ΑΝΑΠΤΥΞΗ ΥΒΡΙΔΙΚΗΣ ΕΦΑΡΜΟΓΗΣ ΕΞΕΙΔΙΚΕΥΜΕΝΩΝ ΑΘΛΗΤΙΚΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ

ΓΕΩΡΓΙΟΣ ΔΟΙΤΣΙΝΗΣ

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

11 Ιουνίου 2020

1 Ευχαριστίες

Για αρχή θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Μιχαήλ Στεφανιδάκη ο οποίος με καθοδήγησε ορθά και ήταν πάντα δίπλα μου προκειμένου να καταφέρω να φέρω εις πέρας την παρούσα εργασία. Ένα μεγάλο ευχαριστώ αξίζει στην οικογένια μου που με στηρίζει κυριολεκτικά από τα πρώτα βήματα μου έως και σήμερα με υπομονή, αγάπη, φροντίδα και συμπαράσταση.

2 Σύνοψη

Με τη μεγάλη εξάπλωση των κινητών τηλεφώνων, το μεγλύτερο μέρος του πληθυσμού, έρχεται πλέον σε επαφή με ένα πλήθος νέων εφαρμογών που λίγα χρόνια πριν ήταν διαθέσιμες μόνο μέσω προσωπικού ηλεκτρονικού υπολογιστή.

Σε αυτήν την εργασία, θα παρουσιαστεί η ανάπτυξη μιας εφαρμογής που διδάσκει το άθλημα της πυγμαχίας. Η εφαρμογή μπορεί να τρέξει σε κινητές συσκευές με λειτουργικό συστημα Android, iOS και Windows.

3 Abstract

With the large spread of mobile phones, most of the population is now in contact with a large number of new applications that a few years ago were only available on a personal computer.

This work will present the development of an application that teach the sport of boxing. The application runs on Android, iOS, and Windows operating systems.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

4 Εισαγωγή

4.1 Περίληψη

Η πυγμαχία, κοινώς μποξ είναι ένα από τα πιο δημοφιλή αγωνίσματα και μαχητικές πολεμικές τέχνες, στηρίζεται στην ικανότητα των αντιπάλων να αντικρούσουν μόνο με τις γροθιές τους ο ένας τον άλλο και μέσω εύστοχων χτυπημάτων να κερδίσουν τη μεταξύ τους αναμέτρηση.

Η παρούσα εργασία έχει ως στόχο τη δημιουργία μιάς υβριδικής εφαρμογής για Android, iOS και Windows Phone, η οποία απευθύνεται στα άτομα που θέλουν να μάθουν τις κινήσεις του αθλήματος της πυγμαχίας και να ακολουθήσουν μια καθημερινή ρουτίνα γυμναστικής και εκπαίδευσης.

4.2 Κίνητρο για τη διεξαγωγή της εργασίας

Υπάρχουν πολλές εφαρμογές που αναφέρονται στη γυμναστική και στη σωστή διατροφή, άλλωστε απο εκεί προέκυψε και η ιδέα της παρούσας εργασίας, παρόλα αυτά δεν υπάρχει κάποια παρόμοια εφαρμογή που να διδάσκει ένα άθλημα ή μία τέχνη και παράλληλα να προάγει έναν υγιεινό τρόπο ζωής.

4.3 Σκοπός και στόχοι εργασίας

Σκοπός της εργασίας είναι, αρχικά να απενοχοποιήσει το παρεξηγημένο άθλημα της πυγμαχίας από την κοινή αίσθηση ότι είναι βάρβαρο και σχετίζεται με σωματικούς τραυματισμούς αλλά και πνευματικά προβλήματα. Η εφαρμογή, έχει σκόπο να δείξει στους χρήστες ότι η πυγμαχία μπορεί να παρέχει σωματικά και πνευματικά οφέλη και να τους δώσει τη δυνατότητα να εξασκηθούν, να καλυτερέψουν τη φυσική τους κατάσταση και να χάσουν βάρος. Τέλος, τους παρέχει οπτικά διαγράμματα ώστε να βλέπουν την πρόοδο τους και έτσι να τους δίνει κίνητρο για να συνεχίσουν να τη χρησιμοποιούν.

5 Υβριδικές mobile εφαρμογές

5.1 Τι είναι μία φορητή εφαρμογή

Μία φορητή εφαρμογή (ή αλλιώς mobile app), είναι η εφαρμογή λογισμικού σχεδιασμένη να τρέχει σε smartphone, tablet και άλλες φορητές συσκευές. Είναι διαθέσιμη στο κοινό μέσω πλατφορμών διανομής εφαρμογών, οι οποίες συνήθως λειτουργούν από τον ιδιοκτήτη του φορητού λειτουργικού συστήματος, όπως το Apple App Store, Google Play, BlackBerry App World κ.λπ. Τα Mobile apps, αρχικά είχαν στόχο την προσφορά στη γενική παραγωγικότητα των χρηστών και την ανάκτηση πληροφοριών, συμπεριλαμβανομένων εφαρμογών για e-mail, ημερολόγιο, κατάλογο επαφών, χρηματιστηριακές αγορές και πληροφορίες για τον καιρό. Ωστόσο, η δημόσια ζήτηση και η διαθεσιμότητα των εργαλείων ανάπτυξης οδήγησε με γρήγορους ρυθμούς σε επέκταση και άλλων κατηγοριών, όπως παιχνίδια, αυτοματισμούς εργοστασίων, GPS και location-based υπηρεσίες, banking, εξέλιξη παραγγελιών, καθώς και στις αγορές εισιτηρίων.

5.2 Τι σημαίνει υβριδική εφαρμογή

Υβριδικό, εξ ορισμού, είναι οτιδήποτε προέρχεται από ετερογενείς πηγές ή αποτελείται από στοιχεία διαφορετικών ή ασυμβίβαστων όρων. Υβριδική εφαρμογή, είναι αυτή που είναι γραμμένη με την ίδια τεχνολογία που χρησιμοποιείται για ιστότοπους και εφαρμογές ιστού για κινητά και που φιλοξενείται ή τρέχει μέσα σε μία κινητή συσκευή.

Η έννοια είναι πολύ απλή. Σχεδόν κάθε mobile λειτουργικό σύστημα διαθέτει ένα API για την ανάπτυξη εφαρμογών. Αυτό το API αποτελείται από ένα στοιχείο που ονομάζεται Web View. Web View είναι συνήθως ένα πρόγραμμα περιήγησης που λειτουργεί εντός του πεδίου μιας εφαρμογής για κινητά. Αυτό το πρόγραμμα περιήγησης εκτελεί τους HTML, CSS και Javascript κώδικες. Αυτό σημαίνει, ότι μπορεί κάποιος να δημιουργήσει μια ιστοσελίδα, χρησιμοποιώντας τις προηγούμενες τεχνολογίες και στη συνέχεια να την εκτελέσει σαν μια εφαρμογή για κινητά τηλέφωνα.

Ο χρήστης μπορεί να χρησιμοποιήσει, τις ίδιες γνώσεις ανάπτυξης ιστοσελίδων για να δημιουργήσει native-hybrid κινητές εφαρμογές, (εδώ ο όρος native αναφέρεται στην εγκατάσταση ενός αρχείου συγκεκριμένης μορφής για τη συγκεκριμένη πλατφόρμα στη συσκευή αφού έχει συσκευαστεί μαζί με άλλα εργαλεία), για παράδειγμα:

- Η Android χρησιμοποιεί Android Application Package (.apk)

- Η iOS χρησιμοποιεί το iPhone Application Archive (.ipa)
- Η Windows Phone χρησιμοποιεί Application Package (.xap)

Το πακέτο / πρόγραμμα εγκατάστασης αποτελείται από ένα κομμάτι native κώδικα που προετοιμάζει την ιστοσελίδα και κάποια στοιχεία που απαιτούνται για την εμφάνιση του περιεχομένου της ιστοσελίδας.

Η έννοια της εμφάνισης μιας ιστοσελίδας μέσα στο κιβώτιο εφαρμογών για κινητά ονομάζεται Υβριδική εφαρμογή.

Παρακάτω υπάρχουν κάποια από τα πλεονεκτήματα ανάπτυξης υβριδικών εφαρμογών:

- Οι υβριδικές εφαρμογές πρόκειται ουσιαστικά για εφαρμογές ιστού που είναι αποθηκευμένες σε επίσημα καταστήματα εφαρμογών.
- Μια υβριδική εφαρμογή μπορεί να χρησιμοποιηθεί μαζί με οποιοδήποτε JavaScript framework / βιβλιοθήκη.
- Ο κώδικας είναι σε μεγάλο βαθμό κοινόχρηστος (shareable) σε όλες τις πλατφόρμες.
- Η πρόσβαση σε μητρικές (native) λειτουργίες, για παράδειγμα κάμερα, επιταχυνσιόμετρο, λίστα επαφών κ.λπ.

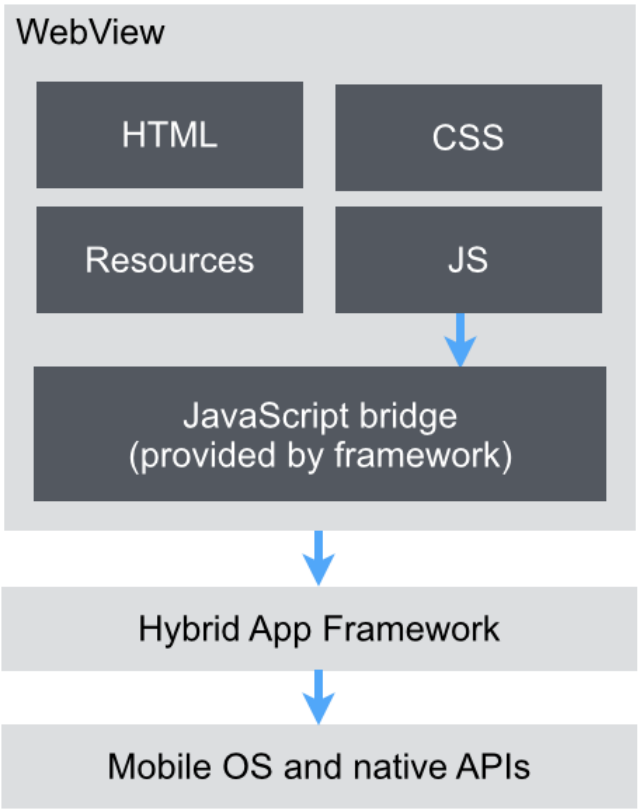
Αλλά όπως πλέονεκτηματα υπάρχουν και μειονεκτήματα:

- Αντιμετωπίζουν προβλήματα επιδόσεων και κατανάλωσης μνήμης, καθώς οι προβολές ιστού (web views) ευθύνονται για οτιδήποτε εμφανίζεται στην οθόνη.
- Πρέπει να μιμούνται όλα τα μητρικά στοιχεία διεπαφής χρήστη (UI - User Interface) πάνω από μία ενιαία προβολή ιστού.
- Είναι δυσκολότερο να γίνουν δεκτές και να δημοσιευτούν στο App Store.
- Συνήθως χρειάζονται περισσότερο χρόνο για να έχουν διαθέσιμες μητρικές λειτουργίες.
- Ορισμένες λειτουργίες ή σχέδια τους δεν υποστηρίζονται και στα δύο λειτουργικά συστήματα (Android και iOS), πράγμα που απαιτεί τροποποίηση.

5.3 Λίγα λόγια για το Apache Cordova

Το Apache Cordova, είναι ένα λογισμικό που ενώνει μια εφαρμογή Web με μια native εφαρμογή. Ο δικτυακός τόπος Apache Cordova αναφέρει ότι: ” Το Apache Cordova είναι μια πλατφόρμα για τη δημιουργία native εφαρμογών κινητής τηλεφωνίας χρησιμοποιώντας HTML, CSS και JavaScript. ”

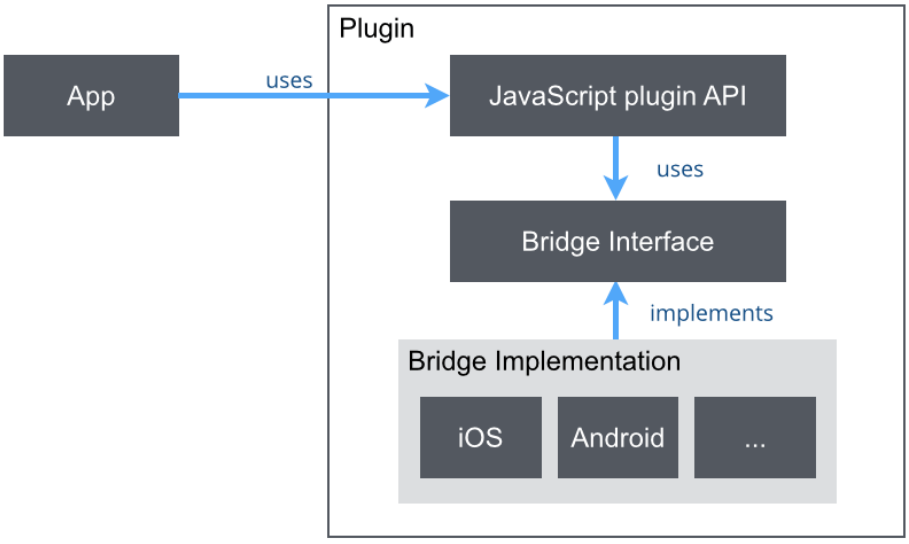
Σχήμα 1: Αρχιτεκτονική Apache Cordova.



Η προηγούμενη εικόνα απεικονίζει την αρχιτεκτονική υψηλού επιπέδου του Apache Cordova. Το Apache Cordova δεν ενώνει μόνο την εφαρμογή Web με τη native εφαρμογή, αλλά επίσης παρέχει ένα σύνολο API γραμμένο σε JavaScript για να αλληλεπιδράσει με τις native λειτουργίες της συσκευής. Μπορεί κανείς να χρησιμοποιήσει τη JavaScript για να αποκτήσει πρόσβαση στην κάμερα, να τραβήξει μια φωτογραφία ή να μιλήσει με το Bluethooth της συσκευής και να πάρει τη λίστα με τις συσκεύες στη γύρω περιοχή. Στην τελευταία περίπτωση, η Cordova διαθέτει κάποια API που αλληλεπιδρούν με το Web View χρησιμοποιώντας τη JavaScript και στη συνέχεια μιλάει στη συσκευή στη μητρική της γλώσσα, παρέχοντας έτσι μια γέφυρα μεταξύ τους.

Η παρακάτω εικόνα απεικονίζει την αρχιτεκτονική Plugin υψηλού επιπέδου:

Σχήμα 2: Αρχιτεκτονική Cordova Plugin.



