση Ξενόγλωσσων Όρων

ΞΕΝΟΓΛΩΣΣΗ	ΑΠΟΔΟΣΗ
Back-end	Παρασκήνιο
Barcode	Γραμμωτός κωδικός
Blank	Κενό
Branch	Κλαδί
Browser	Πρόγραμμα περιήγησης
Calorie counter	Μετρητής θερμίδων
Client side	Πλευρά πελάτη
Cloud based	Βασισμένο σε Cloud
Component	Εργαλείο
Desktop widgets	Γραφικά στοιχεία υπολογιστή
Emulator	Προσομοιωτής
Framework	Δομή
Front end	Προσκήνιο
Internet	Διαδίκτυο
Logo	Λογότυπο
Merging	Συγχώνευση
Mode	Τρόπος
Package	Πακέτο
Password	Κωδικός
Project	Έργο
QR code	Κωδικός QR
Quiz	Ερωτηματολόγιο
Scanner	Σαρωτής
Scripts	Σενάρια
Server-side	Πλευρά του διακομιστή
Smartphone	Έξυπνο κινητό τηλέφωνο
State management tool	Εργαλείο διαχείρισης καταστάσεων
Template	Πρότυπο
Third-party	Τρίτων
Version control	Έλεγχος έκδοσης