ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής σε iOS περιβάλλον

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΠΆΝΤΟΥ ΘΩΜΆ

(2319)

Επιβλέπων : Δημήτριος Βέργαδος

Ιδιότητα

Καστοριά Μάρτιος 2024

Η παρούσα σελίδα σκοπίμως παραμένει λευκή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής σε iOS περιβάλλον

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΠΆΝΤΟΥ ΘΩΜΆ

(2319)

Επιβλέπων : Δημήτριος Βέργαδος

Ιδιότητα

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ημερομηνία εξέτασης

.....	.....	.....
Ον/μο Μέλους	Ον/μο Μέλους	Ον/μο Μέλους
Ιδίοτητα Μέλους	Ιδίοτητα Μέλους	Ιδίοτητα Μέλους

Καστοριά Μάρτιος 2024

Ευχαριστίες

Θέλω να ευχαριστήσω ιδιαιτέρως τον κ. Δημήτριο Βέργαδο , για την εκτίμηση, τις συμβουλές και την καθοδήγησή του κατά την εκπόνηση της συγκεκριμένης πτυχιακής εργασίας και της εμπιστοσύνης που μου έδειξε καθ' όλη την διάρκεια της συγγραφής της. Επιπροσθέτως θέλω να ευχαριστήσω θερμά την μητέρα μου για την ανιδιοτελή στήριξη της κατά την διάρκεια των σπουδών μου.

Περίληψη

Η παρούσα πτυχιακή εργασία αφορά την ανάπτυξη μιας εφαρμογής για iOS για ένα δημοσιογραφικό κανάλι. Η εφαρμογή σχεδιάζεται για να παρέχει στους χρήστες πρόσβαση σε περιεχόμενο που παράγεται από το κανάλι, συμπεριλαμβανομένων ειδήσεων, άρθρων και άλλων πολυμεσικών περιεχομένων. Η εφαρμογή σχεδιάστηκε να προσφέρει μια εύχρηστη διεπαφή (interface) που επιτρέπει στους χρήστες να περιηγηθούν εύκολα στο περιεχόμενο και να αλληλοεπιδράσουν με αυτό, όπως να κοινοποιούν το περιεχόμενο και να αποθηκεύουν άρθρα για αργότερη πρόσβαση. Η ανάπτυξη αυτής της εφαρμογής αντιπροσωπεύει έναν καινοτόμο τρόπο για το δημοσιογραφικό κανάλι να διανέμει το περιεχόμενο του και να αλληλοεπιδρά με το κοινό του μέσω των κινητών συσκευών iOS.

Λέξεις Κλειδιά: iOS εφαρμογή , βάση δεδομένων, χρήστης , ειδήσεις , άρθρα , δημοσιογραφικό κανάλι

Abstract

This thesis is about the development of an iOS application for a media channel. The app is designed to provide users with access to content produced by the channel, including news, articles and other multimedia content. The app is designed to provide an easy-to-use interface that allows users to easily navigate and interact with content, such as sharing content and saving articles for slower access. The development of this app represents an innovative way for the news outlet to distribute its content and interact with its audience via iOS mobile devices.

Key Words: iOS application, database , user , news articles, media channel

Πίνακας Περιεχομένων

Περιεχόμενα

Εισαγωγή *1*

1) Κεφάλαιο 1 Τι είναι το iOS *3*

1.1 Γλώσσα Swift 3

1.2 Γιατί Swift; 3

1.3 Διαφορές UIKit με SwiftUI 4

1.4 Γιατί SwiftUI αντί για UIKit ; 5

1.5 Περιβάλλον υλοποίησης 6

1.6 Δημιουργία εφαρμογής 6

1.7 Θέμα εφαρμογής 6

1.8 Διάγραμμα UML 6

1.9 Αρχιτεκτονική iOS 7

1.9 Design Pattern 10

Κεφάλαιο 2: Δομή της εφαρμογής 1/2 : Αρχικές Οθόνες *10*

2.1 Είσοδος/Εγγραφή στην εφαρμογή: 10

2.2 Κλάση του ViewModel: 14

2.2 Δημιουργία κατηγοριών: 15

2.3 Οθόνη Profile: 16

2.4 Εγγραφή χρήστη: 16

2.5 Εντοπισμός (Localization) 18

2.6 Οφέλη του εντοπισμού εφαρμογών για κινητά τηλέφωνα 19

2.7 Πως κάνουμε localize την εφαρμογή 19

Κεφάλαιο 3: Δομή της εφαρμογής 2/2 : Κεντρικές οθόνες και backend *22*

3.1 Menu Οθόνη 22

3.2 Backend Firebase 26

3.3 Σύνδεση backend με την εφαρμογή 28

Κεφάλαιο 4 Τι είναι το testing? *30*

4.0 Testing 30

4.1 UI Testing 30

4.2 UI Testing Execution 32

Κεφάλαιο 5 Εκτέλεση της εφαρμογής *36*

5.1 Έναρξη του περιβάλλοντος XCode 36

5.2 Εκτέλεση εγγραφής χρήστη 37

5.3 Σύνδεση χρήστη 38

5.4 Welcome Screen & Logout 39

5.5 Αρχικό μενού 43

5.5.1 Home screen 43

5.5.2 Interaction του χρήστη με την εφαρμογή 46

5.5.3 Profile View 47

5.5.4 Voice Over View 48

5.6 Κατηγορίες ειδήσεων 49

Συμπεράσματα..... 50

Βιβλιογραφία..... 51

Λίστα Εικόνων

Εισαγωγή

Στο διαρκώς εξελισσόμενο τοπίο της ανάπτυξης εφαρμογών για κινητά, υπάρχει μια βαθιά ανάγκη για πλαίσια(framework) που όχι μόνο βελτιώνουν τη διαδικασία αλλά και επιτρέπουν τη δημιουργία αισθητικά ευχάριστων και εξαιρετικά λειτουργικών εφαρμογών. Μέσα σε αυτό το πλαίσιο, το SwiftUI αναδεικνύεται ως μια μετασχηματιστική δύναμη, επαναπροσδιορίζοντας τον τρόπο με τον οποίο σχεδιάζουμε και αναπτύσσουμε εφαρμογές για το οικοσύστημα της Apple.

Η παρούσα πτυχιακή εργασία ξεκινάει την εξερεύνηση του SwiftUI, ενός πρωτοποριακού framework που εισήγαγε η Apple, και εισχωρεί βαθιά στον κόσμο της ανάπτυξης εφαρμογών που υποστηρίζεται από αυτή την επαναστατική τεχνολογία. Το SwiftUI, που παρουσιάστηκε το 2019, εγκαινίασε μια νέα εποχή ανάπτυξης εφαρμογών για iOS, macOS, watchOS και tvOS. Ως κύριο αντικείμενο αυτής της διπλωματικής εργασίας, θα αναλύσουμε το SwiftUI, εξετάζοντας τις βασικές αρχές, τη δομή και τις δυνατότητές του. Το SwiftUI ξεχωρίζει υιοθετώντας μια δηλωτική σύνταξη που μειώνει δραματικά την πολυπλοκότητα της ανάπτυξης UI. Αντί για το προστακτικό στυλ που συναντάται σε προηγούμενες προσεγγίσεις, το SwiftUI ενθαρρύνει τους προγραμματιστές να περιγράψουν πώς πρέπει να είναι η διεπαφή χρήστη(user interface), αφήνοντας το framework να χειριστεί τα υπόλοιπα. Αυτή η αλλαγή παραδείγματος φέρνει μαζί της μια πληθώρα πλεονεκτημάτων, όπως η βελτιωμένη αναγνωσιμότητα του κώδικα, η συντήρηση και η δυνατότητα δημιουργίας προσαρμοστικών, διαπλατφορμικών(cross-platform) διεπαφών χρήστη με ευκολία.

1)Κεφάλαιο 1 Τι είναι το iOS

iOS που σημαίνει iPhone Operating System είναι ένα λειτουργικό σύστημα που δημιουργήθηκε από την Apple για όλα τα κινητά της. Η ονομασία iOS εφαρμόστηκε επίσημα στο λογισμικό το 2008 όταν η Apple κυκλοφόρησε το SDK για το iPhone , επιτρέποντας σε οποιονδήποτε κατασκευαστή εφαρμογών να δημιουργήσει εφαρμογές για την πλατφόρμα.

1.1 Γλώσσα Swift

Η Swift είναι η σύγχρονη γλώσσα που δημιούργησε η Apple για το iOS και έκανε το κόσμο των προγραμματιστών της Apple να παραμιλάει τον Ιούνιο του 2014 – αλλά γιατί δημιουργήθηκε η Swift; Αν και αγαπήθηκε από πολλούς προγραμματιστές του iOS η Objective – C(αντικειμενοστραφή C) θεωρήθηκε από ορισμένους ως μια ξεπερασμένη γλώσσα. Μια γλώσσα πάνω από 30 χρόνια και βασισμένη στη C , είχε μια φλύαρη και ιδιόμορφη σύνταξη, με ένα μη ασφαλές σύστημα τύπων. Χτισμένη ως μια σύγχρονη εναλλακτική της Objective- C , η Swift σχεδιάστηκε με συγκεκριμένες βελτιώσεις, συγκεκριμένα:

-Ασφάλεια :

Η Swift εισήγαγε διάφορες έννοιες προγραμματισμού για να μειώσει ορισμένα κοινά λάθη των προγραμματιστών. Αυτά που περιλαμβάνουν την ισχυρή τυποποίηση και το χειρισμό σφαλμάτων.

-Απόδοση:

Η Apple εισήγαγε εσωτερικές βελτιστοποιήσεις για να διασφαλίσει ότι η Swift εκτελείται γρήγορα. Το Xcode(περιβάλλον υλοποίησης) παρέχει επίσης προειδοποιήσεις για να διασφαλίσει ότι η εφαρμογή εκτελείται βέλτιστα.

-Εκφραστικότητα :

Ο εκφραστικός κώδικας διατηρεί τη σωστή ισορροπία μεταξύ της σαφήνειας του νοήματος και της συντομίας. Η Swift αντλεί διδάγματα από την Objective – C και άλλες γλώσσες για να εισάγει αρκετές έννοιες που μπορεί να είναι καινούριες στην αρχή αλλά με το καιρό γίνονται απαραίτητες για τον δημιουργό.

1.2 Γιατί Swift;

Η Swift έχει γνωρίσει μια αλματώδη άνοδο στη δημοτικότητα από την δημιουργία της. Κατατάσσεται τακτικά ως μία από τις πιο αγαπημένες γλώσσες προγραμματισμού στο StackOverFlow. Σύμφωνα με την RedMonk, « η Swift αναπτύσσεται ταχύτερα από οτιδήποτε άλλο παρακολουθούμε». Από τη μία πλευρά , υπάρχουν ορισμένες σημαντικές διαφορές μεταξύ των γλωσσών. Για παράδειγμα , η Objective – C έχει πολύ διαφορετικές προσεγγίσεις από τη Swift όσον αναφορά τις επικεφαλίδες κλάσεων, την ασφάλεια τύπων , τιμές nil και το χειρισμό σφαλμάτων. Από την άλλη πλευρά ,

ορισμένες διαφορές είναι πραγματικά μόνο θέμα σύνταξης. Από κάτω μια σύγκριση ίδιου κώδικα σε Swift και Objective – C

```
UIView.animate(withDuration: 1) {
    self.yellowView.alpha = 0
}

[UIView animateWithDuration:1.0 animations:^{
    self.yellowView.alpha = 0.0;
}];
```

Εικόνα 1.3.1 Σύγκριση Swift με Objective

Η Swift δεν περιορίζεται απαραίτητα στην ανάπτυξη εφαρμογών iOS . Η Swift χρησιμοποιείται σε όλες τις πλατφόρμες της Apple από το macOS έως το iOS και το watchOS, Η Apple ανακοίνωσε το 2015 ότι η Swift θα γινόταν open source. Η IBM ήταν ο πρώτος κολοσσός που υιοθέτησε αυτή την ιδέα καθιστώντας τη Swift διαθέσιμη στους προγραμματιστές εταιρικών εφαρμογών στο IBM Cloud.

Η Swift περιέχει 2 frameworks, το UI Kit και τη SwiftUI. Στη παρακάτω εφαρμογή χρησιμοποιείτε το framework SwiftUI.

1.3 Διαφορές UIKit με SwiftUI

Το UIKit και το SwiftUI είναι και τα δύο frameworks για τη δημιουργία διεπαφών χρήστη σε πλατφόρμες της Apple, όπως το iOS και το iPadOS, αλλά υπάρχουν αρκετές διαφορές:

UIKit: είναι το παλαιότερο και πιο καθιερωμένο πλαίσιο. Υπάρχει από τις πρώτες εκδόσεις του iOS και παρέχει ένα ολοκληρωμένο σύνολο εργαλείων για την κατασκευή user interfaces.

SwiftUI: είναι ένα νεότερο πλαίσιο που εισήχθη το 2019 με το iOS 13. Έχει σχεδιαστεί για να είναι ένας πιο σύγχρονος, διαισθητικός τρόπος κατασκευής user interfaces, με έμφαση στην απλότητα και την προσβασιμότητα.

Στυλ ανάπτυξης: Το UIKit θεωρείται γενικά πιο πολύπλοκο από το SwiftUI, καθώς περιλαμβάνει τη χρήση πολλών διαφορετικών κλάσεων και πρωτοκόλλων για τη δημιουργία μιας διεπαφής. Το SwiftUI έχει σχεδιαστεί για να είναι πιο δηλωτικό και λιγότερο βαρύ με κώδικα, καθιστώντας το πιο κατανοητό και λιγότερο επιρρεπές σε σφάλματα.

Ενσωμάτωση με υπάρχοντα κώδικα: Το UIKit υπάρχει εδώ και πολύ περισσότερο καιρό και χρησιμοποιείται σε πολλές υπάρχουσες εφαρμογές iOS, οπότε είναι καλά ενσωματωμένο με τα υπάρχοντα εργαλεία και διαδικασίες ανάπτυξης iOS. Το SwiftUI είναι ακόμα σχετικά νέο και απαιτεί έναν διαφορετικό τρόπο σκέψης για την κατασκευή user interfaces, οπότε μπορεί να υπάρξει μια καμπύλη εκμάθησης για τους προγραμματιστές που είναι εξοικειωμένοι με το UIKit.

1.4 Γιατί SwiftUI αντί για UIKit ;

Το SwiftUI είναι ένα σύγχρονο πλαίσιο UI που εισήγαγε η Apple το 2019 και παρέχει έναν declarative τρόπο για τη δημιουργία διεπαφών χρήστη για τις πλατφόρμες iOS, macOS, watchOS και tvOS. Το SwiftUI προσφέρει πολλά πλεονεκτήματα σε σχέση με το παραδοσιακό πλαίσιο UIKit, μερικά από τα οποία επισημαίνονται παρακάτω:

1)Declarative σύνταξη: Το SwiftUI χρησιμοποιεί ένα δηλωτικό συντακτικό που επιτρέπει στους προγραμματιστές να περιγράφουν τη διεπαφή χρήστη με απλό και διαισθητικό τρόπο. Αυτή η προσέγγιση εξαλείφει την ανάγκη για χειροκίνητη διαχείριση της διάταξης και καθιστά τον κώδικα ευκολότερο στην ανάγνωση και τη συντήρηση.

2)Μειωμένη πολυπλοκότητα κώδικα: Το SwiftUI μειώνει την πολυπλοκότητα της δημιουργίας διεπαφών χρήστη παρέχοντας ένα σύνολο προκατασκευασμένων στοιχείων UI που μπορούν εύκολα να προσαρμοστούν και να συνδυαστούν για τη δημιουργία σύνθετων διατάξεων. Αυτό μειώνει την ποσότητα του κώδικα που απαιτείται για την κατασκευή μιας διεπαφής και απλοποιεί τη διαδικασία ανάπτυξης.

3) Ζωντανή προεπισκόπηση: Το SwiftUI παρέχει μια λειτουργία ζωντανής προεπισκόπησης που επιτρέπει στους προγραμματιστές να βλέπουν τις αλλαγές που κάνουν στο UI σε πραγματικό χρόνο. Αυτό διευκολύνει την επανάληψη και την τελειοποίηση του σχεδιασμού του UI κατά τη διάρκεια της διαδικασίας ανάπτυξης.

4) Υποστήριξη πολλαπλών πλατφορμών: Το SwiftUI έχει σχεδιαστεί για να λειτουργεί σε όλες τις πλατφόρμες της Apple, συμπεριλαμβανομένων των iOS, macOS, watchOS και tvOS. Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν συνεπείς και ενοποιημένες διεπαφές χρήστη για όλες τις εφαρμογές τους.

5) Απόδοση: Το SwiftUI είναι βελτιστοποιημένο για απόδοση και χρησιμοποιεί μια νέα μηχανή απόδοσης που έχει σχεδιαστεί για να εκμεταλλεύεται το σύγχρονο υλικό. Αυτό έχει ως αποτέλεσμα ταχύτερες και ομαλότερες διεπαφές χρήστη που παρέχουν καλύτερη εμπειρία χρήσης.

Συνολικά, το SwiftUI παρέχει έναν σύγχρονο, δηλωτικό και αποτελεσματικό τρόπο για τη δημιουργία διεπαφών χρήστη για πλατφόρμες της Apple, καθιστώντας το μια ελκυστική επιλογή για προγραμματιστές που επιθυμούν να δημιουργήσουν εφαρμογές υψηλής ποιότητας.

1.5 Περιβάλλον υλοποίησης

Το περιβάλλον που χρησιμοποιήθηκε για την δημιουργία της εφαρμογής είναι το Xcode, ένα εργαλείο σχεδιασμένο για δημιουργία, testing και διανομή εφαρμογών σε όλες τις πλατφόρμες της Apple.