Τα Plugin (πρόσθετα) πάντα αποτελούνται από δύο μέρη. Το πρώτο είναι ένα τμήμα JavaScript που εκτελείται μέσα στο WebView, το οποίο εκθέτει ένα API στην υβριδική εφαρμογή. Το δέυτερο μέρος είναι συγκεκριμένο για κάθε πλατφόρμα και είναι γραμμένο στη μητρική της γλώσσα, π.χ. Java για Android και Objective-C για iOS. Τα native API ελέγχονται από το δέυτερο μέρος.

Τα Plugin αποτελούν αναπόσπαστο μέρος του οικοσυστήματος της Cordova. Παρέχουν μια διασύνδεση για την Cordova και τα native συστατικά για να επικοινωνούν μεταξύ τους και να συνδέονται με τα API της συσκευής. Αυτό επιτρέπει στο χρήστη να εφαρμόζει το native κώδικα από τη JavaScript. Το πρόγραμμα Apache Cordova διατηρεί ένα σύνολο από plugins που ονομάζονται Core Plugins (κεντρικά πρόσθετα). Αυτά τα core plugins παρέχουν στην εφαρμογή πρόσβαση σε δυνατότητες συσκευών όπως μπαταρία, κάμερα, επαφές κ.λπ.

Εκτός από τα core plugins, υπάρχουν πολλά plugins τρίτων που προσφέρουν πρόσθετες συνδέσεις σε λειτουργίες που δεν είναι απαραίτητα διαθέσιμες σε όλες τις πλατφόρμες. Επίσης, κάθε χρήστης μπορεί να δημιουργήσει δικά του Plugins για συγκεκριμένες λειτουργίες που θέλει να έχει η εφαρμογή του, τα οποία στη συνέχεια μπορεί να τα δημοσιέυει στην ιστοσελίδα της Apache Cordova. Πράττοντας έτσι, άλλοι χρήστες μπορούν να τα εισάγουν στην εφαρμογή τους, να διορθώσουν σφάλματα και να τα επεκτείνουν, διευκολύνοντας έτσι τους επόμενους.

5.4 Λίγα λόγια για το Ionic Framework

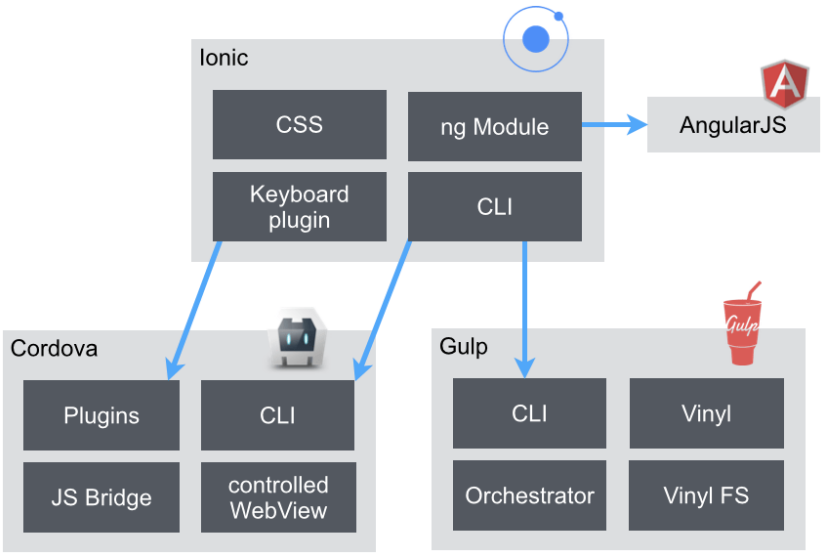
Το Ionic Framework, είναι ένα πλήρες SDK ανοιχτού κώδικα για ανάπτυξη υβριδικών εφαρμογών για κινητά, που δημιουργήθηκε από τους Max Lynch, Ben Sperry και Adam Bradley από την Drifty Co. το 2013. Η αρχική έκδοση κυκλοφόρησε το 2013 και χτίστηκε πάνω από τα AngularJS και Apache Cordova. Ωστόσο, η τελευταία έκδοση δημιουργήθηκε εκ νέου ως ένα σύνολο στοιχείων Web, επιτρέποντας στο χρήστη να επιλέξει οποιοδήποτε Framework, όπως το Angular, React ή Vue.js. Μέσω της Cordova, μπορεί κανείς να έχει πρόσβαση στα API της συσκευής χρησιμοποιώντας μια βιβλιοθήκη όπως η ngCordova και να τα συνδιάζει με στοιχεία διεπαφής χρήστη του Ionic.

Το Ionic Framework, στον πυρήνα του, αποτελείται από τέσσερα μέρη:

- Ένα **stylesheet** που ορίζει μια βελτιστοποιημένη διάταξη για κινητά. Αυτή η διάταξη χρησιμεύει ως βάση για την εφαρμογή.
- Το **AngularJS** module (δομοστοιχείο) ορίζει τις οδηγίες, τα πρότυπα πλοήγησης και τις βέλτιστες πρακτικές, ώστε να μην χρειάζεται να το κάνει ο χρήστης.
- Ενα πρόγραμμα γραμμής εντολών (**CLI**-Command Line Interface) που δέχεται την εισαγωγή κειμένου για την εκτέλεση λειτουργιών του λειτουργικού συστήματος. Στη συγκεκριμένη περίπτωση λειτουργεί σαν ένα είδος μεσολάβησης για το Cordova και το Gulp CLI.
- Το τελευταίο στοιχείο είναι ένα πρόσθετο πληκτρολογίου, **keyboard plugin**. Αν και τεχνικά δεν απαιτείται, το πρόσθετο παρέχει περισσότερες πληροφορίες σχετικά με την τρέχουσα κατάσταση της εφαρμογής.

Παρακάτω υπάρχει μια εικόνα με την αρχιτεκτονική μιας Ionic εφαρμογής:

Σχήμα 3: Αρχιτεκτονική Ionic.



5.5 Τι σημαίνει ο όρος SDK

Το SDK (Software Development Kit) σημαίνει ”Πακέτο ανάπτυξης λογισμικού”. Σκεφτείτε το σαν να κατασκεβάζετε ένα μοντέλο αυτοκινήτου. Κατά την κατασκευή αυτού του μοντέλου, απαιτείται ένα πλήρες σύνολο στοιχείων, συμπεριλαμβανομένων των εξαρτημάτων του αυτοκινήτου, των εργαλείων που απαιτούνται για την τοποθέτησή τους, των οδηγιών συναρμολόγησης και ούτω καθεξής.

Ένα SDK ή devkit λειτουργεί με τον ίδιο τρόπο, παρέχοντας ένα σύνολο εργαλείων προγραμματιστές να δημιουργούν εφαρμογές λογισμικού σε μία συγκεκριμένη πλατφόρμα. Εάν ένα API είναι ένα σύνολο δομικών στοιχείων, βιβλιοθηκών σχετικής τεκμηρίωσης, δειγμάτων κώδικα, διεργασιών ή οδηγών που επιτρέπουν στους χρήστες τη δημιουργία ενός στοιχείου, ένα SDK είναι ένα πλήρες εργαστήριο, διευκολύνοντας τη δημιουργία στοιχείων που είναι αδύνατο να δημιουργηθούν με απλά API.

Τα SDK είναι οι πηγές προέλευσης για σχεδόν κάθε πρόγραμμα με το οποίο μπορεί να αλληλεπιδράσει ένας σύγχρονος χρήστης. Από το πρόγραμμα περιήγησης μέχρι τα παιχνίδια, πολλές εφαρμογές κατασκευάστηκαν πρώτα με ένα SDK, πολύ πριν ένα API χρησιμοποιήθει για την επικοινωνία μεταξύ εφαρμογών.

5.6 Τι σημαίνει ο όρος API

Ένα API είναι απλά μία διασύνδεση που επιτρέπει σε ένα λογισμικό να αλληλεπιδρά με άλλα λογισμικά. Αυτό είναι μέρος του ονόματός του - API, Application Programming Interface - και αποτελεί βασικό στοιχείο της λειτουργικότητάς του. Τα API διατίθενται σε πολλά σχήματα και μεγέθη.

Για οποιονδήποτε ιστότοπο που μπορεί να επισκεφτεί ένας χρήστης, ο φυλλομετρητής, κατά πάσα πιθανότητα, χρησιμοποιεί μία ποικιλία συνόλων API, για να μετατρέπει τις εντολές χρηστών σε χρήσιμες λειτουργίες, να ζητά δεδομένα από διακομιστές (servers), να απεικονίζει αυτά τα δεδομένα σε κατάλληλη μορφή για να μπορεί να δει ο χρήστης και να επικυρώνει την απόδοση των αιτημάτων τους.

Ακόμα και κάτι τόσο απλό όσο η αντιγραφή και η επικόλληση σε έναν υπολογιστή χρησιμοποιεί ένα API. Η αντιγραφή κειμένου μετατρέπει το πάτημα ενός κουμπιού σε μία εντολή, τα δεδομένα αποθηκεύονται στη μνήμη RAM χρησιμοποιώντας ένα API, τα δεδομένα μεταφέρονται στη συνέχεια από μία εφαρμογή σε άλλη χρησιμοποιώντας το ίδιο API και τελικά τα δεδομένα αποδίδονται κατά την επικόλληση χρησιμοποιώντας άλλο API.

Στον παγκόσμιο ιστό, το API αναλαμβάνει μία ελαφρώς διαφορετική λειτουργία. Τα Web API επιτρέπουν την αλληλεπίδραση μεταξύ διαφορετικών συστημάτων, συχνά για συγκεκριμένες περιπτώσεις χρήσης. Για παράδειγμα, όταν οι χρήστες αλληλεπιδρούν με το Facebook, χρησιμοποιούν ένα API για να σχολιάσουν, να αποθηκεύσουν τα δεδομένα τους, να ακολουθήσουν έναν χρήστη, να διαγράψουν τα post κ.ο.κ. Τελικά, ένα Web API είναι απλά ένα σύνολο οδηγιών, όπως ακριβώς το API του προσωπικού υπολογιστή, αλλά βασίζεται στο χώρο του web.

Ίσως το πιο σημαντικό είναι το γεγονός ότι τα API επιτρέπουν τη συνέπεια. Στα πρώτα χρόνια του προγραμματισμού, οι εντολές και οδηγίες ήταν χαλαρά κωδικοποιημένες και σπάνια τεκμηριωμένες. Με την εμφάνιση των σύγχρονων υπολογιστών, τα API επέτρεψαν τη συνεπή κωδικοποίηση σε σταθερά περιβάλλοντα, επιτρέποντας αντιγράψιμες λειτουργίες να παραδίδονται απαράλλακτες κάθε φορά που υποβάλεται μία αίτηση με αξιοπιστία και προβλεψιμότητα.

5.7 Λίγα λόγια για την Angular

Η Angular είναι το πιο δημοφιλές πλαίσιο εφαρμογών ιστού ανοιχτού κώδικα (open-source web application framework) της Google και είναι βασισμένο στην TypeScript. Η πρώτη της έκδοση αναπτύχθηκε το 2010 και ονομάστηκε AngularJs. Η δέυτερη έκδοση αναπτύχθηκε το 2014. Σήμερα όλες οι εκδόσεις από τη δέυτερη μέχρι την τελευταία, που είναι η έβδομη, ονομάζονται Angular. Κυριώς χρησιμοποιείται για τη δημιουργία Εφαρμογών Ενιαίας Σελίδας (SPA-Single Page Application). Με τον όρο SPA αναφερόμαστε σε μία εφαρμογή ιστού ή μία ιστοσελίδα πού αλληλεπιδρά με το χρήστη μέσω της δυναμικής επανεγγραφής της τρέχουσας σελίδας, αντί της φόρτωσης ολόκληρων νέων σελίδων απο το διακομιστή. Παράδειγμα τέτοιας σελίδας είναι η GMAIL, όπου χωρίς ανανέωση, ανοίγει όλες τις λεπτομέρειες της σελίδας. Έχει μόνο μία σελίδα HTML, η οποία ζητά περιεχόμενο άλλων σελίδων από το διακομιστή. Αποτέλεσμα είναι πολύ πιο μικροί χρόνοι φόρτωσης. Οι εφαρμογές Angular κατασκευάζονται από εξαρτήματα τα οποία μπορούν να ενσωματώνονται μεταξύ τους και έτσι η δημιουργία εφαρμογών γίνεται πολύ εύκολη.

6 Φάσεις Ανάπτυξης Εφαρμογής

6.1 Ανάλυση

Στην ενότητα αυτή παρουσιάζεται, η φάση της ανάλυσης του συστήματος καθώς και μια αρχική προσέγγιση της εφαρμογής που πρόκειται να αναπτυχθεί. Αποτελεί το πρώτο και ίσως σημαντικότερο στάδιο ώστε να προχωρήσει κάποιος ορθά στη σχεδίαση και στην υλοποίηση. Στην παρούσα φάση της εφαρμογής καθορίστηκαν οι κύριοι στόχοι της καθώς και συγκεντρωθήκαν οι απαιτήσεις χρηστών.

Παρακάτω ακολουθεί ένας πίνακας ερωταπαντήσεων που συνετέλεσε στην περαιτέρω οργάνωση της δομής του έργου:

Πίνακας 1: Ανάλυση βασικών ερωτημάτων

Ερώτημα	**Απάντηση**
Ποιά είναι η ηλικία χρηστών;	Όλες οι ηλικές επιτρέπονται
Τι γλώσσα μιλούν;	Αγγλικά
Έχουν χρησιμοποιήσει παλιότερα smartphone;	Ναι
Έχουν χρησιμοποιήσει παρόμοιες εφαρμογές;	Πιθανώς
Υφίσταται φυλετικός διαχωρισμός;	Όχι
Χρειάζεται πρόσβαση στο Διαδίκτυο;	Ναι, για την εγγραφή
Απαιτείται πρόσβαση σε τοποθεσία χρήστη;	Όχι
Απαιείται πρόσβαση στις φωτογραφίες ή τις επαφές του χρήστη;	Όχι
Απαιτείται η οπτική αναπαράσταση της προόδου του χρήστη;	Ναι
Πώς θα γίνει η διανομή του τελικού προϊόντος;	Προσωπική χρήση στα πλαίσια πτυχιακής εργασίας
Πόσο χρόνο διαθέτουμε για την ανάπτυξη του έργου;	Έξι μήνες

Έπειτα, καθορίστηκαν τα τμήματα υλοποίησης του έργου και αποφασίστηκε ο χρονοπρογραμματισμός του. Παράλληλα, τα παραπάνω τμήματα τοποθετήθηκαν σε χρονική σειρά και καθορίστηκε η χρονική τους διάρκεια.

Στον παρακάτω πίνακα ακολουθεί το χρονοδιάγραμμα εκπόνησης έργου:

Πίνακας 2: Πλάνο και Χρονοπρογραμματισμός Έργου

Πλάνο Εργασιών	**Απρίλιος**	**Μάιος**	**Ιούνιος**	**Ιούλιος**	**Αύγουστος**	**Σεπτέμβριος**
Φάση Α:Ανάλυση						
Καταγραφή βασικών ιδεών						
Χρονοπρογραμματισμος και πλάνο έργου						
Φάση Β:Σχεδίαση						
Κατηγοριοποίηση και επιλογή περιεχομένου						
Φαση Γ:Υλοποίηση						
Συγκέντρωση και επεξεργασία πρωτογενούς υλικού						
Φαση Δ:Ελεγχος						
Έλεγχος λειτουργίας εφαρμογής και απατήσεων χρηστών						
Διωρθώσεις για την παραγωγή του τελικού προϊόντος						

6.2 Σχεδίαση

Η σχεδίαση αποτελεί τη δεύτερη φάση ανάπτυξης της εφαρμογής. Οι γενικοί στόχοι και οι αρχές που καθορίστηκαν στη φάση της ανάλυσης, μετατρέπονται σε μια ολοκληρωμένη αναλυτική περιγραφή της εφαρμογής. Σε αυτή τη φάση καθορίζεται και η αρχιτεκτονική του συστήματος.

Απαιτήσεις σχεδιασμού:

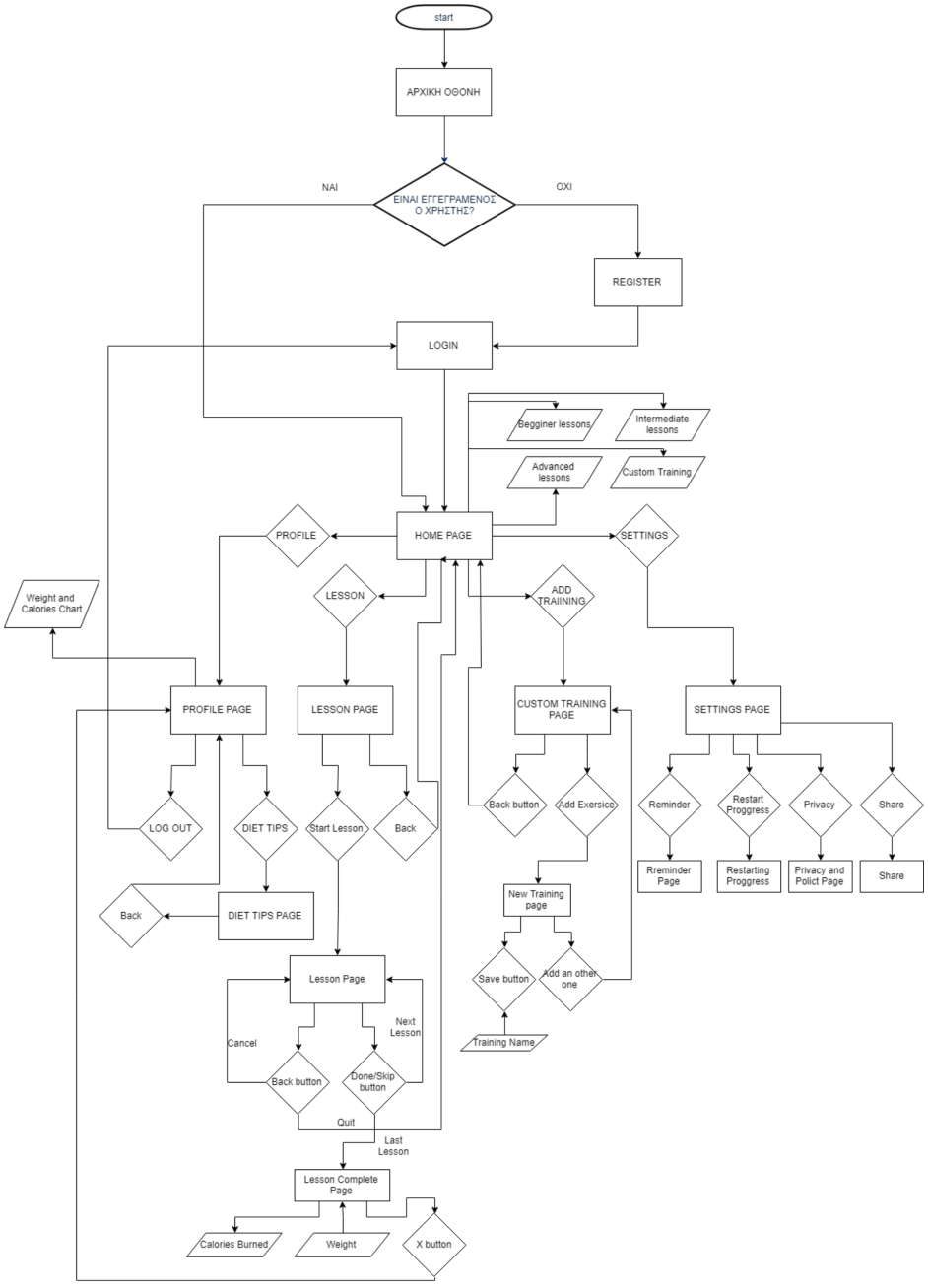
Κατά το σχεδιασμό μιας εφαρμογής υπάρχουν κάποιες απαιτήσεις που πρέπει να έχει. Τα εργαλεία που χρησιμοποιήθηκαν είναι όλα open source. Για όλους τους χρήστες που θα χρησιμοποιήσουν αυτή την εφαρμογή το μόνο που απαιτείται είναι ένα smartphone με οποιοδήποτε λειτουργικό, και σύνδεση στο Διαδίκτυο, για την εγγραφή τους στην εφαρμογή. Στην συνέχεια, δεν είναι απαραίτητη η πρόσβαση στο διαδύκτιο για την χρήση της εφαρμογής. Για την ανάλυση απαιτήσεων έγινε αρχικά μια έρευνα, όπου μελετήθηκαν παρόμοια συστήματα.

Δομή εφαρμογής:

Για τη διαγραμματική απεικόνιση της δομής της εφαρμογής, κατά τη φάση της σχεδίασης, χρησιμοποιήθηκε το εργαλείο λογισμικού **draw.io**, με το οποίο σχεδιάστηκε το αρχικό διάγραμμα ροής προγράμματος (flowchart), για τη δομή της εφαρμογής.

Για την ευκολότερη κατανόηση της δομής της εφαρμογής ακολουθεί η παρακάτω εικόνα:

Σχήμα 4: Διάγραμμα ροής προγράμματος.

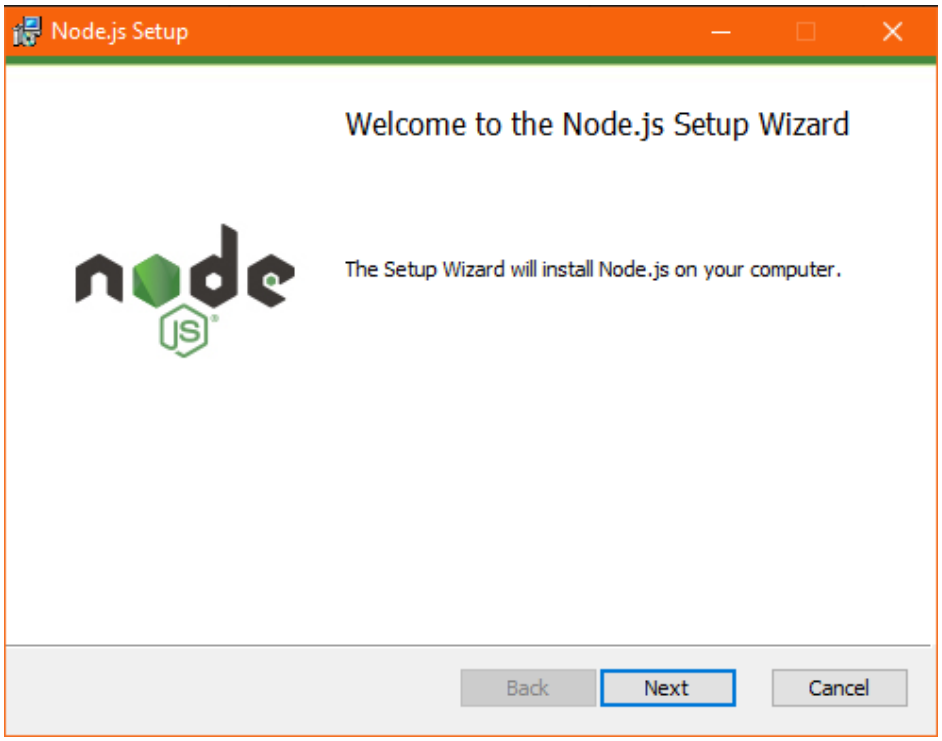


6.3 Υλοποίηση

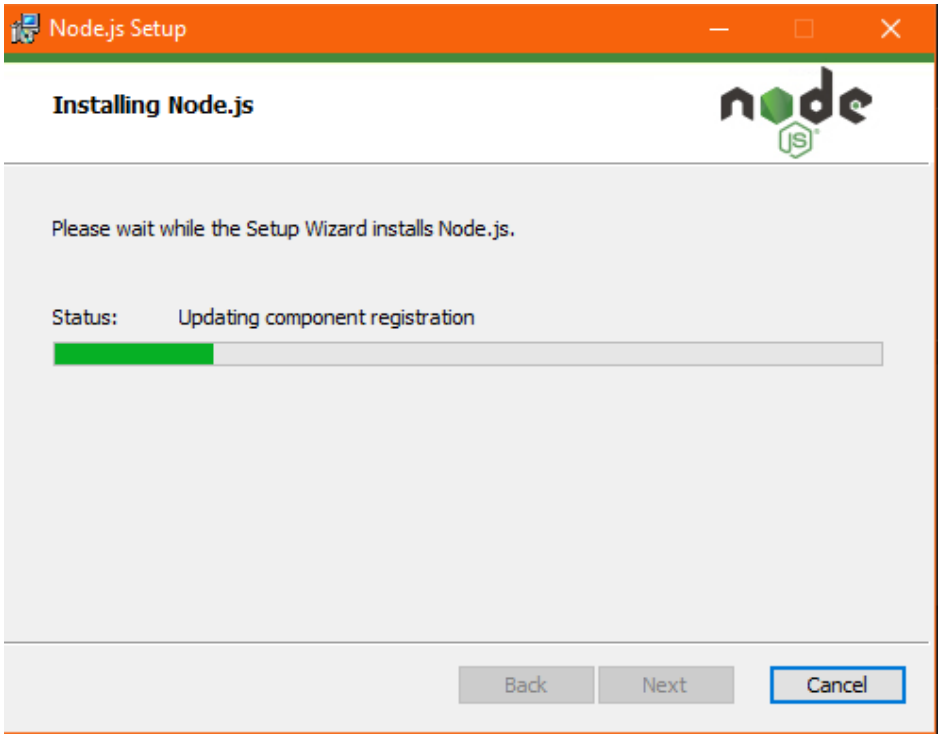
6.3.1 Εγκατάσταση Λογισμικού

Εγκατασταση Node
Node είναι ένα περιβάλλον χρόνου εκτέλεσης που επιτρέπει την εγγραφή της JavaScript στην πλευρά του διακομιστή. Εκτός από το ότι χρησιμοποιείται για υπηρεσίες ιστού, το Node χρησιμοποιείται συχνά για την ανάπτυξη εργαλείων προγραμματιστών, όπως το Ionic CLI.