1.6 Δημιουργία εφαρμογής

Η εφαρμογή δημιουργήθηκε με τη γλώσσα Swift στην οποία γράφεται iOS εφαρμογές και συγκεκριμένα χρησιμοποιήθηκε το framework SwiftUI. Οι βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής είναι η combine που ανήκει στην ήδη υπάρχουσα γλώσσα

1.7 Θέμα εφαρμογής

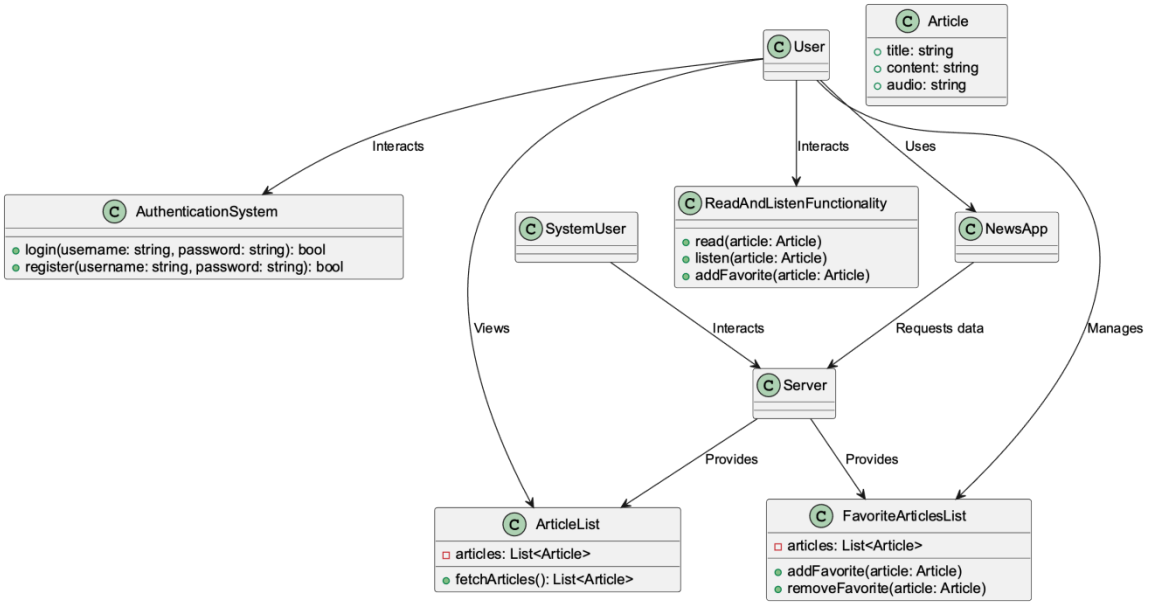
Η εφαρμογή αντιπροσωπεύει ένα δημοσιογραφικό κανάλι δίνοντας την δυνατότητα στο χρήστη εφόσον εγγραφεί να μπορεί να διαβάζει ή να ακούει άρθρα σχετικά με την ενότητα που ενδιαφέρεται.

Διάφορα παραδείγματα αντίστοιχων εφαρμογών είναι:

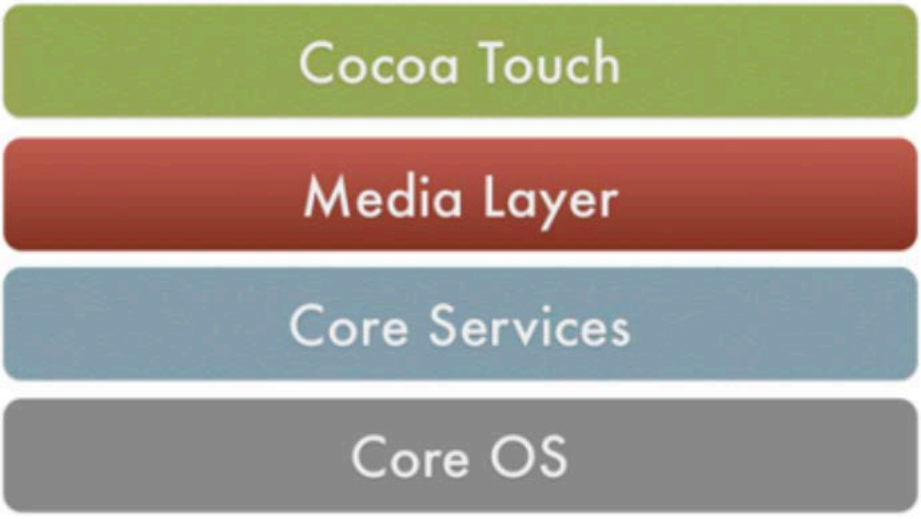
ΑΜΠΕ-ΜΠΕ,Καθημερινή, το Βήμα, ΕΡΤ, Πρώτο Θέμα όπως και αντίστοιχα αθλητικά δημοσιογραφικά κανάλια όπως Sport24, Contra.

1.8 Διάγραμμα UML

Στο παρακάτω διάγραμμα UML περιγράφεται το σύστημα της εφαρμογής , περιλαμβάνοντας τις βασικές κλάσεις όπου περιέχουν το user authentication , τη λίστα με τα άρθρα και την διαχείριση των αγαπημένων άρθρων. Οι χρήστες αλληλοεπιδρούν με το σύστημα μέσω του user interface , ενώ το σύστημα επικοινωνεί με τον server για την ανάκτηση δεδομένων. Οι χρήστες μπορούν να διαχειριστούν την λίστα των αγαπημένων τους άρθρων, όπως και να διαβάσουν ή να ακούσουν τα άρθρα χρησιμοποιώντας τη λειτουργικότητα της ακρόασης.



1.9 Αρχιτεκτονική iOS



Εικόνα 1.6.1 iOS Στρώσεις

1. Στρώση Core iOS

Το επίπεδο Core OS περιέχει τα χαρακτηριστικά χαμηλού επιπέδου πάνω στα οποία βασίζονται οι περισσότερες άλλες τεχνολογίες.

Core Bluetooth Framework.
Πλαίσιο επιτάχυνσης.
Πλαίσιο εξωτερικών εξαρτημάτων.
Πλαίσιο υπηρεσιών ασφαλείας.
Πλαίσιο τοπικού ελέγχου ταυτότητας.
Η υποστήριξη 64-bit από το IOS7 υποστηρίζει την ανάπτυξη εφαρμογών 64-bit και επιτρέπει την ταχύτερη εκτέλεση της εφαρμογής.

2. Στρώμα βασικών υπηρεσιών

Μερικά από τα σημαντικά πλαίσια που είναι διαθέσιμα στα στρώματα βασικών υπηρεσιών παρουσιάζονται λεπτομερώς:

Πλαίσιο βιβλίου διευθύνσεων - Παρέχει προγραμματιστική πρόσβαση σε μια βάση δεδομένων επαφών του χρήστη.
Πλαίσιο Cloud Kit - Παρέχει ένα μέσο για τη μετακίνηση δεδομένων μεταξύ της εφαρμογής σας και του iCloud.
Πλαίσιο Core data - Τεχνολογία για τη διαχείριση του μοντέλου δεδομένων μιας εφαρμογής Model View Controller.
Πλαίσιο Core Foundation - Διεπαφές που δίνουν θεμελιώδεις δυνατότητες διαχείρισης δεδομένων και υπηρεσιών για εφαρμογές iOS.
Πλαίσιο Core Location - Παρέχει πληροφορίες θέσης και κατεύθυνσης σε εφαρμογές.
Core Motion Framework - Πρόσβαση σε όλα τα δεδομένα που βασίζονται στην κίνηση και είναι διαθέσιμα σε μια συσκευή. Με τη χρήση αυτού του πλαισίου πυρήνα κίνησης είναι δυνατή η πρόσβαση σε πληροφορίες που βασίζονται στο επιταχυνσιόμετρο.
Foundation Framework - Objective C που καλύπτει πάρα πολλά από τα χαρακτηριστικά που βρίσκονται στο Core Foundation framework
Πλαίσιο Healthkit - Νέο πλαίσιο για το χειρισμό πληροφοριών σχετικά με την υγεία του χρήστη.
Πλαίσιο Homekit - Νέο πλαίσιο για την επικοινωνία και τον έλεγχο συνδεδεμένων συσκευών στο σπίτι του χρήστη.
Πλαίσιο Social - Απλή διεπαφή για την πρόσβαση στους λογαριασμούς των μέσων κοινωνικής δικτύωσης του χρήστη.
Πλαίσιο StoreKit - Παρέχει υποστήριξη για την αγορά περιεχομένου και υπηρεσιών μέσα από τις εφαρμογές του iOS, μια λειτουργία γνωστή ωςIn-App Purchase.

3. Στρώμα πολυμέσων: Η τεχνολογία γραφικών, ήχου και βίντεο ενεργοποιείται με τη χρήση του επιπέδου πολυμέσων.

Πλαίσιο γραφικών:

UIKit Graphics - Περιγράφει υποστήριξη υψηλού επιπέδου για το σχεδιασμό εικόνων και χρησιμοποιείται επίσης για την εμψύχωση του περιεχομένου των προβολών σας.

Πλαίσιο Core Graphics - Είναι η εγγενής μηχανή σχεδίασης για τις εφαρμογές iOS και παρέχει υποστήριξη για προσαρμοσμένη απόδοση 2D διανυσμάτων και εικόνων.

Core Animation - Είναι μια αρχική τεχνολογία που βελτιστοποιεί την εμπειρία κινούμενων σχεδίων των εφαρμογών σας.

Core Images - παρέχει προηγμένη υποστήριξη για τον έλεγχο βίντεο και εικόνων χωρίς κίνηση με μη καταστροφικό τρόπο.

OpenGl ES και GLKit - διαχειρίζεται την προηγμένη απόδοση 2D και 3D μέσω διεπαφών που επιταχύνονται από υλικό

Metal - Επιτρέπει πολύ υψηλές επιδόσεις για τις εξελιγμένες εργασίες απόδοσης και υπολογισμού γραφικών σας. Προσφέρει πρόσβαση με πολύ χαμηλή επιβάρυνση στην A7 GPU.

Πλαίσιο ήχου:

Είναι ένα πλαίσιο υψηλού επιπέδου που παρέχει απλή χρήση της βιβλιοθήκης iTunes του χρήστη και υποστήριξη για την αναπαραγωγή λιστών αναπαραγωγής.

AV Foundation - Είναι μια διεπαφή Objective C για το χειρισμό της εγγραφής και αναπαραγωγής ήχου και βίντεο.

OpenAL - Είναι μια τυποποιημένη τεχνολογία της βιομηχανίας για την παροχή ήχου.

Πλαίσιο βίντεο

Το AV Kit - πλαίσιο παρέχει μια συλλογή εύχρηστων διεπαφών για την παρουσίαση βίντεο.

AV Foundation - παρέχει προηγμένες δυνατότητες αναπαραγωγής και εγγραφής βίντεο.

Core Media - το πλαίσιο περιγράφει τις διεπαφές χαμηλού επιπέδου και τους τύπους δεδομένων για τη λειτουργία των μέσων.

4 Στρώμα Cocoa Touch

Πλαίσιο EventKit - παρέχει ελεγκτές προβολής για την προβολή των τυπικών διεπαφών του συστήματος για την προβολή και την τροποποίηση των συμβάντων που σχετίζονται με το ημερολόγιο

Πλαίσιο GameKit - υλοποιεί υποστήριξη για το Game Center που επιτρέπει στους χρήστες να μοιράζονται τις πληροφορίες που σχετίζονται με το παιχνίδι τους στο διαδίκτυο

Πλαίσιο iAd - σας επιτρέπει να παραδίδετε διαφημίσεις που βασίζονται σε banner από την εφαρμογή σας.

MapKit Framework - δίνει έναν κυλιόμενο χάρτη που μπορείτε να συμπεριλάβετε στη διεπαφή χρήστη της εφαρμογής σας.

PushKitFramework - παρέχει υποστήριξη εγγραφής για εφαρμογές VoIP.

Twitter Framework - υποστηρίζει ένα UI για τη δημιουργία tweets και υποστήριξη για τη δημιουργία URL για πρόσβαση στην υπηρεσία Twitter.

Πλαίσιο UIKit - παρέχει ζωτικής σημασίας υποδομή για την εφαρμογή γραφικών, οδηγούμενων από συμβάντα εφαρμογών στο iOS. Ορισμένες από τις σημαντικές λειτουργίες του πλαισίου UI Kit:

-Υποστήριξη πολλαπλών εργασιών.

- Βασική διαχείριση και υποδομή εφαρμογών.
- Διαχείριση διεπαφής χρήστη
- Υποστήριξη για συμβάντα αφής και κίνησης.
- Υποστήριξη αποκοπής, αντιγραφής και επικόλλησης και πολλά άλλα.

1.9 Design Pattern

Το μοτίβο που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής:

MVVM που είναι η συντομογραφία για το Model-View-ViewModel.

View – Οθόνη που παρουσιάζει τα δεδομένα προς τον χρήστη.

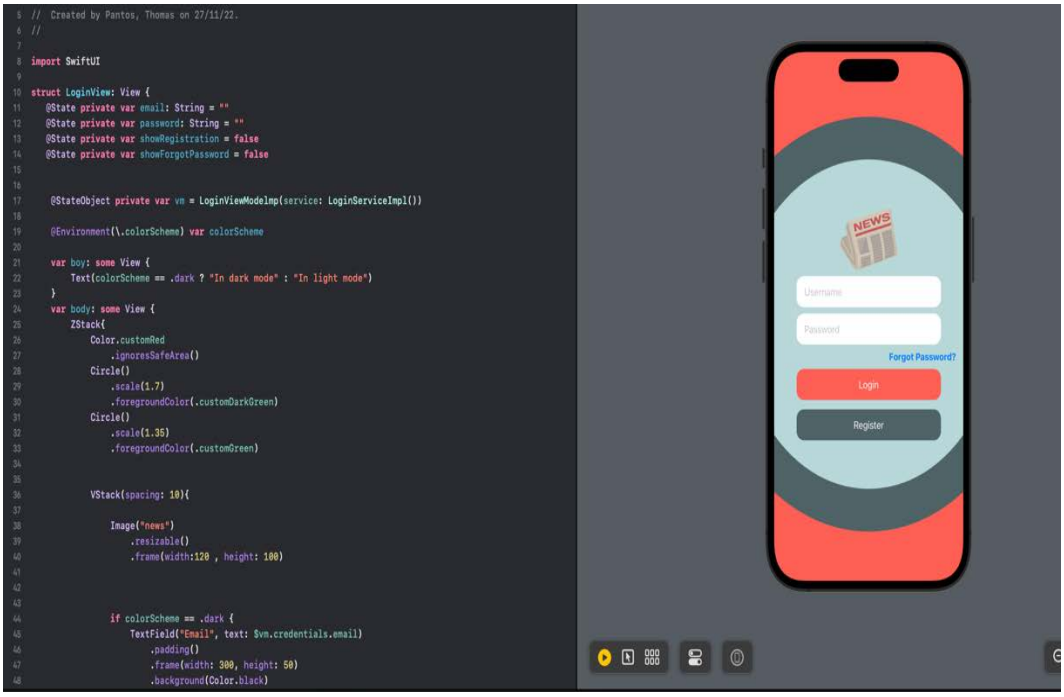
Model- Αντικείμενο που παρουσιάζει τα δεδομένα που έγιναν fetch από την βάση.

ViewModel- Ένα συστατικό που χειρίζεται όλο το business logic και μετατρέπει τα δεδομένα που κατέβηκαν από τη βάση σε δεδομένα που μπορεί να δει ο χρήστης.

Κεφάλαιο 2: Δομή της εφαρμογής 1/2 : Αρχικές Οθόνες

2.1 Είσοδος/Εγγραφή στην εφαρμογή:

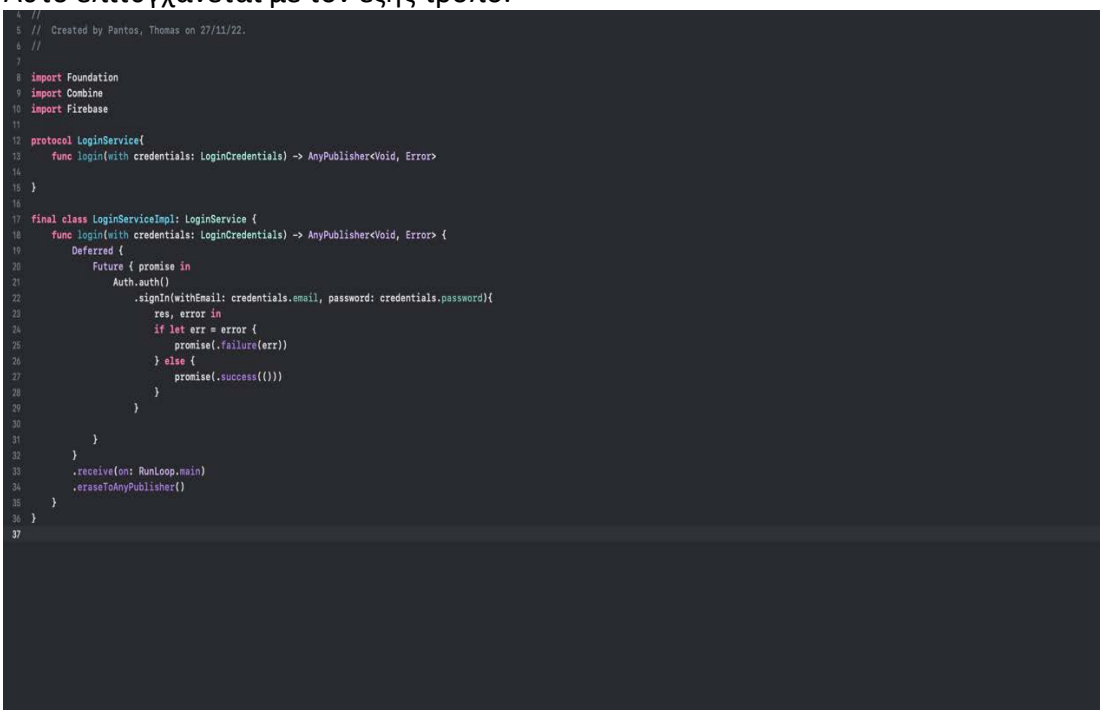
Στη παρακάτω εικόνα βλέπουμε την οθόνη του Login/Register στην εφαρμογή μας.



Εικόνα 2.1.1 Login/Register view

Σε περίπτωση που ο χρήστης έχει βάλει τα σωστά στοιχεία γίνετε ένας έλεγχος στη βάση ώστε να τα επικυρώσει και να «σπρώξει» τον χρήστη στις κυρίως οθόνες της εφαρμογής.

Αυτό επιτυγχάνεται με τον εξής τρόπο.



Εικόνα 2.1.2 Authentication

Στην παραπάνω εικόνα κάνουμε τον έλεγχο με τη βάση ώστε να επιβεβαιώσει η βάση τα σωστά credentials του χρήστη και να τον αφήσει να προχωρήσει στις κυρίως οθόνες.

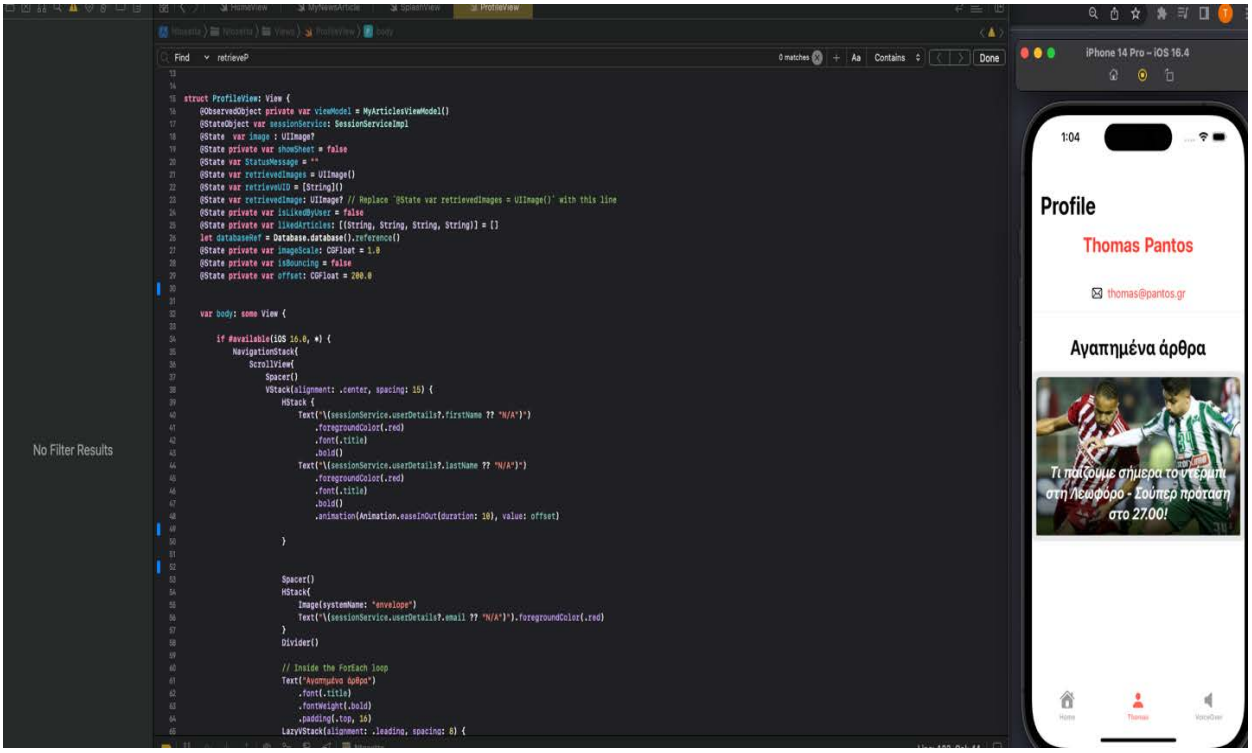
Ο τρόπος αυτός γίνεται με την βιβλιοθήκη Combine.