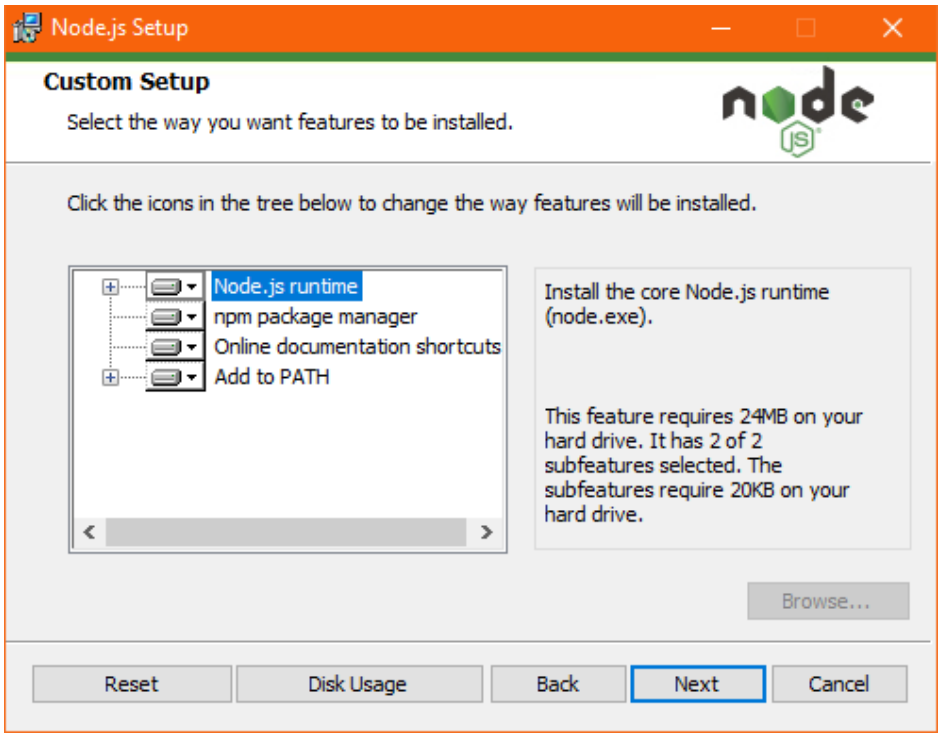
Σχήμα 5: Εγκατάσταση Node.js No.1



Σχήμα 6: Εγκατάσταση Node.js No.2



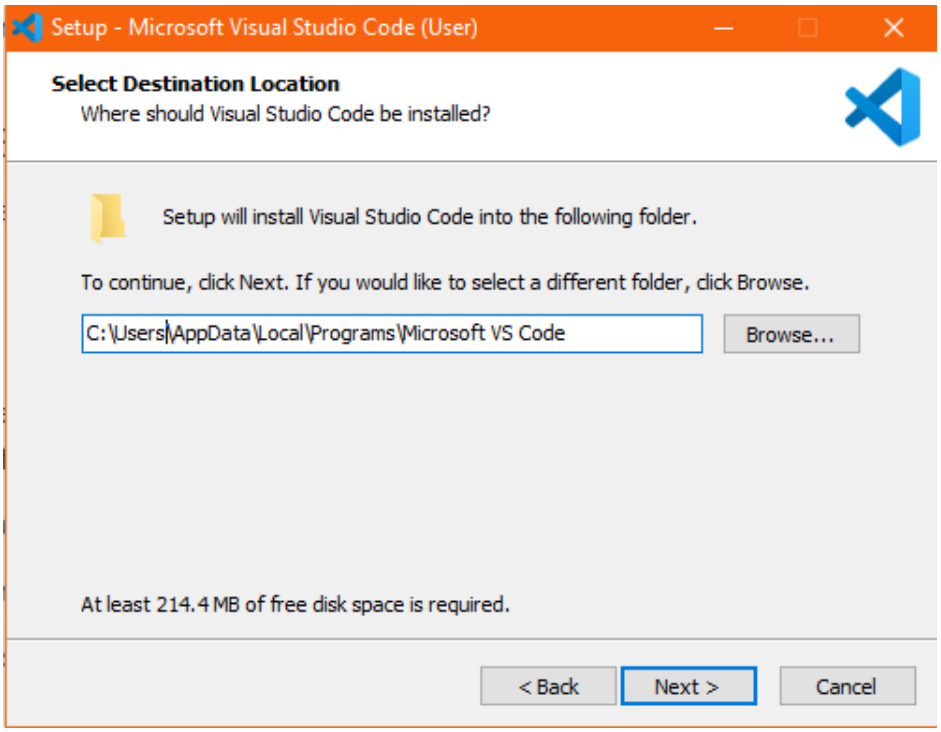
Σχήμα 7: Εγκατάσταση Node.js No.3



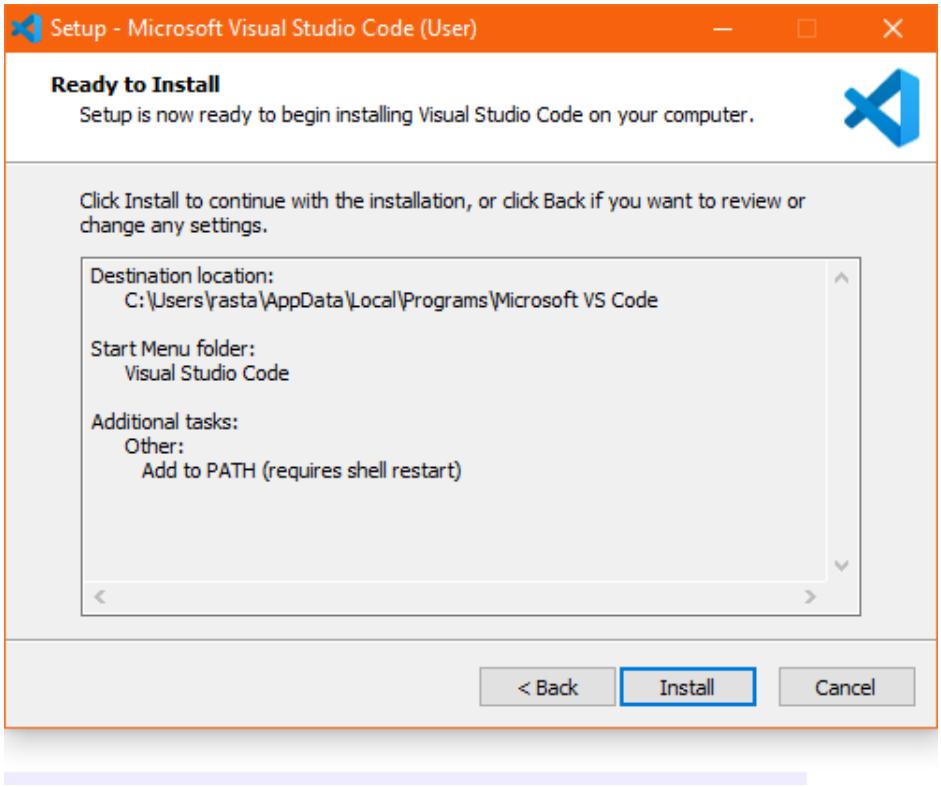
Εγκατάσταση Visual Studio Code

Είναι στο χέρι του καθενός να διαλέξει ποιόν επεξεργαστή κειμένου θα χρησιμοποιήσει. Εγώ διάλεξα το Visual Studio Code το οποίο παρέχει υπέρ αρκετές δυνατότητες στο χρήστη. Παρακάτω φαίνεται πως μπορεί να γίνει η εγκατάσταση του Visual Studio Code.

Σχήμα 8: Εγκατάσταση Visual Studio Code No.1



Σχήμα 9: Εγκατάσταση Visual Studio Code No.2

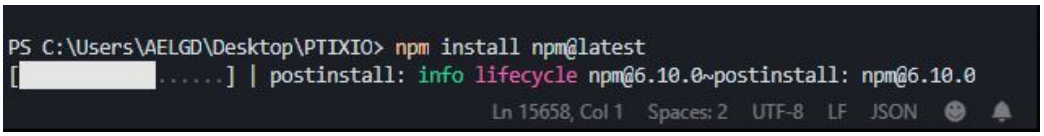


Αφού εγκατασταθεί το Visual Studio Code έρχεται η εγκατάσταση του npm.

Εγκατάσταση npm

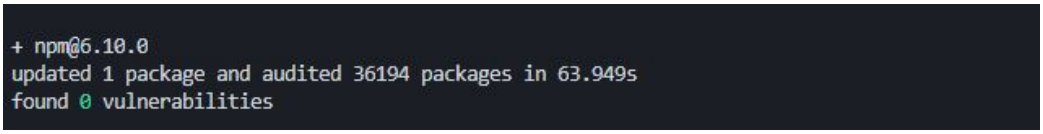
Npm είναι ο διαχειριστής πακέτων του Node. Επιτρέπει στους προγραμματιστές να εγκαταστήσουν, να μοιραστούν και να συσκευάσουν Node modules (δομοστοιχεία). Το Ionic, μαζί με τις εξαρτήσεις του μπορεί να εγκατασταθεί με το npm.

Σχήμα 10: Εγκατάσταση npm No.1



Επόμενο βήμα η εγκατάσταση του Ionic.

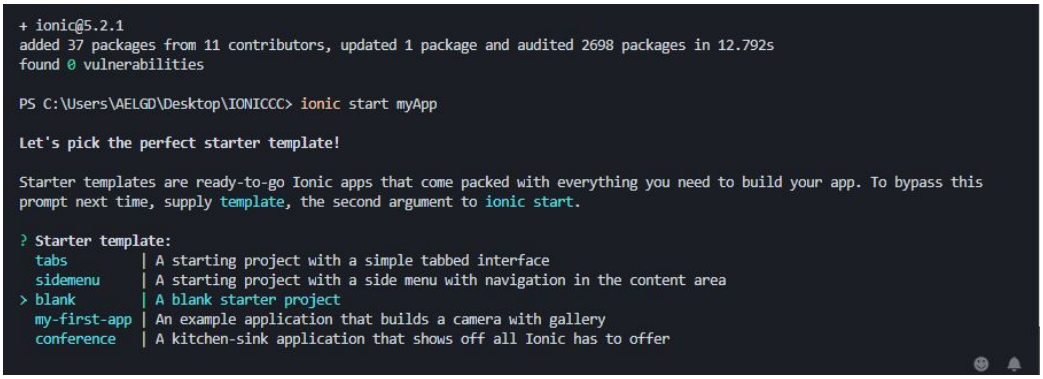
Σχήμα 11: Εγκατάσταση npm No.2



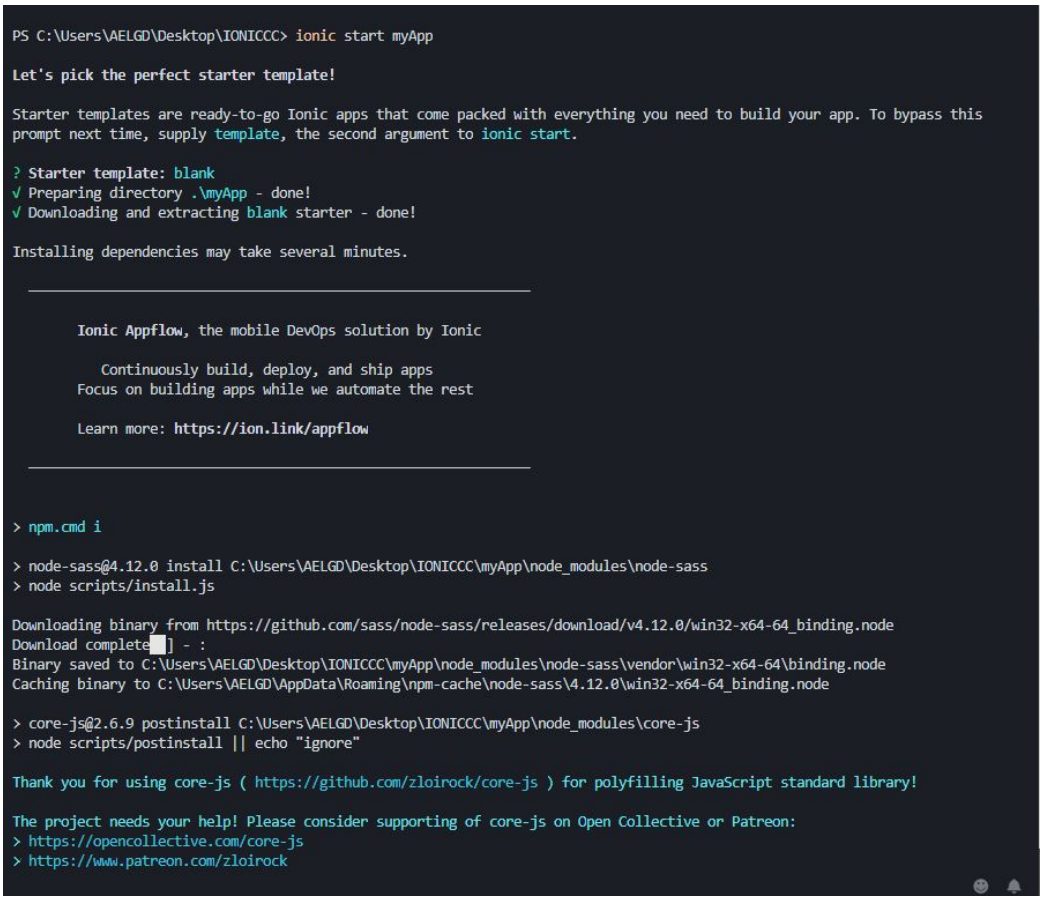
Εγκατάσταση Ionic 4

Η εγκατάσταση του Ionic γίνεται με την εντολή **npm install -g ionic**. Αφού γίνει η εγκατάσταση χωρίς να εμφανιστεί κάποιο σφάλμα, με την εντολή **ionic start myApp** δημιουργείται μια νέα εφαρμογή με το όνομα "**myApp**". Στη συνέχεια το Ionic δίνει τη δυνατότητα να επιλογής template (προτύπου). Αφου γίνει ή όχι η επιλογή κάποιου template, ξεκινάει να γίνεται η λήψη των απαραίτητων στοιχείων του Ionic.

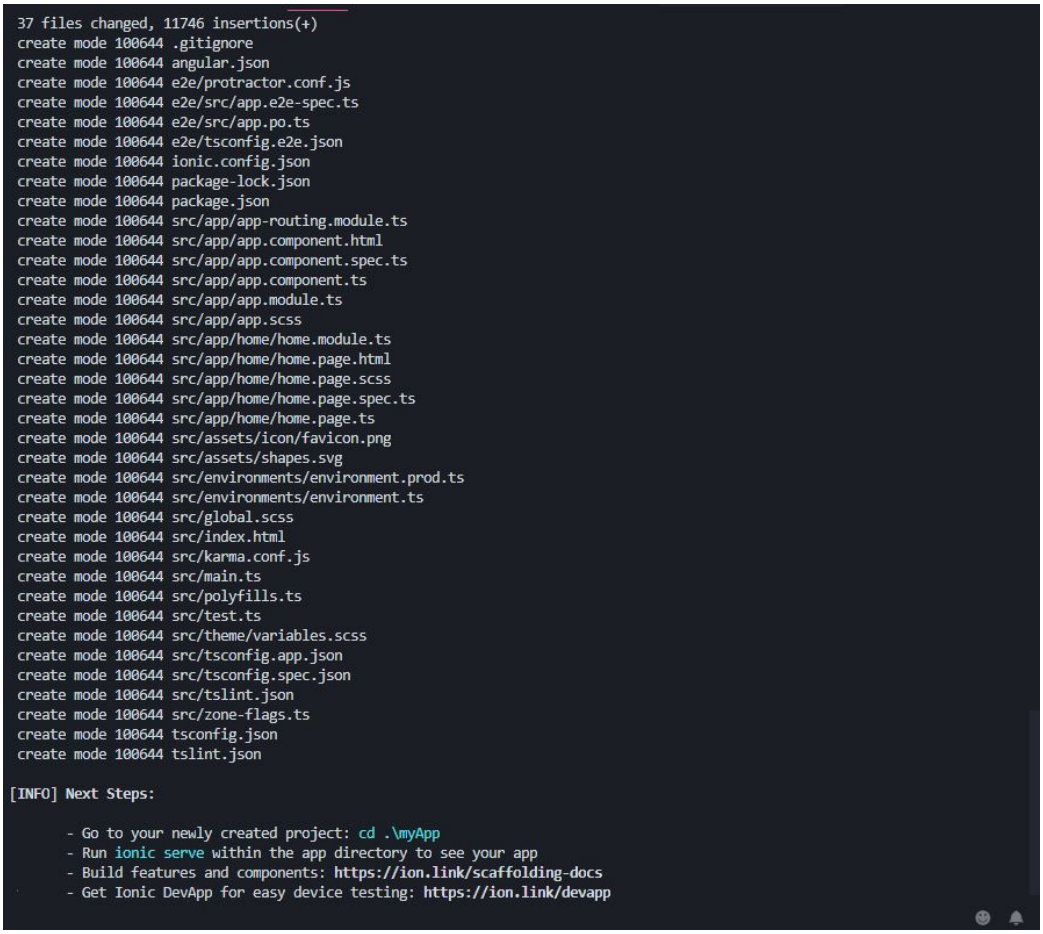
Σχήμα 12: Εγκατάσταση Ionic No.1



Σχήμα 13: Εγκατάσταση Ionic No.2

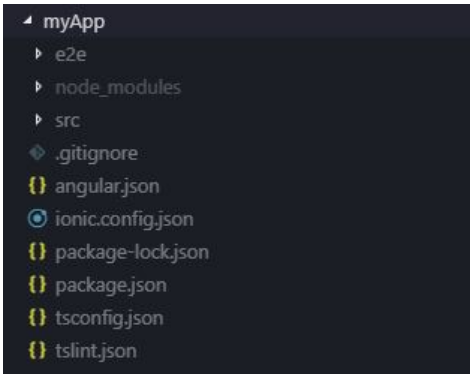


Σχήμα 14: Εγκατάσταση Ionic No.3



Με την εντολή **ionic start myApp** το Ionic εγκαθιστά απαραίτητες εξαρτήσεις και δημιουργεί κάποιους φακέλους. Παρακάτω απεικονίζεται η δομή έργου ενός νέου Ionic project:

Σχήμα 15: Δομή έργου Ionic



e2e είναι τα αρχικά του End to End testing, είναι ένας τρόπος για να βεβαιωθούμε οτι η εφαρμογή μας λειτουργεί σωστά. Συνήθως χρησιμοποιούμε τις δοκιμές E2E για να διασφαλίσουμε οτι τα στοιχεία μας λειτουργούν σωστά μαζί.

node modules παρέχει πακέτα npm σε ολόκληρο το χώρο εργασίας.

src περιέχει αρχεία προέλευσης για το έργο εφαρμογής σε επίπεδο root.

angular.json περιέχει προεπιλεγμένες ρυθμίσεις παραμέτρων CLI για όλα τα έργα στο χώρο εργασίας συμπεριλαμβανομένων των επιλογών διαμόρφωσης για τα εργαλεία δημιουργίας, προβολής και δοκιμής που χρισημοποιεί το CLI όπως το **tslint**.

ionic.cofig.json περιέχει αρχεία διαμόρφωσης του έργου.

package-lock.json παρέχει πληροφορίες έκδοσης για όλα τα πακέτα που έχουν εγκατασταθεί στο φάκελο node modules από το npm.

package.json ρυθμίζει τις εξαρτήσεις πακέτων npm που είναι διαθέσιμες σε όλα τα έργα στο χώρο εργασίας.

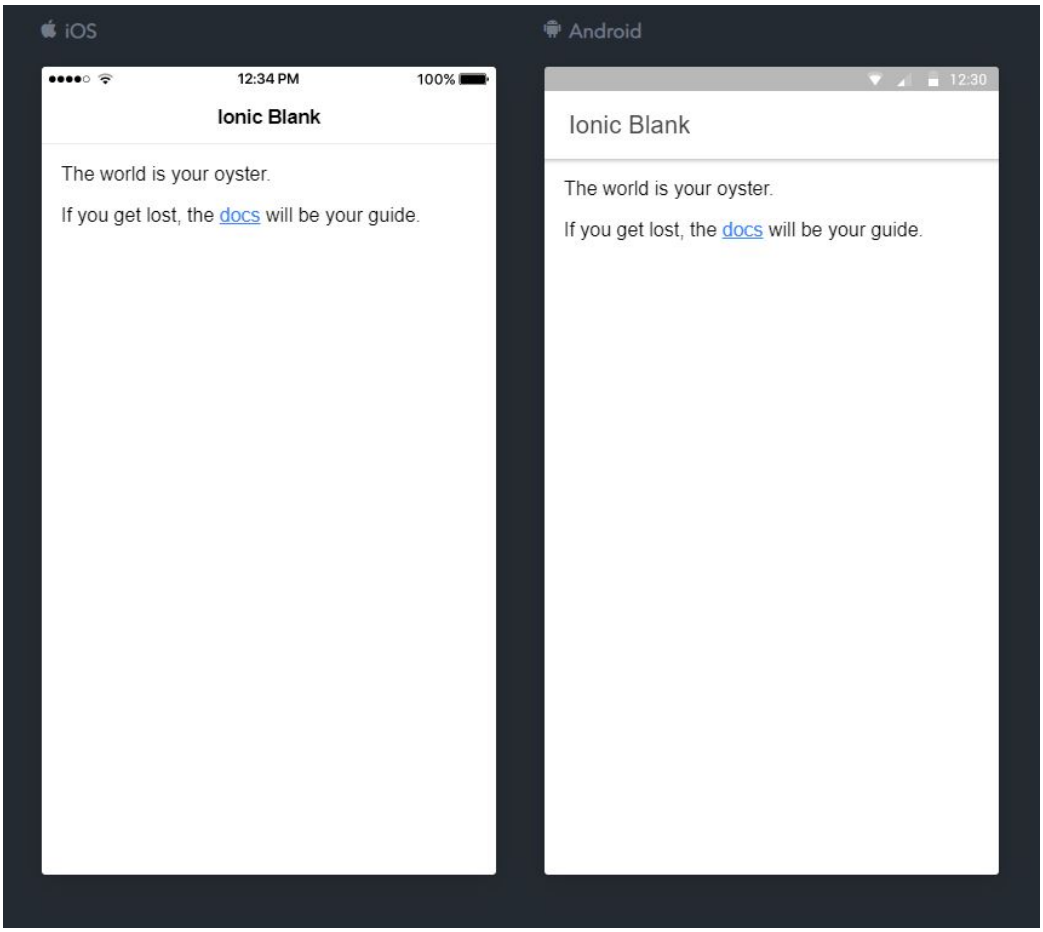
tsconfig.json είναι μία προεπιλεγμένη διαμόρφωση TypeScript για έργα στο χώρο εργασίας.

tslint.json είναι ένα εργαλείο στατικής ανάλυσης που ελέγχει τον κώδικα TypeScript για σφάλματα ευκρίνειας, συντήρησης και λειτουργικότητας.

Με την εντολή **ionic serve**, γίνεται η εκκίνηση ενός τοπικού server για τον έλεγχο της εφαρμογής. Ο server τρέχει στον browser, παρακολουθεί τις αλλαγές στα source files και τον επαναφορτώνει αυτόματα με την ενημερωμένη έκδοση. Επίσης δίνει τη δυνατότητα στο χρήστη να έχει μια εικόνα για το πως φαίνεται η εφαρμογή σε κινητές συσκευές με διαφορετικά λειτουργικά συστήματα. Αυτό είναι πολύ χρήσιμο, διότι πολλές φορές κάτι που λειτουργεί και φαίνεται σωστά στο ένα λειτουργικό στο άλλο παρουσίαζει σφάλματα.

Παρακάτω μια εικόνα με το τι εμφανίζεται στον browser μετά την εντολή:

Σχήμα 16: Ionic serve command



6.4 Έλεγχος

Ο ελέγχος που ακολουθεί μετά από τη φάση της υλοποίησης, γίνεται για να εντοπιστούν εντοπιστούν παραλείψεις ή δυσλειτουργίες. Αν υπάρξει κάτι από τα προηγούμενα γίνεται επιστροφή στη φάση της υλοποίησης ή ακόμα και σε κάποια προγενέστερη φάση αν κρίνεται απαραίτητο. Σκοπός της φάσης αυτής είναι πάντοτε η βέλτιστη τελική κατάσταση που μπορεί να φτάσει το σύστημα.

Στη συγκεκριμένη εφαρμογή ο έλεγχος έγινε χρησιμοποιώντας την εφαρμογή σε Android και iOS συσκευές. Σε όλες τις περιπτώσεις η εφαρμογή λειτούργησε ορθά και δεν παρουσιάστηκαν σφάλματα.

6.5 Διανομή

Η παρούσα εφαρμογή δε θα διατίθεται κάπου προς πώληση ή εγκατάσταση (App Store, Google Play Store, Microsoft Store). Σχεδιάστηκε και υλοποιήθηκε μόνο στα πλάισια της πτυχιακής εργασίας.

7 Παρουσίαση εφαρμογής με περιγραφή κώδικα

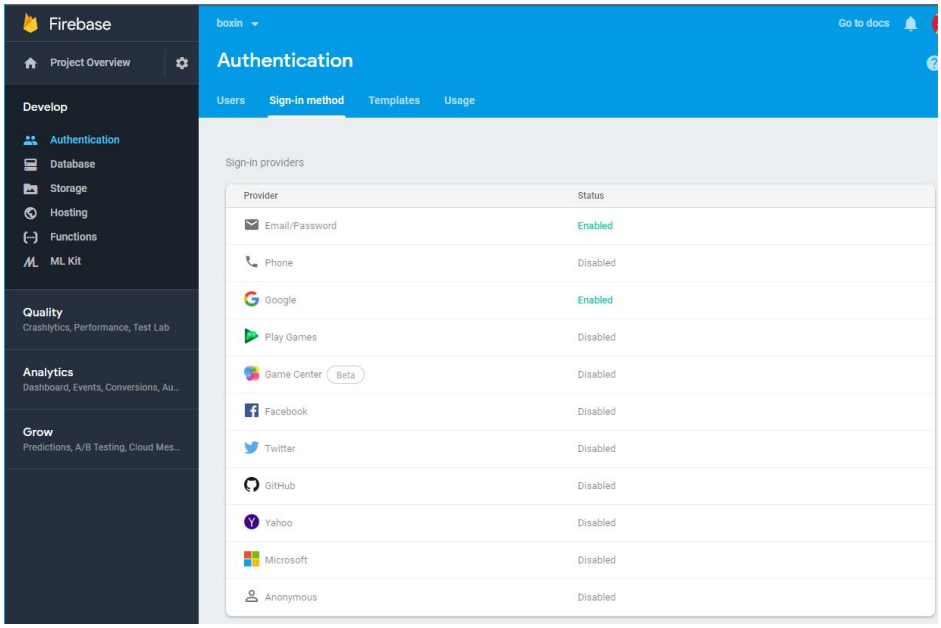
Στο κεφάλαιο αυτό έγινε η περιγραφή του κώδικα καποίων βασικών λειτουργιών της εφαρμογής ώστε να γίνει κατανοητή η υλοποίηση της στον αναγνώστη.

7.1 Δημιουργία συστήματος ελέγχου ταυτότητας χρήστη

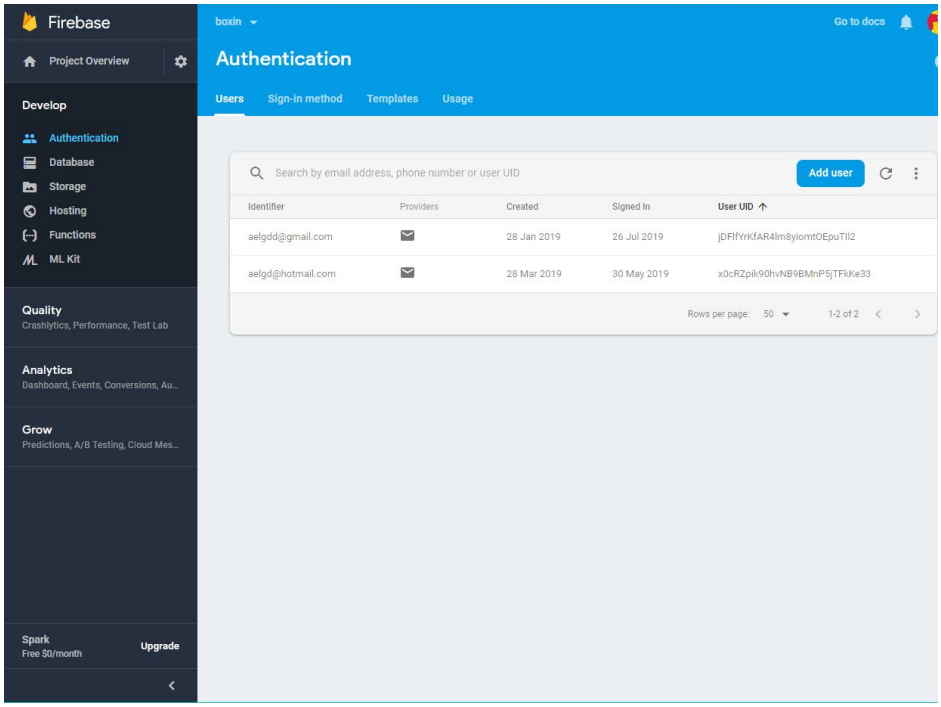
Το σημαντικότερο χαρακτηριστικό για τις περισσότερες εφαρμογές είναι το σύστημα ελέγχου ταυτότητας χρήστη. Για τη λειτουργία αυτή χρησιμοποιήθηκε η Firebase της Google, μια υπηρεσία που παρέχει δυνατότητες βάσης δεδομένων, ελέγχου ταυτότητας χρήστη και πολλών άλλων. Τα βήματα μέσα στη Firebase είναι, αρχικά η δημιουργία ενός νέου project και στη συνέχεια η επιλογή της μεθόδου σύνδεσης χρήστη.

Παρακάτω υπάρχουν εικόνες που δείχνουν ακριβώς αυτό:

Σχήμα 17: Firebase Sign-in method



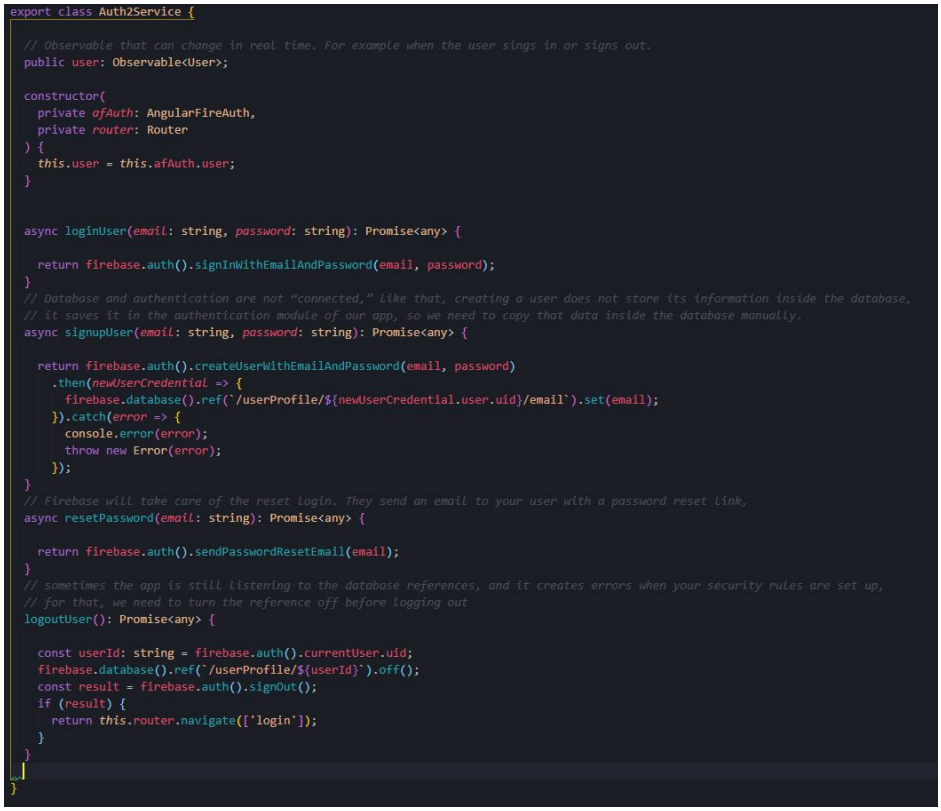
Σχήμα 18: Firebase Authentication



Για τη σύνδεση της Firebase με την εφαρμογή δημιουργήθηκε μια Global Angular Service. Σε αυτή αρχικά γίνεται εισαγωγή διάφορων στοιχείων. Του **Angular Router** για να γίνει η ανακατεύθυνση του χρήστη όταν αποσυνδεθεί. Στη συνέχεια του **AngularFireAuth** για την αλληλεπίδραση με τη Firebase Auth και του Fire Store. Η υπηρεσία αυτή έχει ένα κομμάτι πληροφοριών που μπορεί να μοιράζεται σε όλλα τα εξαρτήματα, αυτό το κομμάτι είναι η καταγραφή του εγγράφου του χρήστη στη βάση δεδομένων. Ορίζεται ως **Observable** γιατί μπορεί να αλλάξει κάθε φορά που ο χρήστης συνδέεται και αποσυνδέεται. Στη λειτουργία **signupUser** γίνεται η επιλογή του τρόπου εγγραφής του χρήστη, στη συγκεκριμένη περίπτωση με Email και Password. Στη συνέχεια δημιουργείται στη βάση δεδομένων νέο πιστοποιητηκό χρήστη με user ID και user Email. Με τη λειτουργία **ressetPassword** ο χρήστης εισάγει το email του και η Firebase του στέλνει αυτόματα ένα email επαναφοράς κωδικού. Τέλος η λειτουργία **logoutUser** αποσυνδέει το χρήστη από τη firebase και τον καθοδιγεί στη σελίδα **login**.