Στη συνέχεια τραβάμε από τη βάση το πλήρες όνομα του χρήστη μέσω του sessionService.

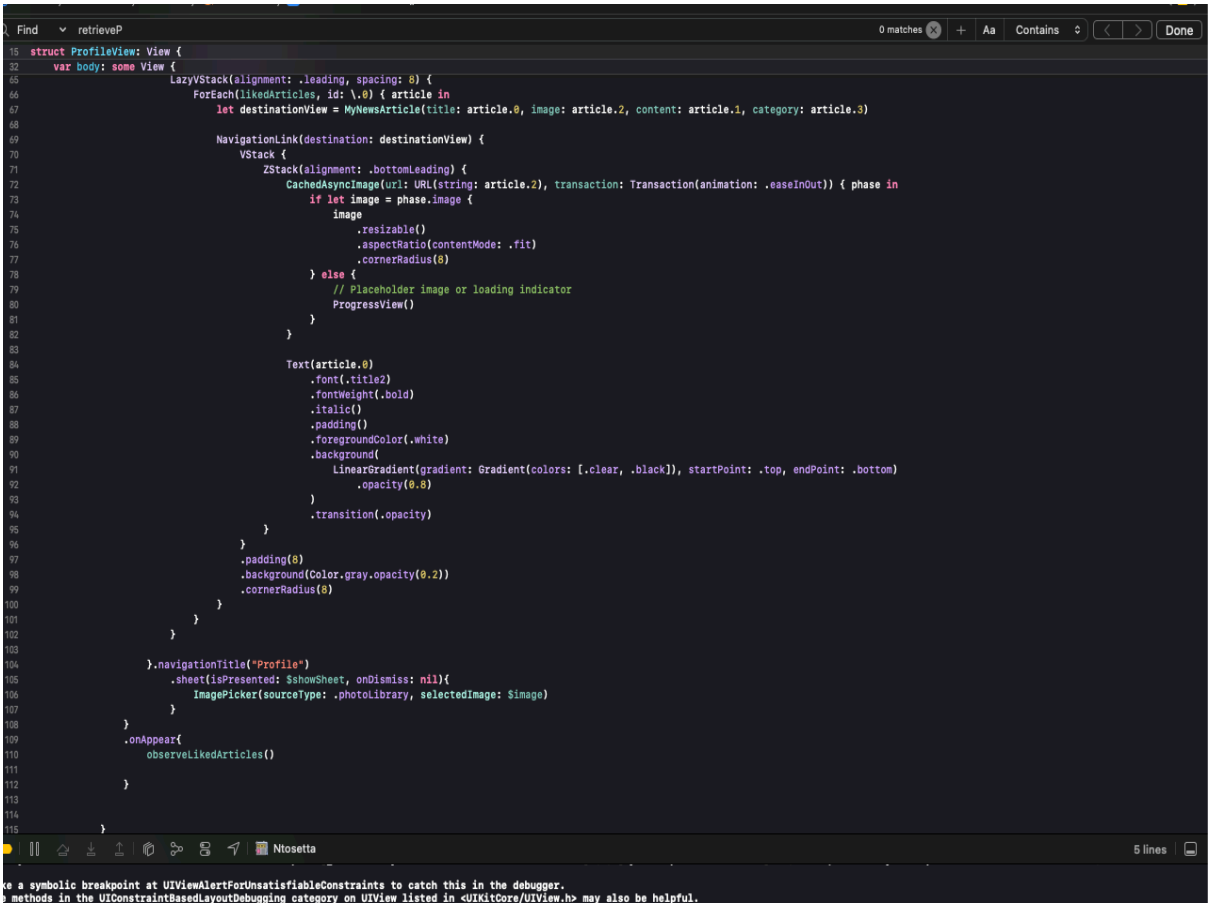
```
    init(){
        setupFirebaseAuthHandler()
    }
    func logout() {
        try? Auth.auth().signOut()
    }
}

private extension SessionServiceImpl {
    func setupFirebaseAuthHandler() {
        handler = Auth.auth().addStateDidChangeListener{
            [weak self] res , user in
            guard let self = self else { return }
            self.state = user == nil ? .loggedOut: .loggedIn
            if let uid = user?.uid {
                self.handleRefresh(with: uid)
            }
        }
    }
    func handleRefresh(with uid: String) {
        Database.database()
            .reference()
            .child("users")
            .child(uid)
            .observe(.value) { [weak self ] snapshot in
                guard let self = self,
                      let value = snapshot.value as? NSDictionary,
                      let firstName = value[RegistrationKeys.firstName.rawValue] as? String,
                      let email = value[RegistrationKeys.email.rawValue] as? String,
                      let lastName = value[RegistrationKeys.lastName.rawValue] as? String else {
                    return
                }
                DispatchQueue.main.async {
                    self.userDetails = UserSessionDetails(firstName: firstName, lastName: lastName, email: email)
                }
            }

    }
}
```

Εικόνα 2.1.3 Τράβηγμα στοιχείων χρήστη

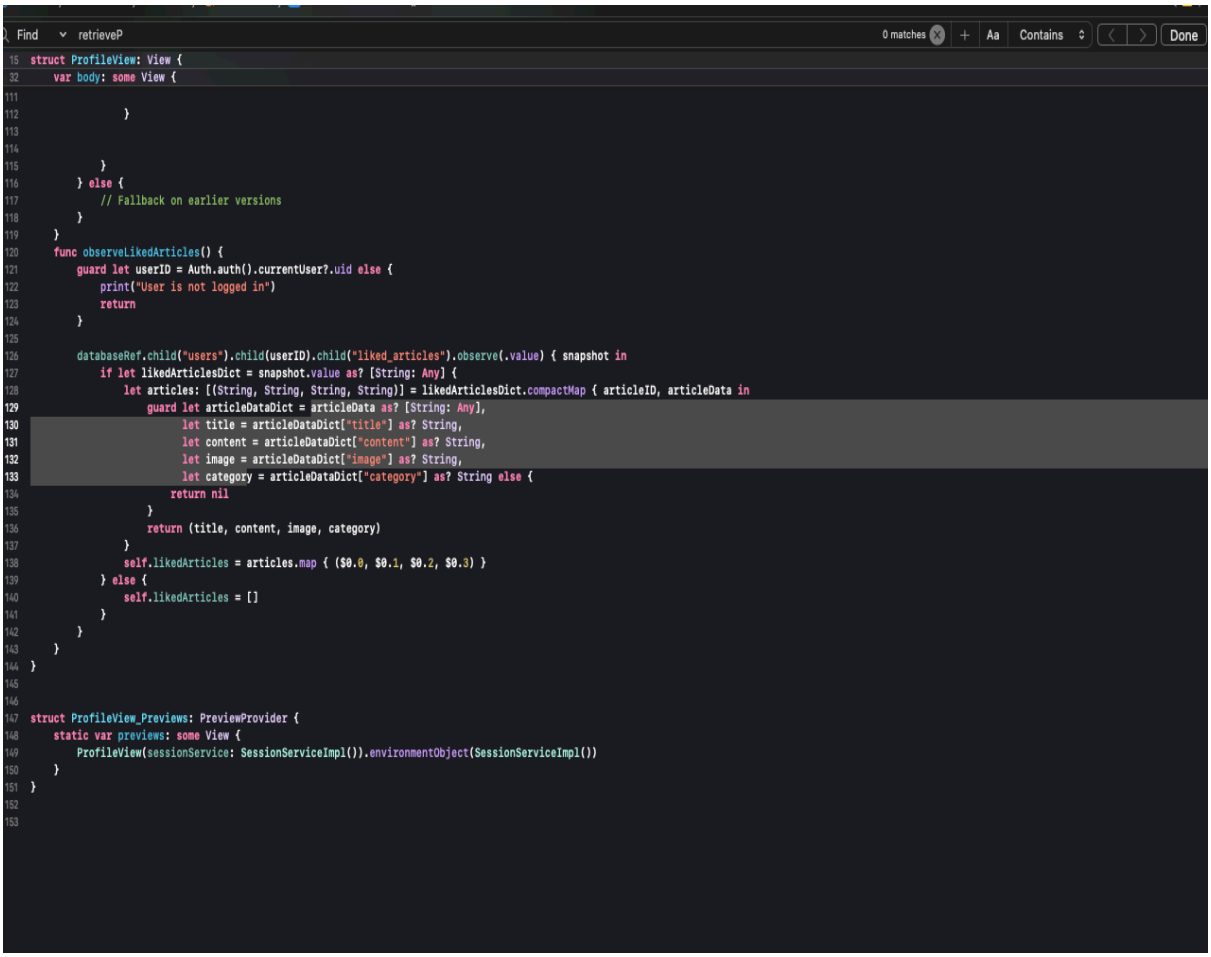
Στο αρχείο sessionService αντιστοιχείτε το uid(user id) που μόλις συνδέθηκε με το uid της βάσης. Εφόσον η αντιστοίχιση γίνει σωστά μας επιστρέφει τα δεδομένα που χρειαζόμαστε ώστε να τα τοποθετήσουμε στην οθόνη profileView που θα εμπεριέχει τα στοιχεία του χρήστη.



Εικόνα 2.1.4 Profile View 1/3

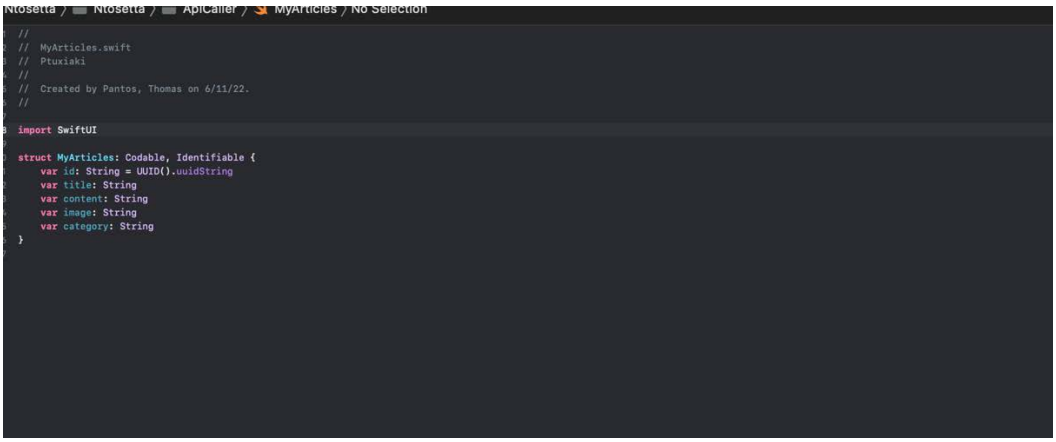


Εικόνα 2.1.5 Profile View 2/3



Εικόνα 2.1.6 Profile View 3/3

2.2 Κλάση του ViewModel:



Εικόνα 2.2.1 Δομή του άρθρου

Δημιουργούμε ένα μοντέλο για να κάνουμε fetch και render τα άρθρα από τη βάση.

```
import Foundation
//import FirebaseFirestore
import Firebase


class MyArticlesViewModel:  ObservableObject {
    @Published var myarticles : [MyArticles] = []
     @Published var likedArticles: [MyNewsArticle] = []
    private var db = Firestore.firestore()

    init(){
        fetchData()
    }

    func fetchData() {
        db.collection("art").addSnapshotListener{ (querySnapshot, error) in
            guard let documents = querySnapshot?.documents else {
                print("no docs")
                return
            }

            self.myarticles = documents.map{ (QueryDocumentSnapshot) -> MyArticles in
                let data = QueryDocumentSnapshot.data()
                let title = data["title"] as? String ?? ""
                let content = data["content"] as? String ?? ""
                let image = data["image"] as? String ?? ""
                let category = data["category"] as? String ?? ""
                return MyArticles(title: title, content: content, image: image, category: category)
            }
        }
    }
```

2.2.2 Κλάση ViewModel

Στην παραπάνω κλάση δηλώνουμε τη κλάση σαν ObservableObject που σημαίνει ότι τα δεδομένα που περιέχονται σε κάθε συνάρτηση εφόσον γίνεται σωστά το fetch θα παρουσιάζει τα δεδομένα αναλογικά με την κάθε κλήση της αντίστοιχης συνάρτησης.

Δηλώνουμε 1 δημόσια μεταβλητή myarticles σαν πίνακας με περιεχόμενα από το μοντέλο MyArticles.

Στην συνάρτηση fetchData() τραβάμε τα δεδομένα απευθείας από τη βάση και τα περνάμε στο πίνακα myarticles. Σε περίπτωση που δεν γίνει σωστά το fetch και δεν υπάρχουν δεδομένα θα εκτυπωθεί το “no docs”.

```
func getSportsCat(){
    db.collection("art").whereField("category", isEqualTo: "sports").addSnapshotListener{(querySnapshot, error ) in
        guard let documents = querySnapshot?.documents else {
            print("no docs")
            return
        }
        self.myarticles = documents.map{(QueryDocumentSnapshot) -> MyArticles in
            let data = QueryDocumentSnapshot.data()
            let title = data["title"] as? String ?? ""
            let content = data["content"] as? String ?? ""
            let image = data["image"] as? String ?? ""
            let category = data["category"] as? String ?? ""
            return MyArticles(title: title, content: content, image: image, category: category)
        }
    }
}
func getMusicCat(){
    db.collection("art").whereField("category", isEqualTo: "music").addSnapshotListener{(querySnapshot, error ) in
        guard let documents = querySnapshot?.documents else {
            print("no docs")
            return
        }
        self.myarticles = documents.map{(QueryDocumentSnapshot) -> MyArticles in
            let data = QueryDocumentSnapshot.data()
            let title = data["title"] as? String ?? ""
            let content = data["content"] as? String ?? ""
            let image = data["image"] as? String ?? ""
            let category = data["category"] as? String ?? ""
            return MyArticles(title: title, content: content, image: image, category: category)
        }
    }
}
```

Εικόνα 2.2.3 Συνάρτηση Κατηγορίας άρθρου

2.2 Δημιουργία κατηγοριών:

Για να τραβήξουμε και να δείξουμε τα άρθρα με βάση τη κατηγορία τους χρησιμοποιούμε τις συναρτήσεις getSportsCat(), getMusicCat κλπ ώστε να ορίσουμε γίνει ο έλεγχος στη βάση σχετικά με την εκάστοστε κατηγορία. Π.χ αν θέλουμε να

δούμε αποκλειστικά άρθρα σχετικά με αθλητικά γεγονότα θα χρησιμοποιήσουμε τη συνάρτηση getSportsCat() μέσω του ViewModel μας όπως θα δούμε στη συνέχεια.

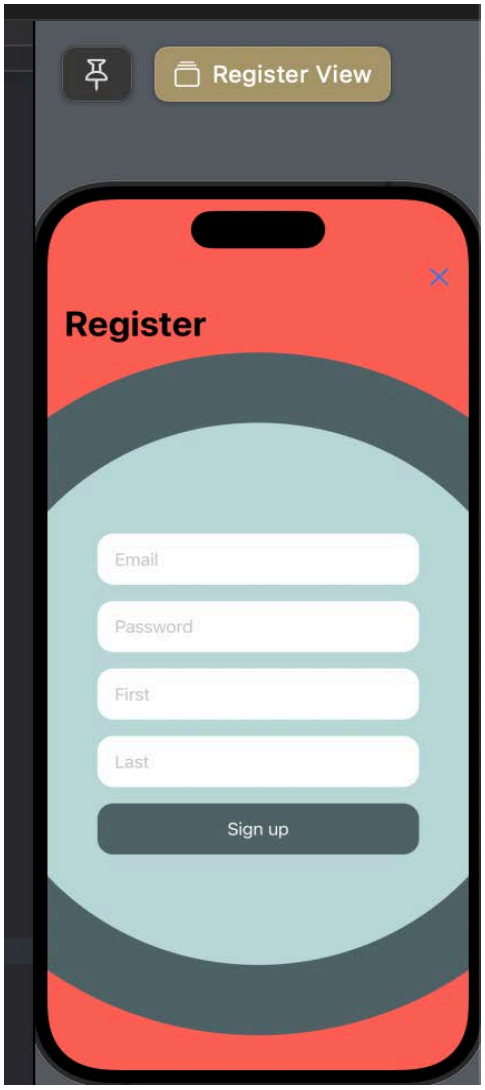
2.3 Οθόνη Profile:

Στις παραπάνω εικόνες (2.1.4 – 2.1.6) βλέπουμε ένα την οθόνη του προφίλ του χρήστη όπου περιέχει τα προσωπικά στοιχεία του χρήστη τα οποία έχουν γίνει fetch από τη βάση.

Δηλώνουμε μια μεταβλητή τύπου StateObject που θα ονομάσουμε sessionService και χρησιμοποιούμε το viewModel του SessionService που δημιουργήσαμε νωρίτερα ώστε να περάσουμε τα στοιχεία σωστά μέσα στο view. Για να γίνει σωστά πρέπει να θέσουμε το viewmodel sessionService.userDetails?.firstName ,sessionService.userDetails?.lastName και sessionService.userDetails?.email ώστε να έχουμε τα στοιχεία. Για να αποφύγουμε προβλήματα σχετικά με τη βάση θα πρέπει να θέσουμε ένα κείμενο σε περίπτωση που δεν πιάσει τα στοιχεία του τύπου "N/A".Ωστόσο ο χρήστης έχει την δυνατότητα να δει τα άρθρα στα οποία έχει πατήσει like καθώς αποθηκεύονται σε μια ξεχωριστή λίστα στη βάση με την συνάρτηση observeLikedArticles() η οποία θα εξηγηθεί παρακάτω.