Παρακάτω υπάρχει ο κώδικας της υπηρεσίας αυτής:

Σχήμα 19: Authendication Service



```
export class Auth2Service {

  // Observable that can change in real time. For example when the user sings in or signs out.
  public user: Observable<User>;

  constructor(
    private afAuth: AngularFireAuth,
    private router: Router
  ) {
    this.user = this.afAuth.user;
  }


  async loginUser(email: string, password: string): Promise<any> {

    return firebase.auth().signInWithEmailAndPassword(email, password);
  }
  // Database and authentication are not "connected," like that, creating a user does not store its information inside the database,
  // it saves it in the authentication module of our app, so we need to copy that data inside the database manually.
  async signupUser(email: string, password: string): Promise<any> {

    return firebase.auth().createUserWithEmailAndPassword(email, password)
      .then(newUserCredential => {
        firebase.database().ref(`/userProfile/${newUserCredential.user.uid}/email`).set(email);
      }).catch(error => {
        console.error(error);
        throw new Error(error);
      });
  }
  // Firebase will take care of the reset login. They send an email to your user with a password reset link,
  async resetPassword(email: string): Promise<any> {

    return firebase.auth().sendPasswordResetEmail(email);
  }
  // sometimes the app is still listening to the database references, and it creates errors when your security rules are set up,
  // for that, we need to turn the reference off before logging out
  logoutUser(): Promise<any> {

    const userId: string = firebase.auth().currentUser.uid;
    firebase.database().ref(`/userProfile/${userId}`).off();
    const result = firebase.auth().signOut();
    if (result) {
      return this.router.navigate(['login']);
    }
  }

}
```

Στη συνέχεια υπάρχει ακόμα μια υπηρεσία, η **Authentication Guard Service**, η οποία ευθύνεται για την ασφάλεια της εφαρμογής. Ελέγχει αν ο χρήστης είναι αποσυνδεδεμένος ή όχι όταν ανοίγει την εφαρμογή και πράττει αναλόγως.

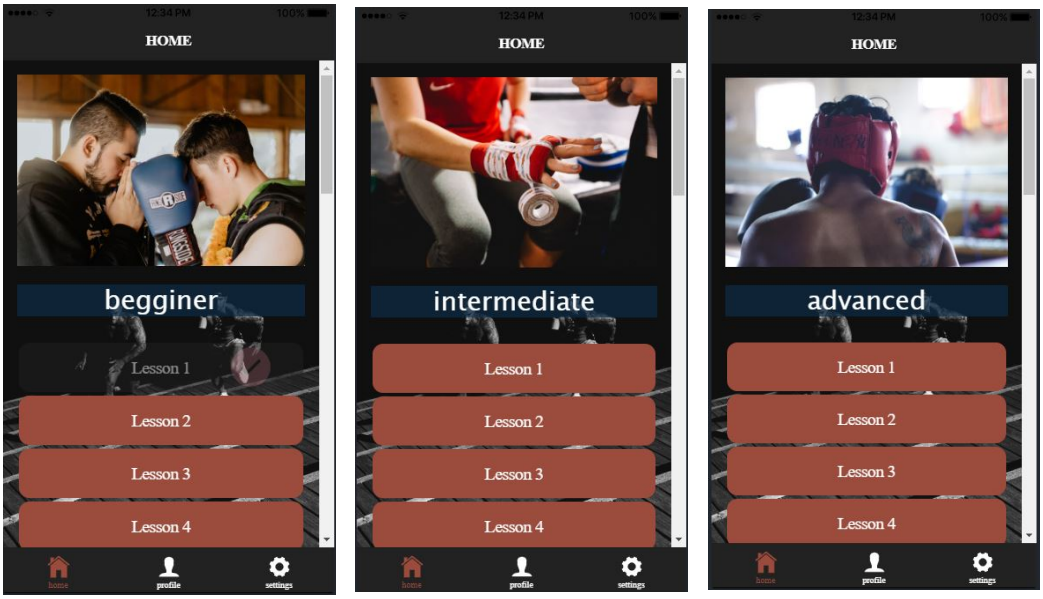
Η μέθοδος **canActivate** επιστρέφει μια boolean ή μια Observable boolean τιμή, εάν είναι true ενεργοποιεί τη διαδρομή (route) και εάν είναι false όχι. Για αυτό έγινε η μετατροπή του user Observable, που ορίστηκε στην authentication service, σε boolean format. Μέσα στη μέθοδο υπάρχει πρώτα ο operator **take(1)** ο οποίος ολοκληρώνει το Observable αφού μεταδωθεί η πρώτη τιμή, γιατί δεν θέλουμε να συνεχίσει να δουλεύει αφού η διαδρομή μπλόκαριστεί. Αμέσως μετά αντιστοιχίζεται το αντικείμενο σε boolean με τον **map** operator. Τέλος αν ο χρήστης δεν είναι συνδεδεμένος καθοδηγείται στη σελίδα σύνδεσης και του εμφανίζεται μια ειδοποίηση με τη **presentAlert** λειτουργία, λέγοντάς του οτι πρέπει να συνδεθεί για να συνεχίσει.

Σχήμα 20: Authentication Guard Service

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';
import { Observable } from 'rxjs';
import { Auth2Service } from '../services/auth2.service';
import { Router } from '@angular/router';
import { tap, map, take } from 'rxjs/operators';
import { AlertController } from '@ionic/angular';


@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor(
    private auth: Auth2Service,
    private router: Router,
    public alertController: AlertController
  ) { }

  async presentAlert() {

    const alert = await this.alertController.create({
      header: 'Alert',
      subHeader: 'You need to login to continue',
      buttons: ['OK']
    });

    await alert.present();
  }

  canActivate(

    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

    return this.auth.user.pipe(
      take(1),
      map(user => !!user),
      tap(loggedIn => {
        if (!loggedIn) {

          this.presentAlert();
          this.router.navigate(['/login']);
        }
      })
    );
  }
}
```

7.2 Home Page

Σχήμα 21: Home Page



Παραπάνω απεικονίζεται η αρχική σελίδα. Αποτελείται από ένα **slider** το οποίο δίνει τη δυνατότητα στο χρήστη να διαλέξει το επίπεδο δυσκολίας των ασκήσεων και να φτιάξει το δικό του πλάνο γυμναστικής. Επίσης υπάρχουν τα κουμπιά **lesson**, τα οποία μεταφέρουν το χρήστη στη σελίδα των ασκήσεων. Στην προκειμένη περίπτωση το lesson 1 του επιπέδου begginer είναι checked και ο χρήστης δεν μπορεί να το επιλέξει, αυτό γιατί έχει ολοκληρώσει ήδη το μάθημα αυτό.

Στη συνέχεια υπάρχουν οι βασικές λειτουργίες την αρχικής σελίδας, από τη μεριά του κώδικα.

Σχήμα 22: Home Page imports

```
import { Component, OnInit } from '@angular/core';
import { Router, NavigationExtras } from '@angular/router';
import { Network } from '@ionic-native/network/ngx';
import { Auth2Service } from 'src/app/services/auth2.service';
import { Storage } from '@ionic/storage';
import { AngularFireAuth } from '@angular/fire/auth';
import { AlertController, ToastController } from '@ionic/angular';
import { DataService } from 'src/app/services/data.service.js';
```

Στην αρχή του κώδικα βλέπουμε τα απαραίτητα στοιχεία που πρέπει να εισάγουμε για τις διάφορες λειτουργίες της αρχικής σελίδας. **Router** για την πλοήγηση του χρήστη απο σελίδα σε σελίδα. **NavigationExtras** για τη μεταφορά δεδομένων από σελίδα σε σελίδα. **Network** για πληροφορίες δικτύου. **Auth2Service** για πληροφορίες χρήστη. **Storage** για τη χρήση του αποθηκευτικού χώρου της συσκευής. **AlertController,**

ToastController για την εμφάνιση ειδοποιήσεων και τέλος **DataService** μια Global Service που δημιουργήθηκε για τη διαχείρηση δεδομένων όλης της εφαρμογής.

Σχήμα 23: Constructor και ngOnInit

Ο **contstructor** είναι η προεπιλεγμένη μέθοδος της κλάσης που εκτελείται όταν η κλάση δημιουργείται και εξασφαλίζει τη σωστή προετοιμασία πεδίων στην κλάση και τις υποκατηγορίες της. Συνήθως η **ngOnInit** χρησιμοποιείται για όλη την αρχικοποίηση / δήλωση. Μέσα στη ngOnInit η πρώτη γραμμή κώδικα καλείται κάθε φορά που έχει γίνει μια αλλαγή στον αποθηκευτικό χώρο της συσκευής, για παράδειγμα όταν ο χρήστης ολοκληρώνει ένα lesson, η αρχική σελίδα πρέπει να γνωρίζει πότε έγινε αυτή η αλλαγή για να απενεργοποιήσει το ανάλογο κουμπί όπως είδαμε παραπάνω, για αυτό καλείται η μέθοδος **watchStorage** της **DataService**. Οι συναρτήσεις **checkForUser** και **checkForPlan** καλούνται κάθε φορά που γίνεται η εμφάνιση της σελίδας και όταν γίνει μια αλλαγή στον αποθηκευτικό χώρο.Τέλος γίνεται ενημέρωση ανάλογα με την κατάσταση του δικτύου.

Σχήμα 24: Συνάρτηση checkForUser

```
checkForUser() {

  this.afAuth.authState.subscribe(user => {

    if (user) {

      this.Uid = user.uid;

      var UidBegginer = this.Uid + '/begginer';

      var UidIntermadiate = this.Uid + '/intermediate';

      var UidAdvanced = this.Uid + '/advanced';


      this.storage.get(UidBegginer).then((val) => {

        this.lessonFinished = JSON.parse(val);

        for (var i = this.firstDay; i <= this.lastDay; i++) {

          this.textArray = {
            lessonID: i,
            disabled: false
          };

          this.BegginerLessons.push(this.textArray);
        }

        if (this.lessonFinished) {

          for (var i = 0; i < this.lessonFinished.length; i++) {

            this.BegginerLessons.find(item => item.lessonID === this.lessonFinished[i]

          }

        }

      });
```

Στη συνάρτηση αυτή πρώτα απο όλα, γίνεται ο έλεγχος χρήστη, αφού έχουμε πάρει τα στοιχεία του χρήστη και πιο συγκεκριμένα το **user.uid**, δημιουργούμε τρείς σταθερές (UidBegginer, UidIntermediate και UidAdvanced) για να ελέγξουμε τον αποθηκευτικό χώρο. Με την εντολή **storage.get(UidBegginer)** παίρνουμε τα δεδομένα για το συγκεκριμένο **ID**. Γεμίζουμε τον πίνακα **BegginerLessons** με τιμές, τέτοιες ώστε ο χρήστης να μπορέι να πατήσει όλα τα κουμπιά **lesson**, για αυτό και αρχικά η τιμή **disabled** είναι **false**. Στη συνέχεια αν το μάθημα έχει ολοκληρωθεί ψάχνουμε μέσα στον πίνακα BegginerLessons και αλλάζουμε την τιμή disabled του συγκεκριμένου κουμπιού σε true, για να μην μπορεί να το επιλέξει ο χρήστης. Επειτα χρησιμοποιούμε τον πίνακα μέσω της HTML .

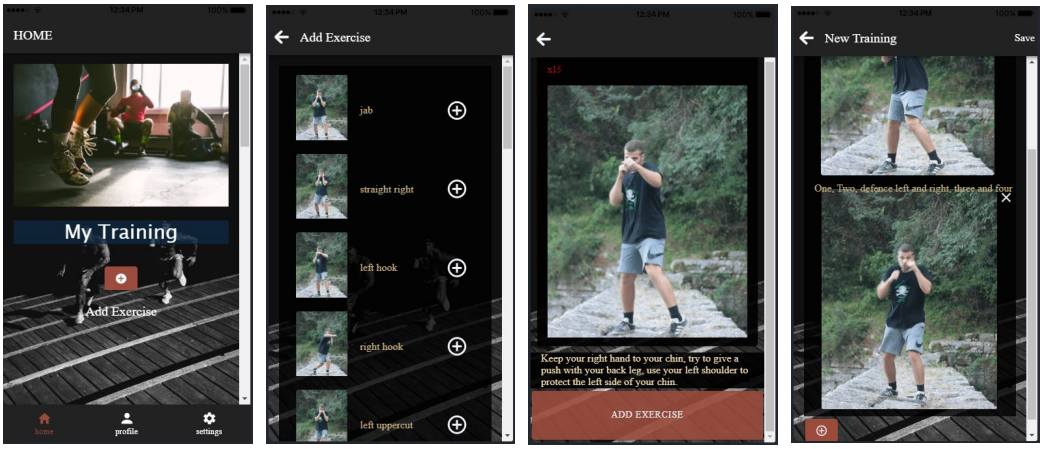
Σχήμα 25: Παράδειγμα HTML της Home Page

```
<div *ngFor="let lesson of BegginerLessons let i=index">
  <ion-button id={{lesson.lessonID}} expand="block" size="large" (click)="navigateTo(lesson.lessonID,slide.title)" ngDefaultControl [(ngModel)]="BegginerLessons" disabled = {{lesson.disabled}} >Lesson {{lesson.lessonID}}
    <div *ngIf="lesson.disabled == true;">
      <ion-icon slot="end" name="checkmark-circle"></ion-icon>
    </div>
  </ion-button>
</div>
```

Παρακάτω βλέπουμε το τέταρτο μέρος του slider, το οποίο δίνει τη δυνατότητα στο χρήστη να φτίαξει το δικό του πλάνο εξάσκησης:

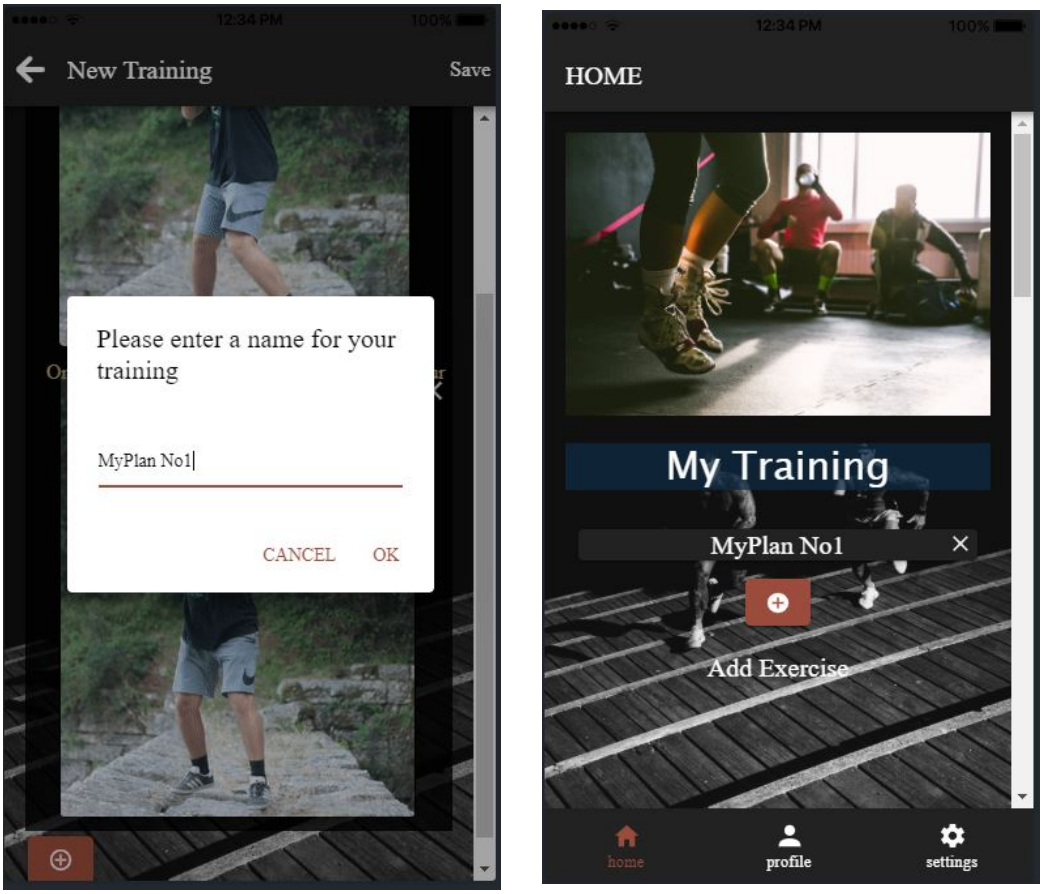
Οταν ο χρήστης πατήσει το κουμπί, μεταφέρεται στη σελίδα **Add Exercise**, εκεί υπάρχουν όλες οι ασκήσεις, τις οποίες μπορεί να προσθέσει στο πλάνο του πατώντας το κουμπί στα δεξία της άσκησης. Στη συνέχεια μεταφέρεται σε μία νέα σελίδα που υπάρχει η συγκεκριμένη άσκηση με την εξήγηση της και αν πατήσει το κουμπί στο κάτω μέρος της σελίδας, την προσθέτει στο πλάνο του. Στο παρασκήνιο, δημιουργείται ένας πίνακας με τη συγκεκριμένη άσκηση στη μνήμη της συσκευής με το κάλεσμα της **DataService**, ο χρήστης μεταφέρεται στη σελίδα **New Training** η οποία ελέγχει τη μνήμη της συσκευής για αλλαγές και αν εντοπίσει κάποια την εμφανίζει αυτόματα στο χρήστη. Επίσης δίνει στο χρήστη τη δυνατότητα να αφαιρέσει και να προσθέσει ασκήσεις στο πλάνο του. Αν επιλέξει να προσθέσει κάποια, πατώντας το κουμπί μεταφέρεται στη σελίδα **Add Exercise** και επαναλαμβάνει τη διαδικασία. Αν επιλέξει να αφαιρέσει, καλείται πάλι η **DataService** και διαγράφει την άσκηση από τον πίνακα.

Σχήμα 26: Home Page user’s custom training No.1



Αποθηκεύει το πλάνο του πατώντας το κουμπί Save και του ζητείται ένα όνομα για το πλάνο. Με το που δώσει κάποιο όνομα μεταφέρεται στη Home Page όπου μπορεί να επιλέξει το πλάνο του ανα πάσα στιγμή.

Σχήμα 27: Home Page user’s custom training No.2

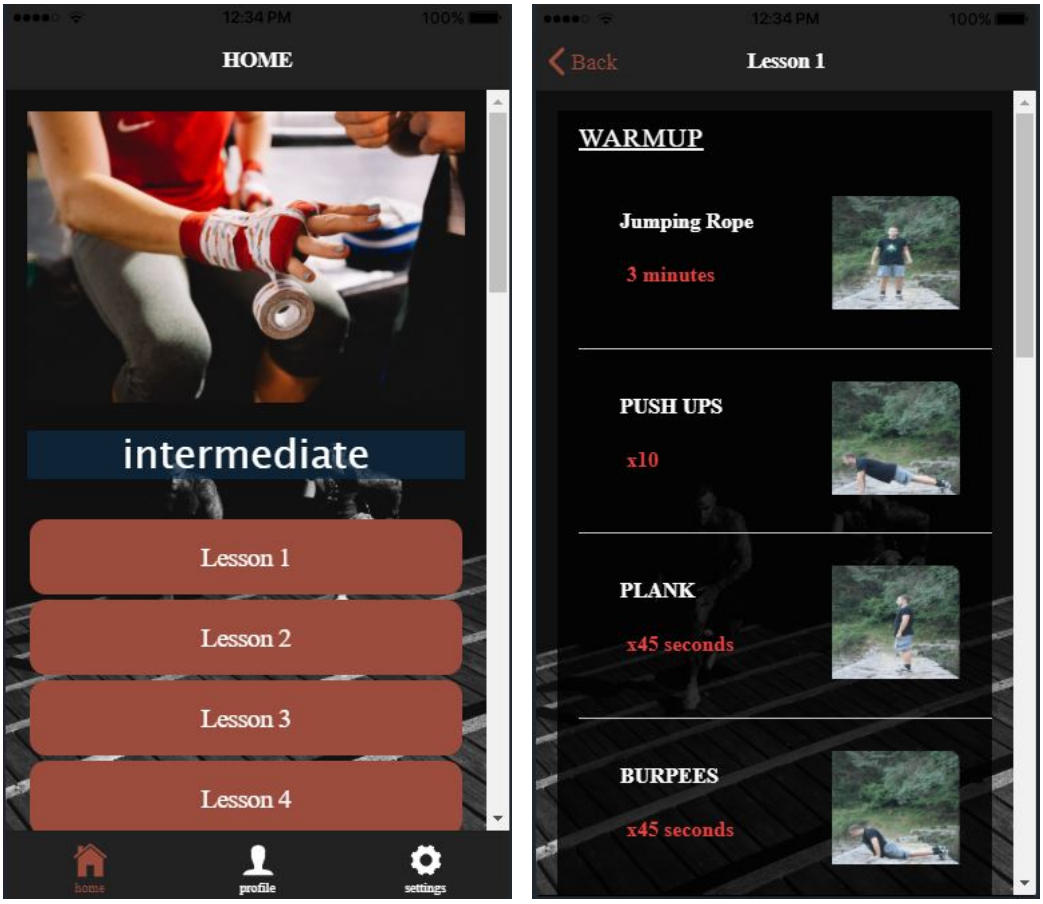


Είναι σημαντικό να αναφερθεί οτι για να γίνουν αυτόματα αυτές οι αλλαγές, έχουν δημιουργηθεί μια σειρά από **Observables** στη **DataService**, τα οποία πυροδοτούνται με κάθε αλλαγή του **Local-Storage** και ενημερώνουν όλα τα στοιχεία της εφαρμογής. Χωρίς αυτά για να έβλεπε της αλλαγές ο χρήστης θα έπρεπε κάθε φορά να ανανεώνει τη σελίδα, κάτι που θα έκανε την εφαρμογή αδύνατη για χρήση.

Περνώντας στο κομμάτι των μαθημάτων. Το πρόσωπο που απεικωνίζεται είμαι εγώ ο ίδιος, για να το επιτύχω αυτό είχα τη βοήθεια δύο καλών μου φίλων από το τμήμα τεχνών ήχου και εικόνας, τον Δημήτρη Γκρίντζο και τον Σπύρο Καβαδία, τους οποίους και ευχαριστώ. Παρακάτω βλέπουμε τη διαδικασία που ακολουθεί ο χρήστης.

7.3 Lesson Page

Σχήμα 28: Lesson Page No.1



Σχήμα 29: Παράδειγμα κώδικα της Home Page

```
async navigateTo(lessonPicked, Navigation) {
  this.NavigateTo = Navigation + '/' + lessonPicked;
  this.router.navigateByUrl(this.NavigateTo);
}
```

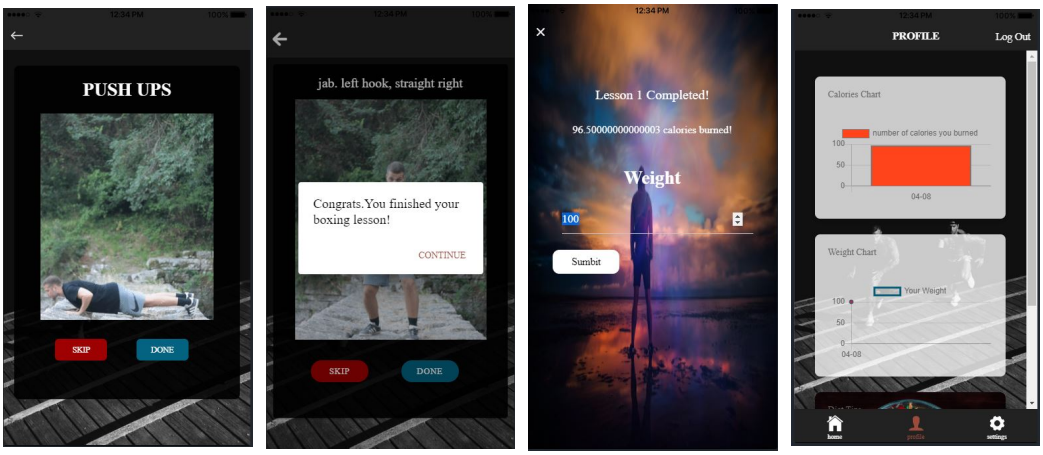
Σχήμα 30: Παράδειγμα κώδικα της Lesson Page

```
ngOnInit() {
  this.lessonId = this.route.snapshot.paramMap.get('id');
```

Οταν ο χρήστης επιλέγει ένα μάθημα από κάποιο πρόγραμμα, μεταφέρεται στην αναλόγη σελίδα (begginer, intermadiate ή advnanced), το περιεχόμενο της οποίας αλλάζει δυναμικά, για παράδειγμα ο αριθμός του μαθήματος στο πάνω μέρος της σελίδας παρέχεται από τη Home Page μέσω της διαδρομής που επιλέγει ο χρήστης. Η σελίδα στην οποία μεταφέρεται ο χρήστης, από τη μερία της, λαμβάνει το id και το χρησιμοποιεί αναλόγως. Για καλύτερη κατανόηση στα αριστερά, βλέπουμε τα σχήματα 29 και 30.

Ο χρήστης ξεκινάει το πρόγραμμα πατώντας το κουμπί στο κάτω μέρος της σελίδας και μεταφέρεται σε μια νέα σελίδα η οποία αποτελείται από ένα slider με της ασκήσεις. Για να συνεχίσει έχει δύο επιλογές, τα κουμπιά Skip και Done, αναλόγως το κουμπί προστίθενται ή όχι θερμίδες μέχρι και την τελευταία άσκηση. Στην συνέχεια μεταφέρεται σε μια σελίδα όπου του ζητείται το βάρος του, αν θέλει το βάζει και μεταφέρεται στην Profile Page, όπου μπορέι να δει τα διαγράμματα των θερμίδων και του βάρους του.

Σχήμα 31: Lesson Page



Στην σελίδα τών ασκήσεων βλέπουμε το κομμάτι του κώδικα που ευθύνεται για την διαχείρηση των θερμίδων. Πατώντας το κουμπί Done, οι θερμίδες κάθε άσκησης προστίθενται σε ένα άθροισμα, με το κουμπί Skip δίνεται στη συνάρτηση **Done** μηδέν ως παράμετρος, για να μην επιρεάζει το άθροισμα.

Σχήμα 32: Παράδειγμα HTML της Lesson Page

```
<ion-card *ngFor="let w of (this.FullWork | slice: noOfItem - 1:noOfItem)">
  <ion-card-header>
    <ion-card-title>
      {{ w.title }}
    </ion-card-title>
    <ion-card-content>
      <img src="{{ w.url }}" />
      <ion-button color="danger" size="small" shape="round" (click)="done(0)">SKIP</ion-button>
      <ion-button slot="end" color="success" size="small"shape="round"(click)="done(w.calories)">DONE</ion-button>
    </ion-card-content>
  </ion-card-header>
</ion-card>
```

Στη μεριά της TypeScript η συνάρτηση **done** καλείται κάθε φορά που πατιέται ένα από τα κουμπιά. Οι θερμίδες προστίθενται σε μια μεταβλητη όσο ο αριθμός τον αντικειμένων είναι μικρότερος από το μήκος του πίνακα των ασκήσεων. Οταν ολοκληρωθεί το πρόγραμμα δημιουργείται ο πίνακας **ChartData** με τις τρείς τιμές που βλέπουμε και δίνεται ως παράμετρος στη συνάρτηση **setData** της **DataService** για να αποθηκευτεί στη μνήμη της συσκευής. Τέλος η **DataService** πυροδοτεί το ανάλογο **Observable** και ενημερώνεται η Home Page για την απενεργοποίηση του κουμπιού, και η Profile Page για την προβολή του διαγράμματος των θερμίδων.

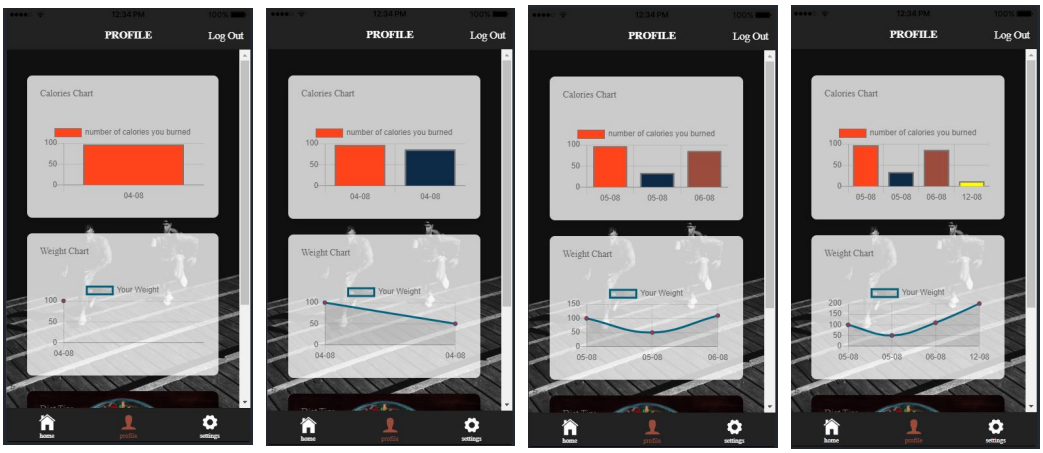
Σχήμα 33: Συνάρτηση done της Lesson Page