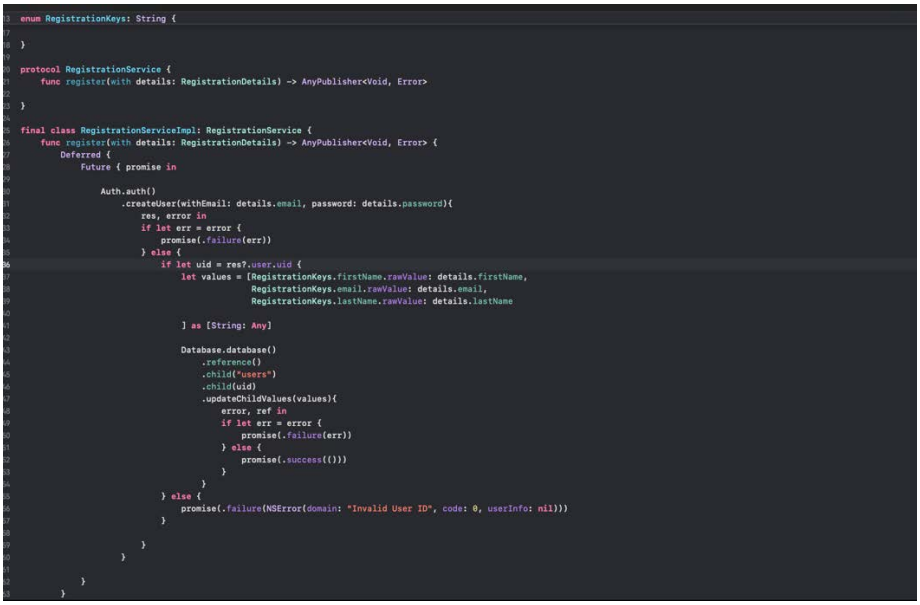
2.4 Εγγραφή χρήστη:

Στην εικόνα 2.4.1 βλέπουμε το Login/Register Screen όπου ο χρήστης θα κάνει την εγγραφή ή την είσοδό του στην εφαρμογή. Όταν πατηθεί το κουμπί register θα ανοίξει ένα modal με μια φόρμα στοιχείων ώστε ο χρήστης να περάσει τα στοιχεία του.



Εικόνα 2.4.1 Φόρμα εγγραφής χρήστη

Στην εικόνα 2.4.2 βλέπουμε τη φόρμα για την τοποθέτηση στοιχείων του χρήστη τα οποία θα περαστούν στη βάση δεδομένων με τη βοήθεια της Combine.



```
enum RegistrationKeys: String {

}

protocol RegistrationService {
    func register(with details: RegistrationDetails) -> AnyPublisher<Void, Error>

}

final class RegistrationServiceImpl: RegistrationService {
    func register(with details: RegistrationDetails) -> AnyPublisher<Void, Error> {
        Deferred {
            Future { promise in

                Auth.auth()
                    .createUser(withEmail: details.email, password: details.password){
                        res, error in
                        if let err = error {
                            promise(.failure(err))
                        } else {
                            if let uid = res?.user.uid {
                                let values = [RegistrationKeys.firstName.rawValue: details.firstName,
                                              RegistrationKeys.email.rawValue: details.email,
                                              RegistrationKeys.lastName.rawValue: details.lastName

                                ] as [String: Any]

                                Database.database()
                                    .reference()
                                    .child("users")
                                    .child(uid)
                                    .updateChildValues(values){
                                        error, ref in
                                        if let err = error {
                                            promise(.failure(err))
                                        } else {
                                            promise(.success(()))
                                        }
                                    }
                            } else {
                                promise(.failure(NSError(domain: "Invalid User ID", code: 0, userInfo: nil)))
                            }

                        }
                    }

            }
        }
```

Εικόνα 2.4.2 Φόρμα εγγραφής μέσω combine

Εφόσον γίνει σωστά η εγγραφή θα μεταφερθεί ο χρήστης στη κυρίως οθόνη(αμέσως μετά το splashScreen) .Επομένως στην οθόνη προφίλ μέσω του SessionService ο χρήστης μπορεί να βλέπει τα στοιχεία του.

2.5 Εντοπισμός (Localization)

Τι είναι ο εντοπισμός εφαρμογών για κινητά;

Εντοπισμός είναι η διαδικασία μετάφρασης και βελτιστοποίησης μιας εφαρμογής με στόχο την καλή λειτουργία της και σε άλλες γλώσσες, εκτός της προεπιλεγμένης. Μπορεί να περιλαμβάνει διαφορετικές παραλλαγές ή ακόμη και διαλέκτους της ίδιας γλώσσας, όπως ομιλείτε σε διαφορετικές χώρες, τις οποίες ονομάζουμε τοπικές γλώσσες.

Μια τοποθεσία είναι πολύ απλά ένας συνδυασμός μιας γλώσσας και μιας συγκεκριμένης χώρας. Για παράδειγμα, τα αγγλικά μιλούνται τόσο στις Ηνωμένες Πολιτείες όσο και στο Ηνωμένο Βασίλειο. Η διαφορά μεταξύ αυτών των δύο έγκειται στο γεγονός ότι μιλούν την ίδια γλώσσα, απλώς με διαφορετική τοπικότητα. Η μορφή της τοπικής γλώσσας είναι απλή – είναι γλώσσα ακολουθούμενη από τον προσδιορισμό της χώρας στην οποία ομιλείτε. Ακολουθεί μια λίστα με μερικές τοπικές γλώσσες και τους αντίστοιχους κωδικούς τους

Ορισμένοι γλωσσικοί κωδικοί δεν έχουν πίσω τους τον προορισμό της χώρας. Αυτό σημαίνει απλώς ότι σε αυτές τις περιπτώσεις, η γλώσσα δεν ομιλείται συνήθως εκτός της συγκεκριμένης χώρας, όπως φαίνεται παρακάτω.

```
en_AS: "English (American Samoa)",
en_AU: "English (Australia)",
en_BE: "English (Belgium)",
en_BZ: "English (Belize)",
fr_BE: "French (Belgium)",
fr_BJ: "French (Benin)",
fr_BF: "French (Burkina Faso)",
fr_BI: "French (Burundi)",
fr_CM: "French (Cameroon)",
fr_CA: "French (Canada)",
es_AR: "Spanish (Argentina)",
es_BO: "Spanish (Bolivia)",
es_CL: "Spanish (Chile)",
es_CO: "Spanish (Colombia)",
es_CR: "Spanish (Costa Rica)",
es_DO: "Spanish (Dominican Republic)",
es_EC: "Spanish (Ecuador)"
```

Εικόνα 2.5.1 Localised Λίστα με κωδικούς χωρών/γλωσσών

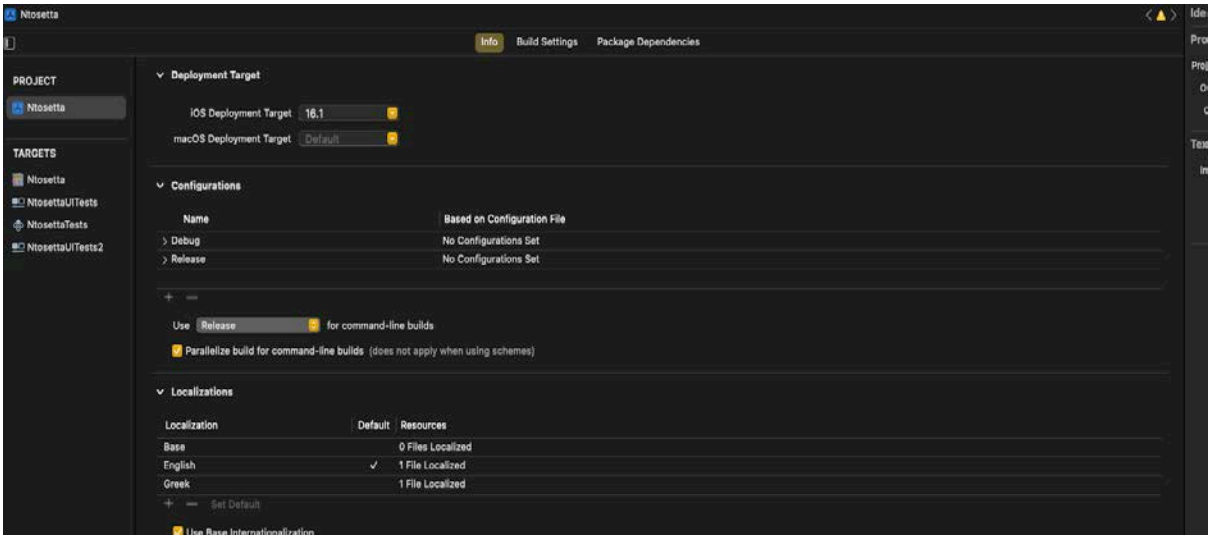
```
fi: Finnish
lit: Lithuanian
tur: Turkish
```

2.6 Οφέλη του εντοπισμού εφαρμογών για κινητά τηλέφωνα

- Οδηγεί σε αύξηση των λήψεων σε μια χρονική περίοδο.
- Ενισχύει τα έσοδα στις χώρες που επηρεάζονται από τον εντοπισμό.
- Φέρνει τις εταιρείες σε επαφή με αγορές που προηγουμένως ήταν ανεκμετάλλευτες.
- Προσφέρει μια μοναδική προοπτική στις διαφορετικές περιπτώσεις χρήσης των εφαρμογών σας.

2.7 Πως κάνουμε localize την εφαρμογή

Μπαίνοντας στο περιβάλλον του Xcode πατώντας πάνω στο root του application μας εμφανίζει την παρακάτω οθόνη:



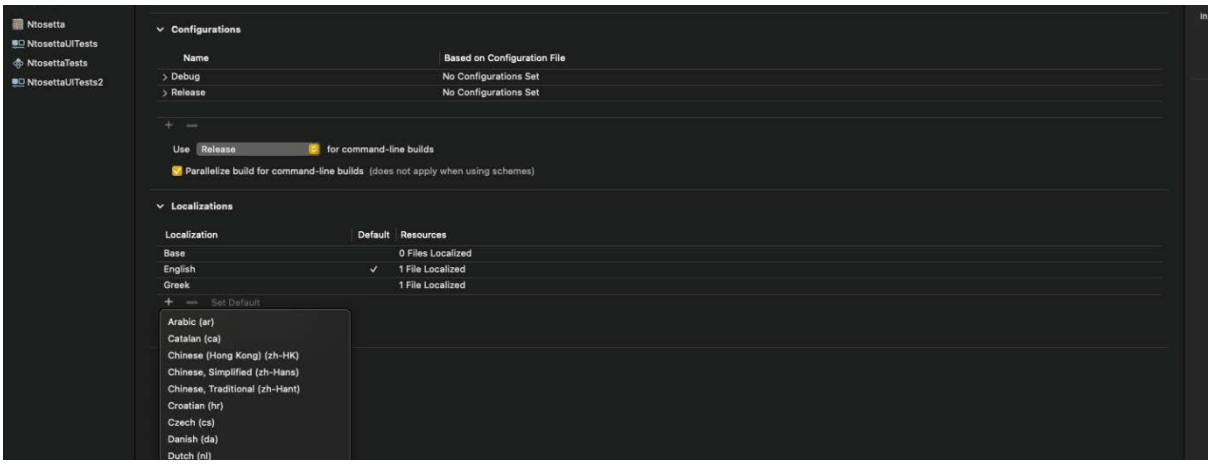
Εικόνα 2.7.1 iOS base SDK 1

Στην παραπάνω εικόνα πατώντας στο τμήμα Info μας εμφανίζει 3 κατηγορίες: Deployment target, configurations και localizations.

Το deployment target: Ο όρος deployment target αναφέρεται στην ελάχιστη έκδοση του λειτουργικού συστήματος ή της πλατφόρμας στην οποία προσδιορίζεται να εκτελεστεί η εφαρμογή .Για παράδειγμα στην εικόνα 2.4.1 το deployment target είναι 16.1 που σημαίνει συσκευές που έχουν το iOS 15 δεν θα είναι συμβατές με την εφαρμογή.

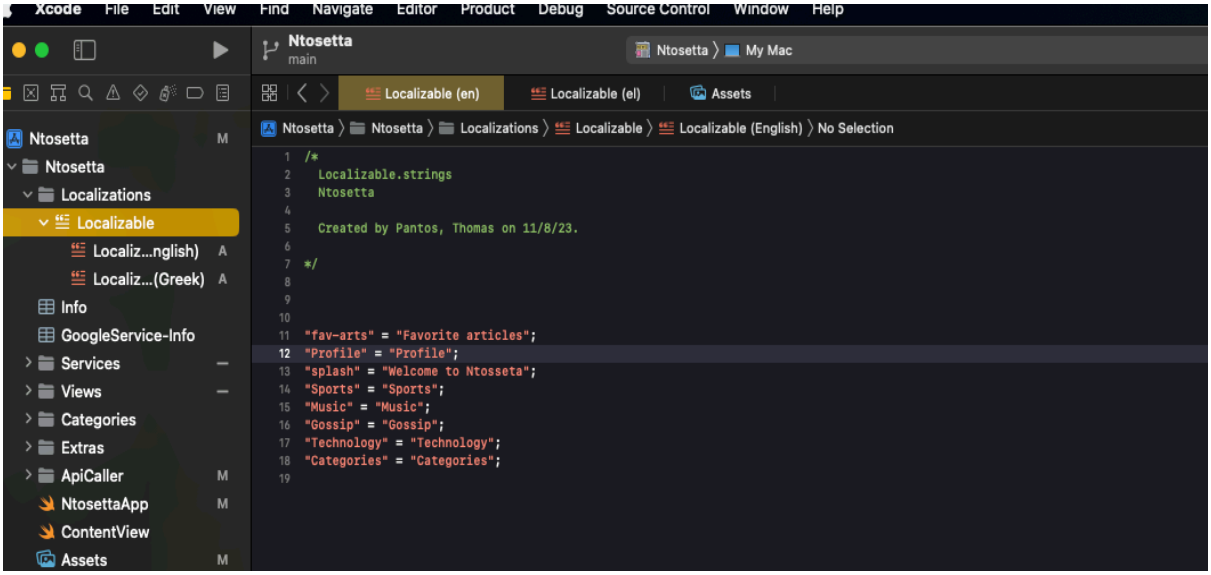
Το configurations : Ένα νέο application σε Xcode ορίζει δύο διαμορφώσεις(configurations), Debug και Release. Τα περισσότερα applications ορίζουν μία ή περισσότερες πρόσθετες διαμορφώσεις κατασκευής για διάφορους λόγους. Αυτό δεν είναι κάτι καινούργιο και είναι καλή πρακτική να χρησιμοποιείτε διαμορφώσεις για να προσαρμόζετε μια δημιουργία στις συγκεκριμένες ανάγκες του περιβάλλοντος στο οποίο πρόκειται να αναπτυχθεί.

Τα localizations που αναφέραμε προηγουμένως επιτυγχάνονται με τον εξής τρόπο :



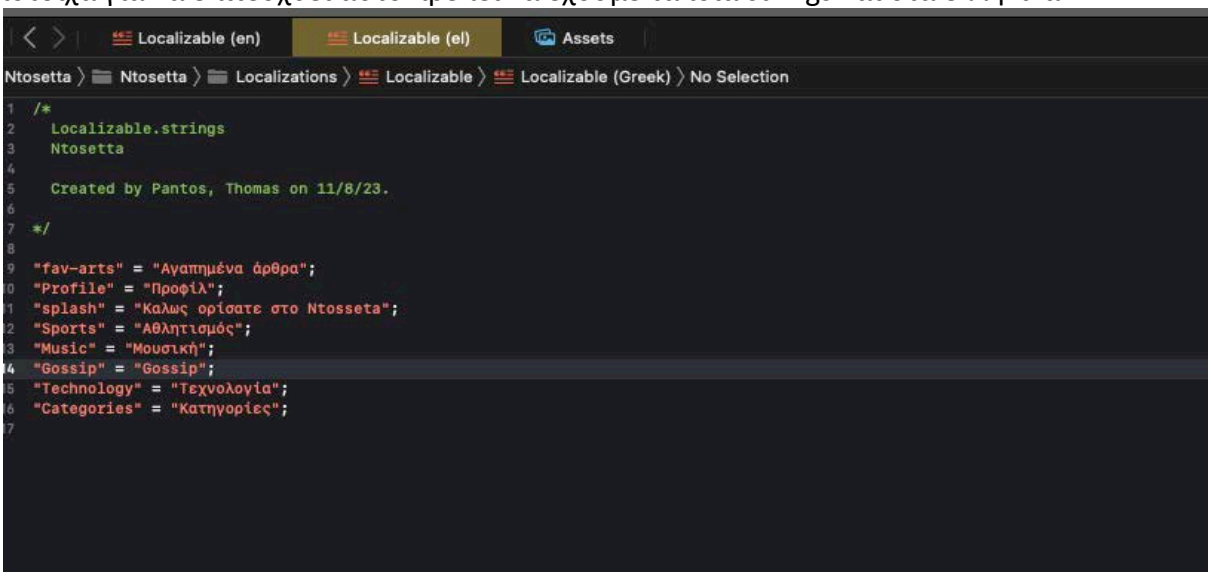
Εικόνα 2.7.2 Προσθήκη γλώσσας

Πρώτον προσθέτουμε στο σχήμα της εφαρμογής τις γλώσσες που θα θέλαμε να έχει η εφαρμογή μας. Για παράδειγμα στην δική μας εφαρμογή θα έχουμε ελληνικά και αγγλικά. Έπειτα θα δημιουργήσουμε ένα αρχείο strings και ύστερα next. Θα θέσουμε την ονομασία Localizable.strings και θα πατήσουμε το κουμπί create. Αμέσως μετα θα δημιουργηθεί ένα αρχείο όπως στην παρακάτω εικόνα.



Εικόνα 2.7.3 Προσθήκη αγγλικής μετάφρασης

Στην εικόνα 2.4.3 έχουμε το αρχείο Localizable το οποίο χωρίζετε σε en και el δηλαδή στις 2 γλώσσες που θέλουμε να έχουμε. Από αυτό το σημείο και μετά γράφουμε όλα τα strings τα οποία θέλουμε να μεταφράζουμε όταν η συσκευή μας. Αντίστοιχα για να επιτευχθεί αυτό πρέπει να έχουμε τα ίδια strings και στα ελληνικά.



Εικόνα 2.7.4 Προσθήκη ελληνικής μετάφρασης

Έπειτα μέσα στο κώδικα πρέπει τα εκάστοτε string να εφαρμοστούν έτσι.

Κεφάλαιο 3: Δομή της εφαρμογής 2/2 : Κεντρικές οθόνες και backend

3.1 Menu Οθόνη

Εφόσον έχει γίνει η ταυτοποίηση επιτυχώς αμέσως μετά την οθόνη Login και τη SplashScreen με το λόγκο της εφαρμογής εμφανίζεται το HomeView. Το HomeView είναι η κεντρική οθόνη της εφαρμογής παρουσιάζοντας στον χρήστη την λίστα από τα άρθρα που θέλει να διαβάσει ή να ακούσει.

```
struct HomeView: View {
    var body: some View {

                                Text(article.content )
                                    .lineLimit(6)
                                    .font(.body)
                                    .padding(8)

                                Text(article.category)
                                    .font(.subheadline)
                                    .padding(8)

                            }
                        }
                    }
                } else {
                    NavigationStack{

                        VStack {
                            List(Category.allCases, id: \.self) { category in
                                        NavigationLink(destination: getView(for: category)) {
                                            Label(category.rawValue, systemImage: category.icon)
                                                .foregroundColor(category.color)
                                        }
                                    }
                                    .navigationTitle("Categories")
                            }

                        }.background(Color.red)

                    }
                        // Group
                }
                .navigationTitle("Ntosetta")

                .toolbar {

                    ToolbarItem(placement: .navigationBarLeading){
                        Button("\(Image(systemName: "sidebar.left"))"){
                            isSideBarOpened.toggle()
                        }




                    }
                    ToolbarItem(placement: .navigationBarTrailing) {
                        Button("\(Image(systemName: "rectangle.portrait.and.arrow.right"))"){
                            logout()
                        }
```

Εικόνα 3.1.1 Home View 1

```
struct HomeView: View {
    var body: some View {
                }
            }
            .onAppear(){
                viewModel.fetchData()
            }
        } else {
        }
    }

    func logout() {
        try? Auth.auth().signOut()
    }
    func getView(for category: Category) -> some View {
        switch category {
        case .sports:
            return AnyView(SportsView())
        case .music:
            return AnyView(MusicView())
        case .gossip:
            return AnyView(GossipView())
        case .technology:
            return AnyView(TechnologyView())
        }
    }

    enum Category: String, CaseIterable {
        case sports = "Sports"
        case music = "Music"
        case gossip = "Gossip"
        case technology = "Technology"

        var icon: String {
            switch self {
            case .sports:
                return "sportscourt"
            case .music:
                return "music.note"
            case .gossip:
                return "person.2.wave.2.fill"
            case .technology:
                return "gear"
```

Εικόνα 3.1.2 Home View 2

```
struct HomeView: View {
    enum Category: String, CaseIterable {
        var icon: String {
            case .gossip:
                return "person.2.wave.2.fill"
            case .technology:
                return "gear"
            }
        }

        var color: Color {
            switch self {
            case .sports:
                return .red
            case .music:
                return .yellow
            case .gossip:
                return .pink
            case .technology:
                return .gray
            }
        }
    }

    struct HomeView_Previews: PreviewProvider {
        static var previews: some View {
            HomeView()
                .environmentObject(SessionServiceImpl())
        }
    }
}
```

Εικόνα 3.1.3 Home View 3

Στις παραπάνω εικόνες βλέπουμε την κεντρική οθόνη HomeView στην οποία υπάρχει ένα side menu στο οποίο ο χρήστης μπορεί να επιλέξει την κατηγορία των ειδήσεων που επιθυμεί. Μέσω του viewModel μπορούμε να τραβήξουμε όλες τις ειδήσεις ή μεμονωμένες ειδήσεις μέσω του πλάγιου μενού.

Το πλάγιο μενού δημιουργήθηκε με μια λίστα και η οποία παρουσιάζει τις κατηγορίες που υπάρχουν. Μέσα σε αυτή τη λίστα υπάρχει η συνάρτηση getView που τραβάει την επιλογή του χρήστη, στην συνέχεια την αντιστοιχεί με την συνάρτηση της ομώνυμης επιλογής και επιστρέφει το επιθυμητό αποτέλεσμα.

Το HomeView είναι βασισμένο σε μια λίστα από components (MyNewsArticle) τα οποία είναι βασισμένα στην δομή του άρθρου(βλ. 2.1.1).

Το component του άρθρου είναι το εξής:

```
//
//  Created by Pantos, Thomas on 22/9/22.
//

import SwiftUI
import CachedAsyncImage
import AVFoundation



struct MyNewsArticle: View {
    let title: String
    let image: String
    let content: String?
    let category: String
    @State var isLiked: Bool = false
    @State var arthra = [MyArticles]()
    @State var isPlaying : Bool = false


    var body: some View {
        let synthesizer = AVSpeechSynthesizer()
        ScrollView(.vertical){
            VStack(alignment: .leading){

                Text(title)
                    .foregroundColor(.blue)
                    .font(.system(size: 30))
                    .bold()
            }



            HStack(alignment: .center) {
                CachedAsyncImage(url: URL(string: image), transaction: Transaction(animation: .easeInOut)) { phase in
                    if let image = phase.image {
                        image
                            .resizable()
                            .scaledToFit()
                            .clipShape(RoundedRectangle(cornerRadius: 25))
                            .transition(.opacity)
                    } else {
                        HStack {
                            // Insert your placeholder here
                            Text("No Image Available")
                        }
                    }
                }
            }
```

Εικόνα 3.1.4 Component Άρθρου 1/2

```
            }



            Text(content ?? "")
                .font(.body)
                .padding(8)

            Text(category )
                .font(.subheadline)
                .padding(8)
            HStack{

                Button(action: {
                    let utterance = AVSpeechUtterance(string: content ??  "no content available" )
                    utterance.voice = AVSpeechSynthesisVoice(language: "el-GR")
                    utterance.rate = 0.45
                    synthesizer.speak(utterance)
                }, label: {
                    Image(systemName: "speaker")

                })
            }
        }
    }
}
```

Εικόνα 3.1.5 Component Άρθρου 2/2

Στον παραπάνω κώδικα βλέπουμε τις παραμέτρους που έχουμε θέσει στο άρθρο οι οποίοι είναι το title, image, content, category και το isPlaying. Αυτές οι παράμετροι παίρνουν το περιεχόμενο από την βάση μας και αποτυπώνονται αντίστοιχα σε κάθε πεδίο. Το isPlaying είναι μια μεταβλητή Boolean που χρησιμοποιείται όταν ο χρήστης επιθυμεί να ακούσει ένα άρθρο πατώντας το κουμπί με το εικονίδιο του ηχείου. Παράλληλα χρησιμοποιούμε την βιβλιοθήκη AVFoundation ώστε να μπορέσουμε να μετατρέψουμε το περιεχόμενο του άρθρου σε ήχο. Μέσω της AVFoundation δημιουργούμε την σταθέρα let synthesizer = AVSpeechSynthesizer() και μέσα στο κουμπί το όποιο έχουμε βάλει το εικονίδιο του ηχείου αναπτύσουμε τον παρακάτω κώδικα. (βλ. Εικόνα 3.2.2)

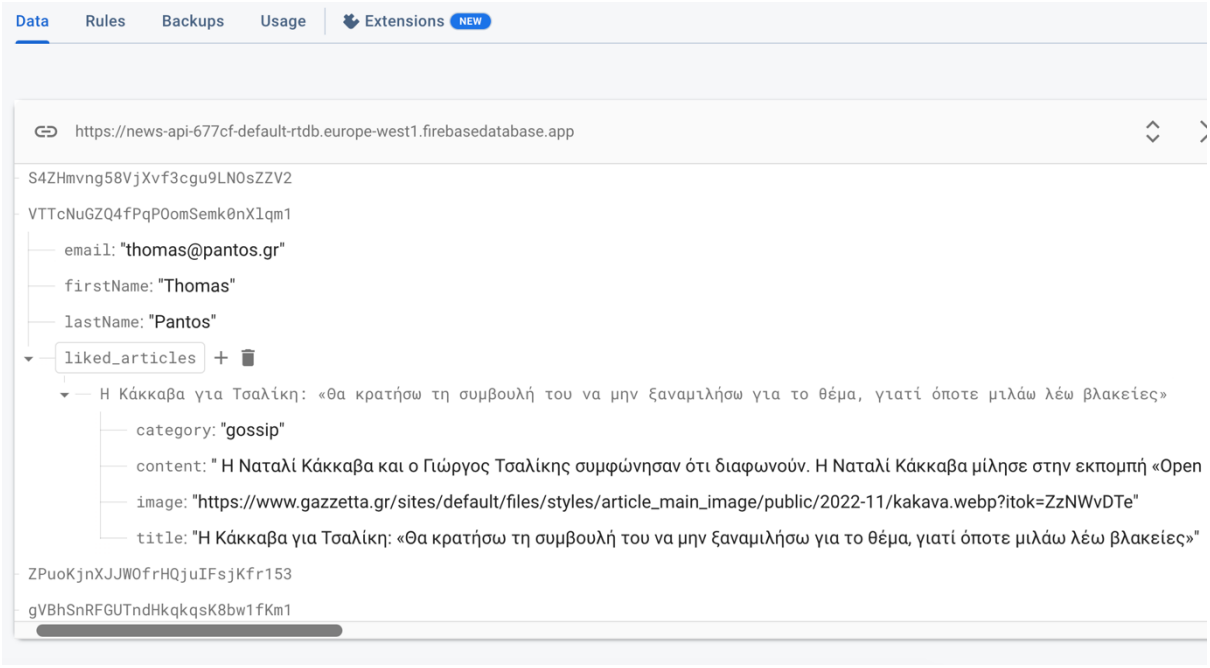
Άλλη μία βιβλιοθήκη που χρησιμοποιήθηκε είναι η CachedAsyncImage για την οποία την χρειαζόμαστε για να κάνουμε render url εικόνες. (βλ. Εικόνα 3.2.1)

Επίσης ο χρήστης μπορεί να κάνει like στο άρθρο που του αρέσει και το συγκεκριμένο άρθρο να αποθηκευτεί στη λίστα με τα αγαπημένα άρθρα μέσω αυτής της συνάρτησης toggleLike() σε συνεργασία με τη συνάρτηση checkIfLikedByUser().

```
    }

    func checkIfLikedByUser() {
        guard let userID = Auth.auth().currentUser?.uid else {
            print("User is not logged in")
            return
        }

        let sanitizedTitle = sanitizeTitle(title)

        databaseRef.child("users").child(userID).child("liked_articles").child(sanitizedTitle).observe(.value) { snapshot in
            isLikedByUser = snapshot.exists()
        }
    }

    func toggleLike() {
        guard let userID = Auth.auth().currentUser?.uid else {
            print("User is not logged in")
            return
        }

        let sanitizedTitle = sanitizeTitle(title)

        if isLikedByUser {
            databaseRef.child("users").child(userID).child("liked_articles").child(sanitizedTitle).removeValue { error, _ in
                if let error = error {
                    print("Error removing liked article: \(error.localizedDescription)")
                } else {
                    print("Successfully removed liked article")
                }
            }
        } else {
            let articleData: [String: Any] = [
                "title": title,
                "image": image,
                "content": content ?? "",
                "category": category
            ]

            databaseRef.child("users").child(userID).child("liked_articles").child(sanitizedTitle).setValue(articleData) { error, _ in
                if let error = error {
                    print("Error saving liked article: \(error.localizedDescription)")
                } else {
                    print("Successfully saved liked article")
                }
            }
        }

        isLikedByUser.toggle()
    }
```

Εικόνα 3.1.6 Συνάρτηση toggleLike & checkIfLikedByUser

Η συνάρτηση toggleLike() είναι υπεύθυνη για την αυθεντικότητα του χρήστη αν είναι συνδεδεμένος σωστά. Ύστερα χρησιμοποιείται η συνάρτηση sanitizeTitle(), που στην ουσία σιγουρεύει ότι ο τίτλος θα αποθηκευτεί στην βάση σαν valid key.Όταν ο χρήστης πατάει like η μεταβλητή isLikedByUser που είναι Boolean γίνεται true και δημιουργεί ένα νέο αντικείμενο στη βάση και στη σχετική λίστα liked_articles. Στην σχετική λίστα όλα τα πεδία του άρθρου αποθηκεύονται σαν strings. Ένα παράδειγμα στην παρακάτω εικόνα από τη βάση.



Εικόνα 3.1.7 Αποθήκευση του άρθρου στη λίστα των αγαπημένων άρθρων στη βάση δεδομένων

.

Η συνάρτηση checkIfLikedByUser() κάνει την ίδια σχεδόν δουλεία με την προηγούμενη με τη διαφορά ότι πηγαίνει απευθείας στην βάση δεδομένων και κάνει observe τη λίστα μέσω του userID ώστε να ελέγξει ότι το άρθρο υπάρχει με τη μεταβλητή isLikedByUser = snapshot.exists() που μεταφράζεται σαν true.

3.2 Backend Firebase

Τι είναι η Firebase;

Η Firebase είναι μια πλατφόρμα ανάπτυξης εφαρμογών για κινητές συσκευές και εφαρμογές ιστού που παρέχει στους προγραμματιστές ένα σύνολο εργαλείων και υπηρεσιών για τη γρήγορη και εύκολη δημιουργία εφαρμογών υψηλής ποιότητας. Η Firebase προσφέρει ένα ευρύ φάσμα χαρακτηριστικών και πόρων, όπως:

1. Βάση δεδομένων πραγματικού χρόνου: μια βάση δεδομένων νέφους NoSQL που σας επιτρέπει να αποθηκεύετε και να συγχρονίζετε δεδομένα σε πραγματικό χρόνο μεταξύ των πελατών.

2. Αυθεντικοποίηση: ένα σύστημα ελέγχου ταυτότητας που σας επιτρέπει να πιστοποιείτε εύκολα τους χρήστες με email και κωδικό πρόσβασης, αριθμούς τηλεφώνου, Google, Facebook, Twitter, GitHub και πολλά άλλα.

3. Cloud Firestore: μια βάση δεδομένων με βάση τα έγγραφα NoSQL που παρέχει κλιμακούμενη, υψηλής απόδοσης και αποθήκευση χωρίς διακομιστή για την εφαρμογή σας.

4. Cloud Functions: ένα backend χωρίς διακομιστή που σας επιτρέπει να γράφετε και να αναπτύσσετε κώδικα που ανταποκρίνεται σε συμβάντα που προκαλούνται από το Firebase ή υπηρεσίες τρίτων.

5. Φιλοξενία: μια υπηρεσία που σας επιτρέπει να φιλοξενείτε την εφαρμογή ιστού ή το στατικό περιεχόμενό σας με γρήγορο και ασφαλή τρόπο με προσαρμοσμένο domain και πιστοποιητικό SSL.

6. Cloud Storage: μια υπηρεσία που σας επιτρέπει να αποθηκεύετε και να εξυπηρετείτε περιεχόμενο που δημιουργείται από χρήστες, όπως εικόνες και βίντεο.

7. Cloud Messaging: μια υπηρεσία που σας επιτρέπει να στέλνετε ειδοποιήσεις και μηνύματα στους χρήστες σας σε πολλαπλές πλατφόρμες, όπως iOS, Android και διαδίκτυο.

8. Παρακολούθηση επιδόσεων: ένα εργαλείο που σας επιτρέπει να μετράτε τις επιδόσεις της εφαρμογής σας και να λαμβάνετε πληροφορίες σχετικά με τον τρόπο με τον οποίο την αντιμετωπίζουν οι χρήστες σας.

9. Test Lab: μια υπηρεσία που σας επιτρέπει να δοκιμάζετε την εφαρμογή σας σε πραγματικές συσκευές στο cloud για να διασφαλίσετε την ποιότητα και τη λειτουργικότητά της.

Στην εφαρμογή μας χρησιμοποιήθηκαν μόνο οι 3 πρώτες κατηγορίες.

Η πρώτη χρησιμοποιήθηκε στη οθόνη εγγραφής χρήστη όπου κάθε νέος χρήστης από τη στιγμή που έκανε την εγγραφή του τα στοιχεία του πέρασαν στην realtime βάση δεδομένων.

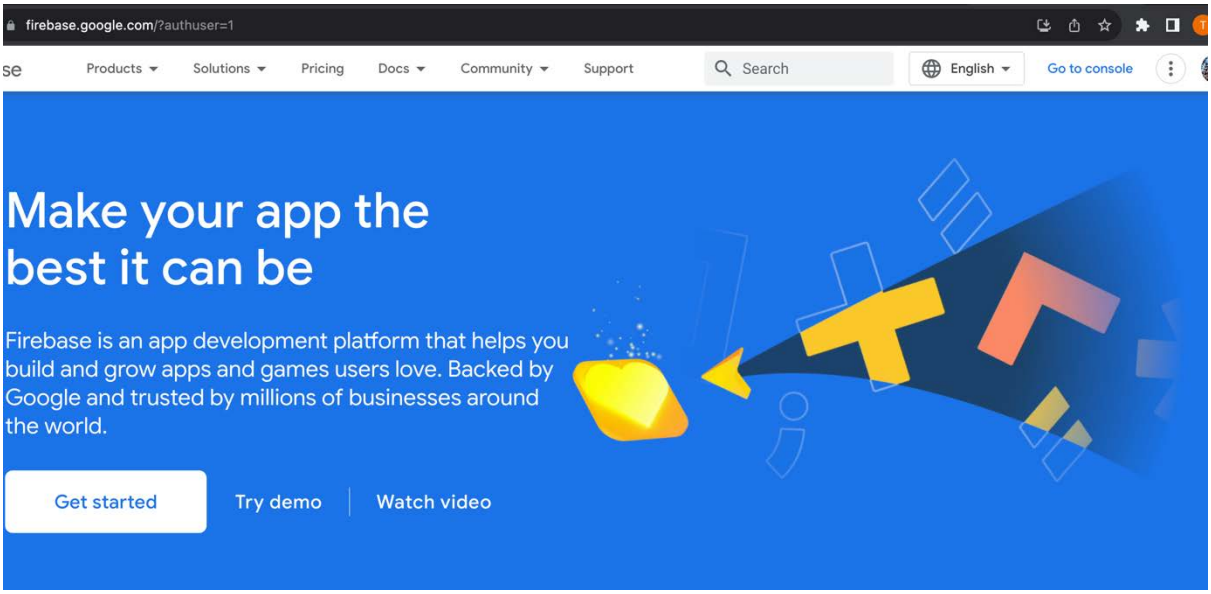
Η δεύτερη χρησιμοποιήθηκε στο authenticate με το email/password method όπου ο χρήστης περνάει το email και τον password για να εισέλθει στο σύστημα.

Η τρίτη χρησιμοποιήθηκε για τα άρθρα της εφαρμογής όπου περάστηκαν τα αντίστοιχα δεδομένα στα ανάλογα πεδία του κάθε άρθρου.