```
async done(calories) {

  var x = +calories;

  this.burned += x;

  if (this.noOfItem < this.FullWork.length) {

    this.noOfItem += 1;

  } else {

    const ChartData = {

      lessonId: this.lessonId,
      calories: this.burned,
      date: this.jstoday

    };

    this.dataService.setData(this.UidBegginer, ChartData);

    this.Finish();

  }
}
```

7.4 Profile Page

Οταν ο χρήστης πλοηγηθεί στην Profile Page βλέπει τα διαγράμματα θερμίδων και βάρους. Στο σχήμα που ακολουθεί βλέπουμε τα διαγράμματα που έχουν δημιουργηθεί για διαφορετικα ενδεχόμενα τεσσάρων ολοκληρωμένων μαθημάτων.

Σχήμα 34: Profile Page



Για τα διαγράμματα χρησιμοποιήθηκε η τεχνολογία Chart.js. Για να λειτουργήσει ένα τέτοιο διαγράμμα απλά του δίνουμε τα απαραίτητα δεδομένα και διαλέγουμε τον τύπο διαγράμματος που θέλουμε. Παρακάτω υπάρχει ενα σχήμα με τον κώδικα των δύο διαγραμμάτων που υπάρχουν στη Profile Page.

Σχήμα 35: Συναρτήσεις διαγραμμάτων της Profile Page

```
getChart(context, chartType, data, options?) {
  return new chartJs(context, {
    data,
    options,
    type: chartType
  });
}

getCaloriesChart(Calories, Dates, BackgroundColors) {
  const data = {
    labels: Dates,
    datasets: [{
      label: 'number of calories you burned',
      data: Calories,
      backgroundColor: BackgroundColors,
      borderWidth: 2,
      borderColor: '#777'}]
  };

  const options = {
    scales: {
      yAxes: [{
        ticks: { beginAtZero: true}
      }],
      xAxes: [{
        ticks: { beginAtZero: true}
      }]
    }
  };

  return this.getChart(this.CaloriesChartCanvas.nativeElement, 'bar', data, options);
}
```

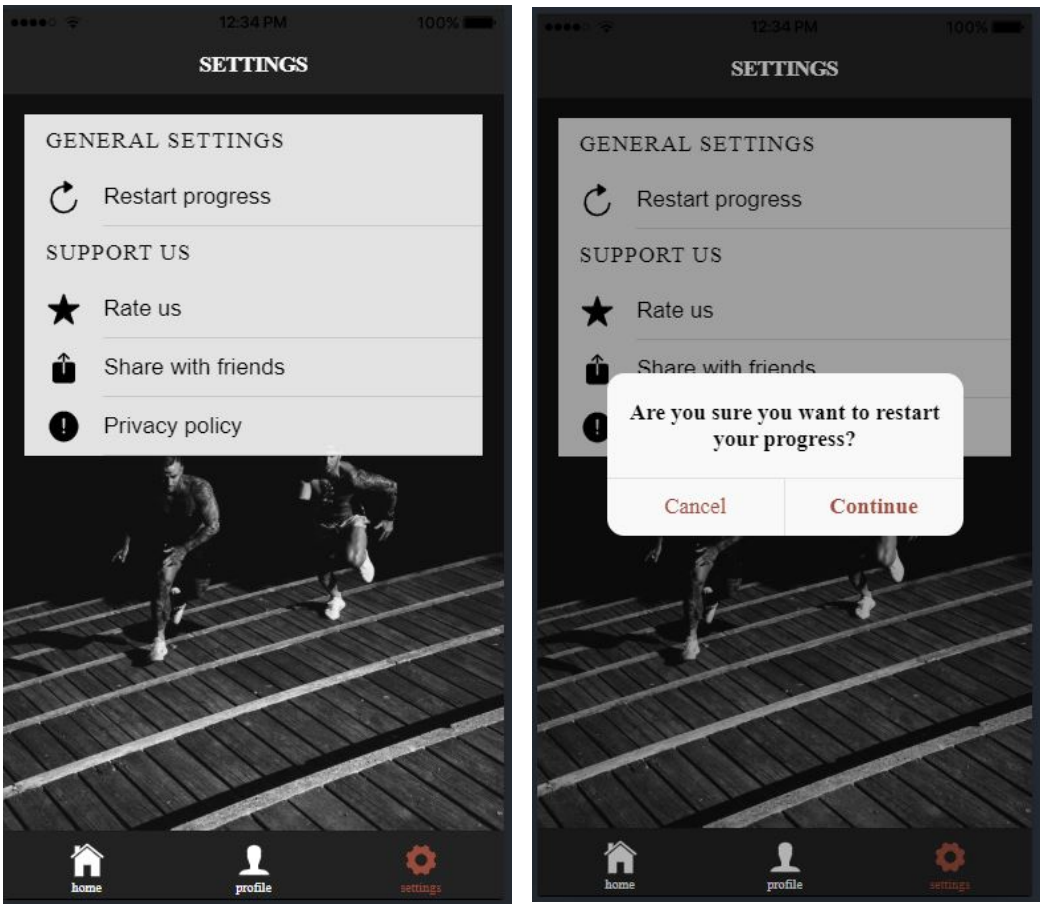
```
getWeightChart(Weight, Dates) {

  const data = {

    labels: Dates,
    datasets: [{
      label: 'Your Weight',
      data: Weight,
      borderColor: 'rgb(4, 99, 128)',
      pointBackgroundColor: 'rgb(192,41,66)',
    }]

  };

  const options = {

    scales: {
      yAxes: [{
        ticks: {
          beginAtZero:
          true
        }
      }],
      xAxes: [{
        ticks: {
          beginAtZero:
          true
        }
      }]
    }
  }

  return this.getChart(this.WeightChartCanvas.nativeElement, 'line', data, options);
}
```

Στο πάνω μέρος της αριστερής εικόνας υπάρχει η συνάρτηση **getChart** η οποία είναι υπεύθυνη για την παρουσίαση του διαγράμματος στην οθόνη. Χρειάζεται τεσσερις τιμές, **περιεχόμενο, τύπο διαγράμματος, δεδομένα και επιλογές**. Η συνάρτηση **getCaloriesChart** καλείται από την **ngOnInit**. Στην ngOnInit πέρνουμε τις τιμές των θερμίδων και ημερομηνιών (**Calories, Dates**) από την μνήμη της συσκεύης (**Local Storage**) και τις εισάγουμε στη συνάρτηση μαζί με έναν πίνακα με RGB τιμές χρωμάτων. Στη συνέχεια βάζουμε αυτές τις τιμές στις απαραίτητες μεταβλητές σύμφωνα με τις οδηγίες χρήσης του Chart.js και τέλος καλούμε τη συνάρτηση **getChart**.

Στη συνάρτηση **getWeightChart** της δεξιά εικόνας, το μόνο που αλλάζει είναι οτι οι τιμές των χρώμάτων του διαγράμματος είναι σταθερές και όταν καλούμε την συνάρτηση **getChart** δίνουμε για τύπο διαγράμματος την τιμή **'line'**

8 Settings Page

Σχήμα 36: Settings Page



Στην σελίδα των ρυθμίσεων υπάρχουν κάποιες επιλογές. Για τα εικονίδια των κουμπιών έχει χρησιμοποιήθει η βιβλιοθήκη **ionicons** της ionic που περιέχει εικονίδια για όλες τις χρήσεις. Επιλέγοντας το κουμπί με την ονομασία Restart progress αρχικά εμφανίζεται ένα παράθηρο που ρωτάει το χρήστη αν θέλει να συνεχίσει, στη συνέχεια γίνεται επανεκίνηση της προόδου του. Παρακάτω υπάρχει η εικόνα με τον κώδικα όπου και εξηγείται η λειτουργία αυτή.

Σχήμα 37: Συνάρτηση Reset της Settings Page

```
ngOnInit() {

  this.afAuth.authState.subscribe(user => {
    if (user) {
      this.Uid = user.uid;
      this.UidWeight = user.uid + '/weight';
      this.UidBegginer = user.uid + '/begginer';
      this.UidIntermediate = user.uid + '/intermediate';
      this.UidAdvanced = user.uid + '/advanced';

      this.keysToRemove = [this.Uid, this.UidWeight, this.UidBegginer, this.UidIntermediate, this.UidA
    }
  });


async Reset() {
  const alert = await this.alertController.create({
    header: 'Are you sure you want to restart your progress?',
    buttons: [
      { text: 'Cancel' },
      {
        text: 'Continue',

        handler: () => {
          for (let key of this.keysToRemove) {
            this.dataService.deleteData(key);
          }
          setTimeout(() => { this.windowReload(); }, 1000);
        }
      }
    ],
    backdropDismiss: false,
  });
  await alert.present();
}

async windowReload() {
  await window.location.reload();
}
```

Αρχικά, χρησιμοποιώντας την συνάρτηση **authState** της **AngularFireAuth**, ελέγχουμε αν είναι συνδεδεμένος ο χρήστης και ανακτούμε το id του, **user.uid**, από τη βάση δεδομένων με τα στοιχεία των χρηστών της firebase. Δημιουργούμε strings με το uid κάθε χρήστη συν μια συγκεκριμένη λέξη, παραδείγματος χάρη ”/weight” γιατί θέλουμε να αναφερθούμε στον πίνακα με αυτή την ονομασία στον αποθηκευτικό χώρο της συσκευής. Στη συνέχεια μεταφέρουμε όλα τα strings στον πίνακα **keysToRemove**. Οταν ο χρήστης πατήσει το κουμπί **Restart progress** τρέχει η συνάρτηση **Reset** και εμφανίζει το alert με την ερώτηση ασφαλείας. Οταν πατήσει το κουμπί **Continue**, τρέχει μια **for loop**, όπου καλείται η συνάρτηση **deleteData** της global **dataService**, για κάθε στοιχείο του πίνακα **keysToRemove**. Τέλος η συνάρτηση **setTimeout**, καλεί τη συνάρτηση **windowReload**, η οποία είναι υπεύθηνη για την επαναφόρτωση της σελίδας, μετά από μια παύση ενός δευτερολέπτου.

9 Συμπεράσματα και Μελλοντικές Επεκτάσεις

Η δημιουργία μιας εφαρμογής είναι μια αρκετά χρονοβόρα διαδικασία που απαιτεί αρκετή έρευνα και πολλή προεργασία, ειδικά σε άτομα που μαθαίνουν της απαραίτητες γλώσσες προγραμματισμού ενώ υλοποιούν την εκάστοτε εφαρμογη. Μέσα από τις δυσκολίες που αντιμετώπισα απέκτησα σημαντικές γνώσεις στον τομέα του προγραμματισμού αλλά και στις τεχνολογίες των έξυπνων κινητών συσκευών, που αναμφίβολα θα με βοηθήσουν στη μετέπειτα σταδιοδρομία μου.

Όσον αφορά μελλοντικές επεκτάσεις της εφαρμογής, είναι αλήθεια ότι θα μπορούσα να τη βελτιώνω συνεχώς προσθέτοντας καινούρια χαρακτηριστικά κάθε φορά. Κάποια από αυτά τα χαρακτηριστικά είναι η μετάφραση της διεπαφής χρήστη της εφαρμογής στις πιο δημοφιλείς ξένες γλώσσες για να γίνει προσιτή και σε χρήστες που δεν γνωρίζουν αγγλικά, καθώς και η δυνατότητα σύνδεσης μέσω λογαριασμού άλλης εφαρμογής κοινωνικής δικτύωσης όπως το Facebook ή η Google. Επίσης θα μπορούσαν να εισαχθούν περισσότερες ασκήσεις και προγράμματα διατροφής απο ειδικούς, όπως και η δυνατότητα ορισμού υπενθυμίσεων σε συγκεκριμένες ώρες από το χρήστη. Τέλος, η εφαρμογή θα ήταν δυνατόν να αναβαθμιστεί έτσι ώστε να καλύπτει περισσότερες πολεμικές τέχνες.

10 Δικτυογραφία-Βιβλιογραφία

Repository της εφαρμογής στο Github: https://github.com/GeorgeDoit/PtixiakiLatest

Panhale, Mahesh. “Beginning Hybrid Mobile Application Development,” n.d., 229.

Ravulavaru, Arvind. Learning Ionic: Build Hybrid Mobile Applications with HTML5, SCSS, and Angular, 2017.

Arvind Ravulavaru Learning Ionic Build Real-Time and Hybrid Mobile Appl ications with Ionic-Packt Publishing 2015

Coury, Felipe, Ari Lerner, Nate Murray, and Carlos Taborda. Ng-Book: The Complete Guide to Angular 4, 2017.

Huber, Thomas Claudius. Getting Started with TypeScript: Includes , Intro duction to Angular ; Write Professional JavaScript Code That Scales, Use Interfac es and Classes to Build Robust Code, Learn about Generics, Modules, Arrow Functi ons, Decorators, Declarations, Npm and Much More. 1st edition. Erscheinu ngsort nicht ermittelbar: Thomas Claudius Huber, 2017.

https://cordova.apache.org/docs/en/latest/guide/overview/

https://www.draw.io/

https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app-

https://nordicapis.com/what-is-the-difference-between-an-api-and-an-sdk/

https://blog.codecentric.de/en/2014/11/ionic-angularjs-framework-on-the-ris e/

https://www.freecodecamp.org/news/a-deeply-detailed-but-never -definitive-guide-to-mobile-development-architecture-6b01ce3b1528/

https://www.webmd.com/diet/ss/slideshow-best-diet-tips-ever

https://medium.com/@javier.ramos1/ionic-4-all-you-need-to-know-d2b9627 aaf03

https://www.quora.com/What-is-AngularJS-in-simple-words