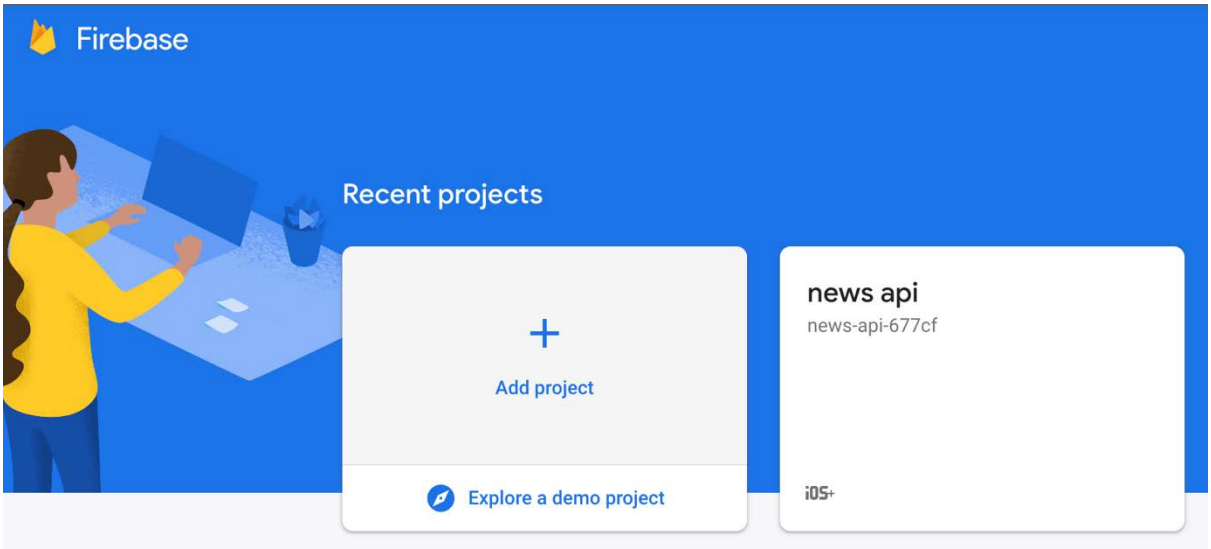
3.3 Σύνδεση backend με την εφαρμογή

Η σύνδεση της Firebase βάσης με την εφαρμογή επιτυγχάνεται με τον εξής τρόπο :

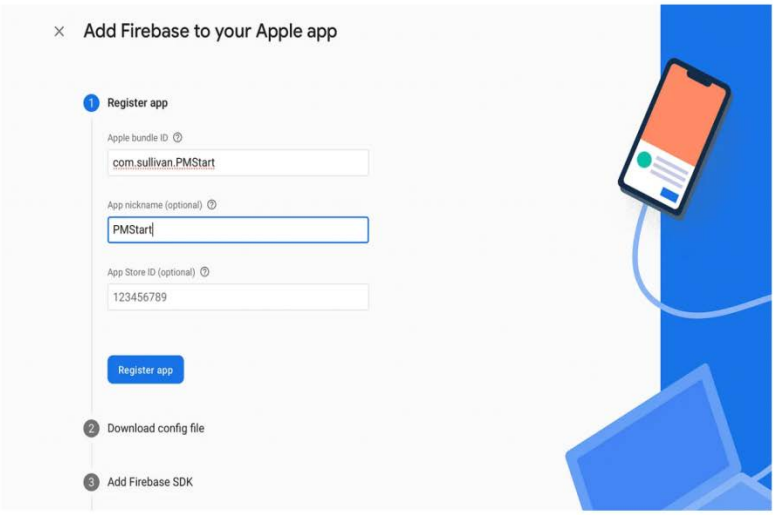
Μπαίνοντας στην αρχική σελιδα της Firebase Firebase.google.com , πατώντας πάνω δεξια στο κουμπί go to console ανοίγεται στον χρήστη μια νέα σελίδα στην οποία μπορεί με την επιλογή add project να ξεκινήσει την δική του βάση δεδομένων εφόσον προσθέσει το όνομα του προτζεκτ. Αμέσως μετά πρέπει να διαλέξει πλατφόρμα στην προκειμένη περίπτωση iOS platform. Ύστερα πρέπει να γίνει η αντιγραφή του bundleID από το Xcode και να την επικολλήσει στο apple bundle ID πεδίο.



Εικόνα 3.3.1 Είσοδος στο Firebase

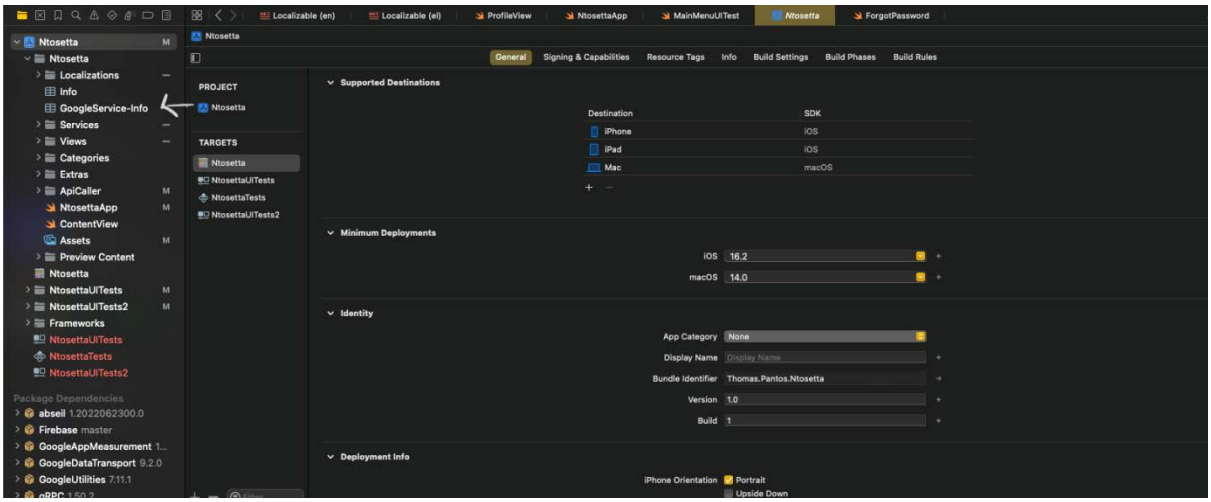


Εικόνα 3.3.2 Επιλογή ή δημιουργία πρότζεκτ



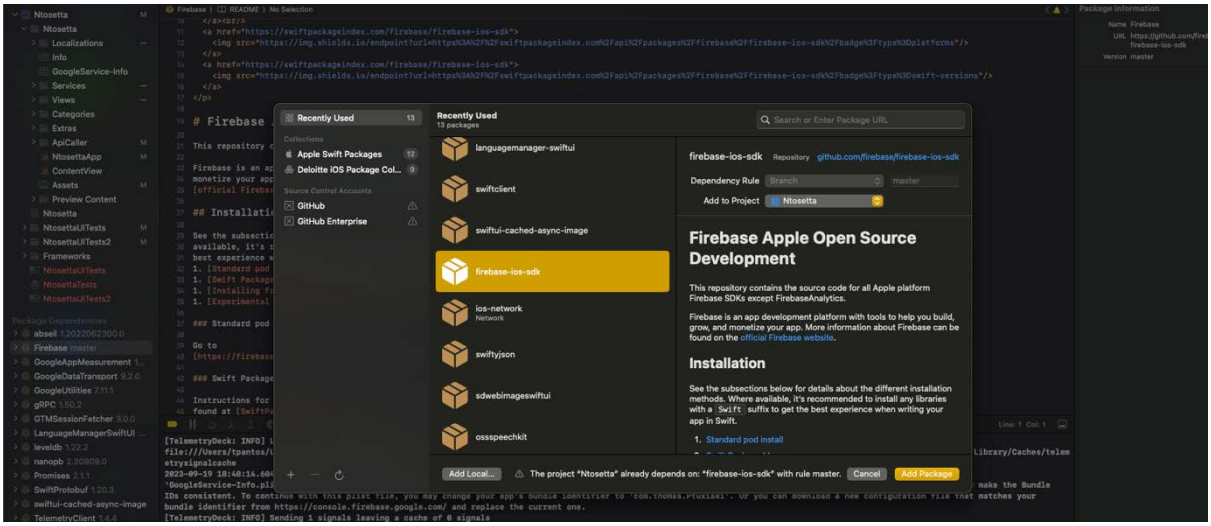
Εικόνα 3.3.3 Προσθήκη apple bundle ID

Έπειτα πρέπει να κατέβει το config αρχείο GoogleService-info.plist και να προστεθεί μέσα στο φάκελο του πρότζεκτ. Είναι σημαντικό να προστεθεί αυτό το αρχείο διότι χωρίς το αρχείο αυτό , η εφαρμογή δεν θα ανταποκρίνεται και θα τερματιστεί με την firebase αρχικοποιημένη.



Εικόνα 3.3.4 Προσθήκη GoogleService-info.plist

Τέλος πρέπει να γίνει η εγκατάσταση της βιβλιοθήκης της Firebase μέσω του Swift Package Manager ώστε να μπορούν να χρησιμοποιηθούν σωστά οι εντολές της αυθεντικότητας του χρήστη όσο και των άρθρων.



Εικόνα 3.3.5 Εγκατάσταση της βιβλιοθήκης

Κεφάλαιο 4 Τι είναι το testing?

4.0 Testing

Το testing στη Swift αναφέρεται στη διαδικασία επαλήθευσης και επικύρωσης της λειτουργικότητας, της ορθότητας και της απόδοσης του κώδικα Swift μέσω της εκτέλεσης test. Περιλαμβάνει τη συγγραφή περιπτώσεων test που ασκούν διάφορα μέρη της βάσης κώδικα για να διασφαλίσουν ότι συμπεριφέρονται όπως προβλέπεται.

Στη Swift, υπάρχουν διάφορα πλαίσια δοκιμών που είναι διαθέσιμα για τη συγγραφή και εκτέλεση δοκιμών, μεταξύ των οποίων:

4.1 UI Testing

Η δοκιμή του χρήστη (UI testing), ή δοκιμή της διεπαφής χρήστη, είναι μια τεχνική δοκιμής λογισμικού που επικεντρώνεται στον έλεγχο της λειτουργικότητας, της χρηστικότητας και της συνοχής μιας γραφικής διεπαφής χρήστη (GUI). Περιλαμβάνει τον έλεγχο της αλληλεπίδρασης μεταξύ του χρήστη και των στοιχείων της διεπαφής χρήστη της εφαρμογής, όπως κουμπιά, μενού, φόρμες και οπτικά στοιχεία.

Ο στόχος της δοκιμής της διεπαφής χρήστη είναι να εξασφαλιστεί ότι η διεπαφή χρήστη συμπεριφέρεται όπως αναμένεται και πληροί τις απαιτήσεις και τις προδιαγραφές σχεδίασης. Περιλαμβάνει τον έλεγχο ότι τα στοιχεία της διεπαφής εμφανίζονται σωστά ,την δυνατότητα του χρήστη να αλληλοεπιδράσει με αυτά όπως προβλέπεται, και την παραγωγή των αναμενόμενων αποτελεσμάτων.

Η δοκιμή της διεπαφής χρήστη μπορεί να καλύπτει διάφορες πτυχές, συμπεριλαμβανομένων:

1. Λειτουργική δοκιμή: Επαληθεύει την σωστή λειτουργία της διεπαφής χρήστη και την εκτέλεση των επιθυμητών ενεργειών. Περιλαμβάνει τον έλεγχο της πλοήγησης, του ελέγχου της εισόδου, των ενεργειών κουμπιών, της υποβολής φόρμας και άλλων αλληλεπιδραστικών στοιχείων.

2. Δοκιμή χρηστικότητας: Αξιολογεί τη χρηστικότητα της διεπαφής. Επικεντρώνεται στην αξιολόγηση της ευκολίας χρήσης, της καταλληλόλητας και της συνολικής εμπειρίας χρήστη. Η δοκιμή χρηστικότητας μπορεί να περιλαμβάνει τη συλλογή σχολίων από πραγματικούς χρήστες για την αναγνώριση οποιωνδήποτε προβλημάτων χρηστικότητας ή πεδίων βελτίωσης.

3. Δοκιμή οπτικής συνοχής: Βεβαιώνει ότι τα οπτικά στοιχεία της διεπαφής χρήστη, όπως γραμματοσειρές, χρώματα, εικονίδια και διάταξη, είναι συνεπή σε διάφορες οθόνες και αναλύσεις. Αυτό βοηθά στη διατήρηση μιας συνεκτικής και οπτικά ελκυστικής εμπειρίας χρήστη.

4. Δοκιμή ανάμεσα σε πλατφόρμες:

Επαληθεύει ότι η διεπαφή χρήστη λειτουργεί σωστά και έχει ομοιόμορφη εμφάνιση σε διάφορες πλατφόρμες και συσκευές. Αυτό περιλαμβάνει τη δοκιμή σε διάφορα λειτουργικά συστήματα, περιηγητές ιστού, κινητές συσκευές και μεγέθη οθονών.

5. Δοκιμή τοπικής προσαρμογής: Ελέγχει ότι η διεπαφή χρήστη έχει μεταφραστεί σωστά και προσαρμοστεί σε διάφορες γλώσσες και πολιτισμικές συμβάσεις. Περιλαμβάνει τον έλεγχο της ακρίβειας και την καταλληλόλητα του τοπικοποιημένου κειμένου, των μορφών ημερομηνίας, των μορφών αριθμών και άλλων στοιχείων που εξαρτώνται από τη γλώσσα.

Η δοκιμή της διεπαφής χρήστης μπορεί να πραγματοποιηθεί με τη χειροκίνητη μέθοδο, όπου οι δοκιμαστές αλληλεπιδρούν με τη διεπαφή της εφαρμογής και παρατηρούν τη συμπεριφορά της, ή μπορεί να αυτοματοποιηθεί χρησιμοποιώντας ειδικά εργαλεία δοκιμής. Η αυτοματοποιημένη δοκιμή της διεπαφής χρήστης περιλαμβάνει τη δημιουργία σεναρίων ή δοκιμαστικών περιπτώσεων που προσομοιώνουν τις αλληλεπιδράσεις του χρήστη και επαληθεύουν τα αναμενόμενα αποτελέσματα αυτόματα.

Με την εκτέλεση πλήρους δοκιμής της διεπαφής χρήστης, οι ομάδες ανάπτυξης λογισμικού μπορούν να ανιχνεύσουν και να διορθώσουν προβλήματα που σχετίζονται με τη διεπαφή χρήστη από τα πρώτα στάδια του κύκλου ανάπτυξης, με αποτέλεσμα να παράγεται μια πιο αξιόπιστη, χρηστική και οπτικά ελκυστική εφαρμογή για τους χρήστες.

4.2 UI Testing Execution

```
        app.launch()

    }


    func testLoginButtonTapped() {
        let emailTextField = app.textFields["Email"]
        let passwordTextField = app.secureTextFields["Password"]
        let loginButton = app.buttons["Login"]


        XCTAssertTrue(emailTextField.exists)
        XCTAssertTrue(passwordTextField.exists)

        XCTAssertTrue(loginButton.exists)


        // When
        emailTextField.tap()
        emailTextField.typeText("tak@rak.gr")

        passwordTextField.tap()
        passwordTextField.typeText("123456")



        loginButton.tap()


    }
```

Εικόνα 4.2.1 1ο UI Test

Στην παραπάνω εικόνα βλέπουμε το πρώτο test της εφαρμογής. Η συνάρτηση func testLoginButtonTapped() είναι η πρώτη συνάρτηση που ολοκληρώνεται σε αυτό το αρχείο, παρακάτω θα δούμε τις υπόλοιπες συναρτήσεις και σε άλλες οθόνες. Στην συνάρτηση ξεκινάμε δηλώνοντας τα στοιχεία που θέλουμε να δοκιμάσουμε τα οποία είναι : δύο πεδία για να εισάγει ο χρήστης τα στοιχεία του (email , password) και ένα κουμπί login. Μετά την δήλωση των στοιχείων χρησιμοποιούμε τη μέθοδο

XCAssertTrue που στην πραγματικότητα επιστρέφει ένα Boolean που πρέπει να είναι πάντα αληθής. Αυτό που κάνουμε είναι να ελέγχουμε αν το στοιχείο υπάρχει στην οθόνη μας. Με τη XCAssertTrue(emailTextField.exists) κλείδώνουμε την ύπαρξη του πεδίου όπου ο χρήστης εισάγει το email του. Έπειτα κάνουμε τα ίδια και παρακάτω με το passwordTextField και το κουμπί login. Αμέσως μετά με το emailTextField.tap() το τεστ αγγίζει πάνω στο πεδίο του email και με το typeText() γράφει αυτόματα αυτά που θέλουμε να δοκιμάσουμε. Αφού το τεστ κλείσει επιτυχώς θα μας βγάλει δίπλα από το όνομα της συνάρτησης ένα πράσινο ρόμβο που σημαίνει ότι το τεστ μας έτρεξε σωστά και ήταν επιτυχής. Το πρώτο παράδειγμα ήταν μόνο για το login του χρήστη , παρακάτω θα δούμε όλες τις περιπτώσεις που ο χρήστης είτε κάνει register ή κάνει κάποιο τυπογραφικό λάθος στο textfield.

```
47
    func testLoginButtonTappedWrongly() {
49
        let errorAlert = app.alerts["Error"]
51
        let emailTextField = app.textFields["Email"]
52
        let passwordTextField = app.secureTextFields["Password"]
53
        let loginButton = app.buttons["Login"]
54
        let okButton = errorAlert.scrollViews.otherElements.buttons["OK"]
55
56
57
        XCTAssertTrue(emailTextField.exists)
58
        XCTAssertTrue(passwordTextField.exists)
59
60
        XCTAssertTrue(loginButton.exists)
61
62
        // When
63
        emailTextField.tap()
64
        emailTextField.typeText("example@rak.gr")
65
66
        passwordTextField.tap()
67
        passwordTextField.typeText("123456")
68
69
        loginButton.tap()
70
71
        okButton.tap()
72
73
74
75
76
    }
77
```

Εικόνα 4.2.2 Περίπτωση λάθος στοιχείων χρήστη

Στη παραπάνω συνάρτηση εισάγουμε στο τεστ λάθος στοιχεία με αποτέλεσμα η εφαρμογή να μας πετάξει ένα errorAlert κουμπί με το OK να αναγράφεται από μέσα.

```
80
    func testForgotButton() throws {
81
        let forgotBtn = app.buttons["Forgot Password?"]
82
        let forgotTxtFld = app.textFields["EmailFgt"]
83
        let senderBtn = app.buttons["SendPasswordReset"]
84
85
        XCTAssertTrue(forgotBtn.exists)
86
87
        forgotBtn.tap()
88
89
        XCTAssertTrue(forgotTxtFld.exists)
90
91
        forgotTxtFld.tap()
92
        forgotTxtFld.typeText("tommyaek@gmail.com")
93
94
        senderBtn.tap()
95
96
97
98
    }
99
100
```

Εικόνα 4.2.3 Περίπτωση Forgot Password

Σε αυτή τη συνάρτηση ο χρήστης πατάει το στην οθόνη το κουμπί του forgot password και κάνει pop up ένα modal στο οποίο συμπληρώνει το email στο πεδίο και έπειτα έρχεται στο email του χρήστη η φόρμα αλλαγής κωδικού.

```
    func testRegisterBtn() throws {

        let registerBtn = app.buttons["Register"]
        let emailTextFieldRegister = app.textFields["TextFieldRegister"]
        let passwordSecureTextFieldRegister = app.secureTextFields["PassReg"]
        let firstNameTextFieldRegister = app.textFields["FirstName"]
        let lastNameTextFieldRegister = app.textFields["LastName"]
        let registerSenderBtn = app.buttons["FinalRegister"]

        XCTAssertTrue(registerBtn.exists)

        registerBtn.tap()

        XCTAssertTrue(emailTextFieldRegister.exists)
        XCTAssertTrue(passwordSecureTextFieldRegister.exists)
        XCTAssertTrue(firstNameTextFieldRegister.exists)
        XCTAssertTrue(lastNameTextFieldRegister.exists)
        XCTAssertTrue(registerSenderBtn.exists)

        emailTextFieldRegister.tap()
        emailTextFieldRegister.typeText("tommyaek@gmail.com")

        passwordSecureTextFieldRegister.tap()
        passwordSecureTextFieldRegister.typeText("123456")

        firstNameTextFieldRegister.tap()
        firstNameTextFieldRegister.typeText("Thomas")

        lastNameTextFieldRegister.tap()
        lastNameTextFieldRegister.typeText("Pantos")

        registerSenderBtn.tap()

    }
```

Εικόνα 4.2.4 Περίπτωση register χρήστη

Στην παραπάνω συνάρτηση βλέπουμε δηλωμένα στοιχεία που στην πραγματικότητα δεν υπάρχουν στην οθόνη login εκτός του register button. Αυτό συμβαίνει διότι η φόρμα της εγγραφής του χρήστη βρίσκεται μέσα στο modal που θα γίνει pop up αφού πατηθέι το κουμπί του register. Έπειτα ο χρήστης θα εισάγει τα στοιχεία του με το typetext και εφόσον δεν υπάρχει κάποιος άλλος χρήστης με αυτό το email θα δημιουργηθεί στη βάση και έπειτα θα γίνει αυτόματα εισαγωγή στην αρχική σελίδα της εφαρμογής .

```
func testArticles() throws {

    let emailTextField = app.textFields["Email"]
    let passwordTextField = app.secureTextFields["Password"]
    let loginButton = app.buttons["Login"]
    let tabBar = app.tabBars["Tab Bar"]
    let profileButton = tabBar.buttons["Profile"]
    let homeTabButton = tabBar.buttons["Home"]
    let first = app.collectionViews.firstMatch
    let backButton = app.navigationBars.buttons.element(boundBy: 0)
    XCTAssertTrue(emailTextField.exists)
    XCTAssertTrue(passwordTextField.exists)

    XCTAssertTrue(loginButton.exists)


    // When
    emailTextField.tap()
    emailTextField.typeText("tak@rak.gr")

    passwordTextField.tap()
    passwordTextField.typeText("123456")



    loginButton.tap()


    XCTAssertTrue(tabBar.waitForExistence(timeout: 10))

    profileButton.tap()
    homeTabButton.tap()
    first.tap()

    XCTAssertTrue(backButton.waitForExistence(timeout: 3))
    backButton.tap()
```

Line: 86 Col: 60

Εικόνα 4.2.5 Περίπτωση ανοιγματος πρώτου άρθρου

Στην παραπάνω εικόνα υλοποιείται η πράξη που ο χρήστης εισέρχεται στην εφαρμογή και πατάει πάνω στο πρώτο άρθρο που ανοίγει στη λίστα. Η εντολή που είναι υπεύθυνη για αυτό είναι η μεταβλητή first η οποία μέσα περιέχει την εντολή app.collectionViews.firstMatch που σημαίνει ότι πιάνει το πρώτο στοιχείο ενός πίνακα και έπειτα γίνεται tap και έτσι προέρχεται η εντολή first.tap() που καλείται στο τέλος της συνάρτησης με σκοπό ο χρήστης να ανοίξει το πρώτο άρθρο και στοιχείο του πίνακα.

```
setta ⟩ NtosettaUITests2 ⟩ MainMenuUITest ⟩ testLikeButtonAndSaveArticles()
class MainMenuUITest: XCTestCase {
    func testArticles() throws {


    }
    func testLikeButtonAndSaveArticles() throws {
        let app = XCUIApplication()

        let tabBar = app.tabBars["Tab Bar"]
        let profileButton = tabBar.buttons["Profile"]
        let homeTabButton = tabBar.buttons["Home"]
        let first = app.collectionViews.firstMatch



        let elementsQuery = app.scrollViews.otherElements
        XCTAssertTrue(tabBar.waitForExistence(timeout: 10))

        first.tap()


        let loveButton = elementsQuery.buttons["Love"]
        loveButton.tap()
        loveButton.tap()

        profileButton.tap()
        XCTAssertTrue(tabBar.waitForExistence(timeout: 2))

        homeTabButton.tap()

        app.navigationBars["_TtGC7SwiftUI32NavigationStackHosting"].buttons["Ntosetta"].tap()
    }

}
```

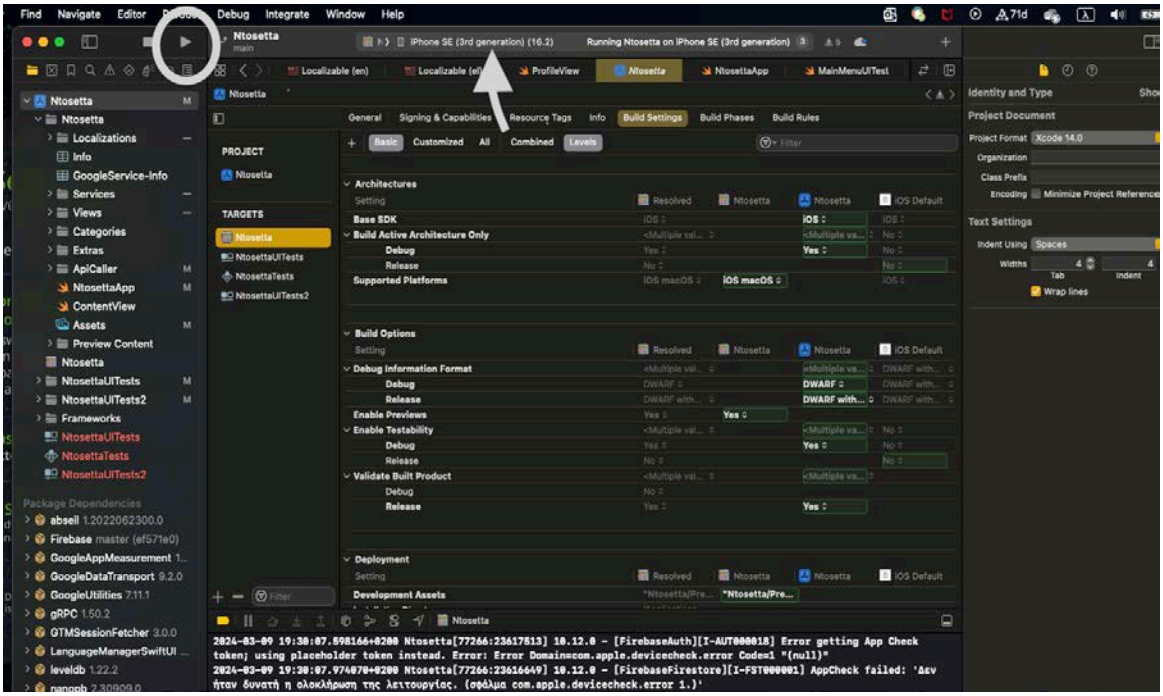
Εικόνα 4.2.6 Περίπτωση πατήματος like στο άρθρο

Στο παραπάνω τεστ η συνάρτηση είναι παρόμοια με την προηγούμενη αλλά εφόσον έχει γίνει ήδη σωστά το login δεν θα χρειαστεί να ξαναπεραστούν τα στοιχεία του εκάστοτε χρήστη. Εφόσον εισέρχεται ο χρήστης στο κεντρικό μενού, το πρώτο άρθρο του πίνακα πατιέται και έτσι ο χρήστης διαβάζει το άρθρο. Έπειτα ο χρήστης πατάει το like button του άρθρου, το άρθρο αποθηκεύεται στην λίστα του εκάστοτε χρήστη με το πίνακα με τα αγαπημένα άρθρα. Αμέσως μετά ο χρήστης πατάει το profileButton το οποίο τον κατευθύνει στην οθόνη με τα προσωπικά στοιχεία του χρήστη. Εκεί γίνεται ένα display με το πίνακα που ο χρήστης έχει πατήσει like. Ύστερα ο χρήστης πατάει το homeButton μεταφέρεται στο κεντρικό μενού πάλι και τερματίζει το τεστ επιτυχώς.

Κεφάλαιο 5 Εκτέλεση της εφαρμογής

5.1 Έναρξη του περιβάλλοντος XCode

Όπου υπάρχει όλη η δομή της εφαρμογής εκτός και αν η εφαρμογή θα γραφτεί σε όχι προσωπικό μας macOS συσκευή τότε υπάρχει η δυνατότητα να έχουμε ήδη δημιουργήσει σε repository στο GitHub την εφαρμογή και να την κάνουμε clone με το εκάστοτε σύνδεσμο του repository. Έπειτα το XCode θα χρειαστεί ένα διάστημα από 1 έως 5 λεπτών αναλογικά με το πόσο φορτισμένη είναι η εφαρμογή από βιβλιοθήκες .Στην παρακάτω εικόνα υπάρχει η αρχική δομή της εφαρμογή και στη συνέχεια για να εκτελεστεί η εφαρμογή πρέπει να πατηθεί το κουμπί run (μέσα στο κύκλο) και να επιλέχθει η αντίστοιχη iOS συσκευή (βελάκι) .

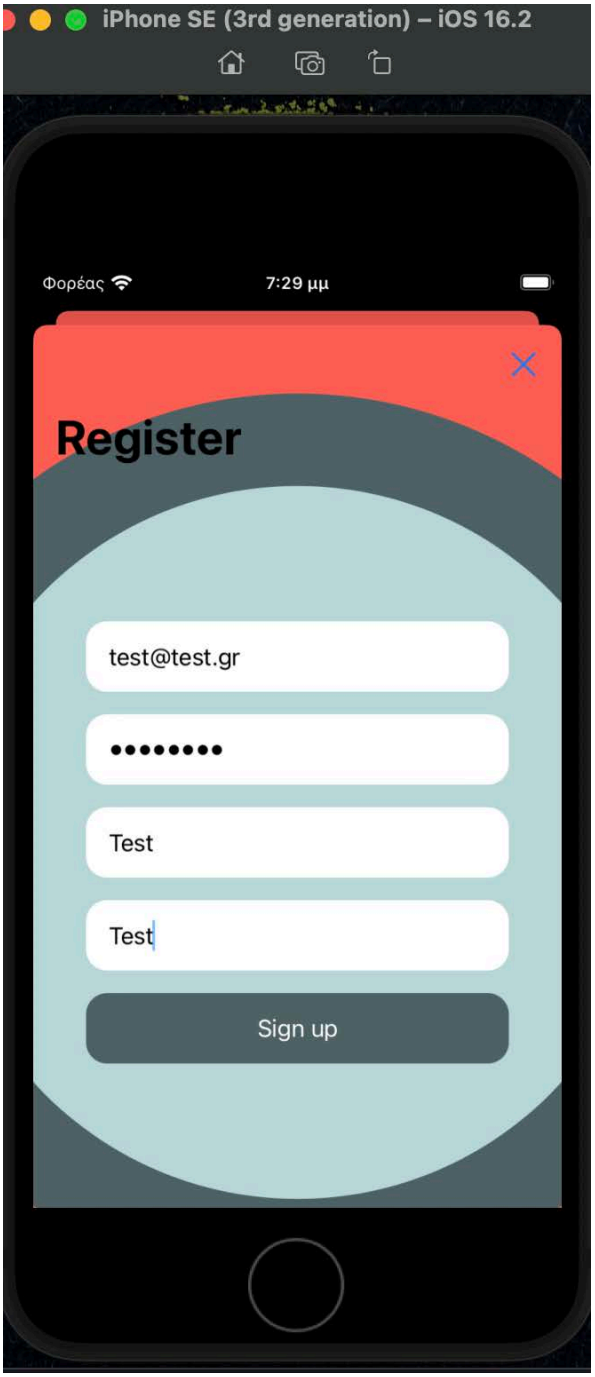


Εικόνα 5.1.1 Εκτέλεση εφαρμογής

5.2 Εκτέλεση εγγραφής χρήστη

Η εγγραφή χρήστη γίνεται με τον εξής τρόπο:

Εμφάνιση ενός εισαγωγικού μενού με τις επιλογές σύνδεση ή εγγραφή όπου ο χρήστης επιλέγει την επιλογή της εγγραφής εφόσον δεν έχει ακόμα λογαριασμό. Μόλις το κουμπί της εγγραφής πατηθεί θα εμφανιστεί ένα modal με μια φόρμα στοιχείων όπου ο χρήστης θα πρέπει να συμπληρώσει το email , το κωδικό και το ονοματεπώνυμό του με την προϋπόθεση το email να μην έχει ξαναχρησιμοποιηθεί ώστε να μην δημιουργηθεί κάποιο error στην οθόνη του. Αμέσως μετά την δημιουργία της φόρμας του χρήστη στη βάση δεδομένων ο χρήστης περνάει απευθείας στο welcome screen της οθόνης (βλ παρακάτω) όπου και θα καταλήξει στο αρχικό μενού της εφαρμογής.

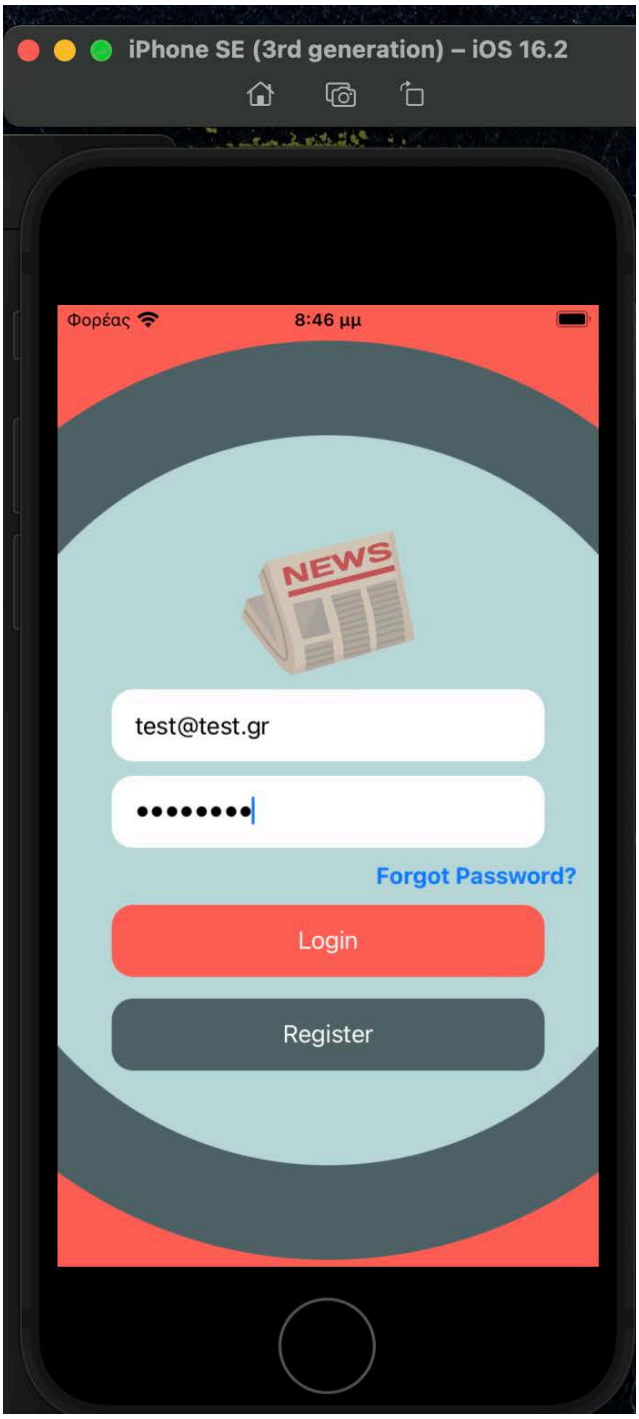


5.2.1 Φόρμα στοιχείων εγγραφής χρήστη

5.3 Σύνδεση χρήστη

Η σύνδεση χρήστη αποτελεί ένα κρίσιμο στοιχείο σε πολλές εφαρμογές, προσφέροντας την δυνατότητα ασφαλούς και προστατευμένης πρόσβασης στο περιεχόμενο. Η οθόνη της σύνδεσης είναι η εισαγωγική οθόνη της εφαρμογής που συμπεριλαμβάνει 2 πεδία κειμένου 1 για το email του χρήστη και 1 για τον κωδικό πρόσβασης. Μόλις ο χρήστης γράψει σωστά τα

credentials του και πατήσει το κουμπί Login θα κατευθυνθεί στο welcome screen και έπειτα όπως και στην εγγραφή θα καταλήξει στο αρχικό μενού της εφαρμογής.



Εικόνα 5.3.1 Σύνδεση χρήστη

5.4 Welcome Screen & Logout

Η οθόνη καλωσορίσματος είναι η οθόνη η οποία εμφανίζεται αμέσως μετά από το authentication της βάσης δεδομένων ανεξαρτήτως εάν ο χρήστης προέρχεται από σύνδεση ή από εγγραφή. Η πρώτη εντύπωση πρέπει να είναι σημαντική , γι’ αυτό το λόγο η οθόνη καλωσορίσματος είναι εξίσου σημαντική διότι δίνει στους χρήστες μια πρώτη επαφή με την εφαρμογή. Επίσης η οθόνη καλωσορίσματος θα μπορούσε να χρησιμοποιηθεί ακόμα και για διαφήμιση. Μέσω του welcome screen μπορεί να παρουσιαστούν νέα προϊόντα, υπηρεσίες ή ακόμα και προσφορές.

Συνολικά η οθόνη καλωσορίσματος διαδραματίζει κρίσιμο ρόλο στην πρώτη εντύπωση που δημιουργείται στους χρήστες και μπορεί να βελτιώσει την χρηστικότητα και την επικοινωνία μεταξύ της εφαρμογής και του κοινού.

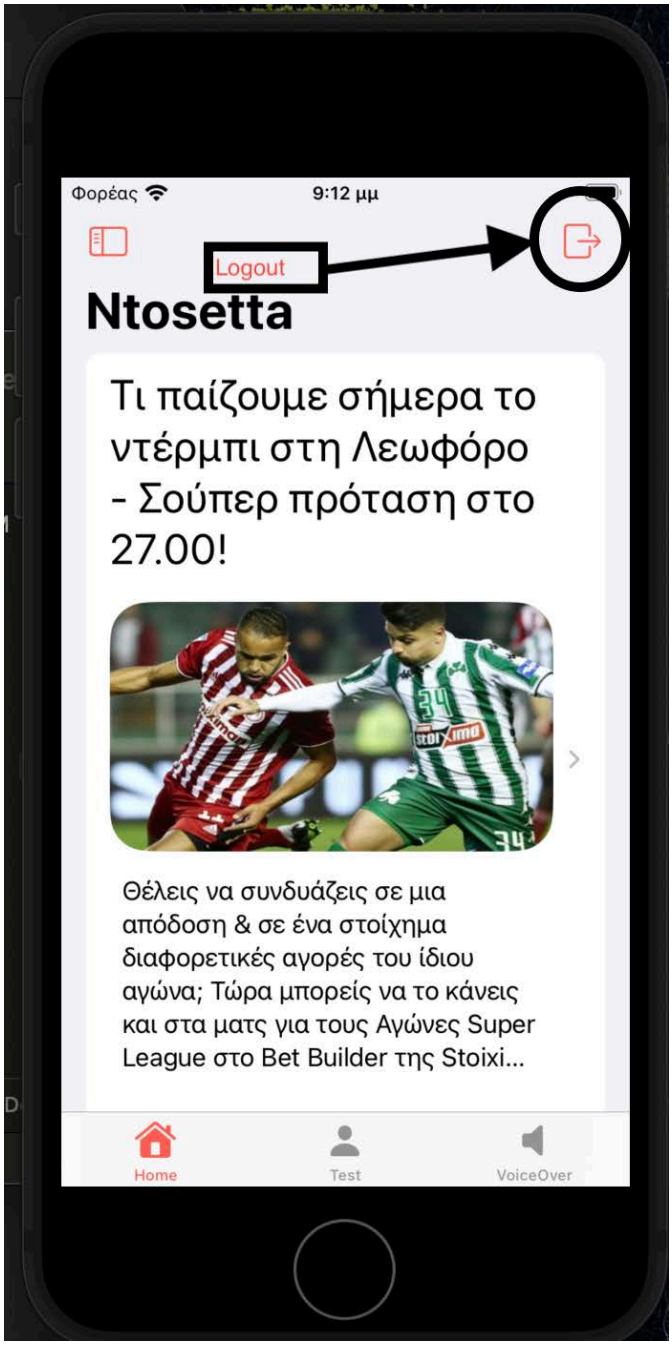
Εικόνα 5.4 Welcome Screen

Logout είναι το κουμπί το οποίο βρίσκεται στην πάνω δεξιά γωνία του αρχικού μενού και ο σκοπός του είναι να κάνει αποσύνδεση τον χρήστη από την εφαρμογή. Γιατί είναι σημαντική η ύπαρξη ενός τέτοιου button;

Το κουμπι αποσύνδεσης είναι σημαντικό για τους εξής λόγους:

1) Προστασία ασφάλειας: Οι χρήστες πρέπει να έχουν τη δυνατότητα να αποσυνδέονται από τον λογαριασμό τους για λόγους ασφαλείας. Αν παραμείνουν συνδεδεμένοι σε ένα δημόσιο ή κοινό υπολογιστή, η ενέργεια αποσύνδεσης προστατεύει τα προσωπικά τους δεδομένα από ανεπιθύμητη πρόσβαση.
2) Διαχείριση Λογαριασμού: Ο κουμπί αποσύνδεσης παρέχει στους χρήστες τη δυνατότητα να διαχειρίζονται τον λογαριασμό τους. Μπορεί να χρησιμοποιηθεί για την εναλλαγή μεταξύ πολλών λογαριασμών, αν υποστηρίζονται, ή για την απλή αποσύνδεση από έναν λογαριασμό πριν την απόλυτη κλείσιμο της εφαρμογής.
3) Διαφάνεια και Εμπιστοσύνη: Η παρουσία ενός κουμπιού αποσύνδεσης δείχνει στους χρήστες ότι έχουν τον έλεγχο του λογαριασμού τους και μπορούν να αποσυνδεθούν ανά πάσα στιγμή. Αυτό δημιουργεί μια αίσθηση εμπιστοσύνης και διαφάνειας μεταξύ των χρηστών και της εφαρμογής.
4) Διαχείριση Συνεδρίας: Η αποσύνδεση επιτρέπει στην εφαρμογή να διαχειρίζεται τις συνεδρίες των χρηστών. Με την αποσύνδεση, οποιαδήποτε ενεργή σύνδεση ή στοιχεία συνεδρίας διαγράφονται, βοηθώντας έτσι στην αποφυγή ανεπιθύμητης πρόσβασης σε προσωπικά δεδομένα

Η παρουσία του logout κουμπιού είναι απαραίτητη για την διασφάλιση της ασφάλειας, της διαχείρισης του λογαριασμού και της εμπιστοσύνης των χρηστών στην εφαρμογή.



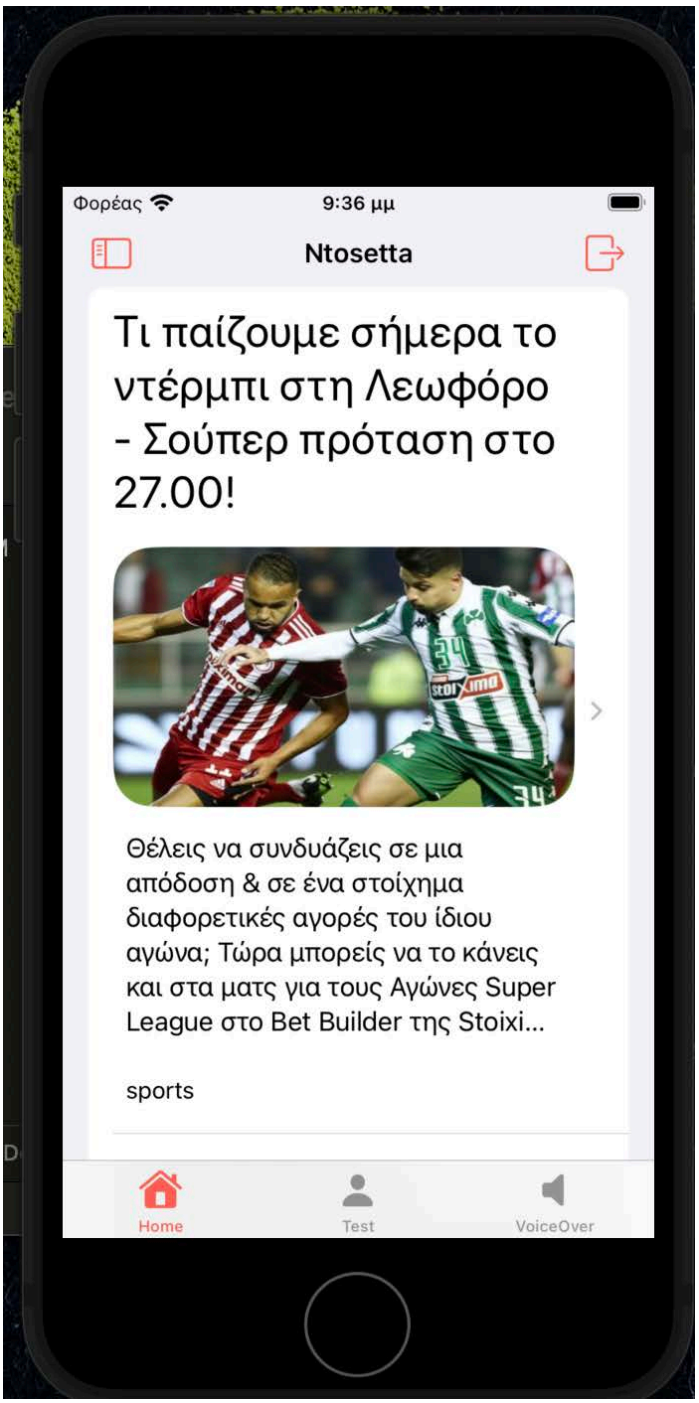
Εικόνα 5.4.2 Κουμπί αποσύνδεσης

5.5 Αρχικό μενού

Το αρχικό μενού της εφαρμογής αποτελείται από 3 tab : το Home, το profile και το voice over.

5.5.1 Home screen

Το Home screen είναι η πρώτη οθόνη που συναντάει ο χρήστης αφού έχει γίνει ταυτοποίηση από τη βάση και έχει περάσει από το welcome screen. Η οθόνη περιέχει την λίστα όλων των άρθρων ανεξαρτήτως κατηγορίας σε ένα πλαίσιο όπου το κάθε άρθρο παρουσιάζεται σαν μικρογραφία με συγκεκριμένης διάστασης εικόνες, κείμενα και φυσικά τον τίτλο και τη κατηγορία του άρθρου.



5.5.1 Εικόνα μικρογραφίας άρθρου

Όταν ο χρήστης πατήσει πάνω στο άρθρο στο οποίο έχει επιλέξει θα μεταφερθεί σε μια πιο λεπτομερείς οθόνη η οποία ξετυλίγει όλο το κείμενο του άρθρου όπως και το μέγεθος της εικόνας. Όταν ο χρήστης τελειώσει την ανάγνωση του μπορεί να κάνει ένα

pop-out με το κουμπί πάνω αριστερά όπου δημιουργείται αυτόματα όταν γίνεται navigate από οθόνη σε οθόνη.

Εικόνα 5.5.2 Λεπτομερής οθόνη

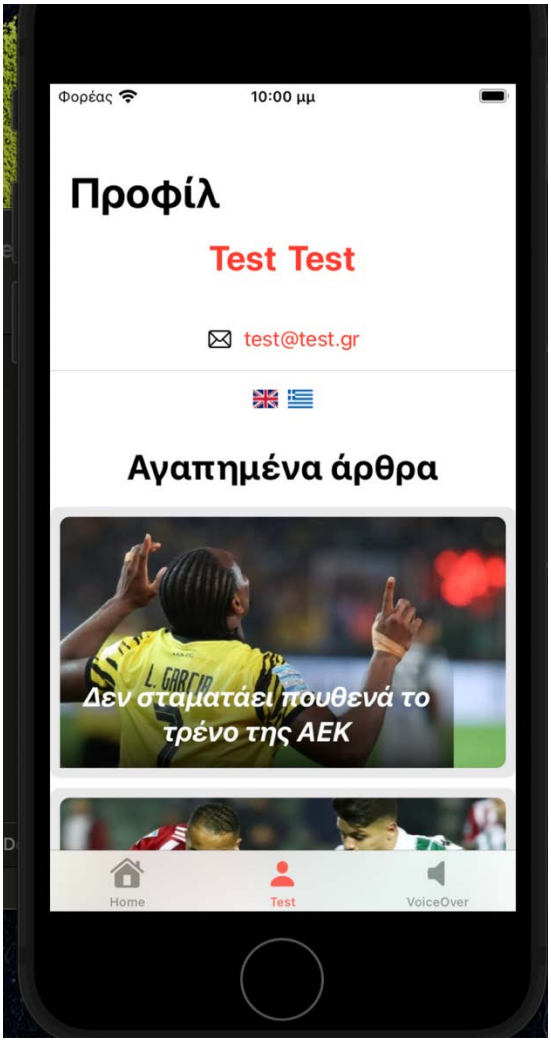
5.5.2 Interaction του χρήστη με την εφαρμογή

Ο χρήστης κατά την διάρκεια της ανάγνωσης του άρθρου στο detailed view μπορεί να κάνει like στο άρθρο και αυτό έχει ως αποτέλεσμα να αποθηκεύει τα άρθρα σε ένα πίνακα που βρίσκεται στο profile view στην ενότητα αγαπημένα άρθρα. Με αυτό το τρόπο ο χρήστης μπορεί να βρει τα άρθρα που θέλει να ξαναδιαβάσει στο μέλλον.

Εικόνα 5.5.2.1 Κάτω μέρος του detailed view

5.5.3 Profile View

Το Profile view είναι η οθόνη στην οποία περιέχονται τα προσωπικά στοιχεία του χρήστη όπως το email και το ονοματεπώνυμο όπως και επίσης τα αγαπημένα άρθρα του χρήστη που έκανε προηγουμένως like ώστε να τα ξαναδιαβάσει αλλά και να τα ακούσει.



Εικόνα 5.5.3.1 Profile View

5.5.4 Voice Over View

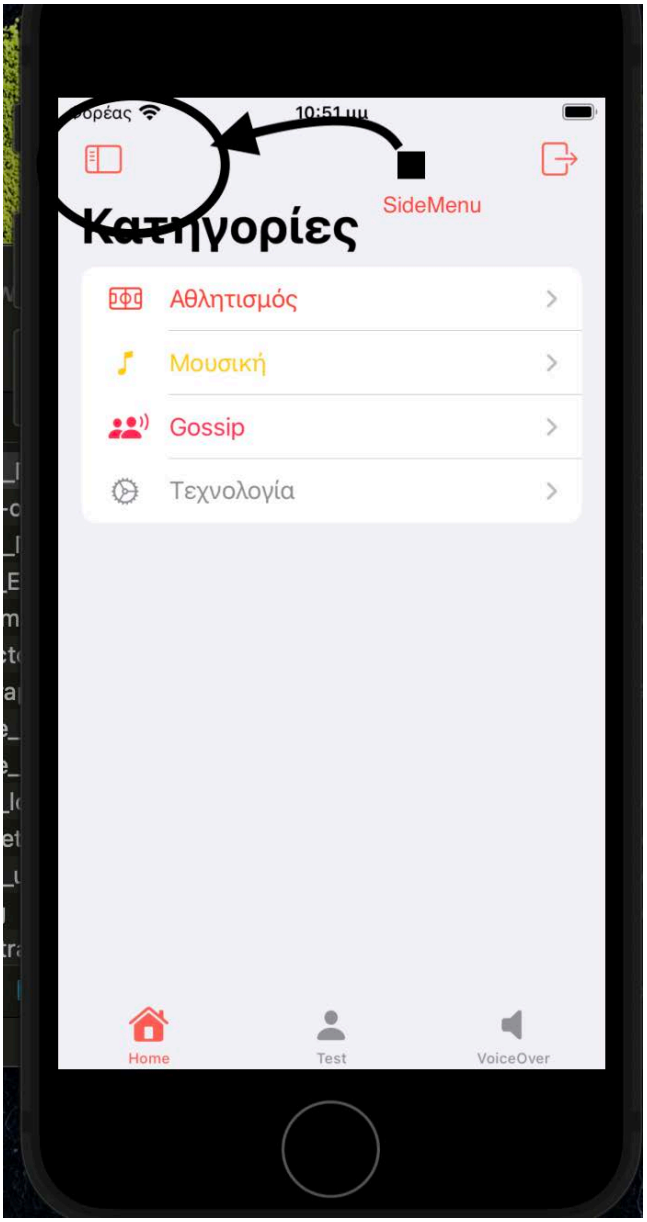
Η μετατροπή κειμένου σε ομιλία είναι μια σημαντική λειτουργία που επιτρέπει στην εφαρμογή να διαβάζει δυναμικά κείμενο και να το μετατρέπει σε ομιλία, προσφέροντας έτσι μια πιο επικοινωνιακή και προσβάσιμη εμπειρία στους χρήστες. Αυτή η λειτουργία είναι χρήσιμη για να διαβάζονται αυτόματα κείμενα όπως άρθρα, ειδήσεις ή μηνύματα σε φωνητική μορφή, κάνοντας την πλοήγηση στην εφαρμογή πιο εύκολη και βοηθώντας τους χρήστες να επωφεληθούν πλήρως από το περιεχόμενο.

Στην συγκριμένη οθόνη ο χρήστης μπορεί να ακούσει όλα τα υπάρχοντα άρθρα που περιέχει η εφαρμογή με το κουμπί play και να σταματήσει να ακούει με το κουμπί stop. Εάν ο χρήστης επιθυμεί να ακούσει συγκεκριμένο άρθρο έχει την δυνατότητα να μπει στην λεπτομερής οθόνη του άρθρου στο κάτω μέρος της οθόνης περιέχει το αριστερό κουμπί με το εικονίδιο ενός ηχείου που σημαίνει ότι το άρθρο μπορεί να γίνει μετατροπή σε ομιλία.

Εικόνα 5.5.4.1 Voice over View

5.6 Κατηγορίες ειδήσεων

Τα άρθρα είναι χωρισμένα σε 4 κατηγορίες άρθρων, Αθλητικά (sports) , κουτσομπολιό(gossip), μουσική(music) και τεχνολογία(technology). Ο χρήστης μπορεί να διαλέξει από το side menu ποια κατηγορία θέλει να επιλέξει ώστε να γίνει η κλήση της ανάλογης συνάρτησης. Για παράδειγμα εάν ο χρήστης θέλει να διαβάζει μόνο αθλητικά θα πατήσει το κουμπί πάνω αριστερά στην αρχική οθόνη ώστε να γίνει trigger το Boolean του sidebar και να εμφανιστεί η ανάλογη οθόνη.



Εικόνα 5.6.1 SideMenu

Μόλις ο χρήστης επιλέξει την κατηγορία θα εμφανιστεί μια φιλτραρισμένη αρχική οθόνη με μόνο τα ανάλογα άρθρα μέσα.

Συμπεράσματα

Το framework SwiftUI με το οποίο υλοποιήθηκε η εφαρμογή είναι η νέα γενιά του iOS development δίνοντας την δυνατότητα στον δημιουργό να κάνει preview κάθε οθόνη χωρίς να τρέξει το πρόγραμμα στο simulator που εξοικονομεί χρόνο σε αντίθεση με το UI Kit το οποίο δημιουργεί native iOS εφαρμογές με τον τρόπο του storyboard είτε με auto layout.

Καθώς η τεχνολογική βιομηχανία συνεχίζει να εξελίσσεται, οι προσδοκίες και οι απαιτήσεις των χρηστών ακολουθούν. Οι χρήστες αναζητούν ολοένα και περισσότερο εφαρμογές που δεν είναι μόνο πλούσιες σε χαρακτηριστικά αλλά και διαισθητικές, οπτικά ελκυστικές και προσβάσιμες. Η ευθυγράμμιση του SwiftUI με αυτές τις εξελισσόμενες προτιμήσεις των χρηστών το καθιστά ένα απαραίτητο εργαλείο για τους προγραμματιστές και μια συναρπαστική επιλογή για τις επιχειρήσεις που επιδιώκουν να προσελκύσουν και να διατηρήσουν τους χρήστες.

Εν κατακλείδι, το SwiftUI αντιπροσωπεύει μια καίρια αλλαγή στο πρότυπο ανάπτυξης εφαρμογών iOS, προσφέροντας μια νέα προοπτική για τον τρόπο με τον οποίο σχεδιάζουμε και δημιουργούμε εφαρμογές. Ο εναγκαλισμός του δηλωτικού προγραμματισμού, η συμβατότητα μεταξύ διαφορετικών πλατφορμών και το ολοκληρωμένο σύνολο χαρακτηριστικών θέτουν τις βάσεις για να διαμορφώσει το SwiftUI το μέλλον του εγγενούς προγραμματισμού του iOS. Είναι κάτι περισσότερο από ένα πλαίσιο- είναι μια δήλωση της δέσμευσης της Apple να δώσει στους προγραμματιστές τα εργαλεία που χρειάζονται για να δημιουργήσουν την επόμενη γενιά εξαιρετικών εφαρμογών με επίκεντρο τον χρήστη. Καθώς κοιτάμε μπροστά, το SwiftUI στέκεται ως φάρος που καθοδηγεί το δρόμο προς ένα πιο αποτελεσματικό, καινοτόμο και φιλικό προς το χρήστη μέλλον για την ανάπτυξη εφαρμογών iOS.

Βιβλιογραφία

XCode : https://developer.apple.com/xcode/

MVVM: https://jayeshkawli.ghost.io/swiftui-in-mvvm-architecture/

iOS Layers: https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/#:~:text=Architecture%20of%20IOS%20is%20a,of%20well%20defined%20system%20interfaces

Διαφορές SwiftUI με UI Kit:
https://www.hackingwithswift.com/quick-start/swiftui/an-introduction-to-swiftui
https://swiftwithmajid.com/2020/09/30/swiftui-components-tutorial/
https://www.appcoda.com/swiftui-preview/
https://www.hackingwithswift.com/quick-start/swiftui/why-swiftui-is-a-game-changer-for-ios- development
https://swiftuibyexample.com/swiftui-performance/

Γλώσσα Swift: https://www.linkedin.com/pulse/what-swift-language-jay-tillu-uydyf/